DOI: 10.26509/frbc-wp-201617corrigendum

Corrigendum to: Large Bayesian Vector Autoregressions with Stochastic Volatility and Non-Conjugate Priors[*]

Andrea Carriero[†] Joshua Chan[‡] Todd E. Clark[§] Massimiliano Marcellino[¶]

November 2021

Abstract

The original algorithm contained a mistake that meant the conditional distributions used for the VAR's coefficients were missing a piece of information. We propose a new algorithm that uses the same factorization but includes the missing term. The new, correct algorithm has the same computational complexity as the old, incorrect one (i.e., $\mathcal{O}(N^4)$), and therefore it still allows the estimation of large VARs.

J.E.L. classifications: C11, C13, C33, C53

Keywords: Big data, forecasting, structural VAR

1 The problem

Consider an N-variable vector autoregression with stochastic volatility

$$y_t = \Pi' x_t + A^{-1} \Lambda_t^{0.5} \epsilon_t, \tag{1}$$

where $t = 1, ..., T$, y_t is an N-dimensional vector, p denotes the lag order, x_t is a $k = Np + 1$ vector containing the lags of y_t and an intercept, $\Pi = (\Pi_0, \Pi_1, ..., \Pi_p)'$ is a $k \times N$ matrix of coefficients, A^{-1} is a lower triangular matrix with ones on its main diagonal, Λ_t is

[*] The views expressed herein are solely those of the authors and do not necessarily reflect the views of the Federal Reserve Bank of Cleveland or the Federal Reserve System.

[†] Queen Mary University of London: School of Economics and Finance, Mile End Road E1 4NS, London, United Kingdom; a.carriero@qmul.ac.uk

[‡] Purdue University: Department of Economics, 100 Grant St., West Lafayette, IN 47907, United States; joshuacc.chan@gmail.com

[§] (Corresponding author) Federal Reserve Bank of Cleveland: Economic Research Department, P.O. Box 6387, Cleveland, OH 44101, United States; todd.clark@researchfed.org

[¶] Bocconi University, IGIER, and CEPR: Department of Economics, Via Roentgen 1, 20136, Milano, Italy; massimiliano.marcellino@unibocconi.it

a diagonal matrix with generic j-th element $\lambda_{j,t}$, and $\epsilon_t \sim iid\ N(0, I_N)$. The reduced-form error covariance matrix is $\Sigma_t = A^{-1}\Lambda_t A^{-1\prime}$. Below we will also make use of the lower-triangular Cholesky factor of Σ_t, with elements $\sigma^*_{j,i,t}$. In this note, although we focus on the model with stochastic volatility, the basic results also apply to a homoskedastic version of the model with non-conjugate priors.

The algorithm used in Carriero, Clark, and Marcellino (2019) aimed to simulate the conditional joint posterior distribution of the VAR autoregressive coefficients Π:

$$vec(\Pi)|y, \Sigma_T \sim N(\bar{\mu}_\Pi, \overline{\Omega}_\Pi), \tag{2}$$

where Σ_T contains the entire history of the error variance matrix Σ_t and y contains the entire history of the observables y_t. The algorithm intended to do so by using the following factorization of the joint distribution:

$$\begin{aligned} p(\Pi \,|\, y, \Sigma_T) \quad = \quad & p(\pi^{(1)} \,|\, y, \Sigma_T) \\ & \times p(\pi^{(2)} \,|\, \pi^{(1)}, y, \Sigma_T) \\ & \vdots \\ & \times p(\pi^{(N)} \,|\, \pi^{(N-1)}, \ldots, \pi^{(1)}, y, \Sigma_T), \end{aligned} \tag{3}$$

with generic element

$$p(\pi^{(j)} \,|\, \pi^{(j-1)}, \ldots, \pi^{(1)}, y, \Sigma_T), \tag{4}$$

where $\pi^{(j)}$ denotes the vector of coefficients for equation j, appearing in the j-th column of the matrix

$$\Pi = \left[\begin{array}{ccccccc} \pi^{(1)} & \vdots & \pi^{(j-1)} & \pi^{(j)} & \pi^{(j+1)} & \vdots & \pi^{(N)} \end{array} \right].$$

This triangular factorization breaks down the joint distribution into a sequence of distributions of the generic coefficients $\pi^{(j)}$, i.e., the coefficients appearing in the j-th equation of the VAR. Therefore, an algorithm drawing in sequence from the distribution at the top of expression (3) and proceeding step by step toward the last distribution at the bottom will obtain the desired joint distribution.

In the paper, this strategy was pursued by drawing $\pi^{(j)}$ from a Gaussian distribution with mean $\bar{\mu}_{\Pi\{j\}} = \overline{\Omega}_{\Pi\{j\}} \left\{ \underline{\Omega}^{-1}_{\Pi\{j\}} \underline{\mu}_{\Pi\{j\}} + \sum_{t=1}^T x_t \sigma^{*-2}_{j,j,t} y^*_{j,t} \right\}$ and variance $\overline{\Omega}^{-1}_{\Pi\{j\}} = \underline{\Omega}^{-1}_{\Pi\{j\}} + \sum_{t=1}^T x_t \sigma^{*-2}_{j,j,t} x'_t$, where $\underline{\mu}_{\Pi\{j\}}$ and $\underline{\Omega}^{-1}_{\Pi\{j\}}$ denote, respectively, the prior mean and prior variance of $\pi^{(j)}$, and $y^*_{j,t} = y_{j,t} - (\sigma^*_{j,1,t}\epsilon_{1,t} + \cdots + \sigma^*_{j,j-1,t}\epsilon_{j-1,t})$.[1] However, this distribution

[1] In the published version of the paper, the inverse error variance $\sigma^{*-2}_{j,j,t}$ was erroneously written as $\sigma^{*-1}_{j,j,t}$. The code and application results did not reflect this typo.

does not have p.d.f. (4), a fact pointed out in Bognanni (2021). Instead, the distribution used has p.d.f.

$$p(\pi^{(j)} \,|\, \pi^{(j-1)}, \dots, \pi^{(1)}, y^{(1)}, ..., y^{(j-1)}, x, \Sigma_T), \tag{5}$$

where $y^{(j)}$ denotes the time series of the j-th variable and x contains the entire history of the variables x_t that appear on the right-hand side of each equation of the VAR. Therefore, the distribution generated by the algorithm described in the paper is:

$$\begin{aligned} & p(\pi^{(1)} \,|\, y^{(1)}, x, \Sigma_T) \\ & \times p(\pi^{(2)} \,|\, \pi^{(1)}, y^{(1)}, y^{(2)}, x, \Sigma_T) \\ & \vdots \\ & \times p(\pi^{(N)} \,|\, \pi^{(N-1)}, \dots, \pi^{(1)}, y^{(1)}, y^{(2)}, ..., y^{(N)}, x, \Sigma_T), \end{aligned} \tag{6}$$

and not (3).

The difference between (3) and (6) lies in the conditioning sets for the j-th equation coefficients $\pi^{(j)}$, which in (3) includes all the dependent variables $(y^{(1)}, y^{(2)} ..., y^{(N)})$, but in (6) includes the dependent variables only up to equation j $(y^{(1)}, y^{(2)} ..., y^{(j)})$. This difference implies that a term goes missing, involving the information about $\pi^{(j)}$ contained in the most recent observations of the dependent variables of equations $j+1, ..., N$. To see this more clearly, consider the simplest case $N = 2$ with the following triangular system:

$$\begin{aligned} y_{1,t} &= x_t' \pi^{(1)} + \sigma^*_{1,1,t} \epsilon_{1,t} \\ y_{2,t} &= x_t' \pi^{(2)} + \sigma^*_{2,1,t} \epsilon_{1,t} + \sigma^*_{2,2,t} \epsilon_{2,t}. \end{aligned}$$

The likelihood is $p(y \,|\, \pi, x, \Sigma_T) \propto p(\epsilon_2 |\, \epsilon_1) p(\epsilon_1) \propto p(y^{(1)} \,|\, \pi^{(1)}, x, \Sigma_T) p(y^{(2)} \,|\, y^{(1)}, \pi^{(1)}, \pi^{(2)}, x, \Sigma_T)$, where ϵ_j denotes the entire time series of the j-th disturbance. Assuming an independent prior $p(\pi^{(1)}, \pi^{(2)}) = p(\pi^{(1)}) p(\pi^{(2)})$, the joint posterior distribution of $(\pi^{(1)}, \pi^{(2)})$ is:

$$\begin{align} & p(\pi^{(1)}, \pi^{(2)} \,| y, x, \Sigma_T) \nonumber \\ & \propto p(y^{(1)} \,|\, \pi^{(1)}, x, \Sigma_T) p(\pi^{(1)}) \times p(y^{(2)} \,|\, y^{(1)}, \pi^{(1)}, \pi^{(2)}, x, \Sigma_T) p(\pi^{(2)}) \nonumber \\ & = p(\pi^{(1)} \,|\, y^{(1)}, x, \Sigma_T) p(y^{(1)} | x, \Sigma_T) \times p(\pi^{(2)} |\, y^{(1)}, y^{(2)}, \pi^{(1)}, x, \Sigma_T) \underline{p(y^{(2)} \,|\, y^{(1)}, \pi^{(1)}, x, \Sigma_T)} \nonumber \\ & \propto p(\pi^{(1)} \,|\, y^{(1)}, x, \Sigma_T) \underline{p(y^{(2)} \,|\, y^{(1)}, \pi^{(1)}, x, \Sigma_T)} \times p(\pi^{(2)} |\, y^{(1)}, y^{(2)}, \pi^{(1)}, x, \Sigma_T) \tag{7} \\ & = p(\pi^{(1)} | y^{(1)}, y^{(2)}, x, \Sigma_T) p(y^{(2)} | y^{(1)}, x, \Sigma_T) \times p(\, \pi^{(2)} |\, y^{(1)}, y^{(2)}, \pi^{(1)}, x, \Sigma_T) \nonumber \\ & \propto p(\pi^{(1)} | y^{(1)}, y^{(2)}, x, \Sigma_T) \times p(\, \pi^{(2)} |\, y^{(1)}, y^{(2)}, \pi^{(1)}, x, \Sigma_T). \tag{8} \end{align}$$

Note that (8) coincides with expression (3), whereas (7) consists of expression (6) *times* the underlined term $\underline{p(y^{(2)} \,|\, y^{(1)}, \pi^{(1)}, x, \Sigma_T)}$, which is therefore missed by the algorithm used in the paper. The missing term contains information about $\pi^{(1)}$ embedded in $y^{(2)}$. Since all the lagged values of all of the variables of the VAR (denoted x) are already in the conditioning set, the extra information that gets lost is that contained in the contemporaneous values $y_t^{(2)}$. This information must be included in order to obtain the target distribution via the correct factorization (3).

2 The solution

The above derivations show one way forward to correct the paper's original algorithm: We could simply add back the missing terms. This is in principle doable, but for general N this approach might get messy and has computational complexity $O(N^5)$. Instead, there is an alternative approach that is cleaner and keeps the complexity at $O(N^4)$.

Consider again the factorization in (3), but this time rather than using it to produce a single Monte Carlo draw from the joint posterior of π, we use it to build a sequence of Gibbs sampler draws from the conditional posteriors of $\pi^{(j)}$, for $j = 1, ...N$. Specifically, one can sample from the joint distribution $\Pi | y, \Sigma_T$ by cycling through the full conditional distributions

$$\pi^{(j)} \,|\, y, \Sigma_T, \pi^{(-j)} \tag{9}$$

for $j = 1, \ldots, N$, where $\pi^{(j)}$ is the j-th column of the $k \times N$ matrix Π, i.e., the vector of coefficients appearing in equation j, and $\pi^{(-j)} = (\pi^{(1)\prime}, \ldots, \pi^{(j-1)\prime}, \pi^{(j+1)\prime}, \ldots, \pi^{(N)\prime})'$ collects all the coefficients in the remaining equations.

Notice that — by virtue of the triangular factorization — at each iteration j, for updating the value of the sampled $\pi^{(j)}$, we only need to use information associated with equations j and higher. Indeed, the kernel of the p.d.f. of (9) can be built by multiplying the $N - j + 1$ densities appearing in the rows $j, ..., N$ of the factorization in (3):

$$\begin{aligned} p(\pi^{(j)} \,|\, y, \Sigma_T, \pi^{(-j)}) \quad &\propto \quad p(\pi^{(j)} \,|\, \pi^{(1)}, ..., \pi^{(j-1)}, y, \Sigma_T) \\ &\quad\;\; \vdots \\ &\quad\;\; \times p(\pi^{(N)} \,|\, \pi^{(N-1)}, \ldots, \pi^{(1)}, y, \Sigma_T). \end{aligned} \tag{10}$$

Cycling through (10) for $j = 1, ..., N$ will deliver draws from the desired joint distribution.

This new algorithm is based on the same triangularization (3) for which the incorrect one was conceived. But there are some key differences: i) Rather than using a sequence of N equations, it uses a sequence of N systems of $N - j + 1$ equations; and ii) rather than jointly drawing the VAR's coefficients Π in a single Gibbs step, it blocks the VAR's coefficients equation-by-equation, using N Gibbs steps. These features imply that the algorithm makes a few more computations than the incorrect one, and that it produces more correlated draws, which might slow down mixing. However, this new algorithm preserves the main feature that made the original (but incorrect) algorithm so appealing: It has the computational complexity of $O(N^4)$, which allows estimating and forecasting with very large VARs.

3 Practical implementation

To make this approach operational, consider the triangular representation of the system:

$$\tilde{y}_t = Ay_t = A\Pi' x_t + \Lambda_t^{0.5}\epsilon_t = A(x_t'\Pi)' + \Lambda_t^{0.5}\epsilon_t, \tag{11}$$

which can be expressed as the following system of equations:

$$\begin{aligned}
\tilde{y}_{1,t} &= x_t'\pi^{(1)} + \lambda_{1,t}^{0.5}\epsilon_{1,t} \\
\tilde{y}_{2,t} &= a_{2,1}x_t'\pi^{(1)} + x_t'\pi^{(2)} + \lambda_{2,t}^{0.5}\epsilon_{2,t} \\
\tilde{y}_{3,t} &= a_{3,1}x_t'\pi^{(1)} + a_{3,2}x_t'\pi^{(2)} + x_t'\pi^{(3)} + \lambda_{3,t}^{0.5}\epsilon_{3,t} \\
&\ \vdots \\
\tilde{y}_{N,t} &= a_{N,1}x_t'\pi^{(1)} + \cdots + a_{N,N-1}x_t'\pi^{(N-1)} + x_t'\pi^{(N)} + \lambda_{N,t}^{0.5}\epsilon_{N,t},
\end{aligned} \tag{12}$$

where $\tilde{y}_t = Ay_t$ is a vector with generic j-th element $\tilde{y}_{j,t} = y_{j,t} + a_{j,1}y_{1,t} + \cdots + a_{j,j-1}y_{j-1,t}$.

The recursive system (12) is a re-parameterization of the recursive system appearing on page 142 of the paper, and it is observationally equivalent to it. It makes clear the problematic feature that was hidden in the terms $\sigma^*_{j,1,t}\epsilon_{1,t}, \cdots, \sigma^*_{j,j-1,t}\epsilon_{j-1,t}$ appearing in the representation used in the paper: The coefficients $\pi^{(j)}$ of equation j influence not only equation j, but also the following equations $j+1, ..., N$, which is yet another way of seeing that these equations have some extra information about $\pi^{(j)}$ that the old algorithm missed.

Importantly though, it remains true that, when conditioning on Σ_T, equations $1, ..., j-1$ have no information about the coefficients of equation j. Indeed the kernel of the joint posterior of Π is:

$$p(\Pi \,|\, y, x, \Sigma_T) \propto \exp\left(-\frac{1}{2}\sum_{t=1}^{T}\left\{\begin{array}{c} \frac{1}{\lambda_{1,t}}(\tilde{y}_{1,t} - x_t'\pi^{(1)})^2 \\ +\frac{1}{\lambda_{2t}}(\tilde{y}_{2,t} - a_{2,1}x_t'\pi^{(1)} - x_t'\pi^{(2)})^2 \\ \vdots \\ +\frac{1}{\lambda_{N,t}}(\tilde{y}_{N,t} - a_{N,1}x_t'\pi^{(1)} - ... - a_{N,N-1}x_t'\pi^{(N-1)} - x_t'\pi^{(N)})^2 \end{array}\right\}\right) p(\Pi).$$

With coefficient priors $\pi^{(j)} \sim N(\underline{\mu}_{\pi^{(j)}}, \underline{\Omega}_{\pi^{(j)}})$, $j = 1, ...N$ that are independent across equations,[2] the first $j-1$ elements in the quadratic term above do not contain $\pi^{(j)}$. It follows that

[2] The assumption of independence across equations covers most of the practical implementations of VARs.

the conditional posterior density $p(\pi^{(j)} \,|\, y, \Sigma_T, \pi^{(-j)})$ can be obtained as in (10), using the subsystem composed of the last $N-j+1$ equations of (12). This density has a multivariate normal distribution

$$(\pi^{(j)} \,|\, y, \Sigma_T, \pi^{(-j)}) \sim \mathcal{N}(\overline{\mu}_{\pi^{(j)}}, \overline{\Omega}_{\pi^{(j)}}), \tag{13}$$

with

$$\overline{\Omega}^{-1}_{\pi^{(j)}} = \underline{\Omega}^{-1}_{\pi^{(j)}} + \textstyle\sum_{i=j}^{N} a_{i,j}^2 \sum_{t=1}^{T} \displaystyle\frac{1}{\lambda_{i,t}} x_t x_t' \tag{14}$$

$$\overline{\mu}_{\pi^{(j)}} = \overline{\Omega}_{\pi^{(j)}} \left(\underline{\Omega}^{-1}_{\pi^{(j)}} \underline{\mu}_{\pi^{(j)}} + \textstyle\sum_{i=j}^{N} a_{i,j} \sum_{t=1}^{T} \displaystyle\frac{1}{\lambda_{i,t}} x_t z_{i,t} \right), \tag{15}$$

where $z_{j+l,t} = \tilde{y}_{j+l,t} - \sum_{i \neq j, i=1}^{j+l} a_{j+l,i} x_t' \pi^{(i)}$, for $l = 0, ..., N-j$, and $a_{i,i} = 1$. The posterior moments as expressed in equations (14) and (15) make clear that, in estimating equation j, information is used from not only that equation but also equations $j+1$ through N. In some software packages, computations may be fastest using sums of moments as in equations (14) and (15).

In what follows we provide an alternative expression for the moments (14) and (15) that works more efficiently in the commonly used Matlab software package. Define

$$\Pi^{[j=0]} = \left[\begin{array}{ccccccc} \pi^{(1)} & \vdots & \pi^{(j-1)} & 0_{k \times 1} & \pi^{(j+1)} & \vdots & \pi^{(N)} \end{array} \right]$$

as the $k \times N$ matrix of coefficients Π , where in column j the coefficient vector $\pi^{(j)}$ has been replaced by a $0_{k \times 1}$ vector. The equations of (12) involving $\pi^{(j)}$ are:

$$\underset{(N-j+1)\times N}{A^{(j:N,1:N)}} \left(\underset{N \times 1}{y_t} - \underset{N \times 1}{\Pi^{[j=0]\prime} x_t} \right) = \underset{(N-j+1)\times 1}{A^{(j:N,j)}} \; \underset{1 \times k}{\pi^{(j)\prime}} \; \underset{k \times 1}{x_t} + \underset{(N-j+1)\times 1}{\varepsilon_t^{(j:N)}},$$

where $\varepsilon_t^{(j:N)}$ denotes the sub-vector of $\Lambda_t^{0.5} \varepsilon_t$ corresponding to variables j through N (here, for simplicity, we subsume the volatilities into the innovation vector). Transposing and stacking all of the equations for $t = 1, ..., T$ yields:

$$(y - X\Pi^{[j=0]}) A^{j:N,1:N)\prime} = X \pi^{(j)} A^{(j:N,j)\prime} + \varepsilon^{(j:N)},$$

where the matrices y, X, and $\varepsilon^{(j:N)}$ have dimensions $T \times N$, $T \times k$, and $T \times (N-j+1)$, respectively. Vectorizing the system yields:

$$vec((y - X\Pi^{[j=0]}) A^{(j:N,1:N)\prime}) = (A^{(j:N,j)} \otimes X) \pi^{(j)} + vec\left(\varepsilon^{(j:N)}\right).$$

If necessary, it could be relaxed while maintaining the same triangular structure. This would entail using the properties of the multivariate normal to derive a triangular factorization for the prior in which $\pi^{(j)}$ appears only in equation $j, ..., N$. While this requires the inversion of large matrices, such inversions need to be performed only once, outside the main MCMC algorithm, and therefore with not much computational cost.

Finally, we need to divide by the standard deviation of the errors. Since $vec\left(\varepsilon^{(j:N)}\right) \sim N(0, I_{N-j+1} \otimes \Lambda_{1:T,j:N})$, we obtain:

$$Y^{(j)} = X^{(j)}\pi^{(j)} + u^{(j)}, \tag{16}$$

with

$$\begin{aligned}
\underset{T(N-j+1)\times 1}{Y^{(j)}} &= vec((y - X\Pi^{[j=0]})A^{(j:N,1:N)\prime})./vec(\Lambda^{0.5}_{(1:T,j:N)}) && (17)\\
\underset{T(N-j+1)\times k}{X^{(j)}} &= (A^{(j:N,j)} \otimes X)./vec(\Lambda^{0.5}_{(1:T,j:N)}), && (18)
\end{aligned}$$

where $u^{(j)} \sim N(0, I_{T(N-j+1)})$ and ./ is the Matlab element-by-element division operator. The model in (16) is a Gaussian linear regression model, and the likelihood moments $X^{(j)\prime}X^{(j)}$ and $X^{(j)\prime}Y^{(j)}$ can be combined with the prior to obtain an equivalent expression for the posterior moments appearing in (14) and (15):

$$\begin{aligned}
\overline{\Omega}^{-1}_{\pi^{(j)}} &= (\underline{\Omega}^{-1}_{\pi^{(j)}} + X^{(j)\prime}X^{(j)}) && (19)\\
\overline{\mu}_{\pi^{(j)}} &= \overline{\Omega}_{\pi^{(j)}}(\underline{\Omega}^{-1}_{\pi^{(j)}}\underline{\mu}_{\pi^{(j)}} + X^{(j)\prime}Y^{(j)}). && (20)
\end{aligned}$$

4 Application results

In the interest of brevity, in this section we limit the presentation of results to those on computational gains and mixing, computed using the monthly macroeconomic data set used in Carriero, et al. (2019). (The data set consists of time series from the FRED-MD data set, for the period January 1960 to December 2014, with 13 lags in each VAR.) A supplemental online appendix provides a full set of updates of the results of the paper based on the correct triangular algorithm, along with some additional results. These updated results are not much different with respect to the ones presented in the paper.

Figure 1 illustrates the computational gains arising from the use of the triangular algorithm. The top panel shows the computational time (on a 3.5 GHz Intel Core i7) needed to perform 10 draws as a function of the size of the cross-section using the (correct) triangular algorithm and the system-wide algorithm. The bottom panel compares the gain in theoretical computational complexity (dashed line — which is equal to N^2) with the actual gain in computational time. Since the computational gains become so large that they create scaling problems, results in this figure are displayed using a logarithmic vertical axis.

Compared to the same Figure 1 of the paper, two things have changed. First, the performance of the system-wide algorithm (SWA) is much improved: It now takes 33 seconds to make 10 draws from a model with $N = 20$, while before it took 255 seconds. This improvement is due to our use of the Matlab function ./ (element-by-element division of a matrix object) in computing the analogues of (17) and (18) used for the full system of equations of the SWA, while before we were using a Kronecker product to perform the same operation.[3] Second, the correct triangular algorithm (CTA) is slower than the earlier incorrect triangular algorithm (TA). For example, for a system of $N = 20$ variables, the correct algorithm takes about 3.25 times as long as the original triangular algorithm. This increased computational cost to the CTA over the original TA comes from the fact that, at each iteration j of the loop across equations, the CTA uses $T \times (N - j + 1)$ observations, while the TA only used T.

However, the main pattern emerging from Figure 1 is still the same as in the paper: The CTA is significantly faster than the SWA, with computational gains growing quadratically, which of course reflects the fact that the SWA and CTA have computational complexity $O(N^6)$ and $O(N^4)$, respectively. For the system of $N = 20$ variables used in the empirical application of the paper (updated in the online appendix to this paper), the CTA is about 13 times faster than the SWA. For $N = 40$, it is about 43 times faster. Finally, note that — in practice, although not documented in Figure 1 — as systems get larger, the CTA becomes the only available option, because the SWA requires matrices of such a size that the storage memory requirements quickly exceed the RAM of the typical desktop computer.

As we mentioned above, because the CTA uses conditional posteriors that block the VAR's coefficients equation-by-equation, whereas the TA did not use such blocking and instead treated the full set of coefficients, the draws generated by the new algorithm are more autocorrelated, which might slow down convergence and reduce mixing. Accordingly, we have run some checks of convergence and mixing. Figure 2 compares the mixing of the CTA algorithm with that of the SWA. The results in these figures are obtained by (1) running the SWA for a total of 22,000 draws, discarding the first 2000 and retaining the remaining 20,000 draws; and (2) running the CTA for the same amount of clock time as the SWA, discarding the first 2000 draws, and then adjusting the skip-sampling of the CTA to reduce the sample to 20,000 retained draws. Note that the scales on the horizontal axes of the SWA (left) and CTA (right) columns differ substantially. These results show that the inefficiency factors obtained by running the two alternative algorithms for the same amount of time are much lower for draws produced by the CTA than those produced by the SWA.

[3] To the best of our knowledge at the time, the Matlab function ./ required the combined use of the function "repmat" for computing the element-by-element ratios in (17) and (18), which ended up being slower than using a Kronecker division. Instead, in the current version of Matlab, the function ./ will fill out columns of a matrix with the single element of a vector, allowing the element-by-element division of every row of a matrix by a vector.

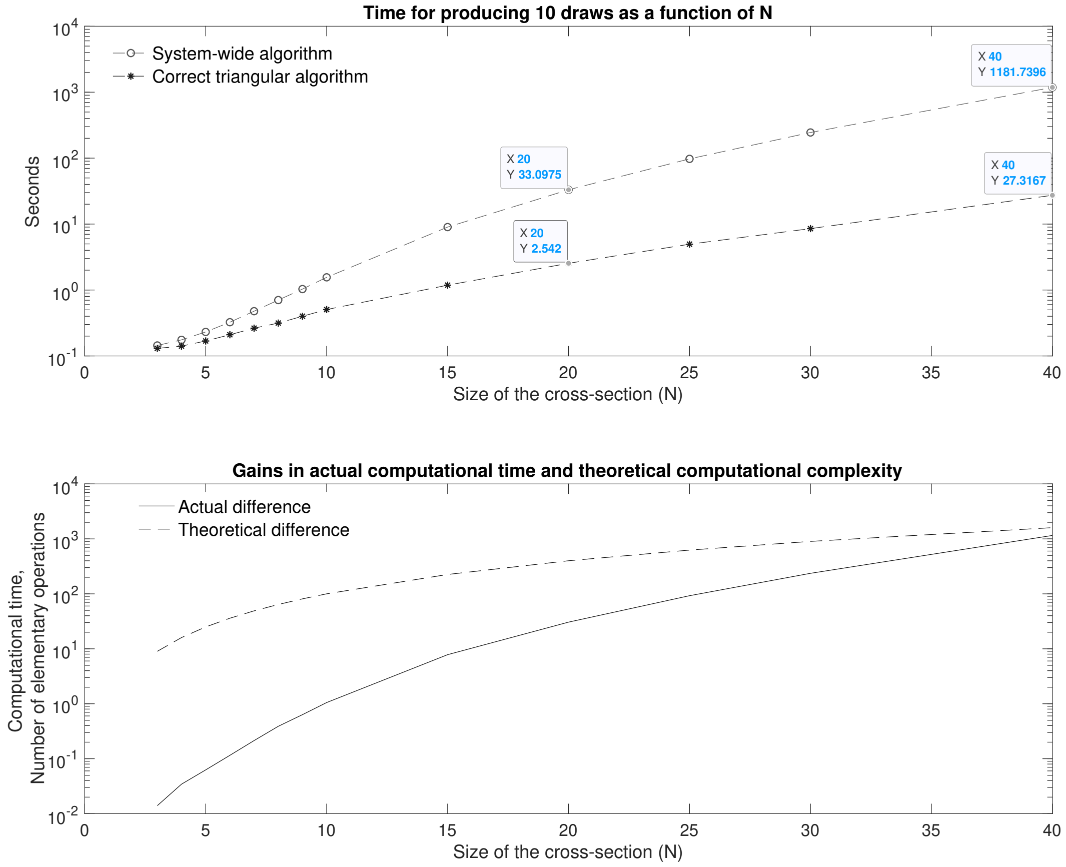


Figure 1: Actual computational time and theoretical computational complexity of the system-wide and correct triangular algorithms. Note that due to the exponential nature of the gains the y-axes are in logarithmic scale. Computational times are computed as the average time (over 10 independent chains) required to make 10 draws on a 3.5 GHz Intel Core i7.

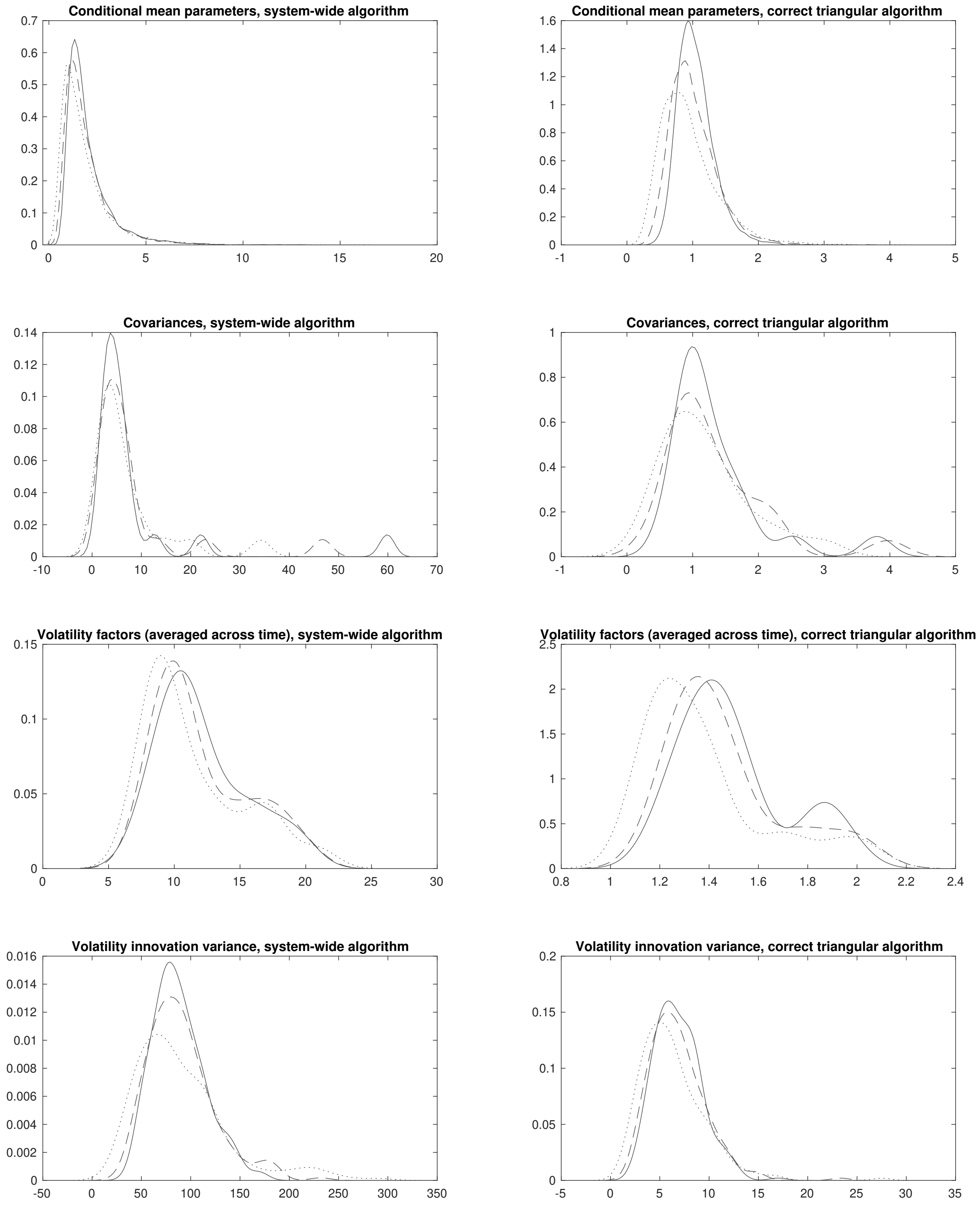


Figure 2: Comparison of inefficiency factors (IF) between the system-wide and correct triangular algorithm. Kernel estimates. Solid, dashed, and dotted lines refer to 4, 8, and 15 percent tapering, respectively. The densities in each sub-plot are computed across the parameters within a given set (from top to bottom: conditional mean coefficients, covariances, states, and covariances of the states). The graphs on the left refer to the system-wide algorithm, while the graphs on the right refer to the correct triangular algorithm.

The CTA with appropriate skip-sampling can produce in the same amount of time draws many times closer to i.i.d. sampling. Instead, the SWA is slower to mix (in a unit of time).

5 Conclusions

We presented a new, correct algorithm that actually draws from the joint posterior distribution that was intended. The new algorithm is based on the same triangularization (3) for which the incorrect one was conceived, but rather than using a sequence of N equations to obtain a joint draw of the VAR's coefficients in a single Gibbs step, it uses a sequence of N Gibbs steps, each based on sub-systems of $N - j + 1$ equations. While in principle these differences can slow down the speed and reduce the mixing, the new algorithm is still faster and mixes better (for a given amount of computation time) than the system-wide alternative. More importantly, the new algorithm preserves the main feature that made the original (but incorrect) algorithm so appealing: It has the computational complexity of $O(N^4)$, which allows handling very large VARs. The empirical results presented in the paper did not change appreciably after re-estimating the model using the correct algorithm. That being said, in other applications, the algorithm correction could yield more of a difference in estimates.

Acknowledgments

We would like to thank editors Serena Ng and Elie Tamer for their guidance, Mark Bognanni for identifying and bringing to our attention the problem with the original algorithm, Elmar Mertens and Tommaso Tornese for many helpful discussions, and Marta Banbura, a referee, and the associate editor for helpful suggestions. Matlab code for the correct triangular algorithm (including data for an application) is available at https://didattica.unibocconi.it/mypage/dwload.php?nomefile=Triangular_Example_new20210715105214.zip.

References

[1] Bognanni, M. 2021. Comment on “Large Bayesian Vector Autoregressions with Stochastic Volatility and Non-Conjugate Priors.” Journal of Econometrics, forthcoming. https://doi.org/10.1016/j.jeconom.2021.10.008

[2] Carriero A., Clark, T. and Marcellino, M. 2019. Large Bayesian Vector Autoregressions with Stochastic Volatility and Non-Conjugate Priors. Journal of Econometrics 212(1), 137-154. https://doi.org/10.1016/j.jeconom.2019.04.024

Appendix to "Corrigendum to: Large Bayesian Vector Autoregressions with Stochastic Volatility and Non-Conjugate Priors"

Andrea Carriero[*] Joshua Chan[†] Todd E. Clark[‡] Massimiliano Marcellino[§]

November 2021

Abstract

This appendix provides additional application results, including figures from the published paper updated to use the correct triangular algorithm and other results mentioned in the corrigendum. It also provides a correct algorithm that can be used for triangular estimation of homoskedastic VARs.

1 Updates of results in the paper

The application results provided in this appendix are the same as in the original paper. They are based on monthly data taken from the FRED-MD data set, for the period January 1960 to December 2014. The model includes the 20 variables indicated in Table 1 and 13 lags.

Figures 1 through 5 provide updates of the results in the paper, using the correct algorithm. As these figures indicate, correcting the algorithm for estimating the VAR's coefficients does not change the conclusions indicated in the paper, also provided below. The results in Figure 1 and Figure 2 are obtained by (1) running the SWA for a total of 22,000 draws, discarding the first 2000 and retaining the remaining 20,000 draws; and (2) running the CTA for the same amount of clock time as the SWA, discarding the first 2000 draws, and then adjusting the skip-sampling of the CTA to reduce the sample to 20,000 retained draws. The results in Figure 3, Figure 4, and Figure 5 are based on a recursive pseudo-out-of-sample forecasting exercise. In each estimation sample of the exercise, the results

[*] Queen Mary University of London, a.carriero@qmul.ac.uk

[†] Purdue University, joshuacc.chan@gmail.com

[‡] Federal Reserve Bank of Cleveland, todd.clark@clev.frb.org

[§] Bocconi University, IGIER and CEPR, massimiliano.marcellino@unibocconi.it

are obtained by running the CTA for a total of 5500 draws, discarding the first 500 and retaining 1 in 5 of the remaining 5000 draws, providing a final sample of 1000 draws.

Updating the paper's Figure 2, Figure 1 illustrates the recursive means for some selected coefficients and shows that the correct triangular algorithm with split-sampling reaches convergence much faster than the system-wide algorithm. This pattern is particularly marked for the volatility component of the model. Figure 2 illustrates the entire posterior distribution of some coefficients, specifically those on the main diagonal of the matrix Π_1. As these estimates indicate, the correct triangular algorithm yields posterior densities that match those obtained with the system-wide algorithm.

Table 1: Variables in the 20-variable forecasting models

Variable	**Mnemonic**
Real personal income	RPI ($\Delta \ln$)
Real PCE	DPCERA3M086SBEA ($\Delta \ln$)
Real manufacturing and trade sales	CMRMTSPLx ($\Delta \ln$)
Industrial production	INDPRO ($\Delta \ln$)
Capacity utilization in manufacturing	CUMFNS
Civilian unemployment rate	UNRATE
Total nonfarm employment	PAYEMS ($\Delta \ln$)
Hours worked: goods-producing	CES0600000007 (ln)
Average hourly earnings: goods-producing	CES0600000008 ($\Delta \ln$)
PPI for finished goods	PPIFGS ($\Delta \ln$)
PPI for commodities	PPICMM ($\Delta \ln$)
PCE price index	PCEPI ($\Delta \ln$)
Federal funds rate	FEDFUNDS
Total housing starts	HOUST (ln)
S&P 500 price index	S&P 500 ($\Delta \ln$)
U.S.-U.K. exchange rate	EXUSUKx ($\Delta \ln$)
1 yr. Treasury - FEDFUNDS spread	T1YFFM
10 yr. Treasury - FEDFUNDS spread	T10YFFM
BAA - FEDFUNDS spread	BAAFFM
ISM: new orders index	NAPMNOI

Updating the paper's Figure 3, the left panels of Figure 3 display the root mean squared forecast error (RMSFE) relative (ratio) to the benchmark (the 20-variable homoskedastic VAR), so that a value below 1 denotes a model outperforming the benchmark. The large homoskedastic model outperforms the small homoskedastic model for all variables at all horizons, suggesting that the inclusion of more data does improve the specification of the conditional means and therefore the point forecasts. The inclusion of time variation in volatilities consistently improves the performance of the small model, and for FEDFUNDS it also outperforms the benchmark at long horizons. However, the small heteroskedastic model is still largely dominated by the benchmark at short forecast horizons. The model with both time-varying volatilities and a large cross-section instead provides systematically better point forecasts than the benchmark (and than the other models).

The right-hand panels of Figure 3 present results for density forecasts, based on the average log scores. The figure displays the average log scores relative (difference) to the benchmark (the 20-variable homoskedastic VAR), so that a value above 0 denotes a model outperforming the benchmark. Both homoskedastic specifications perform quite poorly in density forecasting, while the heteroskedastic ones can achieve very high gains. Moreover, the large heteroskedastic system consistently outperforms the small heteroskedastic system. In combination with the findings for point forecasts, this result suggests that while both heteroskedastic models provide a better assessment of the overall uncertainty around the forecasts, the model based on the large cross-section centers such uncertainty around a more reliable mean, thereby obtaining further gains in predictive accuracy.

Updating the paper's Figures 4 and 5, Figures 4 and 5 compare forecasts for all the variables included in the cross-section of the models with 20 variables, with point forecasts in Figure 4 and density forecasts in Figure 5. In all of the subplots in Figure 4, the x axes measure the RMSFE obtained by the large VAR when we allow for stochastic volatility, while the y axes measure the same loss function (RMSFE) obtained by the homoskedastic specification. Each point corresponds to a different forecast horizon, and when a point is *above* the 45-degree line, this shows that the RMSFE of the heteroskedastic specification is smaller, indicating that the inclusion of variation in the volatility improved point forecasting performance. As is clear in the graph, in several instances the models produce point forecasts of similar accuracy. However, as the forecast horizon increases (which can be indirectly inferred from the graph as in general higher RMSFEs correspond to longer forecast horizons), the specification with variation in the volatilities tends to outperform the homoskedastic version of the model. The mechanism at play is as follows: The heteroskedastic model provides more efficient estimates and therefore a better characterization of the predictive densities, while the homoskedastic model is misspecified and therefore provides an inferior characterization of the predictive densities. At short forecast horizons, this does not have much effect on point forecasts, but as the forecast horizon increases, the predictive densities cumulate nonlinearly and therefore the misspecification of the homoskedastic model increasingly reduces the relative accuracy.

In Figure 5, the x axes measure the (log) density score obtained by the large VAR when we allow for stochastic volatility, while the y axes measure the same gain function (score) obtained by the homoskedastic specification. Each point corresponds to a different forecast horizon, and when a point is *below* the 45-degree line, this shows that the score of the heteroskedastic specification is larger, indicating that the inclusion of variation in the volatility improved density forecasting performance. The improvement coming from the introduction of time variation in the volatilities is striking, and it is common to nearly all variables. Clearly, stochastic volatility improves the overall assessment of uncertainty with respect to the homoskedastic model, and it does so both directly, by simply using a better variance around the point estimates, and indirectly, by centering the densities toward improved point forecasts (as documented in Figure 4).

2 Additional empirical results: Algorithm comparison

Finally, to further compare results across algorithms, Figure 6 shows the impulse response functions for a federal funds rate shock (recursively identified) delivered by the various algorithms. This figure extends and updates results in the online appendix to the original paper. In these new results, response estimates from the system-wide algorithm (SWA) and correct triangular algorithm (CTA) coincide perfectly, while the original triangular algorithm (TA) does show some occasional deviations.

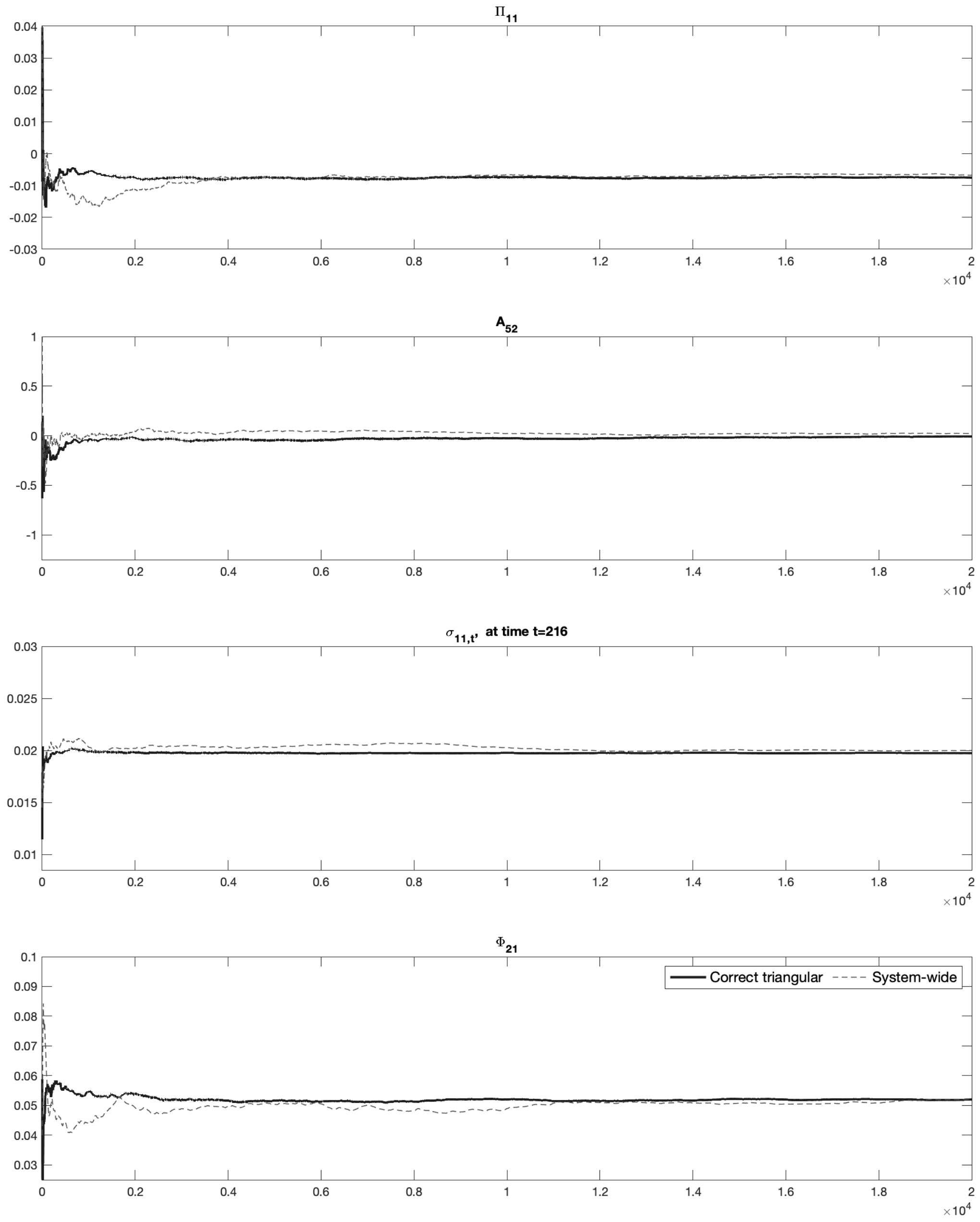


Figure 1: Update of Figure 2 of the paper. Recursive means of selected coefficients. Comparison between the system-wide and correct triangular algorithm. Chains are initialized at the same value (set equal to the priors).

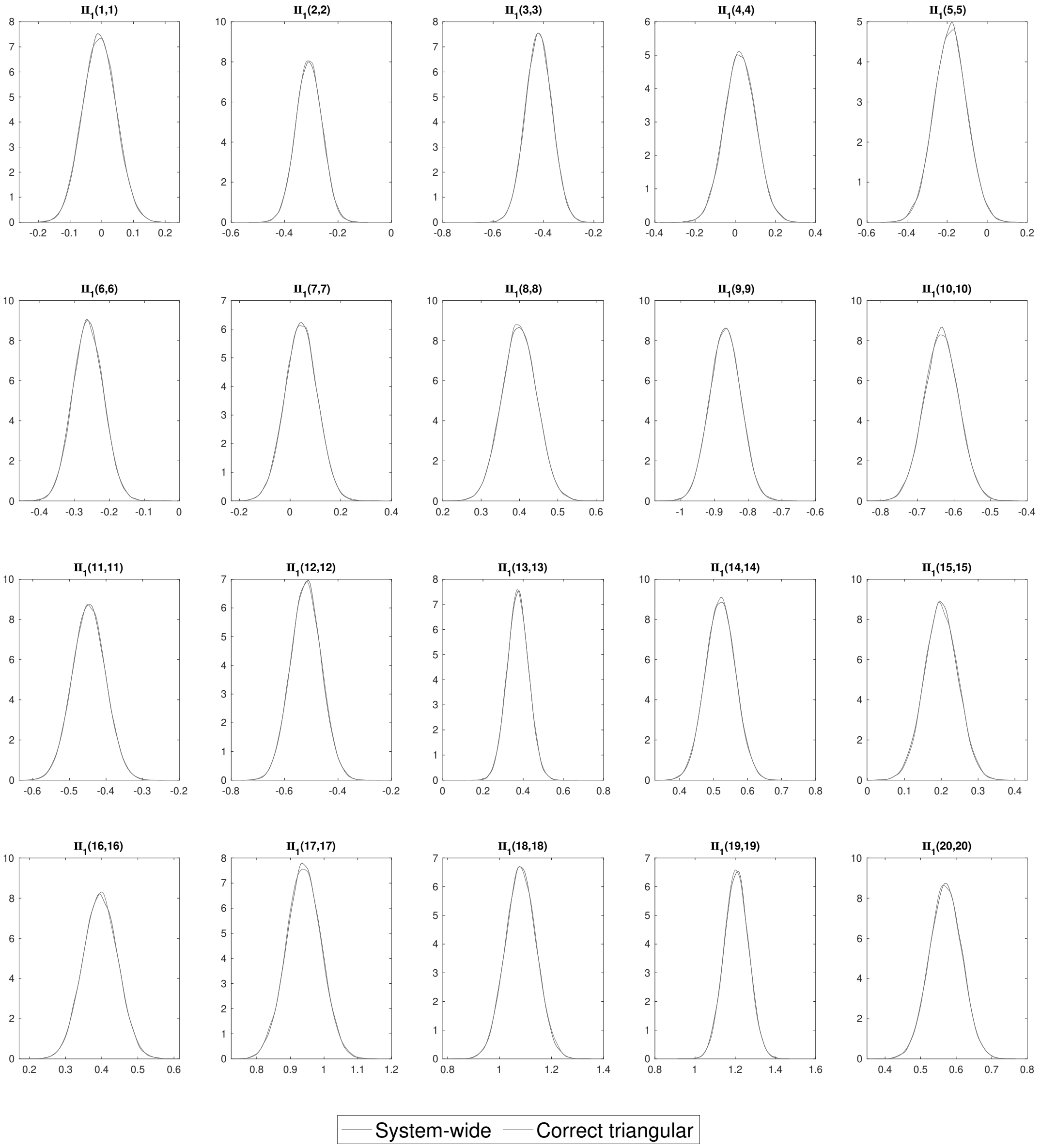
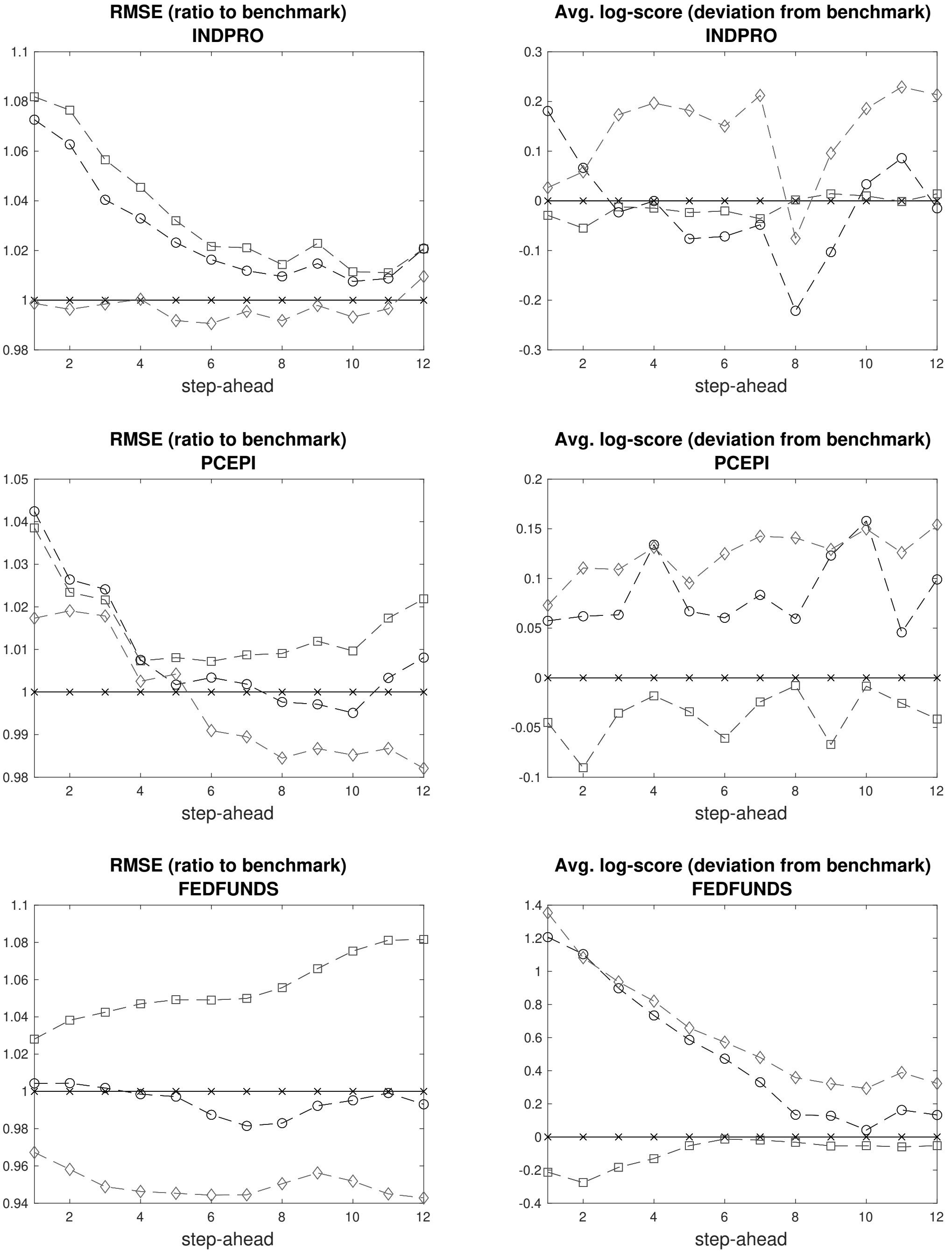


Figure 2: Posterior distribution of the coefficients on the main diagonal of the matrix Π_1, under the SWA and CTA.



RMSE (ratio to benchmark) INDPRO

Avg. log-score (deviation from benchmark) INDPRO

RMSE (ratio to benchmark) PCEPI

Avg. log-score (deviation from benchmark) PCEPI

RMSE (ratio to benchmark) FEDFUNDS

Avg. log-score (deviation from benchmark) FEDFUNDS

Figure 3: Update of Figure 3 of the paper. Panels on the left-hand side contain results for the point forecasts (relative RMSE of different models vs benchmark). The panels on the right-hand side contain results for the density forecasts (log-score gains of different models vs benchmark). In all the panels, crosses represent the homoskedastic VAR with 20 variables (the benchmark model), the squares represent the homoskedastic VAR with 3 variables, the circles represent the heteroskedastic VAR with 3 variables, and the diamonds represent the heteroskedastic VAR with 20 variables.

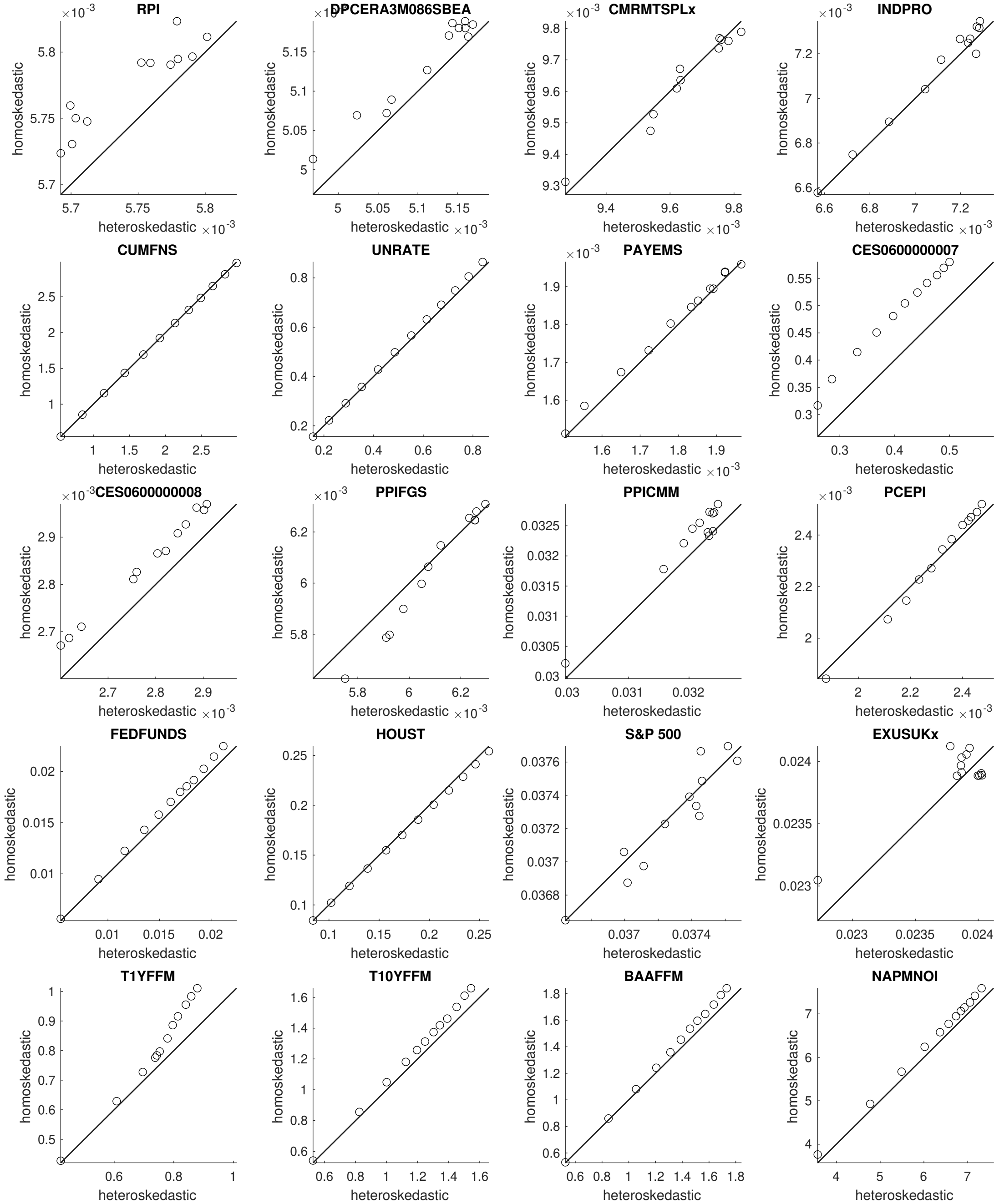


Figure 4: Update of Figure 4 of the paper. Comparison of point forecast accuracy. Each panel describes a different variable. The x axis reports the RMSFE obtained using the BVAR with stochastic volatility (heteroskedastic); the y axis reports the RMSFE obtained using the homoskedastic BVAR. Each point corresponds to a different forecast horizon from 1 to 12 steps ahead (in most cases, a higher RMSFE corresponds to a longer forecast horizon).

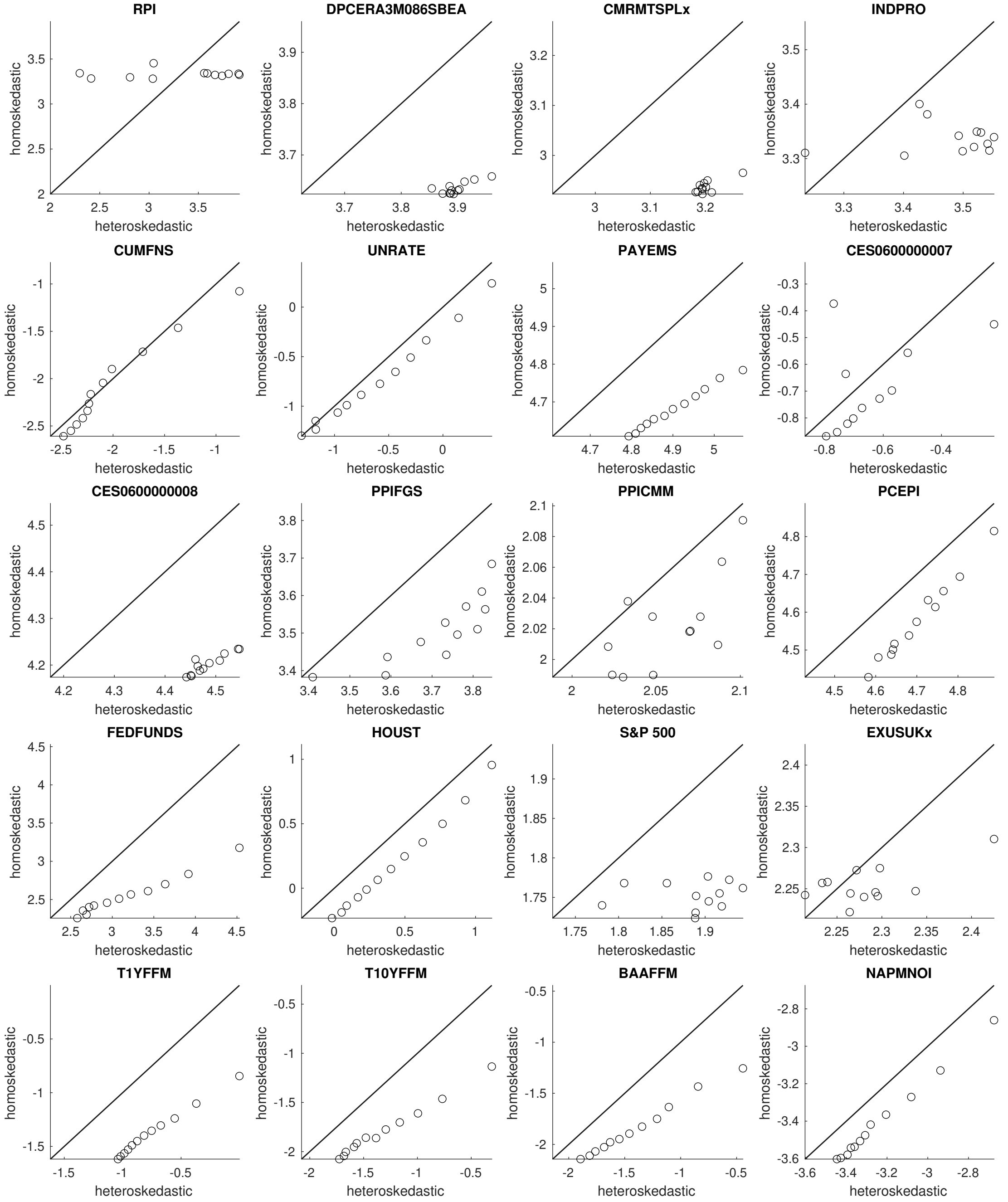


Figure 5: Update of Figure 5 of the paper. Comparison of density forecast accuracy. Each panel describes a different variable. The x axis reports the (log) density score obtained using the BVAR with stochastic volatility (heteroskedastic); the y axis reports the (log) density score obtained using the homoskedastic BVAR. Each point corresponds to a different forecast horizon from 1 to 12 steps ahead.

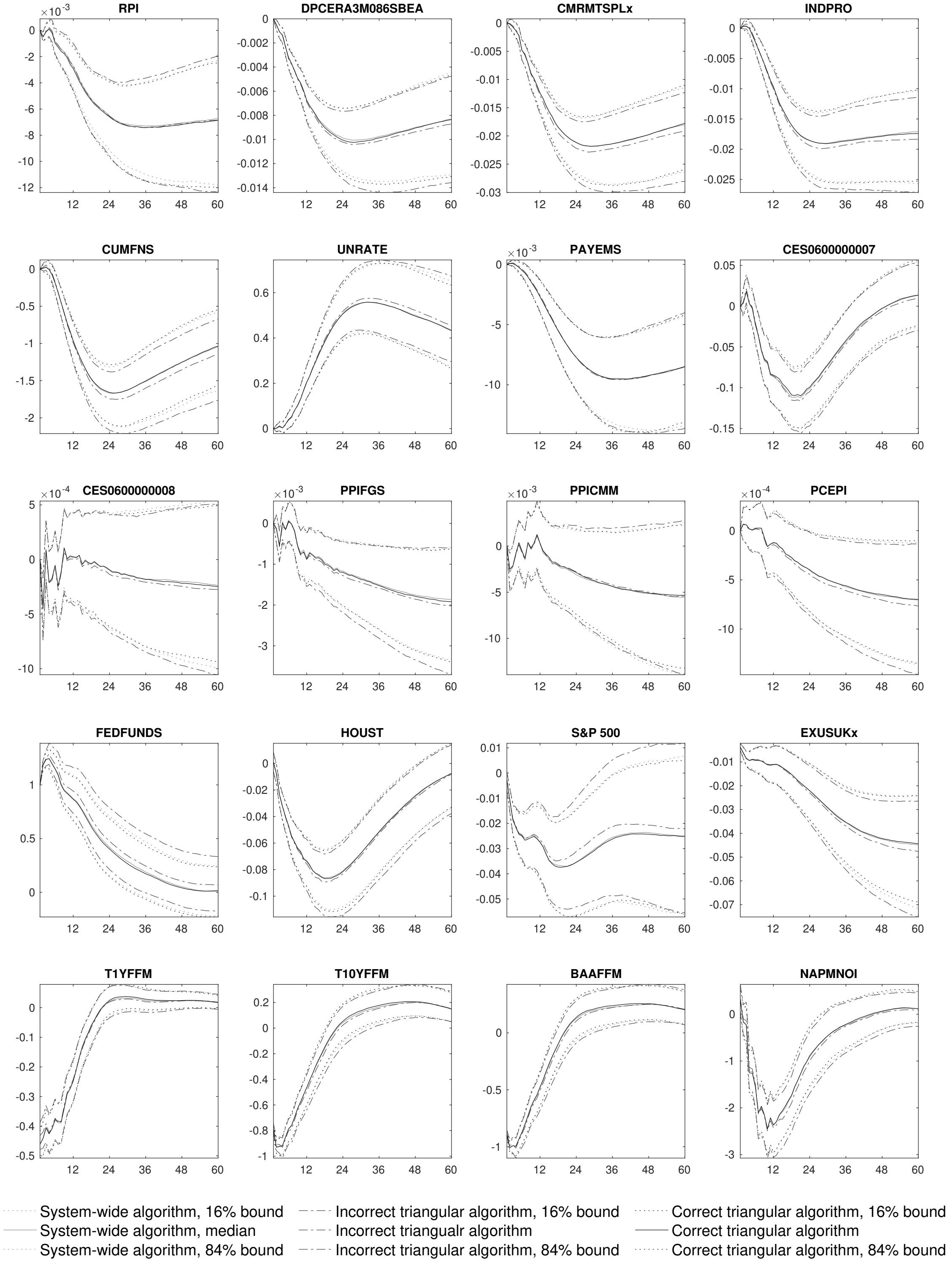


Figure 6: Comparison of impulse responses computed using the alternative algorithms. Model with monthly data, N=20 and p=13.

3 The correct triangular algorithm for homoskedastic VARs

This section outlines the implementation of the correct triangular algorithm to the following homoskedastic VAR:

$$y_t = \Pi' x_t + \epsilon_t,$$

where $\epsilon_t \sim iid\, N(0, \Sigma), t = 1, \ldots, T$. The coefficient priors are Gaussian $\pi^{(j)} \sim N(\underline{\mu}_{\pi^{(j)}}, \underline{\Omega}_{\pi^{(j)}})$, $j = 1, \ldots, N$, and they are independent across equations. We first use an LDL decomposition (sometimes referred to as a square-root-free Cholesky decomposition) to factor the reduced-form error covariance matrix as $\Sigma = A^{-1} \Lambda A^{-1'}$, where A^{-1} is a lower triangular matrix with ones on its main diagonal and Λ is a diagonal matrix with generic j-th element λ_j.[1] Then, the full posterior density of $\pi^{(j)}$ has a multivariate normal distribution

$$(\pi^{(j)} \,|\, y, \Sigma, \pi^{(-j)}) \sim \mathcal{N}(\overline{\mu}_{\pi^{(j)}}, \overline{\Omega}_{\pi^{(j)}}),$$

with

$$\overline{\Omega}^{-1}_{\pi^{(j)}} = \underline{\Omega}^{-1}_{\pi^{(j)}} + \textstyle\sum_{i=j}^{N} \dfrac{a^2_{i,j}}{\lambda_i} \textstyle\sum_{t=1}^{T} x_t x_t' \tag{1}$$

$$\overline{\mu}_{\pi^{(j)}} = \overline{\Omega}_{\pi^{(j)}} \left(\underline{\Omega}^{-1}_{\pi^{(j)}} \underline{\mu}_{\pi^{(j)}} + \textstyle\sum_{i=j}^{N} \dfrac{a_{i,j}}{\lambda_i} \textstyle\sum_{t=1}^{T} x_t z_{i,t} \right), \tag{2}$$

where $z_{j+l,t} = \tilde{y}_{j+l,t} - \sum_{i \neq j, i=1}^{j+l} a_{j+l,i} x_t' \pi^{(i)}$, for $l = 0, \ldots, N - j$, and $a_{i,i} = 1$.

Next, we provide an alternative expression for the moments (1) and (2) that works more efficiently in the commonly used Matlab software package. Let $L^{0.5} = 1_T \otimes (\lambda_j^{0.5}, \ldots, \lambda_N^{0.5})$, where 1_T is a $T \times 1$ column of ones. Moreover, define

$$\begin{array}{rcl} \underset{T(N-j+1)\times 1}{Y^{(j)}} & = & vec((y - X\Pi^{[j=0]}) A^{(j:N,1:N)'}) ./ vec(L^{0.5}) \\ \underset{T(N-j+1)\times k}{X^{(j)}} & = & (A^{(j:N,j)} \otimes X) ./ vec(L^{0.5}), \end{array}$$

where ./ is the Matlab element-by-element division operator. Then, we obtain an equivalent expression for the posterior moments appearing in (1) and (2):

$$\begin{array}{rcl} \overline{\Omega}^{-1}_{\pi^{(j)}} & = & (\underline{\Omega}^{-1}_{\pi^{(j)}} + X^{(j)\prime} X^{(j)}) \\ \overline{\mu}_{\pi^{(j)}} & = & \overline{\Omega}_{\pi^{(j)}} (\underline{\Omega}^{-1}_{\pi^{(j)}} \underline{\mu}_{\pi^{(j)}} + X^{(j)\prime} Y^{(j)}). \end{array}$$

It is also possible to draw from the multivariate normal distribution posterior density of $\pi^{(j)}$ given by $(\pi^{(j)} \,|\, y, \Sigma, \pi^{(-j)}) \sim \mathcal{N}(\overline{\mu}_{\pi^{(j)}}, \overline{\Omega}_{\pi^{(j)}})$ with a Cholesky decomposition of Σ. In

[1] This decomposition can be computed either with a function such as the LDL function in Matlab or from a normalization of the Cholesky decomposition.

this case, let $A^{\dagger}$ denote the Cholesky decomposition of Σ, so that $\Sigma = A^{\dagger}A^{\dagger\prime}$, and define $\tilde{A} = A^{\dagger-1}$, with elements $\tilde{a}_{i,j}$ in row i, column j. In this case, $\tilde{y}_t$ is defined as $\tilde{y}_t = \tilde{A}y_t$, and $z_{j+l,t} = \tilde{y}_{j+l,t} - \sum_{i\neq j,i=1}^{j+l} \tilde{a}_{j+l,i} x_t' \pi^{(i)}$, for $l = 0, \ldots, N-j$. Then the posterior moments of the conditional normal distribution take the form:

$$\overline{\Omega}_{\pi^{(j)}}^{-1} = \underline{\Omega}_{\pi^{(j)}}^{-1} + \sum_{i=j}^{N} \tilde{a}_{i,j}^2 \sum_{t=1}^{T} x_t x_t' \tag{3}$$

$$\overline{\mu}_{\pi^{(j)}} = \overline{\Omega}_{\pi^{(j)}} \left(\underline{\Omega}_{\pi^{(j)}}^{-1} \underline{\mu}_{\pi^{(j)}} + \sum_{i=j}^{N} \tilde{a}_{i,j} \sum_{t=1}^{T} x_t z_{i,t} \right). \tag{4}$$