Inference Using Simulated Neural Moments

Michael Creel

August 7, 2021

Universitat Autònoma de Barcelona, Barcelona GSE, and MOVE, Bellaterra (Barcelona) 08193, Spain. michael.creel@uab.cat.

Abstract

This paper studies method of simulated moments (MSM) estimators that are implemented using Bayesian methods, specifically Markov chain Monte Carlo (MCMC). Motivation and theory for the methods is provided by Chernozhukov and Hong (2003). The paper shows, by Monte Carlo examples, that confidence intervals using these methods may have coverage which is far from the nominal level, a result which has parallels in the literature that studies overidentified GMM estimators. A neural network may be used to reduce the dimension of an initial set of moments to the minimum number that maintains identification, as in Creel (2017). When MSM-MCMC estimation and inference is based on such moments, using a continuously updating criterion function, confidence intervals have statistically correct coverage in all cases studied. The methods are illustrated by application to several test models, including a small DSGE model, and to a jump diffusion model for returns of the S&P 500 index.

Keywords: neural networks; Laplace type estimators; simulated moments; jump diffusion

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, Heaton, and Yaron (1996), Donald, Imbens, and Newey (2009)). In another strand of the literature, 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. A third 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, Trabandt, and Walentin (2010) (see also Christiano, Eichenbaum, and Trabandt (2016)), which proposes a Laplace type estimation methodology that uses simulated moments which are defined in terms of impulse response functions for estimation of macroeconomic modes. Very similar methodologies may be found in the 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. Henceforth, this combination is referred to as MSM-MCMC, because the specific Laplace type method considered here is to use the criterion function of the MSM estimator to define the likelihood that determines acceptance/rejection in Metropolis-Hastings MCMC, as was the focus of Chernozhukov and Hong (2003). This paper provides evidence that confidence intervals derived from such estimators 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 paper concludes with an example that uses the methods to estimate a jump-diffusion model for returns of the S&P 500 index.

Section 2 reviews how Laplace type methods may be used with simulated moments, giving the MSM-MCMC combination, and Section 3 then discusses how neural networks may be used to reduce the dimension of the moment conditions. Section 4 presents four test models, and Section 5 gives results for these models. Section 6 illustrates the methods in the context of an empirical analysis of a model of more complexity, concretely, a jump-diffusion model for

financial returns, and Section 7 summarizes the conclusions. The SNM archive contains all of the code and results reported in this paper. These results were obtained using the Julia package SimulatedNeuralMoments.jl, which provides a convenient way to use the methods for other research projects.

2 Simulated Moments, Indirect Likelihood, and MSM-MCMC Inference

This section summarizes results from the part of the simulation-based estimation literature that bases estimation on a statistic, including McFadden (1989), Gouriéroux, Monfort, and Renault (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 $Z = Z(\theta) = Z(Y(\theta))$ upon which to base estimation, and assume that the statistic satisfies a central limit theorem, uniformly, for all values of θ of interest:

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

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

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

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

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

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

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, Trabandt, and Walentin (2010) and Christiano, Eichenbaum, and Trabandt (2016)). Another possibility is to (1) compute the estimate $\hat{\bar{\Sigma}}(\theta)$ of the covariance matrix in 1 as the sample covariance of R draws of $\sqrt{n}Z^s(\theta)$:

$$\hat{\bar{\Sigma}}(\theta) = \frac{1}{R}\sum_{r=1}^{R}(\sqrt{n}Z^r(\theta) - M)(\sqrt{n}Z^r(\theta) - M)', \tag{5}$$

where $M = \frac{1}{R}\sum_r \sqrt{n}Z^r(\theta)$ is the sample mean of the draws, and then (2) multiply the result by $1 + S^{-1}$ to obtain the estimate

$$\hat{\bar{V}}(\theta) = (1 + S^{-1})\hat{\bar{\Sigma}}(\theta). \tag{6}$$

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 methods[2]. This specific version of Laplace type methods is referred to as MSM-MCMC in this paper. This paper will rely directly on the theory and methods of Chernozhukov and Hong (2003), as MSM-MCMC falls within the class of methods they study. 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.

[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 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, MakeNeuralMoments.jl$W$, 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, according to results collectively known as the universal approximation theorem, will be a very accurate approximation to the posterior mean of θ conditional on W (Hornik, Stinchcombe, and White (1989), Lu et al. (2017)). 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, Imbens, and Newey (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

[3]The function which specifies and trains the neural net is MakeNeuralMoments.jl

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 be unlikely to suceed. However, derivative free methods 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. A major motivation of using Laplace-type estimators in the first place is to overcome problems of local extrema, as Chernozhukov and Hong (2003) emphasize. It is worth noting that the output of the net evaluated at the real sample statistic, $\hat{\theta}$, will also provide an excellent starting value for computing extremum estimators, or for initializing a MCMC chain. Likewise, the covariance estimator of eqn. 6 can be used to define a random walk multivariate normal proposal density for MCMC, by drawing the trial value θ^{s+1} from $N(\theta^s, \hat{\bar{V}})$, where θ^s is the current value of the chain. Experience with this proposal density, as reported below, is that it is easy to tune, by scaling the covariance by a scalar, to achieve an acceptance rate withing the desired limits[4]

Creel (2017) used neural moments to compute a Laplace-type estimator, similarly to what is done here. That paper used nonparametric regression quantiles applied to the set of draws from the Laplace-type posterior in order to compute confidence intervals, and the posterior draws were generated by a procedure similar to sequential Monte Carlo, rather than MCMC. Also, the metric used for selection of particles was different from the GMM criterion, which is what is used here. The use of nonparametric regression quantiles is very costly to study by Monte Carlo. Thus, this paper focuses on straightforward use of the methods that Chernozhukov and Hong (2003) focus on: traditional MCMC using the GMM criterion function, with confidence intervals computed by using the direct quantiles from the posterior sample. These simplifications give a simpler and more tractable procedure that can reasonably be studied and verified by Monte Carlo. For theoretical support, we can note that the methods fall within the class of methods studied by Chernozhukhov and Hong, with the only innovation being the use of statistics filtered through a previously trained neural net. The neural nets used here consist of a finite series of nonstochastic nonlinear mappings to the (-1,1) interval, followed by a final linear transformation. As such, the conjecture that the final statistics that are the output of the net follow a uniform law of large numbers and a uniform central limit theorem seems reasonable, but this is not formally verified in this paper.

[4] See the file MCMC.jl for the details of how this proposal density is implemented.

4 Examples

This section presents example models that are used to investigate the performance of the proposed methods. For all models, the code used (for the Julia Julia language) is available in an archive[5], where the details of each example may be consulted. The 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. For the examples, uniform and fairly uninformative priors were used in all cases. The details regarding priors and statistics, W, may be consulted in the links provided, below.

4.1 Stochastic Volatility

The simple stochastic volatility (SV) 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 same 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 $|y|$.[6]

4.2 ARMA

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

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

with true values $\theta_0 = (\alpha_0, \beta_0, \sigma_0^2) = (0.95,\ 0.5,\ 1.0)$. The sample size is $n = 300$. 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 another AR(1) model fit to the

[5] https://github.com/mcreel/SNM

[6] See the file SVlib.jl for details.

residuals of the first model, plus partial autocorrelations of x_t[7]

4.3 Mixture of Normals

For the mixture of normals example, the variable y is drawn from the distribution $N(\mu_1, \sigma_1^2)$ with probability p and from $N(\mu_1-\mu_2, \sigma_1^2+\sigma_2^2)$ with probability $1-p$. Samples of 1000 observations are drawn. The true parameter values are $\theta_0 = (\mu_1, \sigma_1, \mu_2, \sigma_2, p) = (1.0, 1.0, 0.2, 1.8, 0.4)$, and the prior restricts all parameters to be positive. Thus, the parameterization and the prior together impose that the first component has a larger mean and a lower variance than does the second component, in order to ensure identification. Also, the probability that either component is sampled is restricted to be at least 0.05. The 15 auxiliary statistics are the sample mean, standard deviation, skewness, kurtosis,, and 11 quantiles of y.[8]

4.4 DSGE Model

The previous models are all simple, quickly simulated, and with relatively few parameters. This section presents a model which is more representative of an actual research problem. The model is a simple dynamic stochastic general equilibrium model with two shocks:

At the beginning of period t, the representative household owns a given amount of capital (k_t), and chooses consumption (c_t), investment (i_t) and hours of labor (n_t) to maximize expected discounted utility

$$E_t \sum_{s=0}^{\infty} \beta^s \left(\frac{c_{t+s}^{1-\gamma}}{1-\gamma} + (1-n_{t+s})\eta_t \psi \right)$$

subject to the budget constraint $c_t + i_t = r_t k_t + w_t n_t$, available time $0 \leq n_t \leq 1$, and the accumulation of capital $k_{t+1} = i_t + (1-\delta)k_t$, each of which must hold for all t. The shock, η_t, that affects the desirability of leisure relative to consumption, evolves according to $\ln \eta_t = \rho_\eta \ln \eta_{t-1} + \sigma_\eta \epsilon_t$.

The single competitive firm maximizes profits $y_t - w_t n_t - r_t k_t$ from production of the good (y_t), taking wages (w_t) and the interest rate (r_t) as given, using the constant returns to scale technology

$$y_t = k_t^\alpha n_t^{1-\alpha} z_t. \tag{7}$$

The technology shock, z_t, also follows an AR(1) process in logarithms: $\ln z_t = \rho_z \ln z_{t-1} + \sigma_z u_t$. The innovations to the preference and technology shocks, ϵ_t and u_t, are independent standard normal random variables. Production (y_t) can be allocated by the consumer to consumption or investment: $y_t = c_t + i_t$. The consumer provides capital and labor to the firm, and is paid

[7] Details are in the file ARMAlib.jl.
[8] Details are in the file MNlib.jl.

at the competitive rates r_t and w_t, respectively.

From this model, samples of size 160, which simulate 40 years of quarterly data, are drawn, given the 9 parameters $\alpha, \beta, \gamma, \delta, \rho_z, \sigma_z, \rho_\eta, \sigma_\eta$ and ψ. The variables available for estimation are y, c, n, w, and r. It is possible to recover the parameters α and δ exactly, given the observable variables, so these two parameters are set to fixed values, and the remaining 7 parameters are estimated. To facilitate setting priors, the steady state value of hours (n) is estimated instead of ψ, which may then be recovered. For estimation, 45 statistics are used, including means and standard deviations of the observable variables, and estimates from auxiliary regressions. [9].

5 Monte Carlo Results

This section reports results for MSM-MCMC estimation of each of the test models, using the GMM-like criterion function H (eqn. 4) as the L_n of Chernozhukov and Hong (2003). Results using the criterion L (eqn. 3) were qualitatively very similar in all cases where the two versions were computed, and are thus not reported[10]. In all cases, 500 Monte Carlo replications were done. For all of the test models, the number of artificial samples used to train the neural net was 20,000 times the number of parameters of the model. This is actually a fairly small number, given that generating the samples and training the nets is an operation that takes only 10 minutes or less for the test models, other than the DSGE model. The reason that a larger number of samples was not used is that it was desired to obtain results that may be more relevant for cases where it is more costly to simulate from the model, as is the case of the jump diffusion model studied below.

First, we report results for the SV and ARMA models, where MSM-MCMC estimators were computed using both the overidentifying statistic, W, and the exactly identifying neural moments, Z. For the plain overidentifying statistics, the results are computed using the CUE GMM criterion. For the neural moments, both the two-step and the CUE criteria were used. Table 1 reports RMSE. The three versions of the MSM-MCMC estimators lead to similar RMSEs, generally speaking. The version that uses the raw statistics has somewhat higher RMSE than do the versions based on the neural statistics, Z, in most cases, but the differences are not important.

Tables 2 through 4 address the main point of the paper, inference, reporting confidence interval coverage, which is the proportion of times that the true parameter lies inside the computed confidence interval. Critical coverage proportions that would lead one to reject

[9]The details of the model and priors may be seen at CKlib.jl. The model is solved using third order projection, making use of the SolveDSGE.jl package. The model is discussed in more detail in Chapter 14 of the document econometrics.pdf.

[10]These results are available for the SV and ARMA models, as well as an unreported additional model, in the WP branch of the github archive.

correct coverage may be computed from the binomial(500, p) distribution, where p is the significance level associated with the respective confidence interval. These critical coverage proportions are 0.864 and 0.932 for 90% intervals, 0.924 and 0.974 for 95% intervals, and 0.976 - 1.0 for 99% intervals. Looking at the column labeled W (CUE) in these tables, we see that the results on the unreliability of inferences for overidentified GMM estimators, which were reviewed in the Introduction, carry over to Bayesian MCMC methods, at least for the models considered. In all entries but one, the coverage is significantly different from correct coverage, erring on the side of being too low, and, in many cases, considerably so. This implies that the probability of Type-I error is higher than the associated nominal significance level. For the neural net statistics, Z, coverage is improved. For the two step version, coverage is in all cases closer to the correct proportion than when the raw statistics, W, are used. In a number of cases, correct coverage is also statistically rejected, but now, the error is on the side of conservative confidence intervals, which contain the true parameters more often than the nominal coverage. In this case, the probability of Type-I error will be less than the nominal significance level associated with the confidence intervals. For the CUE version that uses the neural statistics, Z, coverage is very good, and is close to the nominal proportion in all cases. Statistically correct coverage is never rejected when neural moments and the CUE criterion are used.

Table 1: RMSE for SV and ARMA models, using raw (W) or neural net (Z) statistics.

Model	Parameter	true value	W (CUE)	Z (two-step)	Z (CUE)
SV	ϕ	0.692	0.123	0.064	0.076
	ρ	0.90	0.086	0.082	0.086
	σ	0.363	0.138	0.105	0.105
ARMA	α	0.95	0.030	0.028	0.047
	β	0.5	0.078	0.067	0.068
	σ^2	1.0	0.099	0.091	0.084

Table 2: 90% confidence interval coverage for SV and ARMA models, using raw (W) or neural net (Z) statistics. Correct coverage rejected when outside 0.864 - 0.932.

Model	Parameter	W (CUE)	Z (two step)	Z(CUE)
SV	ϕ	0.876	0.884	0.912
	ρ	0.732	0.976	0.910
	σ	0.762	0.956	0.928
ARMA	α	0.786	0.988	0.916
	β	0.814	0.954	0.918
	σ^2	0.808	0.920	0.912

For the other two test models, MN and DSGE, results were computed only for the neural moments, as the results for the SV and ARMA models, as well as the results from the GMM

Table 3: 95% confidence interval coverage for SV and ARMA models, using raw (W) or neural net (Z) statistics. Correct coverage rejected when outside 0.924 - 0.974

Model	Parameter	W (CUE)	Z (two step)	Z (CUE)
SV	ϕ	0.916	0.938	0.954
	ρ	0.796	0.990	0.944
	σ	0.824	0.976	0.958
ARMA	α	0.838	0.994	0.966
	β	0.856	0.984	0.942
	σ^2	0.880	0.966	0.954

Table 4: 99% confidence interval coverage for SV and ARMA models, using raw (W) or neural net (Z) statistics. Correct coverage rejected when outside 0.976 - 1.000.

Model	Parameter	W (CUE)	Z (two step)	Z (CUE)
SV	ϕ	0.936	0.968	0.990
	ρ	0.848	0.998	0.978
	σ	0.888	0.994	0.986
ARMA	α	0.898	1.000	0.988
	β	0.916	0.998	0.986
	σ^2	0.920	0.994	0.990

literature, already indicated that inferences based on the raw statistics, W, were very likely to be unreliable. Table 5 has the RMSE results for the MN and DSGE models. We can see that the use of the two step or CUE criterion makes little difference for RMSE, in common with the above results for the SV and ARMA models. Tables 6 - 8 hold the confidence interval coverage results for these two models. Again, the intervals based on the two step criterion often contain the true parameters more often than they should. The coverage of intervals based on the CUE criterion is very good in all cases, and is never statistically significantly different from correct.

In summary, this section has shown that confidence intervals based on raw overidentifying statistics may be unreliable, rejecting the true parameter values more often than they should. Intervals based on exactly identifying neural moments are more reliable, in general. When the two step version is used, the intervals are often too broad, so the probability of Type-I error is less than what it should be, and power to reject false hypotheses is lower than it could be. When the CUE version is used, coverage is very accurate: correct coverage was never rejected in any of the cases.

It is to be noted that the CUE version is computationally more demanding than is the two step version, as the weight matrix must be estimated at each MCMC trial vector. Each of these estimations requires a reasonably large number of simulations to be drawn, to estimate the covariance matrix accurately. If a researcher is primarily concerned with limiting the

Table 5: RMSE for MN and DSGE models

Model	Parameter	true value	Z (two step)	Z (CUE)
MN	μ_1	1.0	0.019	0.018
	σ_1	0.2	0.087	0.089
	μ_2	0.0	0.021	0.020
	σ_2	2.0	0.064	0.065
	p	0.4	0.024	0.025
DSGE	β	0.99	0.001	0.000
	γ	2.00	0.083	0.085
	ρ_z	0.9	0.009	0.008
	σ_z	0.02	0.001	0.001
	ρ_η	0.7	0.050	0.055
	σ_η	0.01	0.001	0.001
	$n\bar{s}s$	1/3	0.001	0.001

probability of Type-I error, and is willing to accept a loss of power in order to accelerate computations, then the two step version might be preferred. If one is willing to accept more costly computations, all of the examples considered here indicate that the CUE version will lead to accurate confidence intervals.

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

The previous examples are mostly small models that are not costly to simulate, with the exception of the DSGE example. As an example of a more computationally challenging model that may be more representative of actual research problems, this section presents results for estimation of a jump diffusion model of S&P 500 returns. Solving and simulating[11] the model for each MCMC trial parameter acceptance/rejection decision takes about 30 seconds, when the CUE criterion is used, so training a net and estimation by MCMC is somewhat costly, requiring approximately 5 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 moderately complex models.

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

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

Table 6: 90% confidence interval coverage for MN and DSGE models. Correct coverage rejected when outside 0.864 - 0.932.

Model	Parameter	Z (two step)	Z (CUE)
MN	μ_1	0.920	0.914
	σ_1	0.934	0.922
	μ_2	0.906	0.918
	σ_2	0.934	0.920
	p	0.922	0.908
DSGE	β	0.950	0.914
	γ	0.968	0.920
	ρ_z	0.928	0.928
	σ_z	0.910	0.892
	ρ_η	0.892	0.890
	σ_η	0.972	0.906
	$n\bar{s}s$	0.924	0.902

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 10 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. Both RV and BV are informative about volatility , and, because BV is somewhat robust to jumps, while RV is not, the difference between the two can help to identify the frequency and size of jumps (Barndorff-Nielsen and Shephard (2002)). 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 model studied here is quite similar to that studied in Creel and Kristensen (2015) and Creel (2017), except that the drift process is simplified to be constant, and the jump process is modeled somewhat differently, with constant intensity, and with the magnitude of a jump depending on the current instantaneous volatility. These changes were motivated by the results of the previous papers, and by the better tractability of the present specification.

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

Table 7: 95% confidence interval coverage for MN and DSGE models. Correct coverage rejected when outside 0.924 - 0.974.

Model	Parameter	Z (two step)	Z (CUE)
MN	μ_1	0.956	0.962
	σ_1	0.976	0.962
	μ_2	0.944	0.952
	σ_2	0.964	0.958
	p	0.960	0.958
DSGE	β	0.972	0.962
	γ	0.990	0.962
	ρ_z	0.960	0.958
	σ_z	0.952	0.946
	ρ_η	0.950	0.938
	σ_η	0.996	0.952
	$n\bar{s}s$	0.966	0.956

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[12]. The details of the 25 statistics which are used are found in the file JDlib.jl (this same file also gives details of the priors, which are uniform over fairly broad supports, for all parameters). The importances of the statistics are seen in Figure 1. We can observe that statistics 1,4 and 9 appear to have little importance, and could potentially be excluded. This was not done, however, as experience with the test models, reported in the previous section, has shown that the neural net is effective at learning which statistics to ignore, and leaving in the uninformative statistics does seems to impact neither goodness of fit nor the reliability inferences. The remaining 22 statistics all have standardized weights of roughly 1.0 or higher, and, thus, do have an impact on the output of the neural net. Statistic 14, which is the kurtosis of returns (see JDlib.jl), is highly informative. This statistic should be informative for the parameters α and σ. Statistic 16 is also highly informative. This statistic is the kurtosis of the residuals of an artificial regression designed to detect jumps. As such, this statistic should be informative for the parameters λ_0 and λ_1. One can analyze this sort of information to try to ensure that each of the model's parameters is well identified.

The model was fit, using MSM-MCMC and the CUE criterion, to S&P500 data[13] from 16 Dec. 2013 to 05 Dec. 2017, which is an interval of 1000 trading days, the same as was used to train the neural net. The data may be seen in Figure 2, where we observe typical volatility clusters and some jumps. For example, the Brexit drop of June, 2016 is clearly seen, and the more extreme spike in RV versus BV at this point illustrates the fact that jumps can be

[12] See the file Importance.jl for how to do this, as well as discussion in Creel (2017).

[13] The data source is the Oxford-Man Institute's realized library, v. 0.2, https://realized.oxford-man.ox.ac.uk/images/oxfordmanrealizedvolatilityindices-0.2-final.zip

Table 8: 99% confidence interval coverage for MN and DSGE models. Correct coverage rejected when outside 0.976 - 1.000.

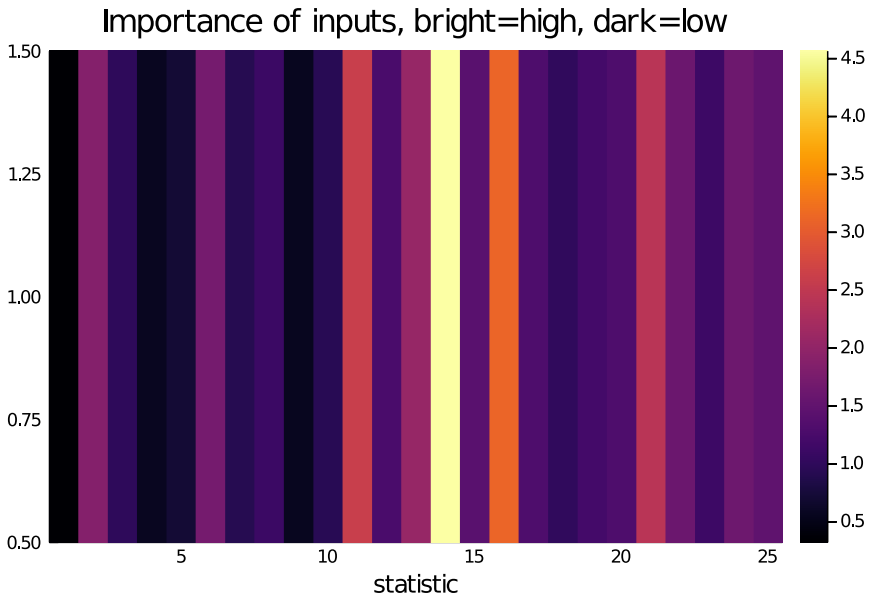
Model	Parameter	Z (two step)	Z (CUE)
MN	μ_1	0.990	0.990
	σ_1	0.996	0.992
	μ_2	9.986	0.984
	σ_2	0.992	0.994
	p	0.996	0.986
DSGE	β	0.996	0.990
	γ	1.000	0.986
	ρ_z	0.976	0.980
	σ_z	0.986	0.990
	ρ_η	0.990	0.982
	σ_η	1.000	0.992
	$n\bar{s}s$	0.988	0.988

identified by comparing the two.

Twenty MCMC chains of length 1000 were drawn independently using threads-based parallelization, for a final chain of length 20,000. The estimation results are in Figure 3, which shows 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. All posteriors are considerably more concentrated than are the priors. Drift (μ) is not significantly different from zero. There is quite a bit of persistence in volatility, as mean reversion, κ, is estimated to be quite low, about 0.075. Leverage (ρ) is quite strong, estimated to be about -0.8. The jump probability per day (λ_0) is estimated to be about 0.036, and is significantly different from zero. So, jumps are a statistically important feature of the model. When a jump does occur, its magnitude (λ_1) is approximately 3.1 times the current instantaneous standard deviation. An interesting result is that τ, the standard deviation of measurement error of log price, is estimated to be approximately 0.028, and it is significantly different from zero. Thus, it appears that it is a safer option to allow for measurement error in the model, as its omission could bias the estimates of the other parameters.

Creel (2017) uses similar methods to analyze the S&P 500 data over the Jan. 2015 - May 2016 interval. That paper found that mean volatility (α) was higher than the estimate here, the variance of volatility (σ) was lower than that found here, and that mean reversion (κ) was faster than that estimated here. These results are consistent with what is seen in Figure 2. The Jan. 2015 - May 2016 interval is one of relatively high volatility, with less pronounced clusters. For this shorter data window, volatility is higher on average and less variable, with less clustering. The differences in the estimated parameters between the earlier paper and the results here are consistent with these facts.

Figure 1: Jump-diffusion model, importances of raw statistics, W



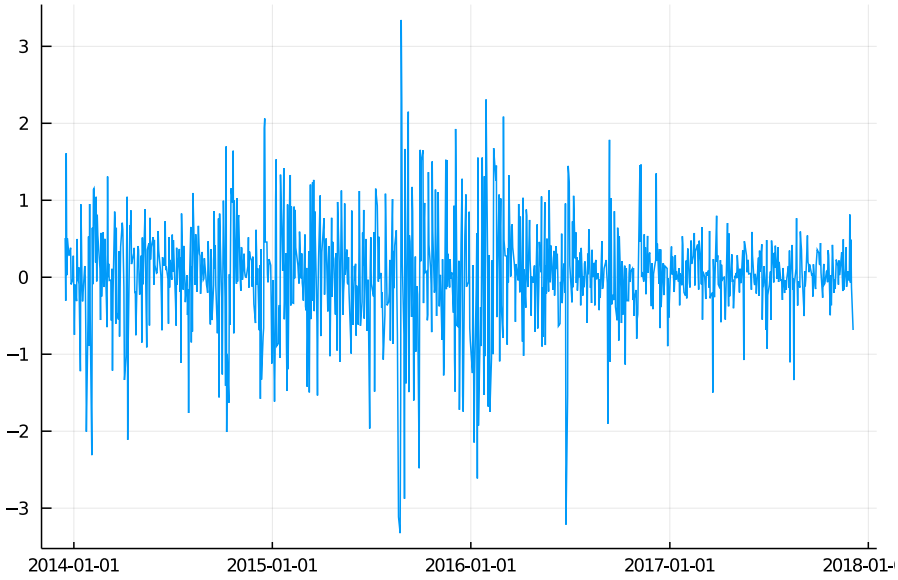
7 Conclusions

This paper has shown, through Monte Carlo experimentation, that confidence intervals based upon quantiles of a tuned MCMC chain may have coverage which is far from the nominal level, even for simple models with few parameters. The results on poor reliability inferences when using overidentified GMM estimators, that were referenced in the Introduction, seem to carry over to a Bayesian version of overidentified MSM, implemented using the Chernozhukov and Hong (2003) methodology. This methodology is what we have termed MSM-MCMC in this paper. The paper proposes to use neural networks to reduce the dimension of an initial set of moments to the minimum number of moments needed to maintain identification, following Creel (2017). When estimation and inference using the MSM-MCMC methodology is based on neural moments, which are exactly identifying, confidence intervals have statistically correct coverage in all cases studied by Monte Carlo, when the CUE version of MSM is used. Thus, there seems to be no generic problems with the MSM-MCMC methodology for the purpose of inference. The problem has to do with the choice of moments upon which MSM is based. Too much overidentification seems to result in poor inferences, for the models studied. The use of neural moments solves this problem, by reducing the number of moments, without loosing the information that they contain. The fact that RMSE does not rise when one moves from raw to neural moments illustrates that the neural moments do not loose the information that is contained in the larger set. The methods have been illustrated by the estimation of a jump diffusion model for S&P 500 data.

It is to be noted that the step of filtering moments though a neural net is very easy and

Figure 2: Plot of returns, RV and BV, S&P 500, 16 Dec. 2013 - 05 Dec. 2017.

(a) Returns



(b) RV and BV

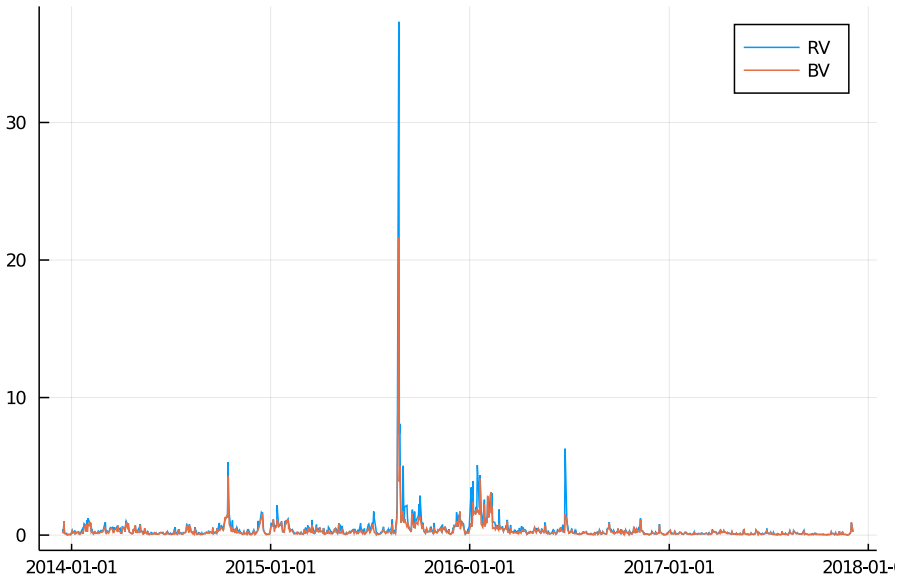
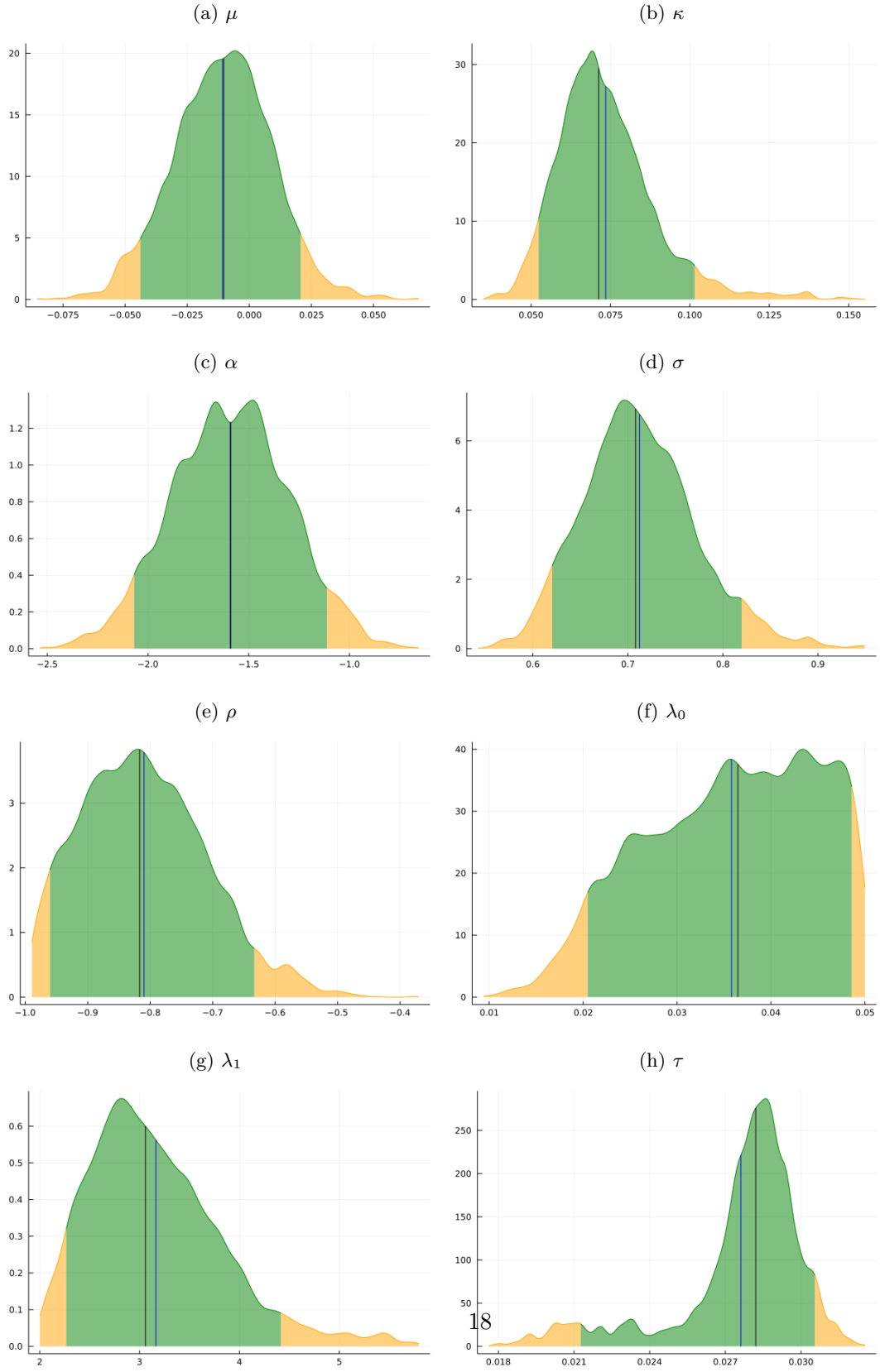


Figure 3: MCMC results for the jump-diffusion model of S&P 500 data. Posterior mean in blue, posterior median in black. The green-yellow borders define the limits of a 90% confidence interval.



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 functions that provide simulated moments computed using data drawn from the model at parameter values drawn from the prior. Filtering moments through a neural net gives an informative, minimal dimension statistic as the output. This provides a convenient and automatic alternative to moment selection procedures. Uninformative moments are essentially removed, and correlated moments are combined.

This paper has examined how inference using quantiles of traditional 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 Hamiltonian Monte Carlo and sequential Monte Carlo, among others, may be made more reliable if neural moments are used, as dimension reduction while maintaining relevant information is likely to be generally beneficial.

Support from the Spanish Ministry of Science, Innovation and Universities and FEDER through a Severo Ochoa Center of Excellence award to the Barcelona GSE and through grant PGC2018-094364-B-I00 is gratefully acknowledged.

References

Barndorff-Nielsen, Ole E and Neil Shephard (2002). "Econometric analysis of realized volatility and its use in estimating stochastic volatility models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 64.2, pp. 253–280.

Chernozhukov, Victor and Han Hong (Aug. 2003). "An MCMC approach to classical estimation". In: *Journal of Econometrics* 115.2, pp. 293–346. DOI: 10.1016/s0304-4076(03)00100-3. URL: http://dx.doi.org/10.1016/s0304-4076(03)00100-3.

Christiano, Lawrence J, Martin S Eichenbaum, and Mathias Trabandt (2016). "Unemployment and business cycles". In: *Econometrica* 84.4, pp. 1523–1569.

Christiano, Lawrence J, Mathias Trabandt, and Karl Walentin (2010). "DSGE models for monetary policy analysis". In: *Handbook of Monetary Economics*. Vol. 3. Elsevier, pp. 285–367.

Corsi, Fulvio (2009). "A simple approximate long-memory model of realized volatility". In: *Journal of Financial Econometrics* 7.2, pp. 174–196.

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

Creel, Michael and Dennis Kristensen (2015). "ABC of SV: Limited information likelihood inference in stochastic volatility jump-diffusion models". In: *Journal of Empirical Finance* 31, pp. 85–108. ISSN: 0927-5398. DOI: http://dx.doi.org/10.1016/j.jempfin.2015.01.002. URL: http://www.sciencedirect.com/science/article/pii/S0927539815000031.

Donald, Stephen G., Guido W. Imbens, and Whitney K. Newey (2009). "Choosing instrumental variables in conditional moment restriction models". In: *Journal of Econometrics* 152.1. Recent Adavances in Nonparametric and Semiparametric Econometrics: A Volume Honouring Peter M. Robinson, pp. 28–36. ISSN: 0304-4076. DOI: 10.1016/j.jeconom.2008.10.013. URL: http://www.sciencedirect.com/science/article/pii/S0304407609000566.

Gallant, A. Ronald and George Tauchen (1996). "Which moments to match?" In: *Econometric Theory* 12, pp. 363–390. DOI: 10.2139/ssrn.37760. URL: http://dx.doi.org/10.2139/ssrn.37760.

— (2002). "EMM: A program for efficient method of moments estimation, Version 1.6, User's Guide". In: *Manuscript, University of North Carolina*.

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

Hall, Peter and Joel L Horowitz (1996). "Bootstrap critical values for tests based on generalized-method-of-moments estimators". In: *Econometrica*, pp. 891–916.

Hansen, Lars Peter, John Heaton, and Amir Yaron (July 1996). "Finite-sample properties of some alternative GMM estimators". In: *Journal of Business & Economic Statistics* 14.3, pp. 262–280. DOI: 10.1080/07350015.1996.10524656. URL: http://dx.doi.org/10.1080/07350015.1996.10524656.

Hornik, K., M. Stinchcombe, and H. White (July 1989). "Multilayer feedforward networks are universal approximators". In: *Neural Netw.* 2.5, pp. 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8. URL: http://dx.doi.org/10.1016/0893-6080(89)90020-8.

Jiang, Wenxin and Bruce Turnbull (2004). "The indirect method: inference based on intermediate statistics a synthesis and examples". In: *Statistical Science* 19.2, pp. 239–263.

Lu, Zhou et al. (2017). "The expressive power of neural networks: A view from the width". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6232–6240. URL: https://papers.nips.cc/paper/2017/file/32cbf687880eb1674a07bf717761dPaper.pdf.

Marjoram, Paul et al. (2003). "Markov chain Monte Carlo without likelihoods". In: *Proceedings of the National Academy of Sciences* 100.26, pp. 15324–15328.

McFadden, Daniel (1989). "A method of simulated moments for estimation of discrete response models without numerical integration". In: *Econometrica* 57.5, pp. 995–1026. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/1913621.

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

Tauchen, George (Oct. 1986). "Statistical properties of generalized method-of-moments estimators of structural parameters obtained from financial market data". In: *Journal of Business & Economic Statistics* 4.4, p. 397. DOI: 10.2307/1391493. URL: http://dx.doi.org/10.2307/1391493.