sstvars: Structural Smooth Transition Vector Autoregressive Models in R

Savi Virolainen
University of Helsinki

Abstract

We describe the R package **sstvars**, which provides tools for estimating and analyzing the reduced form and structural smooth transition vector autoregressive (STVAR) models. The package implements various transition weight functions, conditional distributions, identification methods, and parameter restrictions. The model parameters are estimated with the method of maximum likelihood by running multiple rounds of a two-phase estimation procedure in which a genetic algorithm is used to find starting values for a gradient based method. For evaluating the adequacy of the estimated models, **sstvars** utilizes residuals based diagnostics and provides functions for graphical diagnostics and for calculating formal diagnostic tests. **sstvars** also accommodates the estimation of linear impulse response functions, nonlinear generalized impulse response functions, and generalized forecast error variance decompositions. Further functionality includes hypothesis testing, plotting the profile log-likelihood functions about the estimate, simulation from STVAR processes, and forecasting, for example. We illustrate the use of **sstvars** with a quarterly series consisting of two U.S. variables: the percentage change of real GDP and the percentage change of GDP implicit price deflator.

Keywords: smooth transition vector autoregressive model, structural smooth transition vector autoregressive model, regime-switching, SVAR, STVAR, maximum likelihood estimation.

1. Introduction

Linear vector autoregressive (VAR) models are a standard tools in time series econometrics. They can be employed to answer questions about the statistical relationships of different variables or to forecast future values of the process, for example. Structural VAR model allows to trace out the effects of economic shocks that have been identified by the researcher. With an appropriate choice of the autoregressive order p, a linear VAR is often able to filter out autocorrelation from the series very well. If the errors are assumed to follow an autoregressive conditional heteroskedasticity (ARCH) process, the model is also often able to adequately filter out conditional heteroskedasticity from the series.

In some cases, linear VAR models are not, however, capable to capture all the relevant characteristics of the series. This includes shifts in the mean or volatility, and changes in the autoregressive dynamics of the process. Such nonlinear features frequently occur in economic time series when the underlying data generating dynamics vary in time, for example, depending the specific state of the economy.

Various types of time series models capable of capturing this kind of regime-switching behav-

ior have been proposed, one of them is the smooth transition vector autoregressive (STVAR) models that allow to capture gradual shifts in the dynamics of the data. They consist of a finite number of regimes, each of which are linear vector autoregressions that are characterized by different autoregressive coefficients or error term covariance matrices. The package **sstvars** considers STVAR models in which, at each point of time, the observation is a weighted average of the conditional means of the regimes plus a random error whose covariance matrix is a weighted average of the covariance matrices of the regimes. The weights, in turn, are expressed in terms of time-varying transition weights that depend on the preceding observations. Different STVAR models can be created by specifying the transition weights or the error distribution in various ways. See Hubrich and Teräsvirta (2013) for a survey on STVAR literature, including formulations more general than our framework.

This manuscript describes the R package **sstvars** providing a set of easy-to-use tools for STVAR modeling, including unconstrained and constrained maximum likelihood (ML) estimation of the model parameters, residual based model diagnostics, estimation of linear impulse response functions, nonlinear generalized impulse response functions, and generalized forecast error variance decompositions. Further functionality includes hypothesis testing, plotting the profile log-likelihood functions about the estimate, simulation from STVAR processes, and forecasting, for example. Various transition weight functions are accommodated, including logistic weights Anderson and Vahid (1998), multinomial logit weights, exponential weights (e.g., Hubrich and Teräsvirta 2013), threshold weights (Tsay 1998), and transition weights that defined as weighted relative likelihoods of the regimes corresponding to the preceding p observations (FILL IN REFERENCE). Currently, the accommodated conditional distributions include Student's t distribution and Gaussian distribution, whereas the accommodated identification methods include recursive identification and identification by heteroskedasticity. More conditional distributions and identification methods are probably added in the future, however.

The estimation of the model parameters can, in some cases, be rather tricky. This is because the transition weights are determined endogenously, which induces a large number of modes to the log-likelihood function and large areas of the parameter space where the log-likelihood function is flat in multiple directions. Therefore, the model parameters are estimated by running multiple rounds of a two-phase estimation procedure in which a modified genetic algorithm is used to find starting values for a gradient based variable metric algorithm. Because of the multimodality of the log-likelihood function, some of the estimation rounds may end up in different local maximum points, thereby enabling the researcher to build models not only based on the global maximum point but also on the local ones. The estimated models can be conveniently examined with the `summary` and `plot` methods.

The remainder of this paper is organized as follows. Section 2 defines the implemented STVAR models and discusses some of their properties. For structural models, identification of the shocks is also covered. Section 3 discusses estimation of the model parameters. We also illustrates how the STVAR models can be estimated and examined with **sstvars** and how various parameter restrictions can be imposed in the estimation. In Section **??**, we describe demonstrate the use residual based diagnostics to evaluate model adequacy. Section 5 discusses impulse response analysis, including generalized impulse response functions and generalized forecast error variance decompositions. Section **??** shows how the STVAR models can be built with given parameter values. In Section 7, we first show how to simulate observations from a STVAR process, and then we illustrate how to forecast future values of a STVAR process

with a simulation-based Monte Carlo method. Finally, Section 8 concludes and collects some useful functions in **sstvars** to a single table for convenience.

Throughout this paper, we illustrate the use of **sstvars** with a quarterly series consisting of two U.S. variables: the percentage change of real GDP and the percentage change of GDP implicit price deflator, covering the period from 1959Q1 to 2019Q4. We deploy the notation $n_d(\boldsymbol{\mu}, \boldsymbol{\Gamma})$ for the d-dimensional normal distribution with mean $\boldsymbol{\mu}$ and (positive definite) covariance matrix $\boldsymbol{\Gamma}$, and $t_d(\boldsymbol{\mu}, \boldsymbol{\Gamma}, \nu)$ for the d-dimensional t-distribution with mean $\boldsymbol{\mu}$, (positive definite) covariance matrix $\boldsymbol{\Gamma}$, and $\nu > 2$ degrees of freedom. The corresponding density functions are denoted as $n_d(\cdot; \boldsymbol{\mu}, \boldsymbol{\Gamma})$ and $t_d(\cdot; \boldsymbol{\mu}, \boldsymbol{\Gamma}, \nu)$, respectively. By $\mathbf{1}_p = (1, ..., 1)$ $(p \times 1)$, we denote p-dimensional vector of ones. Finally, $\otimes$ denotes Kronecker product.

2. Smooth Transition Vector Autoregressive Models

This section describes the STVAR models implemented in **sstvars**. First, we describe the general framework of STVAR models accommodartes by **sstvars** and present a sufficient condition for their ergodic stationarity. Then, we present the implemented specifications transition weight functions and conditional distributions. Finally, we discuss structural STVAR models and implemented methods for identification of the shocks.

2.1. General framework for STVAR models

Let y_t, $t = 1, 2, ...$, be the d-dimensional time series of interest and $\mathcal{F}_{t-1}$ denote the σ-algebra generated by the random vectors $\{y_{t-j}, j > 0\}$. We consider STVAR models with M regimes and autoregressive order p assumed to satisfy

$$y_t = \sum_{m=1}^{M} \alpha_{m,t}\mu_{m,t} + B_t e_t, \quad e_t \sim IID(0, I_d) \tag{1}$$

$$\mu_{m,t} = \phi_{m,0} + \sum_{i=1}^{p} A_{m,i} y_{t-i}, \quad m = 1, ..., M, \tag{2}$$

$$B_t B_t' = \sum_{m=1}^{M} \alpha_{m,t}\Omega_m, \tag{3}$$

where $\phi_{m,0} \in \mathbb{R}^d$ are intercept parameters, $\Omega_1, ..., \Omega_M$ are the positive definite $(d \times d)$ covariance matrices of the regimes, B_t is an invertible $(d \times d)$ impact matrix that governs the contemporaneous relationships of the shocks and is a function of $\{y_{t-j}, j = 1, ..., p\}$. For the reduced form model, any invertible matrix B_t that satisfies Equation (3) can be assumed without loss of generality, whereas structural models assume a specific structure on B_t. The structural errors e_t $(d \times 1)$ are independent and identically distributed with mean zero and identity covariance matrix, and they are independent of $\mathcal{F}_{t-1}$. The transition weights $\alpha_{m,t}$ are assumed to be $\mathcal{F}_{t-1}$-measurable functions of $\{y_{t-j}, j = 1, ..., p\}$ and to satisfy $\sum_{m=1}^{M} \alpha_{m,t} = 1$ at all t. They express the proportions of the regimes the process is on at each point of time. Through, a STVAR model with autoregressive order p and M regimes is referred to as a STVAR(p, M) model, whenever the order of the model needs to be emphasized.

Conditional on $\mathcal{F}_{t-1}$, the conditional mean of the above described process is $\mu_{y,t} \equiv E[y_t|\mathcal{F}_{t-1}] = \sum_{m=1}^{M} \alpha_{m,t}\mu_{m,t}$, and the conditional covariance matrix is $\Omega_{y,t} \equiv \text{Cov}(y_t|\mathcal{F}_{t-1}) = \sum_{m=1}^{M} \alpha_{m,t}\Omega_m$.

In other words, the conditional mean is a weighted sum the regime-specific means $\mu_{m,t}$ with the weights given by the transition weights $\alpha_{m,t}$, whereas the conditional covariance matrix is a weighted sum of the regime-specific conditional covariance matrices Ω_m. Different STVAR models are obtained by specifying the transition weights or the error distribution in various ways. See Hubrich and Teräsvirta (2013) for a survey on STVAR literature, including formulations more general than our framework.

The package **sstvars** accommodates models in which the transition weights are functions of $\{y_{t-j}, j = 1, ..., p\}$, which is required for applicability of the stationarity condition presented below. Moreover, $\mathcal{F}_{t-1}$-measurability of the transition weights ensures that the true generalized impulse responses functions (Koop, Pesaran, and Potter 1996) can be easily estimated, as completely exogenous switching-variables are excluded from affecting the weights. We also assume that the transition weights are identical for all the individual equations in (1) and (3), which is also required for applicability of the stationarity condition. Consequently, at each t, the process can be described as a weighted sum of linear vector autoregressions.

Stationarity condition

It can be shown that a sufficient condition for the ergodic stationarity of the STVAR model (1)-(3) can expressed in therms of the joint spectral radius (JSR) of certain matrices (Kheifets and Saikkonen 2020). The JSR of a finite set of square matrices $\mathcal{A}$ is defined by

$$\rho(\mathcal{A}) = \limsup_{j\to\infty} \left(\sup_{A\in\mathcal{A}^j} \rho(A) \right)^{1/j}, \tag{4}$$

where $\mathcal{A}^j = \{A_1 A_2 ... A_j : A_i \in \mathcal{A}\}$ and $\rho(A)$ is the spectral radius of the square matrix A. Consider the companion form AR matrices of the regimes defined as

$$\boldsymbol{A}_m = \underset{(dp\times dp)}{\begin{bmatrix} A_{m,1} & A_{m,2} & \cdots & A_{m,p-1} & A_{m,p} \\ I_d & 0 & \cdots & 0 & 0 \\ 0 & I_d & & 0 & 0 \\ \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I_d & 0 \end{bmatrix}}, \; m = 1, ..., M. \tag{5}$$

Kheifets and Saikkonen (2020, Theorem 1) and FILL IN REFERENCE TO LANNE AND VIROLAINEN WORKING PAPER show that if the following condition holds, the STVAR process is ergodic stationary (both strictly and second-order).

Condition 1 $\rho(\{\boldsymbol{A}_1, ..., \boldsymbol{A}_M\}) < 1$.

Condition 1 is, however, computationally demanding the check in practice with a reasonable accuracy (e.g., Chang and Blondel 2013), making it impractical to use in the estimation. Therefore, we consider a necessary condition for Condition 1 that is easier to check in practice, which is that the usual stability condition is satisfied for each of the regimes. Specifically, we assume the following condition, which is analogous to Corollary 1 of Kheifets and Saikkonen (2020), is necessary for Condition 1.

Condition 2 $\max\{\rho(\boldsymbol{A}_1), ..., \rho(\boldsymbol{A}_M)\} < 1$,

where $\rho(\boldsymbol{A}_m)$ is the spectral radius of $\boldsymbol{A}_m$, $m = 1, ..., M$.

Note that validity Condition 2 does not imply the validity of Condition 1 that guarantees stationarity of the model. However, in practice models that satisfy Condition 2 and are not very close to breaking this condition should most often satisfy Condition 1. For checking the validity of Condition 1, (Virolainen 2024) implements the formula of Parrilo and Jadbabaie (2008) for the upper bound proposed by Blondel and Nesterov (2005) (the function `bound_JSR`. But unfortunately the calculations required are computationally very demanding in practice when the model is not small and even a relatively tight upper bound is required to verify Condition 1. Therefore, in the cases where Condition 1 needs to be formally verified, we suggest using the MATLAB toolbox **JSR** by Jungers (2023), as automatically combines several methods for bound the JSR and finds an accurate upper bound with a more reasonable computational effort.

2.2. Structural G-StMVAR model

gmvarkit currently supports two types of structural models: structural models identified recursively by the lower-triangular Cholesky decomposition and structural models identified by conditional heteroskedasticity. Recursive identification of the structural shocks e_t is automatically assumed for reduced form models. In that case, the (time-varying) impact matrix B_t is simply the lower-triangular Cholesky decomposition of the conditional covariance matrix of the reduced form error $u_t = B_t e_t = \sum_{m=1}^{M} s_{m,t}\Omega_{m,t}^{1/2}\varepsilon_{m,t}$, that is, of $\sum_{m=1}^{M} \alpha_{m,t}\Omega_{m,t}$. The rest of this section discusses structural models identified by conditional heteroskedasticity as in Virolainen (2021).

We write the structural G-StMVAR model identified by conditional heteroskedasticity (Virolainen 2021) as

$$y_t = \sum_{m=1}^{M} s_{m,t}(\phi_{m,0} + \sum_{i=1}^{p} A_{m,i}y_{t-i}) + B_t e_t \tag{6}$$

and

$$u_t \equiv B_t e_t = \begin{cases} u_{1,t} \sim n_d(0, \Omega_{1,t}) & \text{if} \quad s_{1,t} = 1 \quad \text{(with probability } \alpha_{1,t}\text{)} \\ \vdots \\ u_{M_1,t} \sim n_d(0, \Omega_{M_1,t}) & \text{if} \quad s_{M_1,t} = 1 \quad \text{(with probability } \alpha_{M_1,t}\text{)} \\ u_{M_1+1,t} \sim t_d(0, \Omega_{M_1+1,t}, \nu_{M_1+1} + dp) & \text{if} \quad s_{M_1+1,t} = 1 \quad \text{(with probability } \alpha_{M_1+1,t}\text{)} \\ \vdots \\ u_{M,t} \sim t_d(0, \Omega_{M,t}, \nu_M + dp) & \text{if} \quad s_{M,t} = 1 \quad \text{(with probability } \alpha_{M,t}\text{)} \end{cases} \tag{7}$$

where the probabilities are expressed conditionally on $\mathcal{F}_{t-1}$ and e_t $(d \times 1)$ an orthogonal structural error. For the GMVAR type regimes, $m = 1, ..., M_1$, $\Omega_{m,t} = \Omega_m$. For the StMVAR type regimes, $m = M_1 + 1, ..., M$, $\Omega_{m,t} = \omega_{m,t}\Omega_m$, where

$$\omega_{m,t} = \frac{\nu_m - 2 + (\boldsymbol{y}_{t-1} - \boldsymbol{1}_p \otimes \mu_m)'\Sigma_{m,p}^{-1}(\boldsymbol{y}_{t-1} - \boldsymbol{1}_p \otimes \mu_m)}{\nu_m - 2 + dp}. \tag{8}$$

The invertible $(d \times d)$ "B-matrix" B_t, which governs the contemporaneous relations of the shocks, is time-varying and a function of $y_{t-1}, ..., y_{t-p}$. With a particular choice of B_t, the conditional covariance matrix of the structural error can be normalized to an identity matrix.

Consequently, a constant sized structural shock will be amplified according to the conditional variance of the reduced form error, thereby reflecting the specific state of the economy.

We have $\Omega_{u,t} \equiv \text{Cov}(u_t|\mathcal{F}_{t-1}) = \sum_{m=1}^{M_1} \alpha_{m,t}\Omega_m + \sum_{m=M_1+1}^{M} \alpha_{m,t}\omega_{m,t}\Omega_m$, while the conditional covariance matrix of the structural error $e_t = B_t^{-1}u_t$ (which are not IID but are martingale differences and therefore uncorrelated) is obtained as

$$\text{Cov}(e_t|\mathcal{F}_{t-1}) = \sum_{m=1}^{M_1} \alpha_{m,t}B_t^{-1}\Omega_m B_t^{\prime -1} + \sum_{m=M_1+1}^{M} \alpha_{m,t}\omega_{m,t}B_t^{-1}\Omega_m B_t^{\prime -1}. \tag{9}$$

Therefore, we need to choose the B-matrix so that the structural shocks are orthogonal regardless of which regime they come from.

We employ the following decomposition to simultaneously diagonalize all the error term covariance matrices:

$$\Omega_m = W\Lambda_m W', \quad m = 1, ..., M, \tag{10}$$

where the diagonal of $\Lambda_m = \text{diag}(\lambda_{m1}, ..., \lambda_{md})$, $\lambda_{mi} > 0$ $(i = 1, ..., d)$, contains the eigenvalues of the matrix $\Omega_m\Omega_1^{-1}$ and the columns of the nonsingular W are the related eigenvectors (that are the same for all m by construction). When $M = 2$, the decomposition (10) always exists, but for $M \geq 3$ its existence requires that the matrices share the common eigenvectors in W. This is, however, testable.

Lanne, Lütkepohl, and Maciejowska (2010, Proposition 1) show that for a given ordering of the eigenvalues, W is unique apart from changing all signs a column, as long as for all $i \neq j \in \{1, ..., d\}$ there exists an $m \in \{2, ..., M\}$ such that $\lambda_{mi} \neq \lambda_{mj}$ (for $m = 1$, $\Lambda_m = I_d$ and $\lambda_{m1} = \cdots = \lambda_{md} = 1$). A locally unique B-matrix that amplifies a constant sized structural shock according to the conditional variance of the reduced form error is therefore obtained as

$$B_t = W(\sum_{m=1}^{M_1} \alpha_{m,t}\Lambda_m + \sum_{m=M_1+1}^{M} \alpha_{m,t}\omega_{m,t}\Lambda_m)^{1/2}. \tag{11}$$

Since $B_t^{-1}\Omega_m B_t^{\prime -1} = \Lambda_m(\sum_{n=1}^{M_1} \alpha_{n,t}\Lambda_n + \sum_{n=M_1+1}^{M} \alpha_{n,t}\omega_{n,t}\Lambda_n)^{-1}$, the B-matrix (11) simultaneously diagonalizes $\Omega_1, ..., \Omega_M$, and $\Omega_{u,t}$ (and thereby also $\Omega_{1,t}, ..., \Omega_{M,t}$) for each t so that $\text{Cov}(e_t|\mathcal{F}_{t-1}) = I_d$.

2.3. Identification of the structural shocks

With the decomposition (10) of $\Omega_1, ..., \Omega_M$ and the B-matrix (11), a statistical identification of the shocks is achieved as long as each pair of the eigenvalues is distinct for some m. In order to identify structural shocks with economic interpretations, they need to be uniquely related to the economic shocks through the constraints on the B-matrix (or equally W) that only the shock of interest satisfies. **?**, Proposition 1 gives formal conditions for global identification of any subset of the shocks when the relevant pairs eigenvalues are distinct in some regime. He also derives conditions for globally identifying some of the shocks when one of the relevant pairs of the eigenvalues is identical in all regimes. For convenience, we repeat the conditions in the former case below, but in the latter case, we refer to **?**, where also the following Proposition is proven.

Proposition 1 *Suppose* $\Omega_m = W\Lambda_m W'$, $m = 1, ..., M$, *where the diagonal of* $\Lambda = diag(\lambda_{m1}, ..., \lambda_{md})$, $\lambda_{mi} > 0$ $(i = 1, ..., d)$, *contains the eigenvalues of the matrix* $\Omega_m\Omega_1^{-1}$ *and the columns of the*

nonsingular W are the related eigenvectors. Then, the last d_1 structural shocks are uniquely identified if

1. *for all $j > d - d_1$ and $i \neq j$ there exists an $m \in \{2, ..., M\}$ such that $\lambda_{mi} \neq \lambda_{mj}$,*
2. *the columns of W in a way that for all $i \neq j > d - d_1$, the ith column cannot satisfy the constraints of the jth column as is nor after changing all signs in the ith column, and*
3. *there is at least one (strict) sign constraint in each of the last d_1 columns of W.*

Condition 3 fixes the signs in the last d_1 columns of W, and therefore the signs of the instantaneous effects of the corresponding shocks. However, since changing the signs of the columns is effectively the same as changing the signs of the corresponding shocks, and the structural shock has a distribution that is symmetric about zero, this condition is not restrictive. The assumption that the last d_1 shocks are identified is not restrictive either, as one may always reorder the structural shocks accordingly.

For example, if $d = 3$, $\lambda_{m1} \neq \lambda_{m3}$ for some m, and $\lambda_{m2} \neq \lambda_{m3}$ for some m, the third structural shock can be identified with the following constraints:

$$B_t = \begin{bmatrix} * & * & * \\ + & + & - \\ + & + & + \end{bmatrix} \text{ or } \begin{bmatrix} - & * & + \\ - & + & - \\ * & + & + \end{bmatrix} \text{ or } \begin{bmatrix} + & 0 & - \\ * & * & * \\ + & * & + \end{bmatrix} \tag{12}$$

and so on, where "$*$" signifies that the element is not constrained, "$+$" denotes strict positive and "$-$" a strict negative sign constraint, and "0" means that the element is constrained to zero. Because imposing zero or sign constraints on W equals to placing them on B_t, they may be justified economically. Furthermore, besides a single sign constraint in each column, the constraints are over-identifying and can thus be also justified statistically. Sign constraints, however, don't reduce the dimension of the parameter space, making some of the measures such as the conventional likelihood ratio test and information criteria unsuitable for testing them. Quantile residual diagnostics, on the other hand, can be used to evaluate how well the restricted model is able to encapsulate the statistical properties of the data compared to the unrestricted alternative.

If condition 1 of Proposition 1 is strengthened to state that for all $i \neq j$ there exists an $m \in \{2, ..., M\}$ such that $\lambda_{mi} \neq \lambda_{mj}$, the model is statistically identified even though only the last d_1 structural shocks have been identified with the proposition. Consequently, the constraints imposed in condition 2 become testable. If it cannot be assumed that all the pairs of the eigenvalues are distinct in some regime, then the testing problem is nonstandard and the conventional asymptotic distributions of likelihood ratio and Wald test statistics become unreliable. Note, however, that since placing zero or sign constraints on W equals to placing them on the B-matrix (11), the constraints imposed in condition 2 can be justified economically as usual.

3. Estimation

gmvarkit collects the parameters of a G-StMVAR model to the $((M(d+d^2p+d(d+1)/2+2)-M_1-1)\times 1)$ vector $\boldsymbol{\theta} = (\boldsymbol{\vartheta}_1, ..., \boldsymbol{\vartheta}_M, \alpha_1, ..., \alpha_{M-1}, \boldsymbol{\nu})$, where $\boldsymbol{\vartheta}_m = (\phi_{m,0}, vec(\boldsymbol{A}_{m,p}), vech(\Omega_m))$

and $\boldsymbol{\nu} = (\nu_{M_1+1}, ..., \nu_M)$. The last mixing weight parameter α_M is not parametrized because it is obtained from the restriction $\sum_{m=1}^{M} \alpha_m = 1$. A G-StMVAR model with autoregressive order p, and M_1 GMVAR type and M_2 StMVAR type mixture components is referred to as G-StMVAR(p, M_1, M_2) model, whenever the order of the model needs to be emphasized. If the model imposes constraints or is a structural model, the parameter vector is different. For details, see the documentation.

3.1. Log-likelihood function

gmvarkit employs the method of maximum likelihood (ML) for estimating the parameters of the G-StMVAR model. Even the exact log-likelihood function is available, as the stationary distribution p consecutive observations in known. Suppose the observed time series is $y_{-p+1}, ..., y_0, y_1, ..., y_T$ and that the initial values are stationary. Then, the log-likelihood function of the G-StMVAR model takes the form

$$L(\boldsymbol{\theta}) = \log\left(\sum_{m=1}^{M} \alpha_m d_{m,dp}(\boldsymbol{y}_0; \mathbf{1}_p \otimes \mu_m, \Sigma_{m,p}, \nu_m)\right) + \sum_{m=1}^{M} l_t(\boldsymbol{\theta}), \tag{13}$$

where $d_{m,dp}(\cdot; \mathbf{1}_p \otimes \mu_m, \Sigma_{m,p}, \nu_m)$ is defined in (**??**) and

$$l_t(\boldsymbol{\theta}) = \log\left(\sum_{m=1}^{M_1} \alpha_{m,t} n_d(y_t; \mu_{m,t}, \Omega_m) + \sum_{m=M_1+1}^{M} \alpha_{m,t} t_d(y_t; \mu_{m,t}, \Omega_{m,t}, \nu_m + dp)\right). \tag{14}$$

If stationarity of the initial values seems unreasonable, one can condition on the initial values and base the estimation on the conditional log-likelihood function, which is obtained by dropping the first term on the right hand side of (13).

Virolainen (2021, Theorem 3) shows that the ML estimator of the G-StMVAR model is strongly consistent and has the conventional limiting distribution under the conventional high-level conditions. In the case of a GMVAR model ($M_1 = M$), however, establishing asymptotic normality of the ML estimator requires less unverified assumptions (Kalliovirta, Meitz, and Saikkonen 2016, Theorem 3).

Structural models identified by the lower-triangular Cholesky decomposition use the same parametrization as the reduced form models (more specifically, reduced form models used to structural analysis assume the recursive identification automatically). If there are two regimes in the model ($M = 2$), the structural G-StMVAR model identified by conditional heteroskedasticity is obtained from estimated reduced form model by decomposing the covariance matrices $\Omega_1, ..., \Omega_M$ as in (10). If $M \geq 3$ or overidentifying constraints are imposed on B_t through W, the model can be reparametrized with W and Λ_m ($m = 2, ..., M$) instead of $\Omega_1, ..., \Omega_M$, and the log-likelihood function can be maximized subject to the new set of parameters and constraints.[1] In this case, the decomposition (10) is plugged in to the log-likelihood function and $vech(\Omega_1), ..., vech(\Omega_M)$ are replaced with $vec(W)$ and $\boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_M$ in the parameter vector $\boldsymbol{\theta}$, where $\boldsymbol{\lambda}_m = (\lambda_{m1}, ..., \lambda_{md})$.

3.2. Two-phase estimation procedure

[1] Namely, instead of constraining $vech(\Omega_1), ..., vech(\Omega_M)$ so that $\Omega_1, ..., \Omega_M$ are positive definite, we impose the constraints $\lambda_{mi} > 0$ for all $m = 2, ..., M$ and $j = 1, ..., d$.

Finding the ML estimate amounts maximizing the log-likelihood function (13) (and (14)) over a high dimensional parameter space satisfying the constraints summarized in Virolainen (2021, Assumption 1). Due to the complexity of the log-likelihood function, numerical optimization methods are required. The maximization problem can, however, be challenging in practice. This is particularly due to the mixing weights' complex dependence on the preceding observations, which induces a large number of modes to the surface of the log-likelihood function, and large areas to the parameter space where it is flat in multiple directions. Also, the popular EM algorithm (Redner and Walker 1984) is virtually useless here, as at each maximization step one faces a new optimization problem that is not much simpler than the original one. Following **?**, **?**, and **?**Virolainen (2018), we therefore employ a two-phase estimation procedure in which a genetic algorithm is used to find starting values for a gradient based method.

The genetic algorithm in **gmvarkit** is, at core, mostly based on the description by Dorsey and Mayer (1995) but several modifications have been deployed to improve its performance. The modifications include the ones proposed by Patnaik and Srinivas (1994) and Smith, Dike, and Stegmann (1995) as well as further adjustments that take into account model specific issues related to the mixing weights' dependence on the preceding observations. For a more detailed description of the genetic algorithm and its modifications, see **?**, Appendix A, where the genetic algorithm is discussed in the univariate context. After running the genetic algorithm, the estimation is finalized with a variable metric algorithm (Nash 1990, algorithm 21, implemented by R Core Team 2022) using central difference approximation for the gradient of the log-likelihood function (see Section 3.4).

3.3. Examples of unconstrained estimation

In this section, we demonstrate how to estimate GSMVAR models with **gmvarkit** and provide several examples in order to illustrate various frequently occurring situations. In addition to the ordinary estimation, we particularly show how a GSMVAR model can be built based on a local-only maximum point when the ML estimate seems unreasonable. We also consider the estimation of the appropriate G-StMVAR model when the estimated StMVAR model contains overly large degrees of freedom estimates (see the related discussion in Virolainen 2021). In the examples, we only consider $p = 1$ models for simplicity and because then the code outputs fit in the margins better, estimation times are shorter, etc. This order may not be best in the modeling perspective, however.

In **gmvarkit**, the GSMVAR models are defined as class `gsmvar` S3 objects, which can be created with given parameter values using the constructor function `GSMVAR` (see Section 6) or by using the estimation function `fitGSMVAR`, which estimates the parameters and then builds the model. For estimation, `fitGSMVAR` needs to be supplied with a univariate time series and the arguments specifying the model. The necessary arguments for specifying the model include the autoregressive order `p`, the number of mixture components `M`, and `model`, which should be either `"GMVAR"`, `"StMVAR"`, or `"G-StMVAR"`. For GMVAR and StMVAR models, the argument `M` is a positive integer, whereas for the G-StMVAR model it is a length two numeric vector specifying the number of GMVAR type regimes in the first element and the number of StMVAR type regimes in the second.

Additional arguments may be supplied to `fitGSMVAR` in order to specify, for example, whether the exact log-likelihood function should be used instead of the conditional one (`conditional`),

how many estimation rounds should be performed (`ncalls`), and how many central processing unit (CPU) cores should be used in the estimation (`ncores`). Some of the estimation rounds may end up in local-only maximum points or saddle points, but reliability of the estimation results can be improved by increasing the number of estimation rounds. A large number of estimation rounds may be required particularly when the number of mixture components is large, as the surface of the log-likelihood function becomes increasingly more challenging. It is also possible to adjust the settings of the genetic algorithm that is used to find the starting values. The available options are listed in the documentation of the function `GAfit` to which the arguments adjusting the settings will be passed.

In general, we recommend being conservative with choice of M due to the identifation problems induced if the number of regimes is chosen too large. Also, estimation of models that contain more than two regimes can be extremely challenging. Another important thing to know about estimation is that the estimation algorithm performs very poorly if some of the AR coefficients very large, signifcantly larger than one. This means that you need scale each component time series so that they vary approximately in the same magnitude. For instance, typically in macroeconomic time series, log-differences should be multiplied by hundred. If the suitable scales are not obvious, you can try out different scales and estimate linear VARs with your favorite package to see whether the AR coeffients are in a reasonable range. When a suitable scale is found, proceed to the GSMVAR models.

We illustrate the use of **gmvarkit** with a quarterly series consisting of two U.S. variables: the percentage change of real GDP and the percentage change of GDP implicit price deflator, covering the period from 1959Q1 to 2019Q4. The following code fits a StMVAR($p = 1, M = 2$) this model to this series (`gdpdef`) using the conditional log-likelihood function and performing 16 estimation rounds with 8 CPU cores. **In practice, hundreds or even thousands of estimation rounds is often required to obtain reliable results. The larger the dimension of the series is and the larger the order of the model is, the more estimation rounds is required.** We use only 16 estimation rounds in this simplistic example to shorten the estimation time, knowing beforehand that the given seeds produce the desired result (in this simplistic case, majority the estimation rounds end up in the MLE anyway, though).

The argument `seeds` supplies the seeds that initialize the random number generator at the beginning of each call to the genetic algorithm, thereby yielding reproducible results.

```
R> library(gmvarkit)
R> data("gdpdef", package="gmvarkit")
R> fit12t <- fitGSMVAR(gdpdef, p=1, M=2, model="StMVAR", ncalls=16, ncores=8,
+    seeds=1:16)

Using 8 cores for 16 estimations rounds...
Optimizing with a genetic algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=26s
Results from the genetic algorithm:
The lowest loglik:  -277.038
The mean loglik:    -258.416
```

```
The largest loglik: -252.168
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=06s
Results from the variable metric algorithm:
The lowest loglik:  -276.974
The mean loglik:    -252.439
The largest loglik: -243.719
Calculating approximate standard errors...
Finished!
Warning messages:
1: In warn_df(p = p, M = M, params = params, model = model) :
  The model contains overly large degrees of freedom parameters. Consider
  switching to the appropriate G-StMVAR model by setting the corresponding
  regimes to GMVAR type with the function 'stmvar_to_gstmvar'.
2: In warn_eigens(ret) :
  Regime 2 has near-singular error term covariance matrix! Consider
  building a model from the next-largest local maximum with the function
  'alt_gsmvar' by adjusting its argument 'which_largest'.
```

The progression of the estimation process is reported with a progress bar giving an estimate of the remaining estimation time. Also statistics on the spread of the log-likelihoods are printed after each estimation phase. The progress bars are generated during parallel computing with the package **pbapply** (Solymos and Zawadzki 2020).

The function throws a warning (the second warning) because at least one the regimes contains a near-singular covariance matrix. This kind of unreasonable boundary points can often be disregarded, and the model can be built based on a reasonable estimate found from a local maximum that is clearly in the interior of the parameter space. Models based on the next-best local maximum can be built with the function `alt_gsmvar` by adjusting its argument `which_largest`.

The following code builds a StMVAR model based on the second-largest local maximum found in the estimation:

```
R> fit12t_alt <- alt_gsmvar(fit12t, which_largest=2)
```

```
Warning message:
In warn_df(p = p, M = M, params = params, model = model) :
  The model contains overly large degrees of freedom parameters. Consider
  switching to the appropriate G-StMVAR model by setting the corresponding
  regimes to GMVAR type with the function 'stmvar_to_gstmvar'.
```

The estimates can be examined with the `print`.

```
R> print(fit12t_alt)
```

```
Reduced form StMVAR model:
 p = 1, M = 2, d = 2, #parameters = 21, #observations = 244 x 2,
```

```
 conditional log-likelihood, intercept parametrization, no AR parameter
 constraints

Regime 1
Mixing weight: 0.83
Regime means: 0.78, 0.54
Df parameter:  7.57

   Y     phi0          A1                                Omega          1/2
1 y1 = [ 0.55 ] + [  0.33 -0.04 ] y1.1 + (         [  0.42 0.00 ] )     eps1
2 y2   [ 0.12 ]   [  0.05  0.71 ] y2.1   ( ARCH_mt [  0.00 0.04 ] )     eps2

Regime 2
Mixing weight: 0.17
Regime means: 0.66, 1.67
Df parameter:  92497.86

   Y     phi0          A1                                Omega           1/2
1 y1 = [ 1.60 ] + [  0.13 -0.61 ] y1.1 + (         [  1.21 -0.04 ] )     eps1
2 y2   [ 0.48 ]   [ -0.03  0.72 ] y2.1   ( ARCH_mt [ -0.04  0.14 ] )     eps2
```

The parameter estimates are reported for each mixture component separately so that the estimates can be easily interpreted. Each regime's autoregressive formula is presented in the form

$$y_t = \varphi_{m,0} + A_{m,1}y_{t-1} + ... + A_{m,p}y_{t-p} + \Omega_{m,t}^{1/2}\varepsilon_{m,t}. \tag{15}$$

If $\Omega_{m,t}^{1/2}$ is time varying, it printed in the form $\Omega_{m,t}^{1/2} = (\text{ARCH\_mt}\Omega_m)^{1/2}$ where ARCH_mt is the $\omega_{m,t}$ defined in (8). No numerical value is given to the ARCH scalar, as it is time-varying. The other statistics are listed above the formula, including the mixing weight pameter α_m, the unconditional mean μ_m, and the degrees freedom parameter ν_m.

The above printout shows that the second regime's degrees of freedom parameter estimate is very large, which might induce numerical problems. However, since a StMVAR model with some degrees of freedom parameters tending to infinity coincides with the G-StMVAR model with the corresponding regimes switched to GMVAR type, one may avoid the problems by switching to the appropriate G-StMVAR model (see Virolainen 2021). Switching to the appropriate G-StMVAR model is recommended also because it removes the redundant degrees of freedom parameters from the model, thereby reducing its complexity. The function `stmvar_to_gstmvar` does this switch automatically by first removing the large degrees of freedom parameters and then estimating the G-StMVAR model with a variable metric algorithm (Nash 1990, algorithm 21) using the induced parameter vector as the initial value.

To exemplify, the following code switches all the regimes of the StMVAR model `fit12t_alt` with a degrees of freedom parameter estimate larger than 100 to GMVAR type, and then estimates the corresponding G-StMVAR model.

```
R> fit12gs <- stmvar_to_gstmvar(fit12t_alt, maxdf=100)
```

We use the `summary` method to obtain a more detailed printout of the estimated the G-StMVAR model:

```
R> summary(fit12gs)

Reduced form G-StMVAR model:
 p = 1, M1 = 1, M2 = 1, d = 2, #parameters = 20, #observations = 244 x 2,
 conditional log-likelihood, intercept parametrization, no AR parameter
 constraints

 log-likelihood: -247.50, AIC: 534.99, HQIC: 563.13, BIC: 604.85

Regime 1 (GMVAR type)
Moduli of 'bold A' eigenvalues:  0.75, 0.10
Cov. matrix 'Omega' eigenvalues: 1.21, 0.14
Mixing weight: 0.17
Regime means: 0.66, 1.67

   Y     phi0          A1                   Omega         1/2
1 y1 = [ 1.60 ] + [  0.13 -0.61 ] y1.1 + [  1.21 -0.04 ]     eps1
2 y2   [ 0.48 ]   [ -0.03  0.72 ] y2.1   [ -0.04  0.14 ]     eps2

Error term correlation matrix:
       [,1]   [,2]
[1,]  1.000 -0.087
[2,] -0.087  1.000

Regime 2 (StMVAR type)
Moduli of 'bold A' eigenvalues:  0.70, 0.34
Cov. matrix 'Omega' eigenvalues: 0.42, 0.04
Mixing weight: 0.83
Regime means: 0.78, 0.54
Df parameter:  7.57

   Y     phi0          A1                             Omega          1/2
1 y1 = [ 0.55 ] + [  0.33 -0.04 ] y1.1 + (         [  0.42 0.00 ] )     eps1
2 y2   [ 0.12 ]   [  0.05  0.71 ] y2.1   ( ARCH_mt [  0.00 0.04 ] )     eps2

Error term correlation matrix:
      [,1]  [,2]
[1,] 1.000 0.014
[2,] 0.014 1.000

Print approximate standard errors with the function 'print_std_errors'.
```

In the G-StMVAR model, estimates for GMVAR type regimes are reported before StMVAR type regimes, in a decreasing order according to the mixing weight parameter estimates. As shown above, the model `fit12gs` incorporates one GMVAR type regime and one StMVAR type regime. Estimates of the unconditional mean, the first p autocovariances and autocorrelations (including the unconditional covariance matrix) can be obtained from the element

`$uncond_moments` of the model object. The conditional moments calculated using the data are available for the process (`$total_cmeans` and `$total_ccovs`) as well as for the regimes separately (`$regime_cmeans` and `$regime_ccovs`). These conditional moments can be conveniently plotted along the series with the function `cond_moment_plot`.

Approximate standard errors can be printed with the function `print_std_errors`, which prints the standard errors in the same form as the print method prints the estimates. Note that the last mixing weight parameter estimate does not have an approximate standard error because it is not parametrized. Likewise, there is no standard error for the intercepts if mean parametrization is used (by setting `parametrization = "mean"` in `fitGSMVAR`) and vice versa. In order to obtain standard errors for the regimewise unconditional means or intercepts, one can easily swap between the mean and intercept parametrizations with the function `swap_parametrization`.

To exemplify, the following code prints approximate standard errors for the model `fit12gs`:

```
R> print_std_errors(fit12gs)

Reduced form model:
p = 1, M = 2, conditional log-likelihood, intercept parametrization,
no AR parameter constraints

APPROXIMATE STANDARD ERRORS

Regime 1 (GMVAR type)
Mixing weight: 0.124

   Y      phi0          A1                   Omega         1/2
1 Y1 = [ 0.759 ] + [ 0.149 0.386 ] Y1.1 + [ 0.264 0.061 ]     eps1
2 Y2   [ 0.234 ]   [ 0.050 0.118 ] Y2.1   [ 0.061 0.029 ]     eps2

Regime 2 (StMVAR type)
Df parameter:  2.740

   Y      phi0          A1                              Omega         1/2
1 Y1 = [ 0.127 ] + [ 0.079 0.196 ] Y1.1 + (          [ 0.070 0.011 ]     eps1
2 Y2   [ 0.037 ]   [ 0.024 0.064 ] Y2.1   ( ARCH_mt [ 0.011 0.008 ]     eps2
```

Missing values are reported when **gmvarkit** is not able to calculate the standard error. This typically happens either because there is an overly large degrees of freedom parameter estimate in the model or because the estimation algorithm did not stop a local maximum. In the former case, switch to the appropariate G-StMVAR with the function `stmvar_to_gstmvar`. In the latter case, make sure the estimate in not an unreasonable near-the-boundary point. If it is, it might appropriate the consider the next-best local maximum with the function `alt_gsmvar`. If it is not a near-the-boundary point, try running more iterations of the variable metric algorithm with the function `iterate_more`. Section 3.4 discusses how to evaluate with **gmvarkit** whether the estimate is a local maximum (and how to improve the reliability that it is the global maximum).

Other statistics reported in the summary printout include the log-likelihood and values of the information criteria, moduli of the eigenvalues of the 'bold A' matrix (see (**??**)) and eigenvalues of the covariance matrix Ω_m. If some of the moduli are very close to one, the related estimates are near the boundary of the stationarity region. If some of the eigenvalues of Ω_m close to zero, the related estimates are near the boundary of positive-definiteness region. As mentioned already multiple times, this kind of near-the-boundary point might be unreasonable and maximize the log-likelihood function for a technical reason, so it might be more appropriate to consider the next-best local maximum with the function `alt_gsmvar`.

This is possible in **gmvarkit**, because the estimation function `fitGSMVAR` stores the estimates from all the estimation rounds so that a GSMVAR model can be built based on any one of them, most conveniently with the function `alt_gsmvar`. The desired estimation round can be specified either with the argument `which_round` or `which_largest`. The former specifies the round in the estimation order, whereas the latter specifies it in a decreasing order of the log-likelihoods.

It is also possible to automatically filter out inappropriate estimates by setting the argument `filter_estimates=TRUE` in `fitSGMVAR`. Then, the function will automatically filter out estimates that it deems "inappropriate". That is, estimates that are not likely solutions of interest. Specifically, solutions that incorporate a near-singular error term covariance matrix (any eigenvalue less than 0.002), mixing weights such that they are close to zero for almost all t for at least one regime, or mixing weight parameter estimate close to zero (or one). It also filters out estimates with any modulus "bold A" eigenvalues larger than 0.9985, as the solution is near the boundary of the stationarity region and likely not a local maximum. `fitSGMVAR` then returns the solution based on the largest log-likelihood that is not filtered out. Other solutions can be studied by using the function `alt_gsmvar` as usual.

3.4. Further examination of the estimates

In addition to examining the summary printout, it is often useful to visualize the model by plotting the mixing weights together with the time series and the model's (marginal) stationary density together with a kernel density estimate of the time series. That is exactly what the plot method for GSMVAR models does. The following command creates the time series plot along with estimated mixing weights:

```
R> plot(fit12gs, type="series")
```

The resulting plot is presented in Figure **??**.

And the following command creates the stationary density plot:

```
R> plot(fit12gs, type="density")
```

The resulting plot is presented in Figure **??**. If the argument `type` is not specified, both of the figures will be plotted.

It is also sometimes interesting to examine the time series of (one-step) conditional means and variances of the process along with the time series the model was fitted to. This can be done conveniently with the function `cond_moment_plot`, where the argument `which_moment` should be specified with `"mean"` or `"variance"` accordingly. In addition to the conditional moment of the process, `cond_moment_plot` also displays the conditional means or variances of

the regimes multiplied by the mixing weights. Note, however, that the conditional variance of the process is not generally the same as the weighted sum of regimewise conditional variances, as it includes a component that encapsulates heteroskedasticity caused by variation in the conditional mean.

The variable metric algorithm employed in the final estimation does not necessarily stop at a local maximum point. The algorithm might also stop at a saddle point or near a local maximum, when the algorithm is not able to increase the log-likelihood, or at any point, when the maximum number of iterations has been reached. In the latter case, the estimation function throws a warning, but saddle points and inaccurate estimates need to be detected by the researcher.

It is well known that in a local maximum point, the gradient of the log-likelihood function is zero, and the eigenvalues of the Hessian matrix are all negative. In a local minimum, the eigenvalues of the Hessian matrix are all positive, whereas in a saddle point, some of them are positive and some negative. Nearly numerically singular Hessian matrices occur when the surface of the log-likelihood function is very flat about the estimate in some directions. This particularly happens when the model contains overly large degrees of freedom parameter estimates or the mixing weights $\alpha_{m,t}$ are estimated close to zero for all $t = 1, ..., T$ for some regime m.

gmvarkit provides several functions for evaluating whether the estimate is a local maximum point. The function `get_foc` returns the (numerically approximated) gradient of the log-likelihood function evaluated at the estimate, and the function `get_soc` returns eigenvalues of the (numerically approximated) Hessian matrix of the log-likelihood function evaluated at the estimate. The numerical derivatives are calculated using a central difference approximation

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_i} \approx \frac{f(\boldsymbol{\theta} + \boldsymbol{h}^{(i)}) - f(\boldsymbol{\theta} - \boldsymbol{h}^{(i)})}{2h}, \quad h > 0, \tag{16}$$

where θ_i is the ith element of $\boldsymbol{\theta}$ and $\boldsymbol{h}^{(i)} = (0, ..., 0, h, 0, ..., 0)$ contains h as its ith element. By default, the difference $h = 6 \cdot 10^{-6}$ is used for all parameters except for overly large degrees of freedom parameters, whose partial derivatives are approximated using larger differences. The difference is increased for large degrees of freedom parameters, because the limited precision of the float point presentation induces artificially rugged surfaces to the their profile log-likelihood functions, and the increased differences diminish the related numerical error. On the other hand, as the surface of the profile log-likelihood function is very flat about a large degrees of freedom parameter estimate, large differences work well for the approximation.

For example, the following code calculates the first order condition for the G-StMVAR model `fit12gs`:

```
R> get_foc(fit12gs)

 [1]  1.475392e-03  2.520058e-03  1.593868e-03 -4.325443e-03
 [5]  3.426033e-03  1.272343e-03 -7.762371e-04 -4.553748e-03
 [9]  3.181280e-03  3.919771e-03 -2.321276e-02  4.170381e-03
[13] -1.782989e-02  3.166647e-03 -3.229744e-03 -3.738130e-03
[17] -9.082465e-03  2.232360e-02 -7.895506e-03  9.746278e-06
```

and the following code calculates the second order condition:

```
R> get_soc(fit12gs)

 [1] -1.329154e-01 -1.389172e+00 -1.273009e+01 -1.865806e+01
 [5] -2.067262e+01 -5.037907e+01 -8.094178e+01 -1.070554e+02
 [9] -1.715455e+02 -2.124879e+02 -2.769413e+02 -3.371079e+02
[13] -4.467047e+02 -1.104002e+03 -1.130339e+03 -1.261037e+03
[17] -1.820865e+03 -9.262083e+03 -1.302900e+04 -3.456585e+04
```

All eigenvalues of the Hessian matrix are negative, which points to a local maximum, but the gradient of the log-likelihood function seems to somewhat deviate from zero. The gradient might be inaccurate, because it is based on a numerical approximation. It is also possible that the estimate is inaccurate, because it is based on approximative numerical estimation, and the estimates are therefore not expected to be exactly accurate. Whether the estimate is a local maximum point with accuracy that is reasonable enough, can be evaluated by plotting the graphs of the profile log-likelihood functions about the estimate. In **gmvarkit**, this can be done conveniently with the function `profile_logliks`.

The exemplify, the following command plots the graphs of profile log-likelihood functions of the estimated G-StMVAR model `fit12gs`:

```
R> profile_logliks(fit12gs, scale=0.02, precision=200)
```

The resulting plot is presented in Figure **??**.

The output shows that the estimate's accuracy is reasonable, as changing any individual parameter value marginally would not increase the log-likelihood much. The argument `scale` can be adjusted to shorten or lengthen the interval shown in the horizontal axis. If one zooms in enough by setting `scale` to a very small number, it can be seen that the estimate is not exactly at the local maximum, but it is so close that moving there would not increase the log-likelihood notably. The argument `precision` can be adjusted to increase the number of points the graph is based on. For faster plotting, it can be decreased, and for more precision, it can be increased. The argument `which_pars` is used to specify the parameters whose profile log-likelihood functions should be plotted. This argument is particularly useful when creating as many plots as there are parameters in the model to a single figure would cause the individual plots to be very small. In such a case, profile log-likelihood functions for subsets of the parameters can be plotted separately by specifying this argument accordingly.

We have discussed tools that can be utilized to evaluate whether the found estimate is a local maximum with a reasonable accuracy. It is, however, more difficult to establish that the estimate is the global maximum. With **gmvarkit**, the best way to increase the reliability that the found estimate is the global maximum, is to run more estimation rounds by adjusting the argument `ncalls` of the estimation function `fitGSMVAR`.

If the model is very large, a very large number of estimation rounds may be required to find the global maximum. If there are two regimes in the model, p is reasonable, and the dimension of the time series at most four, the required number of estimation rounds typically varies from several hundred to several thousand depending on the model and the data. In the simpler models, less estimation rounds are required. In the larger models, and in particular if $M > 2$ or $d > 4$, a significantly large number of estimation rounds may be required obtain the MLE. Another thing that makes the estimation more challenging, are exotic parameter

constraints that do not reduce the dimension of the parameter much. Constraints that greatly reduce complexity of the parameter space (such as constraining the autoregressive matrices to be identical in all regimes[2]), on the other hand, make the estimation easier and reliable estimation of such models thereby require less estimation rounds.

3.5. Estimation of the structural GSMVAR model

As explained, **gmvarkit** currently supports two types of structural models: structural models identified recursively by the lower triangular Cholesky decomposition and structural models identified by conditional heteroskedasticity. Recursive identification is assumed for reduced form models, and thus do not require any further estimation. Generalized impulse response functions and generalized forecast error variance decompositions can be estimated for recursively identified models by using the reduced form models directly in the functions `GIRF` and `GFEVD`. The rest of this section discusses estimation of structural models identified by conditional heteroskedasticity as in Virolainen (2021), which employ a parametrization different to the reduced form models.

The structural GSMVAR models are estimated similarly to the reduced form version, expect that the model is parametrized with W and λ_{mi}, $m = 2, ..., M$, $i = 1, ..., d$ instead of the covariance matrices Ω_m, $m = 1, ..., M$. The estimation is can be done with the function `fitGSMVAR` but now the argument `structural_pars` needs to be supplied with a list providing the constraints on W (which equally imposes the constraints on the B-matrix), and optionally, linear constraints on the λ_{mi} parameters or constraints restricting λ_{mi} to fixed values.

The list `structural_pars` should contain at least the element `W` which is a (dxd) matrix matrix with its entries imposing constraints on W: `NA` indicating that the element is unconstrained, a positive value indicating strict positive sign constraint, a negative value indicating strict negative sign constraint, and zero indicating that the element is constrained to zero. The elements named `C_lambda` and `fixed_lambdas` are optional (and alternative to each other).

If `C_lambda` is specified, it should be a $(d(M-1) \times r)$ constraint matrix that satisfies $(\lambda_2, ..., \lambda_M) = C_\lambda \gamma$ where $\lambda_m = (\lambda_{m1}, ..., \lambda_{md})$ and γ is the new $(rx1)$ parameter subject to which the model is estimated (similarly to AR parameter constraints). The entries of `C_lambda` must be either positive or zero. Ignore (or set to `NULL`) if the eigenvalues λ_{mi} should not be constrained. Note that other constraints than constraining some of the λ_{mi} to be identical are not recommended but if such constraints are imposed, the argument `lambda_scale` in the genetic algorithm (see `?GAfit`) should be adjusted accordingly. If some of the λ_{mi} are constrained to be identical, make sure the appropriate zero constraints placed in the W matrix, because otherwise the MLE does not identify and you probably won't obtain any useful estimates (see Virolainen 2021, Proposition 2).

If `fixed_lambdas` is specified, it should be a $d(M-1)$ length numeric vector $(\lambda_2, ..., \lambda_M)$ specifying fixed values for the eigenvalue parameters λ_{mi}. They should be strictly larger than zero. Ignore (or set to `NULL`) if the eigenvalues λ_{mi} should not be constrained. Note that you cannot use both `C_lambda` and `fixed_lambdas` at the same time.

Reliable estimation of structural GSMVAR models typically requires much more estimation rounds than the estimation of the reduced form models. However, when $M = 2$, every reduced form model has an implied statistically identified

[2]Models constrained in this way can often be reliably estimated with a reasonable number of estimation rounds even when $M > 2$

structural model, which can be built without any additional estimation (this will be discussed next). We recommend considering this implied model first. Then, if overidentifying constraints are to be imposed on the B-matrix (or equally W), we recommend using the unrestricted estimate to create an initial guess for the constrained parameter vector and pass this to the genetic algorithm as an initial population. See the help page `?GAfit` for the arguments that can be passed by `fitGSMVAR` to the genetic algorithm. Create the initial guess for the parameter vector by using the form given in documentation of the argument `initpop`. If $M \neq 2$, the structural model needs to be estimated in the normal way with `fitGSMVAR`, however.

Building structural model based on a reduced form model

If the number of regimes is two ($M = 2$), a structural model can be built based on a reduced form model, because the matrix decomposition used in the simultaneous diagonalization of the error term covariance matrices always exists. This can be done with function `gsmvar_to_sgsmvar` which should be supplied with the reduced form model, and it then returns a corresponding structural model. After creating the structural model, the columns of W can be reordered with the function `reorder_W_columns` which also reorders all λ_{mi} accordingly (and hence the resulting model will coincide with the original reduced form model). Also, all signs any column of W can be swapped with the function `swap_W_signs`.

The exemplify, the following code creates statistically identified structural model based on the reduced form model `fit12gs` and then prints the estimates.

```
R> fit12gss <- gsmvar_to_sgsmvar(fit12gs)
R> fit12gss

Structural G-StMVAR model:
 p = 1, M1 = 1, M2 = 1, d = 2, #parameters = 20, #observations = 244 x 2,
 conditional log-likelihood, intercept parametrization, no AR parameter
 constraints

Regime 1 (GMVAR type)
Mixing weight: 0.17
Regime means: 0.66, 1.67

   Y     phi0          A1                   Omega          1/2
1 y1 = [ 1.60 ] + [  0.13 -0.61 ] y1.1 + [  1.21 -0.04 ]     eps1
2 y2   [ 0.48 ]   [ -0.03  0.72 ] y2.1   [ -0.04  0.14 ]     eps2

Regime 2 (StMVAR type)
Mixing weight: 0.83
Regime means: 0.78, 0.54
Df parameter:  7.57

   Y     phi0          A1                              Omega           1/2
1 y1 = [ 0.55 ] + [  0.33 -0.04 ] y1.1 + (         [  0.42 0.00 ] )     eps1
2 y2   [ 0.12 ]   [  0.05  0.71 ] y2.1   ( ARCH_mt [  0.00 0.04 ] )     eps2
```

```
Structural parameters:
        W             lamb2
1 [  0.95 -0.55 ]   [  0.36 ]
2 [  0.16  0.34 ] , [  0.28 ]

The B-matrix (or equally W) is subject to 0 zero constraints and 2 sign
constraints. The eigenvalues lambda_{mi} are not subject to linear constraints.
```

Estimates for the structural parameters, W and the eigenvalues, are printed last.

If there is only one mixture component, i.e., `M == 1`, `gsmvar_to_sgsmvar` returns a symmetric and pos. def. square root matrix of the error term covariance matrix by default. But one may also employ lower triangular Cholesky identification by setting `cholesky = TRUE` in the arguments.

Estimating overidentified structural GSMVAR models

Sometimes is appropriate to impose overidentifying constraints on W (or equally the B-matrix). With preliminary estimates from the just-identified model in the case $M > 1$ or any model in the case $M = 1$, it is convenient to use the function `estimate_sgsmvar`. It takes in a reduced form or structural GSMVAR model as a class 'gsmvar' object and new constraints in the argument `new_W` as a matrix expressing the sign or zero constraints. Strictly positive or negative elements signify strict sign constraints, zeros zero constraints, and `NA` values that the element is unconstrained.

`estimate_sgsmvar` then creates preliminary estimate based on the supplied model and the constraints, and then runs the two-phase estimation with settings of the genetic algorithm such that the search is focused on the neighbourhood of the preliminary estimate. Thus, it will lead to the correct ML estimate only if the unconstrained estimate is close to the constrained one in the first place. It is therefore useful for imposing zero constraints for elements that are close to zero in the unrestricted estimate, for instance.

It is important to make sure that supplied model readily satisfies the sign constraints that are imposed. To achieve this, you can swap the signs in each column of the W matrix with the function `swap_W_signs`. If the sign constraints are not not readily satisfie, the preliminary estimate switches the signs and will probably lead to incorrect estimate.

`estimate_sgsmvar` can also be used to estimate models that are not identified, i.e., one regime models. If it supplied with a reduced form model, it will first apply the function `gsmvar_to_sgsmvar`, then impose the constraints and finally estimate the model.

3.6. Constrained estimation

Linear constraints on the autoregressive parameters

Imposing linear constraints on the autoregressive parameters of GMVAR model is straightforward in **gmvarkit**. The constraints are expressed in a somewhat general form which allows to impose a wide class of constraints but one needs to take the time to construct the constraint matrix carefully for each particular case.

We consider constraints of form

$$(\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_M) = \boldsymbol{C}\boldsymbol{\psi}, \tag{17}$$

$$\boldsymbol{\phi}_m = (vec(A_{m,1}), ..., vec(A_{m,p})) \ \ (pd^2x1), \ \ m = 1, ..., M, \tag{18}$$

$\boldsymbol{C}$ is known (Mpd^2xq) constraint matrix (of full column rank) and $\boldsymbol{\psi}$ is unknown $(qx1)$ parameter vector.

The parameter vector for constrained model has the size $((M(d + d(d + 1)/2 + 1) + q - 1)x1)$ and the form

$$\boldsymbol{\theta} = (\phi_{1,0}, ..., \phi_{M,0}, \boldsymbol{\psi}, \alpha_1, ..., \alpha_{M-1}, \boldsymbol{\nu}), \tag{19}$$

where $\boldsymbol{\psi}$ is the $(qx1)$ parameter vector containing constrained autoregressive parameters. As in the case of regular models, instead of the intercept parametrization that takes use of intercept terms $\phi_{m,0}$, one may use the mean parametrization with regimewise means μ_m instead $(m = 1, ..., M)$.

Examples of linear constraints

Consider the following two common uses of linear constraints: restricting the autoregressive matrices to be the same for all regimes and constraining some AR parameters to zero. Of course also some other constraints may be useful, but we chose to show illustrative examples of these two, as they are taken use of in Kalliovirta *et al.* (2016).

Restricting AR matrices to be the same for all regimes

To restrict the AR matrices to be the same for all regimes, we want $\boldsymbol{\phi}_m$ to be the same for all $m = 1, ..., M$. The parameter vector $\boldsymbol{\psi}$ $(qx1)$ then corresponds to any $\boldsymbol{\phi}_m = \boldsymbol{\phi}$, and therefore $q = pd^2$. For the constraint matrix we choose

$$\boldsymbol{C} = [I_{pd^2} : \cdots : I_{pd^2}]' \ \ (Mpd^2xpd^2), \tag{20}$$

that is, M pieces of (pd^2xpd^2) diagonal matrices stacked on top of each other, because then

$$\boldsymbol{C}\boldsymbol{\psi} = (\boldsymbol{\psi}, ..., \boldsymbol{\psi}) = (\boldsymbol{\phi}, ..., \boldsymbol{\phi}). \tag{21}$$

For instance, if there are two regimes in the model, the appropriate constraint matrix then created with R as

```
R> p <- 1 # Any autoregressive order
R> d <- 2 # Whatever the dimension of the time series is
R> I_pd2 <- diag(p*d^2) # The appropriate diagonal matrix
R> (C1 <- rbind(I_pd2, I_pd2)) # Stack them on top of each other
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
[5,]    1    0    0    0
[6,]    0    1    0    0
[7,]    0    0    1    0
[8,]    0    0    0    1
```

The command `fitGSMVAR(gdpdef, p=1, M=2, model="GMVAR", constraints=C1)` would then estimate a GMVAR$(1, 2)$ model with the AR matrices constrained to be the same in both regimes. In practice, you might want to adjust the number of CPU cores used, the of estimation rounds, and set seeds. Notably, with the dimension of the time series being only two and $p = 1$ with two regimes, almost all of the estimation rounds end up in the MLE. Also, because model has the AR matrices constrained to be the same for all regimes, the estimation is much easier than with freely estimated models.

Restricting AR parameters to be the same for all regimes and constraining non-diagonal elements of coefficient matrices to be zero

The previous example shows how to restrict the AR parameters to be the same for all regimes, but say we also want to constrain the non-diagonal elements of coefficient matrices $A_{m,i}$ $(m = 1, ..., M, i = 1, ..., p)$ to be zero. We have the constrained parameter $\boldsymbol{\psi}$ $(qx1)$ representing the unconstrained parameters $(\boldsymbol{\phi_1}, ..., \boldsymbol{\phi}_M)$, where by assumption $\boldsymbol{\phi}_m = \boldsymbol{\phi} = (vec(A_1), ..., vec(A_p))$ (pd^2x1) and the elements of $vec(A_i)$ $(i = 1, ..., p)$ corresponding to the diagonal are zero.

For illustrative purposes, let's consider a GMVAR model with autoregressive degree $p = 2$, number of mixture components $M = 2$ and number of time series in the system $d = 2$. Then we have

$$\boldsymbol{\phi} = (A_1(1,1), 0, 0, A_1(2,2), A_2(1,1), 0, 0, A_2(2,2)) \ (8x1) \text{ and} \tag{22}$$

$$\boldsymbol{\psi} = (A_1(1,1), A_1(2,2), A_2(1,1), A_2(2,2)) \ (4x1). \tag{23}$$

By a direct calculation, we can see that choosing the constraint matrix

$$\boldsymbol{C} = \begin{bmatrix} \tilde{\boldsymbol{c}} \\ \tilde{\boldsymbol{c}} \end{bmatrix} \ (Mpd^2x4), \tag{24}$$

where

$$\tilde{\boldsymbol{c}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ (pd^2x4) \tag{25}$$

satisfies $\boldsymbol{C\psi} = (\boldsymbol{\phi}, ..., \boldsymbol{\phi})$.

The above constraint matrix can be created with R as

```
R> c_tilde <- matrix(0, nrow=2*2^2, ncol=4)
R> c_tilde[c(1, 12, 21, 32)] <- 1
R> C2 <- rbind(c_tilde, c_tilde)
R> C2
```

```
      [,1] [,2] [,3] [,4]
 [1,]    1    0    0    0
```

```
 [2,]    0    0    0    0
 [3,]    0    0    0    0
 [4,]    0    1    0    0
 [5,]    0    0    1    0
 [6,]    0    0    0    0
 [7,]    0    0    0    0
 [8,]    0    0    0    1
 [9,]    1    0    0    0
[10,]    0    0    0    0
[11,]    0    0    0    0
[12,]    0    1    0    0
[13,]    0    0    1    0
[14,]    0    0    0    0
[15,]    0    0    0    0
[16,]    0    0    0    1
```

The command `fitGSMVAR(gdpdef, p=2, M=2, model="GMVAR", constraints=C2)` would then estimate a GMVAR$(2, 2)$ model with the AR matrices constrained to be the same in both regimes and the off-diagonal elements constrained to zero (again, you may want to adjust the arguments `ncalls`, `ncored`, and `seeds`).

Constraining the unconditional means of some regimes to be the same

In addition to constraining the autoregressive parameters, **gmvarkit** allows to constrain the unconditional means of some regimes to be the same. This feature is, however, only available for models that are parametrized with the unconditional means instead of intercepts (because some of the estimation is always done with mean-parametrization and one cannot generally swap the parametrization when constraints are imposed on means/intercepts). With the mean-parametrization employed (by setting `parametrization="mean"`), one may define groups of regimes that have the same mean parameters using the argument `same_means`. For instance, with three regime model ($M = 3$) the argument `same_means=list(c(1, 3), 2)` sets the unconditional means of the first and third regimes to be the same while allows the second regime to have different mean.

One can also combine linear constraints on the AR parameters with constraining some of the means to be the same. This allows, for instance, to estimate a model in which only the covariance matrix varies in time. To exemplify, the following code (which is not executed in this vignette) estimates a GMVAR$(p = 4, M = 2)$ model such that the unconditional means and autoregression matrices are constrained be the same in both regimes. The resulting model thereby has time-varying covariance matrix but otherwise it is linear.

```
R> I_pd2 <- diag(4*2^2) # The appropriate diagonal matrix for the constraint matrix
R> C3 <- rbind(I_pd2, I_pd2) # Stack them on top of each other
R> fit42cm <- fitGSMVAR(gdpdef, p=4, M=2, model="GMVAR", parametrization="mean",
+    same_means=list(1:2), constraints=C3, ncalls=16, ncores=8, seeds=1:16)

Using 8 cores for 16 estimations rounds...
Optimizing with a genetic algorithm...
```

```
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=31s
Results from the genetic algorithm:
The lowest loglik:  -227.251
The mean loglik:    -225.743
The largest loglik: -224.648
Optimizing with a variable metric algorithm...
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=03s
Results from the variable metric algorithm:
The lowest loglik:  -223.311
The mean loglik:    -223.311
The largest loglik: -223.311
Calculating approximate standard errors...
Finished!
```

Constraining the mixing weight parameters alphas to fixed values

It is also possible to constrain the mixing weight parameters $\alpha_1, ..., \alpha_{M-1}$ to some fixed values. To do so, specify the fixed values in the argument `weight_constraints` as a $(M-1 \times 1)$ vector $(\alpha_1, ..., \alpha_{M-1})$. Each element should be strictly between zero and one, and the sum of all of them should be strictly less than one. For instance, when $M = 2$, specifying `weight_constraints=0.6` constraints the mixing weights parameters of the first regime as $\alpha_1 = 0.6$ (and hence $\alpha_2 = 0.4$).

3.7. Testing parameter constraints

One way to asses the validity of the imposed constraints is to compare the values of information criteria of the constrained and unconstrained models. **gmvarkit**, however, also provides functions for testing the constraints with the likelihood ratio test, Wald test, and Rao's test, which are applicable as the ML estimator of a GSMVAR model has the conventional asymptotic distribution (as long as the model is correctly specified and one is willing to assume the validity of the required unverified assumptions; see Virolainen 2021, Theorem 3, and Kalliovirta *et al.* (2016), Theorem 3). For a discussion on the likelihood ratio and Wald tests, see Buse (1982) and the references therein, for example.

The likelihood ratio test considers the null hypothesis that the true parameter value $\boldsymbol{\theta}_0$ satisfies some constraints imposed on these parameters (such that the constrained parameter space is a subset of the parameter space, which is presented in Virolainen 2021, Assumption 2 for the GSMVAR models). Denoting by $\hat{L}_U$ and $\hat{L}_C$ the (maximized) log-likelihoods based on the unconstrained and constrained ML estimates, respectively, the test statistic takes the form

$$LR = 2(\hat{L}_U - \hat{L}_C). \tag{26}$$

Under the null, the test statistic is asymptotically χ^2-distributed with the degrees of freedom given by the difference in the dimensions of the unconstrained and constrained parameter spaces. With **gmvarkit**, the likelihood ratio test can be calculated with the function `LR_test`, which takes the unconstrained model (a class `gsmvar` object) as its first argument and the constrained model as the second argument.

gmvarkit implements the Wald test of the null hypothesis

$$A\boldsymbol{\theta}_0 = c, \tag{27}$$

where A is a $(k \times d)$ matrix with full row rank, c is a $(k \times 1)$ vector, $\boldsymbol{\theta}_0$ is the true parameter value, d is the dimension of the parameter space, and k is the number of constraints. The Wald test statistic takes the form

$$W = (A\hat{\boldsymbol{\theta}} - c)'[A\mathcal{J}(\hat{\boldsymbol{\theta}})^{-1}A']^{-1}(A\hat{\boldsymbol{\theta}} - c), \tag{28}$$

where $\mathcal{J}(\hat{\boldsymbol{\theta}})$ is the observed information matrix evaluated at the ML estimate $\hat{\boldsymbol{\theta}}$. Under the null, the test statistic is asymptotically χ^2-distributed with k degrees of freedom (which is the difference in dimensions of the constrained and unconstrained parameter spaces). With **gmvarkit**, the Wald test can be calculated with function `Wald_test`, which takes the estimated unconstrained model (as a class `gsmvar` object) as the first argument, the matrix A as the second argument, and the vector c as the third argument.

Rao's test is implemented to the function `Rao_test` (see function documentation on how to use it).

Note that the standard tests are not applicable if the number of GMVAR or StMVAR type regimes is chosen too large, as then some of the parameters are not identified, causing the result of the asymptotic normality of the ML estimator to break down. This particularly happens when one tests for the number of regimes in the model, as the under the null some of the regimes are reduced from the model[3] (see the related discussion in Virolainen 2021). Similar caution applies for testing whether a regime is of the GMVAR type against the alternative that it is of the StMVAR type: then $\nu_m = \infty$ under the null for the regime m to be tested, which violates the assumption that the parameter value is in the interior of a compact subset of the parameter space (see Virolainen 2021, Theorem 3 and Assumption 1).

4. Quantile residual based model diagnostics

In the GSMVAR models, the empirical counterparts of the error terms $\varepsilon_{m,t}$ in (**??**) cannot be calculated, because the regime that generated each observation is unknown, making the conventional residual based diagnostics unavailable. Therefore, **gmvarkit** utilizes so called quantile residuals, which are suitable for evaluating adequacy of the GSMAR models.

Denote by y_t, $t = 1, 2, ...$, the time series of interest and $\mathcal{F}_{t-1}$ the σ-algebra generated by the random variables or vectors $\{y_{t-j}, j > 0\}$. Moreover, let $\boldsymbol{\theta}$ denote the relevant parameter vector. Kalliovirta (2012) defines univariate quantile residuals as

$$R_{t,\boldsymbol{\theta}} = \Phi^{-1}(F(y_t; \boldsymbol{\theta} \mid \mathcal{F}_{t-1})), \tag{29}$$

where $\Phi(\cdot)^{-1}$ is the standard normal quantile function and $F(\cdot \mid \mathcal{F}_{t-1})$ is the conditional distribution function of the considered model.

Kalliovirta and Saikkonen (2010) define multivariate quantile residuals analogously to the univariate ones but by taking into account the dependence of the component time series

[3] Meitz and Saikkonen (2021) have, however, recently developed such tests for mixture models with Gaussian conditional densities

from each other. Denote $\mathcal{A}_{j-1} = \sigma(y_{1,t}, ..., y_{j-1,t})$ and by $f(\cdot|\sigma(\mathcal{F}_{t-1}, \mathcal{A}_{j-1})) = f_{j-1,t-1}(\cdot)$ the conditional density function conditional on the σ-algebra $\sigma(\mathcal{F}_{t-1}, \mathcal{A}_{j-1})$

The conditional density function of the random vector y_t can be expressed in a product form by conditioning to the components y_t in addition to the history $\mathcal{F}_{t-1}$ as

$$f(y_t; \theta|\mathcal{F}_{t-1}) = \prod_{j=1}^{d} f_{j-1,t-1}(y_{j,t}; \boldsymbol{\theta}), \tag{30}$$

where $y_{j,t}$ is the jth component of y_t and $f_{0,t-1}(y_{1,t}; \boldsymbol{\theta}) = f_{1,t-1}(y_{1,t}; \boldsymbol{\theta})$ is the marginal conditional density function of $y_{1,t}$ conditional on $\mathcal{F}_{t-1}$.

The conditional distribution functions corresponding to the density functions $f_{j-1,t-1}(\cdot; \boldsymbol{\theta})$ in (30) are of the form

$$F_{j-1,t-1}(y_{j,t}; \boldsymbol{\theta}) = \int_{-\infty}^{y_{j,t}} f_{j-1,t-1}(u; \boldsymbol{\theta}) du. \tag{31}$$

The multivariate quantile residuals are then defined as

$$R_{t,\boldsymbol{\theta}} = \begin{bmatrix} R_{1t,\boldsymbol{\theta}} \\ R_{2t,\boldsymbol{\theta}} \\ \vdots \\ R_{dt,\boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} \Phi^{-1}(F_{0,t-1}(y_{1,t}; \boldsymbol{\theta})) \\ \Phi^{-1}(F_{1,t-1}(y_{2,t}; \boldsymbol{\theta})) \\ \vdots \\ \Phi^{-1}(F_{d-1,t-1}(y_{d,t}; \boldsymbol{\theta})) \end{bmatrix}, \tag{32}$$

and its empirical counterpart, $r_{t,\hat{\boldsymbol{\theta}}}$, is obtained by replacing the parameter $\boldsymbol{\theta}$ with its maximum likelihood estimate $\hat{\boldsymbol{\theta}}$. Closed form expressions for the quantile residuals of the G-StMVAR model (which encompasses GMVAR and StMVAR models as special cases) are derived in Appendix B

For a correctly specified GSMVAR model employing the ML estimator, the empirical counterparts of multivariate quantile residuals are asymptotically multivariate standard normal (Kalliovirta and Saikkonen 2010, Lemma 3). They can therefore be utilized in graphical diagnostic simalarly to the conventional Pearson's residuals. For the graphical diagnostics, **gmvarkit** provides function `diagnostic_plot` which plots the quantile residual time series, auto- and crosscorrelations of the quantile residuals, auto- and crosscorrelations of the squared quantile residuals, and normal quantile-quantile plots as well as histrograms of the quantile residuals.

Kalliovirta and Saikkonen (2010) also propose three diagnostic tests for testing normality, autocorrelation, and conditional heteroskedasticity of the quantile residuals. The tests can be based either on the data or on a simulation procedure. If the sample is short, tests based on the data can be too forgiving, so to obtain more reliable test results the simulation procedure is recommended (with sample size of at least several thousand). The tests can be calculated with **gmvarkit** by using the function `quantile_residual_tests`. The simulation procedure is employed if the argument `nsim` is set larger the number of observations (in each component time series). In this case, `nsim` sets the length of the sample path used in the simulation procedure. If one is concerned about autocorrelation or conditional heteroskedasticity in a specigic lag, the (standardized) individual statistics discussed in Kalliovirta and Saikkonen (2010) can be examined. The function `quantile_residual_tests` returns them automatically for the specified lags.

5. Impulse response analysis

5.1. Generalized impulse response function

We consider the generalized impulse response function (GIRF) (Koop *et al.* 1996) defined as

$$\text{GIRF}(n, \delta_j, \mathcal{F}_{t-1}) = \text{E}[y_{t+n}|\delta_j, \mathcal{F}_{t-1}] - \text{E}[y_{t+n}|\mathcal{F}_{t-1}], \tag{33}$$

where n is the chosen horizon, $\mathcal{F}_{t-1} = \sigma\{y_{t-j}, j > 0\}$ as before, the first term in the RHS is the expected realization of the process at time $t + n$ conditionally on a structural shock of magnitude $\delta_j \in \mathbb{R}$ in the jth element of e_t at time t and the previous observations, and the second term in the RHS is the expected realization of the process conditionally on the previous observations only. GIRF thus expresses the expected difference in the future outcomes when the specific structural shock hits the system at time t as opposed to all shocks being random.

Due to the p-step Markov property of the GSMVAR model, conditioning on (the σ-algebra generated by) the p previous observations $\boldsymbol{y}_{t-1} \equiv (y_{t-1}, ..., y_{t-p})$ is effectively the same as conditioning on $\mathcal{F}_{t-1}$ at the time t and later. The history $\boldsymbol{y}_{t-1}$ can be either fixed or random, but with random history the GIRF becomes a random vector, however. Using fixed $\boldsymbol{y}_{t-1}$ makes sense when one is interested in the effects of the shock in a particular point of time, whereas more general results are obtained by assuming that $\boldsymbol{y}_{t-1}$ follows the stationary distribution of the process. If one is, on the other hand, concerned about a specific regime, $\boldsymbol{y}_{t-1}$ can be assumed to follow the stationary distribution of the corresponding component model.

In practice, the GIRF and its distributional properties can be approximated with a Monte Carlo algorithm that generates independent realizations of the process and then takes the sample mean for point estimate. If $\boldsymbol{y}_{t-1}$ is random and follows the distribution G, the GIRF should be estimated for different values of $\boldsymbol{y}_{t-1}$ generated from G, and then the sample mean and sample quantiles can be taken to obtain the point estimate and confidence intervals. The algorithm implemented in **gmvarkit** is presented in Appendix C.

Because the GSMVAR model allows to associate specific features or economic interpretations for different regimes, it might be interesting to also examine the effects of a structural shock to the mixing weights $\alpha_{m,t}$, $m = 1, ..., M$. We then consider the related GIRFs $E[\alpha_{m,t+n}|\delta_j, \boldsymbol{y}_{t-1}] - E[\alpha_{m,t+n}|\boldsymbol{y}_{t-1}]$ for which point estimates and confidence intervals can be constructed similarly to (33).

In **gmvarkit**, the GIRF can be estimated with the function `GIRF` which should be supplied with the estimated GSMVAR model or a GSMVAR built with hand-specified parameter values using the function `GSMVAR`. If a reduced form model is supplied to `GIRF`, recursive identification by lower-triangular Cholesky decomposition is automatically assumed. The size of the structural shock can be set with the argument `shock_size`. If not specified, the size of one standard deviation is used; that is, the size one. Among other arguments, the function may also be supplied with the argument `init_regimes` which specifies from which regimes' stationary distributions the initial values are generated from (if more than one regime is specified, the initial values will be generated from a mixture of the stationary distributions with the relative mixing proportions given by the mixing weight parameters). If more than one regime is specified, a mixture distribution with weights given by the mixing weight parameters is used. Alternatively, one may specify fixed initial values with the argument `init_values`.

Note that the confidence intervals (whose confidence level can be specified with the argument `ci`) reflect uncertainty about the initial value only and not about the parameter estimates.

Due to the nonlinear nature of the model, GIRFs estimated from different starting values, or with different sign or magnitude of the shock, generally move the variables differently. Sometimes it is, however, of interest to scale the impulse responses so that they correspond to movement of some specific sign and magnitude of some specific variable. In **gmvarkit**, this is most conveniently achieved with the arguments `scale` and `scale_type`. The argument `scale` can be specified in order to scale the GIRFs to some of the shocks so that they correspond to a specific magnitude of instantaneous or peak response of some specific variable. For a single shock, it should a length three vector where the shock of interest is given in the first element (an integer in $1, ..., d$), the variable according to which the GIRFs should be scaled in the second element (an integer in $1, ..., d$), and the magnitude of the given variable's instantaneous or peak response in the third element (a non-zero real number). If the GIRFs of multiple shocks should be scaled, provide a matrix which has one column for each of the shocks with the columns being the length three vectors described above. The argument `scale_type` should be either `"instant"` or `"peak"` specifying whether you want to scale according to the instantaneous movement of peak response. If `"peak"`, the scale is based on the largest magnitude of peak response in absolute value. Scaling according to peak response won't based on values after the horizon specified in the argument `"scale_horizon"`. Note that if you scale the GIRFs, the scaled GIRFs of mixing weights can be outside the interval from zero to one.

Because estimating the GIRF and their confidence intervals is computationally demanding, parallel computing is taken use of to shorten the estimation time. The number of CPU cores used can be set with the argument `ncores`. The objects returned by the `GIRF` function have their own `plot` and `print` methods. Also, cumulative impulse responses of the specified variables can be obtained directly by specifying the argument `which_cumulative`.

5.2. Generalized forecast error variance decomposition

We consider the generalized forecast error variance decomposition (GFEVD) (Lanne and Nyberg 2016) that is defined for variable i, shock j, and horizon n as

$$\text{GFEVD}(n, y_{it}, \delta_j, \mathcal{F}_{t-1}) = \frac{\sum_{l=0}^{n} \text{GIRF}(l, \delta_j, \mathcal{F}_{t-1})_i^2}{\sum_{k=1}^{d}\sum_{l=0}^{n} \text{GIRF}(l, \delta_k, \mathcal{F}_{t-1})_i^2}, \tag{34}$$

where n is the chosen and $\text{GIRF}(l, \delta_j, \mathcal{F}_{t-1})_i$ is the ith element of the related GIRF (see also the notation described for GIRF in the previous section). That is, the GFEVD is otherwise similar to the conventional forecast error variance decomposition but with GIRFs in the place of conventional impulse response functions. Because the GFEVDs sum to unity (for each variable), they can be interpreted in a similar manner to the conventional FEVD.

In **gmvarkit**, the GFEVD can be estimated with the function `GFEVD`. As with the GIRF, the GFEVD is dependent on the initial values. The type of the initial values is set with the argument `initval_type`, and there are three options:

1. `"data"` which estimates the GIRFs for all possible length p histories in the data and then the GIRFs in the GFEVD are obtained as the sample mean over those GIRFs.

2. `"random"` which generates the initial values from the stationary distribution of the process or from the mixture of the stationary distributions of some specific regime(s) with the relative mixing proportions given by the mixing weight parameters. The initial regimes can be set with the argument `init_regimes`. The GIRFs in the GFEVD are obtained as the sample mean over the GIRFs estimated for the different random initial values.

3. `"fixed"` which estimates the GIRFs for a single fixed initial value that is set with the argument `init_values`.

The shock size is the same for all scalar components of the structural shock and it can be adjusted with the argument `shock_size`. If the GIRFs for some variables should be cumulative before calculating the GFEVD, specify them with the argument `which_cumulative`. Finally, note that the GFEVD objects have their own plot and print methods.

5.3. Linear impulse response functions

It is also possible to calculate linear impulse response functions (IRF) based on a specific regime of the estimated model by using the function `linear_IRF`. If the autoregressive dynamics of the model are linear (i.e., either $M = 1$ or mean and AR parameters are constrained identical across the regimes), confidence bounds can be estimated based on a type of a fixed-design wild residual bootstrap method. `gmvarkit` implements the method proposed Herwartz and Lütkepohl (2014).

6. Building a GSMVAR model with specific parameter values

The function `GSMVAR` facilitates building GSMVAR models without estimation, for instance, in order to simulate observations from a GSMVAR process with specific parameter values. The parameter vector (of length $M(d + d^2p + d(d + 1)/2 + 2) - M_1 - 1$ for unconstrained reduced form models) has the form $\boldsymbol{\theta} = (\boldsymbol{\vartheta}_1, ..., \boldsymbol{\vartheta}_M, \alpha_1, ..., \alpha_{M-1}, \boldsymbol{\nu})$ where

$$\boldsymbol{\vartheta}_m = (\varphi_{m,0}, \text{vec}(A_{m,1}), ..., \text{vec}(A_{m,p}), \text{vech}(\Omega_m)), \quad m = 1, ..., M, \text{ and} \tag{35}$$

$$\boldsymbol{\nu} = (\nu_{M_1+1}, ..., \nu_M). \tag{36}$$

In the GMVAR model (when $M_1 = M$), the vector $\boldsymbol{\nu}$ is omitted, as the GMVAR model does not contain degrees of freedom parameters. For models imposing additional constraints on the paremeters, the parameter vectors are expressed in a different way. They are only presented in the package documentation for brevity, because the hand-specified parameter values can be set to satisfy any constraints as is.

In a structural GSMVAR model, the parameter vector has the form

$$\boldsymbol{\theta} = (\varphi_{1,0}, ..., \varphi_{M,0}, \text{vec}(\boldsymbol{A}_1), ..., \text{vec}(\boldsymbol{A}_M), \text{vec}(W), \boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_M, \alpha_1, ..., \alpha_{M-1}, \boldsymbol{\nu}), \tag{37}$$

$$\boldsymbol{A}_m = (\text{vec}(A_{m,1}), ..., \text{vec}(A_{m,p})) \tag{38}$$

$$\boldsymbol{\lambda}_m = (\lambda_{m1}, ..., \lambda_{md}). \tag{39}$$

For constrained structural models (including constraints on the structural parameters), see the documentation of `GSMVAR` (or any other relevant function).

In addition to the parameter vector, `GSMVAR` should be supplied with arguments `p` and `M` specifying the order of the model similarly to the estimation function `fitGSMVAR` discussed in Section 3.3. If one wishes to parametrize the model with the regimewise unconditional means (μ_m) instead of the intercepts ($\varphi_{m,0}$), the argument `parametrization` should be set to `"mean"` in which case the intercept parameters $\varphi_{m,0}$ are replaced with μ_m in the parameter vector. By default, **gmvarkit** uses intercept parametrization.

To exemplify, we build a reduced form StMVAR $p = 1$, $M = 1$, $d = 2$ model. The model has intercept parametrization and parameter values $\varphi_{1,0} = (0, 1)$ $\text{vec}(A_{1,1}) = (0.2, 0.2, 0.2, -0.2)$, $\text{vech}(\Omega_2) = (1, 0.1, 1)$, and $\nu_1 = 3$. After building the model, we use the `print` method to examine it:

```
R> params112 <- c(0, 1, 0.2, 0.2, 0.2, -0.2, 1, 0.1, 1, 3)
R> mod112 <- GSMVAR(p=1, M=1, d=2, params=params112, model="StMVAR")
R> mod112

Reduced form StMVAR model:
 p = 1, M = 1, d = 2, #parameters = 10,
 conditional log-likelihood, intercept parametrization, no AR parameter
 constraints

Regime 1
Mixing weight: 1.00
Regime means: 0.22, 0.87
Df parameter:  3.00

   Y     phi0          A1                                Omega          1/2
1 y1 = [ 0.00 ] + [  0.20  0.20 ] y1.1 + (         [  1.00 0.10 ] )     eps1
2 y2   [ 1.00 ]   [  0.20 -0.20 ] y2.1   ( ARCH_mt [  0.10 1.00 ] )     eps2
```

It is possible to include data in the models built with `GSMVAR` by either providing the data in the argument `data` when creating the model or by adding the data afterwards with the function `add_data`. When the model is supplied with data, the mixing weights, one-step conditional means and variances, and quantile residuals can be calculated and included in the model. The function `add_data` can also be used to update data to an estimated GSMVAR model without re-estimating the model.

7. Simulation and forecasting

7.1. Simulation

gmvarkit implements the S3 method `simulate` for simulating observations from GSMVAR processes (see `?simulate.gsmvar`). The method requires the process to be given as a class `gsmvar` object, which are typically created either by estimating a model with the function `fitGSMVAR` or by specifying the parameter values by hand and building the model with the constructor function `GSMVAR`. The initial values required to simulate the first p observations can be either set by hand (with the argument `init_values`) or drawn from the stationary

distribution of the process (by default) or from a (mixture of) stationary distribution(s) of given regime(s). The argument `nsim` sets the length of the sample path to be simulated.

To give an example, the following code sets the random number generator seed to one and simulates 500 observations long sample from the StMVAR process built in Section 6:

```
R> mysim <- simulate(mod112, nsim=500, seed=1)
```

Our implementation of `simulate` returns a list containing the simulated sample path in `$sample`, the mixture component that generated each observation in `$component`, and the mixing weights in `$mixing_weights`.

7.2. Simulation based forecasting

Deriving multiple-steps-ahead point predictions and prediction intervals analytically for the GSMVAR models is very complicated, so **gmvarkit** employs the following simulation-based method. By using the last p observations of the data up to the date of forecasting as initial values, a large number of sample paths for the future values of the process are simulated. Then, sample quantiles from the simulated sample paths are calculated to obtain prediction intervals, and the median or mean is used for point predictions. A similar procedure is also applied to forecast future values of the mixing weights, which might be of interest because the researcher can often associate specific characteristics to different regimes.

Forecasting is most conveniently done with the `predict` method (see `?predict.gsmvar`). The available arguments include the number of steps ahead to be predicted (`n_ahead`), the number sample paths the forecast is based on (`nsim`), possibly multiple confidence levels for prediction intervals (`pi`), prediction type (`pred_type`), and prediction interval type (`pi_type`). The prediction type can be either `median`, `mean`, or for one-step-ahead forecasts also the exact conditional mean, `cond_mean`. The prediction interval type can be any of `"two-sided"`, `"upper"`, `"lower"`, or `"none"`.

To exemplify, the following code forecasts the two-dimensional time-series of U.S. GDP and GDP deflator growth using the G-StMVAR$(1, 1, 1)$ model `fit12gs` estimated in Section 3.3. The forecast is 10 steps (quarters in this case) ahead, based on 10000 Monte Carlo repetitions with the point forecast based on the mean of those repetitions. The prediction intervals are two-sided with confidence levels 0.95 and 0.90. Finally, the argument `mix_weights` states that also future values of the the mixing weights should be forecasted. After completing the forecast, the function plots the results by default.

```
R> mypred <- predict(fit12gs, n_ahead=10, nsim=10000, pred_type="mean",
+                    pi_type="two-sided", pi=c(0.95, 0.90),
+                    mix_weights=TRUE)
```

The resulting plot is presented in Figure **??**.

8. Summary

Mixture vector autoregressive models are a valuable tool in modeling multivariate time series in which the data generating dynamics vary in time. We described the R package **gmvarkit**,

Related to	Name	Description
Estimation	`fitGSMVAR`	Estimate a GSMVAR model.
	`alt_gsmvar`	Build a GSMVAR model based on results from any estimation round.
	`stmvar_to_gstmvar`	Estimate a G-StMVAR model based on a StMVAR (or G-StMVAR) model with large degrees of freedom parameters.
	`gsmvar_to_sgsmvar`	Obtain a structural GSMVAR model identified by heteroskedasticity based on a reduced form model.
	`iterate_more`	Run more iterations of the variable metric algorithm for a preliminary estimated GSMVAR model.
Estimates	`summary` (method)	Detailed printout of the estimates.
	`plot` (method)	Plot the series with the estimated mixing weights and a kernel density estimates of the (marginal) series with the (marginal) stationary densities of the model.
	`get_foc`	Calculate numerically approximated gradient of the log-likelihood function evaluated at the estimate.
	`get_soc`	Calculate eigenvalues of numerically approximated Hessian of the log-likelihood function evaluated at the estimate.
	`profile_logliks`	Plot the graphs of the profile log-likelihood functions.
	`cond_moment_plot`	Plot the model implied one-step conditional means or variances.
Diagnostics	`quantile_residual_tests`	Calculate quantile residual tests.
	`diagnostic_plot`	Plot quantile residual diagnostics.
Forecasting	`predict` (method)	Forecast future observations and mixing weights of the process.
Simulation	`simulate` (method)	Simulate from a GSMVAR process.
Create model	`GSMVAR`	Construct a GSMVAR model based on specific parameter values.
Hypothesis testing	`LR_test`	Calculate likelihood ratio test.
	`Wald_test`	Calculate Wald test.
	`Rao_test`	Calculate Rao's test.
Impulse response analysis	`GIRF`	Estimate generalized impulse response functions.
	`GFEVD`	Estimate generalized forecast error variance decomposition.
	`linear_IRF`	Estimate linear impulse response functions.
Other	`add_data`	Add data to a GSMVAR model
	`swap_parametrization`	Swap between mean and intercept parametrizations

Table 1: Some useful functions in **gmvarkit** sorted according to their usage. The note "method" in parentheses after the name of a function signifies that it is an S3 method for a class `gsmvar` object (often generated by the function `fitGSMVAR` or `GSMVAR`).

which accommodates the GMVAR model (Kalliovirta *et al.* 2016), the StMVAR model (Virolainen 2021), and the G-StMVAR model (Virolainen 2021) - an appealing family of mixture vector autoregressive models that call the GSMVAR models. We discussed several features provided by **gmvarkit** for GSMVAR modeling: unconstrained and constrained maximum likelihood estimation of the model parameters, hypothesis testing, quantile residual based model diagnostics, estimation of generalized impulse response function and generalized forecast error variance decomposition, simulation, forecasting, and more. For convenience, we have collected some useful functions in **gmvarkit** to Table 1. For all the exported functions and their usage, see the reference manual.

Computational details

The results in this paper were obtained using R 4.1.2 and **uGMAR** 3.4.1 package running on MacBook Pro 14", 2021, with Apple M1 Pro processor, 16 Gt of unified RAM, and macOS Monterey 12.1 operating system.

Some of the estimation results (and thereby everything that is calculated based on the estimates) may vary slightly when running the code on different computers. This is likely due to the numerical error caused by the limited precision of the float point representation.

References

Anderson H, Vahid F (1998). "Testing multiple equation systems for common nonlinear components." *Journal of Econometrics*, **84**(1), 1–36. doi:10.1016/S0304-4076(97)00076-6.

Aomoto K, Kita M (2011). *Theory of Hypergeometric Functions*. 1st edition. Springer-Verlag, Tokyo. doi:10.1007/978-4-431-53938-4.

Blondel V, Nesterov Y (2005). "Computationally Efficient Approximations of the Joint Spectral Radius." *SIAM Journal on Matrix Analysis and Applications*, **27**(1), 256–272. doi:10.1137/040607009.

Buse A (1982). "The Likelihood Ratio, Wald, and Lagrange Multiplier Tests: An Expository Note." *The American Statistician*, **36**(3a), 153–157. doi:10.1080/00031305.1982.10482817.

Chang CT, Blondel V (2013). "An experimental study of approximation algorithms for the joint spectral radius." *Numerical Algorithms*, **64**, 181–202. doi:10.1007/s11075-012-9661-z.

Ding P (2016). "On the Conditional Distribution of the Multivariate t Distribution." *The American Statistician*, **70**(3), 293–295. doi:10.1080/00031305.2016.1164756.

Dorsey R, Mayer W (1995). "Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features." *Journal of Business and Economic Statistics*, **13**(1), 53–66. doi:10.1080/07350015.1995.10524579.

Hankin R, Clausen A, Murdoch D (2006). "Special functions in R: introducing the gsl package." *R News*, **6**(4).

Herwartz H, Lütkepohl H (2014). "Structural vector autoregressions with Markov switching: Combining conventional with statistical identification of shocks." *Journal of Econometrics*, **183**(1), 104–116. doi:doi.org/10.1016/j.jeconom.2014.06.012.

Hubrich K, Teräsvirta T (2013). "Thresholds and Smooth Transitions in Vector Autoregressive Models." *CREATES Research Paper 2013-18, Aarhus University*. URL https://pure.au.dk/ws/files/54473123/rp13_18.pdf.

Jungers R (2023). *The JSR toolbox (https://www.mathworks.com/matlabcentral/fileexchange/33202-the-jsr-toolbox), MATLAB Central File Exchange*.

Kalliovirta L (2012). "Misspecification tests based on quantile residuals." *The Econometrics Journal*, **15**(2), 358–393. doi:10.1111/j.1368-423X.2011.00364.x.

Kalliovirta L, Meitz M, Saikkonen P (2016). "Gaussian mixture vector autoregression." *Journal of Econometrics*, **192**(2), 465–498. doi:10.1016/j.jeconom.2016.02.012.

Kalliovirta L, Saikkonen P (2010). "Reliable Residuals for Multivariate Nonlinear Time Series Models." *Unpublished revision of HECER discussion paper No. 247*. URL https://blogs.helsinki.fi/lkvaisan/files/2010/08/ReliableResiduals.pdf.

Kheifets I, Saikkonen P (2020). "Stationarity and ergodicity of vector STAR models." *Econometric Reviews*, **39**(407–414), 1311–1324. doi:10.1080/07474938.2019.1651489.

Kilian L, Lütkepohl H (2017). *Structural Vector Autoregressive Analysis*. 1st edition. Cambridge University Press, Cambridge. doi:10.1017/9781108164818.

Koop G, Pesaran M, Potter S (1996). "Impulse response analysis in nonlinear multivariate models." *Journal of Econometrics*, **74**(1), 119–147. doi:10.1016/0304-4076(95)01753-4.

Lanne M, Lütkepohl H, Maciejowska K (2010). "Structural vector autoregressions with Markov switching." *Journal of Economic Dynamics and Control*, **34**(2), 121–131. doi:10.1016/j.jedc.2009.08.002.

Lanne M, Nyberg H (2016). "Generalized Forecast Error Variance Decomposition for Linear and Nonlinear Multivariate Models." *Oxford Bulletin of Economics and Statistics*, **78**(4), 595–603. doi:10.1111/obes.12125.

Meitz M, Saikkonen P (2021). "Testing for observation-dependent regime switching in mixture autoregressive models." *Journal of Econometrics*, **222**(1), 601–624. doi:10.1016/j.jeconom.2020.04.048.

Nash J (1990). *Compact Numerical Methods for Computers. Linear Algebra and Function Minimization*. 2nd edition. Adam Hilger, Bristol and New York. doi:10.1201/9781315139784.

Parrilo P, Jadbabaie A (2008). "Approximation of the joint spectral radius using sum of squares." *Linear Algebra and its Applications*, **428**(10), 2385–2402. doi:10.1016/j.laa.2007.12.027.

Patnaik L, Srinivas M (1994). "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms." *Transactions on Systems, Man and Cybernetics*, **24**(4), 656–667. doi:10.1109/21.286385.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Redner R, Walker H (1984). "Mixture Densities, Maximum Likelihood and the Em Algorithm." *Society for Industrial and Applied Mathematics*, **26**(2), 195–239. doi:10.1137/1026034.

Smith R, Dike B, Stegmann S (1995). "Fitness inheritance in genetic algorithms." *Proceedings of the 1995 ACM symbosium on Applied Computing*, pp. 345–350. doi:10.1145/315891.316014.

Solymos P, Zawadzki Z (2020). ***pbapply**: Adding Progress Bar to '*apply' Functions*. R package version 1.4-3, URL https://CRAN.R-project.org/package=pbapply.

Tsay R (1998). "Testing and Modeling Multivariate Threshold Models." *Journal of the American Statistical Association*, **93**(443), 1188–1202. doi:10.1080/01621459.1998.10473779.

Virolainen S (2018). *uGMAR: Estimate Univariate Gaussian and Student's t Mixture Autoregressive Models*. R package version 3.4.2 available at CRAN: https://CRAN.R-project.org/package=uGMAR, URL https://CRAN.R-project.org/package=uGMAR.

Virolainen S (2021). "Structural Gaussian Mixture vector autoregressive model with application to the asymmetric effects of monetary policy shocks." *Unpublished working paper, available as arXiv:2007.04713*. URL https://arxiv.org/abs/2007.04713.

Virolainen S (2024). *sstvars: Toolkit for Structural Smooth Transition Vector Autoregressive models*. R package version 0.1.0 available at GitHub: https://github.com/saviviro/sstvars, URL https://github.com/saviviro/sstvars.

A. Properties of multivariate Gaussian and Student's t distribution

Denote a d-dimensional real valued vector by y. It is well known that the density function of a d-dimensional Gaussian distribution with mean μ and covariance matrix Σ is

$$n_d(y;\mu,\Sigma) = (2\pi)^{-d/2}\det(\Sigma)^{-1/2}\exp\left\{-\frac{1}{2}(y-\mu)'\Sigma^{-1}(y-\mu)\right\}. \tag{40}$$

Similarly to **?** but differing from the standard form, we parametrize the Student's t-distribution using its covariance matrix as a parameter together with the mean and the degrees of freedom. The density function of such a d-dimensional t-distribution with mean μ, covariance matrix Σ, and $\nu > 2$ degrees of freedom is

$$t_d(y;\mu,\Sigma,\nu) = C_d(\nu)\det(\Sigma)^{-1/2}\left(1+\frac{(y-\mu)'\Sigma^{-1}(y-\mu)}{\nu-2}\right)^{-(d+\nu)/2}, \tag{41}$$

where

$$C_d(\nu) = \frac{\Gamma\left(\frac{d+\nu}{2}\right)}{\sqrt{\pi^d(\nu-2)^d}\Gamma\left(\frac{\nu}{2}\right)}, \tag{42}$$

and $\Gamma(\cdot)$ is the gamma function. We assume that the covariance matrix Σ is positive definite for both distributions.

Consider a partition $X = (X_1, X_2)$ of either Gaussian or t-distributed (with ν degrees of freedom) random vector X such that X_1 has dimension $(d_1 \times 1)$ and X_2 has dimension $(d_2 \times 1)$. Consider also a corresponding partition of the mean vector $\mu = (\mu_1, \mu_2)$ and the covariance matrix

$$\Sigma = \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{12}' & \Sigma_{22}\end{bmatrix}, \tag{43}$$

where, for example, the dimension of Σ_{11} is $(d_1 \times d_1)$. In the Gaussian case, X_1 then has the marginal distribution $n_{d_1}(\mu_1, \Sigma_{11})$ and X_2 has the marginal distribution $n_{d_2}(\mu_2, \Sigma_{22})$. In the Student's t case, X_1 has the marginal distribution $t_{d_1}(\mu_1, \Sigma_{11}, \nu)$ and X_2 has the marginal distribution $t_{d_2}(\mu_2, \Sigma_{22}, \nu)$ (see, e.g., Ding (2016), also in what follows).

When X has Gaussian distribution, the conditional distribution of the random vector X_1 given $X_2 = x_2$ is

$$X_1 \mid (X_2 = x_2) \sim n_{d_1}(\mu_{1|2}(x_2), \Sigma_{1|2}(x_2)), \tag{44}$$

where

$$\mu(x_2) \equiv \mu_{1|2}(x_2) = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2) \quad \text{and} \tag{45}$$

$$\Omega \equiv \Sigma_{1|2}(x_2) = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}'. \tag{46}$$

When X has t-distribution, the conditional distribution of the random vector X_1 given $X_2 = x_2$ is

$$X_1 \mid (X_2 = x_2) \sim t_{d_1}(\mu_{1|2}(x_2), \Sigma_{1|2}(x_2), \nu + d_2), \tag{47}$$

where

$$\mu(x_2) = \mu_{1|2}(x_2) = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2-\mu_2) \quad \text{and} \tag{48}$$

$$\Omega(x_2) \equiv \Sigma_{1|2}(x_2) = \frac{\nu-2+(x_2-\mu_2)'\Sigma_{22}^{-1}(x_2-\mu_2)}{\nu-2+d_2}(\Sigma_{11}-\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}'). \tag{49}$$

In particular, we have

$$n_d(x;\mu,\Sigma) = n_{d_1}(x_1;\mu_{1|2}(x_2),\Sigma_{1|2}(x_2))t_{d_2}(x_2;\mu_2,\Sigma_{22}) \quad \text{and} \tag{50}$$

$$t_d(x;\mu,\Sigma,\nu) = t_{d_1}(x_1;\mu_{1|2}(x_2),\Sigma_{1|2}(x_2),\nu+d_2)t_{d_2}(x_2;\mu_2,\Sigma_{22},\nu). \tag{51}$$

B. Quantile residuals of the G-StMVAR model

The conditional density function of the d-dimensional G-StMVAR process y_t conditional on $\mathcal{F}_{t-1}$ is

$$f_{t-1}(y_t;\boldsymbol{\theta}) = \sum_{m=1}^{M_1}\alpha_{m,t}n_d(y_t;\mu_{m,t},\Omega_m) + \sum_{m=M_1+1}^{M}\alpha_{m,t}t_d(y_t;\mu_{m,t},\Omega_{m,t},\nu_m+dp), \tag{52}$$

where $n_d(\cdot;\mu_{m,t},\Omega_m,\nu_m+dp)$ is the density function of d-dimensional normal distribution with mean $\mu_{m,t}$ and covariance matrix Ω_m; and $t_d(\cdot;\mu_{m,t},\Omega_m,\nu_m+dp)$ is the density function of d-dimensional t-distribution with mean $\mu_{m,t}$, covariance matrix $\Omega_{m,t}$, and ν_m+dp degrees of freedom.

Denote $y_t^{(k)} = (y_{1,t},...,y_{k,t})$ $(k\times 1)$, $k\leq d$, $\mu_{m,t}^{(k)} = (\mu_{1,m,,t},...,\mu_{k,m,t})$ $(k\times 1)$, $k\leq d$, and by $\Omega_{m,t}^{(k)}$ $(\Omega_m^{(k)})$ the upper left $(k\times k)$ block matrix of $\Omega_{m,t}$ (Ω_m). Then, the properties of the marginal distributions of multivariate Gaussian and t-distributions (see Appendix A) show that conditional on $\mathcal{F}_{t-1}$, the random vectors $y_t^{(j)}$, $j=1,..,d$, follow the distribution that is a mixture M_1 j-dimensional normal distributions (with means $\mu_{m,t}^{(j)}$ and covariance matrices $\Omega_m^{(j)}$) and $M_2\equiv M-M_1$ j-dimensional t-distributions (with means $\mu_{m,t}^{(j)}$, covariance matrices $\Omega_{m,t}^{(j)}$, and ν_m+dp degrees of freedom). The mixing weights $\alpha_{m,t}$ are not affected, as they are $\mathcal{F}_{t-1}$-measurable. Therefore, the marginal density function of $y_t^{(j)}$ is

$$f_{t-1}(y_t^{(j)};\boldsymbol{\theta}) = \sum_{m=1}^{M_1}\alpha_{m,t}n_j(y_t^{(j)};\mu_{m,t}^{(j)},\Omega_m^{(j)}) + \sum_{m=M_1+1}^{M}\alpha_{m,t}t_j(y_t^{(j)};\mu_{m,t}^{(j)},\Omega_{m,t}^{(j)},\nu_m+dp), \tag{53}$$

The conditional density function $f_{0,t-1}(y_{1,t};\boldsymbol{\theta})$ in (30) is obtained from (53) by choosing $j=1$. For $j=2,...,d$, the conditional density functions $f_{j-1,t-1}(y_{j,t};\boldsymbol{\theta})$ are obtained by substituting the equation (53) to the formula of conditional density function:

$$f_{j-1,t-1}\left(y_{j,t};\boldsymbol{\theta}\right) = \frac{f_{t-1}(y_t^{(j)};\boldsymbol{\theta})}{f_{t-1}(y_t^{(j-1)};\boldsymbol{\theta})} = \frac{\sum_{m=1}^{M_1}\alpha_{m,t}n_j(y_t^{(j)};\mu_{m,t}^{(j)},\Omega_m^{(j)}) + \sum_{m=M_1+1}^{M}\alpha_{m,t}t_j(y_t^{(j)};\mu_{m,t}^{(j)},\Omega_{m,t}^{(j)},\nu_m+dp)}{\sum_{n=1}^{M_1}\alpha_{n,t}n_{j-1}(y_t^{(j-1)};\mu_{n,t}^{(j-1)},\Omega_n^{(j-1)}) + \sum_{n=M_1+1}^{M}\alpha_{n,t}t_{j-1}(y_t^{(j-1)};\mu_{n,t}^{(j-1)},\Omega_n^{(j-1)},\nu_m+dp)}. \tag{54}$$

It follows from the properties of the conditional distributions of multivariate normal distribution that we may express the j-dimensional normal distributions as

$$n_j(y_t^{(j)};\mu_{m,t}^{(j)},\Omega_m^{(j)}) = n_1(y_{j,t};\mu_{m,t,j|j-1},\Omega_{m,j|j-1})n_{j-1}(y_t^{(j-1)};\mu_{m,t}^{(j-1)},\Omega_m^{(j-1)}), \tag{55}$$

where $\mu_{m,t,j|j-1}$ and $\Omega_{m,j|j-1}$ are the conditional mean and covariance matrix of $y_{j,t}$ conditional on $\sigma(\mathcal{A}_{j-1}, \mathcal{F}_{t-1})$. Likewise, it follows from the properties of the conditional distributions of multivariate t-distribution that we may express the j-dimensional t-distributions as

$$\begin{aligned} t_j(y_t^{(j)}; \mu_{m,t}^{(j)}, \Omega_{m,t}^{(j)}, \nu_m + dp) =& t_1(y_{j,t}; \mu_{m,t,j|j-1}, \Omega_{m,t,j|j-1}, \nu_m + dp + j - 1) \\ & \times t_{j-1}(y_t^{(j-1)}; \mu_{m,t}^{(j-1)}, \Omega_{m,t}^{(j-1)}, \nu_m + dp), \end{aligned} \tag{56}$$

where $\mu_{m,t,j|j-1}$ and $\Omega_{m,t,j|j-1}$ are the conditional mean and covariance matrix of $y_{j,t}$ conditional on $\sigma(\mathcal{A}_{j-1}, \mathcal{F}_{t-1})$.

By denoting

$$\beta_{m,t,j} \equiv \frac{\alpha_{m,t} n_{j-1}(y_t^{(j-1)}; \mu_{m,t}^{(j-1)}, \Omega_m^{(j-1)})}{\sum_{n=1}^{M_1} \alpha_{n,t} n_{j-1}(y_t^{(j-1)}; \mu_{n,t}^{(j-1)}, \Omega_n^{(j-1)}) + \sum_{n=M_1+1}^{M} \alpha_{n,t} t_{j-1}(y_t^{(j-1)}; \mu_{n,t}^{(j-1)}, \Omega_{n,t}^{(j-1)}, \nu_n + dp)} \tag{57}$$

for $m = 1, .., M_1$, $j = 2, ..., d$, and

$$\beta_{m,t,j} \equiv \frac{\alpha_{m,t} t_{j-1}(y_t^{(j-1)}; \mu_{m,t}^{(j-1)}, \Omega_{m,t}^{(j-1)}, \nu_m + dp)}{\sum_{n=1}^{M_1} \alpha_{n,t} n_{j-1}(y_t^{(j-1)}; \mu_{n,t}^{(j-1)}, \Omega_n^{(j-1)}) + \sum_{n=M_1+1}^{M} \alpha_{n,t} t_{j-1}(y_t^{(j-1)}; \mu_{n,t}^{(j-1)}, \Omega_{n,t}^{(j-1)}, \nu_n + dp)} \tag{58}$$

for $m = M_1 + 1, ..., M$, $j = 2, ..., d$, and using the expressions (55) and (56), we can express the conditional density function (54) as

$$\begin{aligned} f_{j-1,t-1}(y_{j,t}; \boldsymbol{\theta}) = & \sum_{m=1}^{M_1} \beta_{m,t,j} n_1(y_{j,t}; \mu_{m,t,j|j-1}, \Omega_{m,j|j-1}) \\ & + \sum_{m=M_1+1}^{M} \beta_{m,t,j} t_1(y_{j,t}; \mu_{m,t,j|j-1}, \Omega_{m,t,j|j-1}, \nu_m + dp + j - 1), \quad j = 2, .., d. \end{aligned} \tag{59}$$

For $m = 1, ..., M_1$, the conditional means $\mu_{m,t,j|j-1}$ and covariance matrices $\Omega_{m,j|j-1}$ are as in (45) and (46) when for each $j = 2, ..., d$ and $m = 1, ..., M$ we consider the partition $y_t^{(j)} = (y_t^{(j-1)}, y_{j,t})$, $\mu_{m,t}^{(j)} = (\mu_{m,t}^{(j-1)}, \mu_{j,m,t})$, and

$$\Omega_m^{(j)} = \begin{bmatrix} \Omega_m^{(j-1)} & \Omega_{(j-1),j,m} \\ \Omega'_{(j-1),j,m} & \Omega_m(j,j) \end{bmatrix}, \tag{60}$$

where $\Omega_m(j,j)$ is the jjth elementh of Ω_m and $\Omega_{(j-1),j,m}$ ($(j-1) \times 1$) consists of the rows $1, ..., j-1$ of the jth column of Ω_m. In particular, we have

$$\mu_{m,t,j|j-1} = \mu_{j,m,t} + \Omega'_{(j-1),j,m} (\Omega_m^{(j-1)})^{-1} (y_t^{(j-1)} - \mu_{m,t}^{(j-1)}), \tag{61}$$

$$\Omega_{m,j|j-1} = \Omega_m(j,j) - \Omega'_{(j-1),j,m} (\Omega_m^{(j-1)})^{-1} \Omega_{(j-1),j,m}. \tag{62}$$

For $m = M_1 + 1, .., M$, the conditional means $\mu_{m,t,j|j-1}$ and covariance matrices $\Omega_{m,t,j|j-1}$ are as in (48) and (49) when for each $j = 2, ..., d$ and $m = 1, ..., M$ we consider the partition $y_t^{(j)} = (y_t^{(j-1)}, y_{j,t})$, $\mu_{m,t}^{(j)} = (\mu_{m,t}^{(j-1)}, \mu_{j,m,t})$, and

$$\Omega_{m,t}^{(j)} = \begin{bmatrix} \Omega_{m,t}^{(j-1)} & \Omega_{(j-1),j,m,t} \\ \Omega'_{(j-1),j,m,t} & \Omega_{m,t}(j,j) \end{bmatrix}, \tag{63}$$

where $\Omega_{m,t}(j,j)$ is the jjth elementh of $\Omega_{m,t}$ and $\Omega_{(j-1),j,m,t}$ $((j-1)\times 1)$ consists of the rows $1,...,j-1$ of the jth column of $\Omega_{m,t}$. In particular, taking use of the relation $\Omega_{m,t}=\omega_{m,t}\Omega_m$ (where $\omega_{m,t}$ is scalar), we have

$$\begin{aligned}\mu_{m,t,j|j-1} &= \mu_{j,m,t}+\Omega'_{(j-1),j,m,t}(\Omega_{m,t}^{(j-1)})^{-1}(y_t^{(j-1)}-\mu_{m,t}^{(j-1)})\\ &= \mu_{j,m,t}+\Omega'_{(j-1),j,m}(\Omega_{m}^{(j-1)})^{-1}(y_t^{(j-1)}-\mu_{m,t}^{(j-1)}),\end{aligned} \tag{64}$$

and

$$\begin{aligned}\Omega_{m,t,j|j-1} &= \frac{\nu_m+dp+(y_t^{(j-1)}-\mu_{m,t}^{(j-1)})'(\Omega_{m,t}^{(j-1)})^{-1}(y_t^{(j-1)}-\mu_{m,t}^{(j-1)})}{\nu_m+dp+j-3}\tilde{\Omega}_{m,t,j|j-1}\\ &= \frac{\nu_m+dp+\omega_{m,t}^{-1}(y_t^{(j-1)}-\mu_{m,t}^{(j-1)})'(\Omega_{m}^{(j-1)})^{-1}(y_t^{(j-1)}-\mu_{m,t}^{(j-1)})}{\nu_m+dp+j-3}\tilde{\Omega}_{m,t,j|j-1},\end{aligned} \tag{65}$$

where

$$\begin{aligned}\tilde{\Omega}_{m,t,j|j-1} &\equiv \Omega_{m,t}(j,j)-\Omega'_{(j-1),j,m,t}(\Omega_{m,t}^{(j-1)})^{-1}\Omega_{(j-1),j,m,t}\\ &= \omega_{m,t}(\Omega_m(j,j)-\Omega'_{(j-1),j,m}(\Omega_m^{(j-1)})^{-1}\Omega_{(j-1),j,m}).\end{aligned} \tag{66}$$

It then remains to find expressions for the conditional distribution functions $F_{j-1,t-1}(y_{j,t};\boldsymbol{\theta})$, $j=1,...,d$, in (32). For notational convenience, we write

$$\begin{aligned}f_{j-1,t-1}(y_{j,t}\boldsymbol{\theta}) = &\sum_{m=1}^{M_1}\beta_{m,t,j}n_1(y_{j,t};\mu_{m,t,j|j-1},\Omega_{m,j|j-1})\\ &+\sum_{m=M_1+1}^{M}\beta_{m,t,j}t_1(y_{j,t};\mu_{m,t,j|j-1},\Omega_{m,t,j|j-1},\nu_m+dp+j-1)\end{aligned} \tag{67}$$

for all $j=1,...,d$ by defining $\beta_{m,t,1}\equiv\alpha_{m,t}$, $\mu_{m,t,1|0}\equiv\mu_{m,t}^{(1)}$, $\Omega_{m,1|0}\equiv\Omega_m^{(1)}$, and $\Omega_{m,t,1|0}\equiv\Omega_{m,t}^{(1)}$. For $j=2,...,d$, these quantities are defined in (58), (61), (62), (64), and (65). Then,

$$\begin{aligned}F_{j-1,t-1}(y_{j,t};\boldsymbol{\theta}) = &\sum_{m=1}^{M_1}\beta_{m,t,j}\int_{-\infty}^{y_{j,t}}n_1(u;\mu_{m,t,j|j-1},\Omega_{m,j|j-1})du\\ &+\sum_{m=M_1+1}^{M}\beta_{m,t,j}\int_{-\infty}^{y_{j,t}}t_1(u;\mu_{m,t,j|j-1},\Omega_{m,t,j|j-1},\nu_m+dp+j-1)du,\end{aligned} \tag{68}$$

where we seek to solve the integrals inside the sums.

Regarding the first sum, for $m=1,...,M_1$, it is easy to see that the integrals can be expressed using the stardard normal distribution function $\Phi(\cdot)$ as

$$\int_{-\infty}^{y_{j,t}}n_1(u;\mu_{m,t,j|j-1},\Omega_{m,j|j-1})du=\Phi\left(\frac{y_{j,t}-\mu_{m,t,j|j-1}}{\sqrt{\Omega_{m,j|j-1}}}\right). \tag{69}$$

Next, consider the second sum, $m=M_1+1,...,M$. By taking use of the symmetry of the t-distribution about its mean, we obtain

$$\begin{aligned}&\int_{-\infty}^{y_{j,t}}t_1(u;\mu_{m,t,j|j-1},\Omega_{m,t,j|j-1},\nu_m+dp+j-1)du\\ &=\frac{1}{2}+\int_{\mu_{m,t,j|j-1}}^{y_{j,t}}t_1(u;\mu_{m,t,j|j-1},\Omega_{m,t,j|j-1},\nu_m+dp+j-1)du.\end{aligned} \tag{70}$$

By applying the change of variables $\tilde{u}_{m,t,j} = u - \mu_{m,t,j|j-1}$ in the integral, the RHS of (70) can be expressed as

$$\frac{1}{2} + \frac{\Gamma\left(\frac{\nu_m+dp+j}{2}\right)}{\sqrt{\pi(\nu_m+dp+j-3)}\Gamma\left(\frac{\nu_m+dp+j-1}{2}\right)}\Omega_{m,t,j|j-1}^{-1/2}\int_0^{\tilde{y}_{m,t,j}}\left(1+\frac{\tilde{u}_{m,t,j}^2}{a_{m,t,j}}\right)^{-b_{m,j}} d\tilde{u}_{m,t,j}, \quad (71)$$

where $\tilde{y}_{m,t,j} \equiv y_{j,t} - \mu_{m,t,j|j-1}$, $a_{m,t,j} \equiv (\nu_m + dp + j - 3)\Omega_{m,t,j|j-1}$, and $b_{m,j} \equiv (\nu_m + dp + j)/2$. Then, by applying the change of variables $z_{m,t,j} = \tilde{u}_{m,t,j}^2/\tilde{y}_{m,t,j}$, we can express the integral in (71) as

$$\int_0^{\tilde{y}_{m,t,j}}\left(1+\frac{\tilde{u}_{m,t,j}^2}{a_{m,t,j}}\right)^{-b_{m,j}} d\tilde{u}_{m,t,j} = \frac{1}{2}\int_0^{\tilde{y}_{m,t,j}}\left(\frac{\tilde{y}_{m,t,j}}{z_{m,t,j}}\right)^{1/2}\left(1+\frac{z_{m,t,j}\tilde{y}_{m,t,j}}{a_{m,t,j}}\right)^{-b_{m,j}} dz_{m,t,j}. \quad (72)$$

By applying the third change of variables $x_{m,t,j} = z_{m,t,j}/\tilde{y}_{m,t,j}$ and using the properties of the gamma function, the RHS of (72) can be expressed as

$$\frac{\tilde{y}_{m,t,j}}{2}\int_0^1 x_{m,t,j}^{-1/2}\left(1-x_{m,t,j}\left(-\frac{\tilde{y}_{m,t,j}^2}{a_{m,t,j}}\right)\right)^{-b_{m,j}} dx_{m,t,j} = \tilde{y}_{m,t,j} \times {}_2\mathrm{F}_1\left(\frac{1}{2}, b_{m,j}, \frac{3}{2}; -\frac{\tilde{y}_{m,t,j}^2}{a_{m,t,j}}\right), \quad (73)$$

where the hypergeometric function is defined as (Aomoto and Kita 2011, Section 1.3.1)

$${}_2F_1(a,b,c;x) = \frac{\Gamma(c)}{\Gamma(a)\Gamma(c-a)}\int_0^1 s^{a-1}(1-s)^{c-a-1}(1-sx)^{-b}ds, \quad (74)$$

when $|x| < 1$, $a > 0$, and $c - a > 0$ (when $a, c \in \mathbb{R}$).
Using the above result, we have

$$\begin{aligned}&\int_{-\infty}^{y_{j,t}} t_1(u; \mu_{m,t,j|j-1}, \Omega_{m,t,j|j-1}, \nu_m + dp + j - 1)du \\ &= \frac{1}{2} + \frac{\Gamma\left(\frac{\nu_m+dp+j}{2}\right)}{\sqrt{\pi(\nu_m+dp+j-3)}\Gamma\left(\frac{\nu_m+dp+j-1}{2}\right)}\Omega_{m,t,j|j-1}^{-1/2}\tilde{y}_{m,t,j} \times {}_2\mathrm{F}_1\left(\frac{1}{2}, b_{m,j}, \frac{3}{2}; -\frac{\tilde{y}_{m,t,j}^2}{a_{m,t,j}}\right)\end{aligned} \quad (75)$$

whenever $\left|-\frac{\tilde{y}_{m,t,j}^2}{a_{m,t,j}}\right| < 1$. That is, the closed form expression (75) exists when

$$|y_{j,t} - \mu_{m,t,j|j-1}| < \sqrt{(v_m + dp + j - 3)\Omega_{m,t,j,|j-1}}. \quad (76)$$

If this condition does not hold, the quantile residual are obtained by numerically integrating the conditional density function $t_1(u; \mu_{m,t,j|j-1}, \Omega_{m,t,j|j-1}, \nu_m + dp + j - 1)$. For the hypergeometric function, **gmvarkit** uses the package **gsl** (Hankin, Clausen, and Murdoch 2006).

C. Monte Carlo algorithm

We present a Monte Carlo algorithm that produces point estimates and with random initial value $\boldsymbol{y}_{t-1} = (y_{t-1}, ..., y_{t-p})$ confidence intervals for the generalized impulse response function

defined in (33). Our algorithm is adapted from Koop *et al.* (1996, pp. 135-136) and Kilian and Lütkepohl (2017, pp. 601-602). We assume that the history $\boldsymbol{y}_{t-1}$ follows a known distribution G, which may be such that it produces a single outcome with probability one (corresponding to a fixed $\boldsymbol{y}_{t-1}$), or it can be the stationary distribution of the process or of a specific regime. In the following, $y_{t+h}^{(i)}(\delta_j, \boldsymbol{y}_{t-1})$ denotes a realization of the process at time $t+h$ conditional on the structural shock of magnitude δ_j in the jth element of e_t hitting the system at time t and on the p observations $\boldsymbol{y}_{t-1} = (y_{t-1}, ..., y_{t-p})$ preceding the time t, whereas $y_{t+h}^{(i)}(\boldsymbol{y}_{t-1})$ denotes an alternative realization conditional on the history $\boldsymbol{y}_{t-1}$ only.

The algorithm proceeds with the following steps.

0. Decide the horizon H, the numbers of repetitions R_1 and R_2, and the magnitude δ_j for the jth structural shock that is of interest.

1. Draw an initial value $\boldsymbol{y}_{t-1}$ from G.

2. Draw $H+1$ independent realizations of a shock ε_t from $N(0, I_d)$. If the models contains Student's t regime, the Gaussian shocks are used obtain shocks from the appropriate Student's t distributions (in this case, $H+1$ independent realizations of shocks from $\chi^2_{\nu_m+dp}$-distributions are also drawn for the t-regimes). Also, draw an initial regime $m \in \{1, ..., M\}$ according to the probabilities given by the mixing weights $\alpha_{1,t}, ..., \alpha_{M,t}$ and compute the reduced form shock u_t. If identification by heteroskedasticity is used, $u_t = W\Lambda_m^{1/2}\varepsilon_t$, where $\Lambda_1 = I_d$. If the shocks are, instead, identified recursively, $u_t = L_t\varepsilon_t$, where L_t is the lower triangular matrix obtained from the Cholesky decomposition $\Omega_t = L_t L_t'$ and Ω_t is the conditional covariance matrix of the process. Then, compute the structural shock $e_t = B_t^{-1}u_t$ and impose the size δ_j on its jth element to obtain e_t^*. Finally, calculate the modified reduced form shock $u_t^* = B_t e_t^*$.

3. Use the modified reduced form shock u_t^* and the rest H standard normal shocks ε_t obtained from Step 2 to compute realizations $y_{t+h}^{(i)}(\delta_j, \boldsymbol{y}_{t-1})$ for $n = 0, 1, ..., H$, iterating forward so that in each iteration the regime m that generates the observation is first drawn according to the probabilities given by the mixing weights. At $h = 0$, the initial regime and the modified reduced form shock u_t^* calculated from the structural shock in Step 2 is used. From $h = 1$ onwards, the $h+1$th standard normal shock ε_t is used to calculate the reduced form shock $u_{t+h} = W\Lambda_m^{1/2}\varepsilon_{t+h}$, where $\Lambda_1 = I_d$ and m is the selected regime (note that only the reduced form shocks are of interest with $h > 0$, so the same formula can be used for recursively identified shocks here).

4. Use the reduced form shock u_t and the rest H the standard normal shocks ε_t obtained from Step 2 to compute realizations $y_{t+h}^{(i)}(\boldsymbol{y}_{t-1})$ for $h = 0, 1, ..., H$, so that the reduced form shock u_t (calculated in Step 2) is used to compute the time $h = 0$ realization. Otherwise proceed similarly to the previous step.

5. Calculate $y_{t+h}^{(i)}(\delta_j, \boldsymbol{y}_{t-1}) - y_{t+h}^{(i)}(\boldsymbol{y}_{t-1})$.

6. Repeat Steps 2-5 R_1 times and calculate the sample mean of $y_{t+h}^{(i)}(\delta_j, \boldsymbol{y}_{t-1}) - y_{t+n}^{(i)}(\boldsymbol{y}_{t-1})$ for $h = 0, 1, ..., H$ to obtain an estimate of the $\text{GIRF}(h, \delta_j, \boldsymbol{y}_{t-1})$.

7. Repeat Steps 1-6 R_2 times to obtain estimates of $\text{GIRF}(h, \delta_j, \boldsymbol{y}_{t-1})$ with different starting values $\boldsymbol{y}_{t-1}$ generated from the distribution G. Then, take the sample mean and

sample quantiles over the estimates to obtain point estimate and confidence intervals for the GIRF with random initial value.

Notice that if a fixed initial value $\boldsymbol{y}_{t-1}$ is used, Step 7 is redundant.

Affiliation:

Savi Virolainen
Faculty of Social Sciences
University of Helsinki
P. O. Box 17, FI-0014 University of Helsinki, Finland
E-mail: savi.virolainen@helsinki.fi