Notes on Bayesian Estimation of Unobserved Components Models

May 15, 2021

Contents

1 Read Me

The idea of this collection of classes is to allow for quick Bayesian estimation of unobserved components (UC) models. Taken together, these classes cover virtually all the 'usual' UC models in the current literature. More exotic ones can be incorporated easily based on the object-oriented design. The main result that drives all of these classes is given in the subsection below: following the general philosophy in Chan et al. (2019), potentially complex models are reduced into applications of conditional Bayesian OLS. Therefore, the conceptual part just requires identifying basic conditional linear regression patterns.

Some notes on usage:

- Unfortunately, draws are not stored within the individual blocks due to very severe performance issues when doing so
- Code is highly automised– almost no initial values have to be supplied since they are set equal to the expectation of the unconditional prior. This can sometimes lead to problems in early parts of the Gibbs sampler. The issue is usually avoided by simply running the code again (but be sure to check convergence diagnostics). If it persists it suggests that the priors are bad, and that you should revise them
- The philosophy is fundamentally object-oriented, but I do not use the Matlab template for two reasons
    1. Very little is gained by using the Matlab version compared to the object orientation based on structs achieved here (also in terms of awkwardness of passing an object to a method manually)
    2. Using structs leaves the door open to 'tweak' (abuse?) these objects based on the requirements of a particular estimation exercise.

To illustrate the ease with which models can be adapted in this modular framework, examples that replicate Chan et al. (2019) are provided (see the "examples" directory).

1.1 The Crucial Result

The 'workhorse' result that drives all the methods is a simple one on Bayesian regression. Suppose you have a relation

$$y = X\beta + \varepsilon, \varepsilon \sim \mathcal{N}[0, \Sigma],$$

where Σ is some arbitrary covariance matrix with some arbitrary prior, and β has a standard prior given by

$$\beta \sim \mathcal{N}[\mu_\beta, V_\beta].$$

Then you can still sample from the conditional posterior of β given by

$$\beta | y, \Sigma \sim \mathcal{N}[\hat{\beta}, K_\beta^{-1}],$$

where

$$\begin{aligned} K_\beta &= V_\beta^{-1} + X'\Sigma^{-1}X \\ \hat{\beta} &= K_\beta^{-1}(V_\beta^{-1}\mu_\beta + X'\Sigma^{-1}y). \end{aligned}$$

See Exercise 12.2 in “Bayesian Econometric Methods Second Edition” by Chan, Koop, Poirier, and Tobias for a proof of this (and remember that their ‘h’ is simply unity, and to substitute their expression ‘$\hat{\beta}(\Omega)'$’).

Finally, when dealing with the ‘H matrices’ (see below) it is helpful to remember that for invertible matrices A, B, C, it is the case that $(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$ (this property does not depend on any matrix being diagonal), and that $\left((A^{-1})'\right)^{-1} = A'$ since $(A^{-1})' = (A')^{-1}$ (i.e., you can flip the $'$ and $^{-1}$).

Note: since everything ultimately boils down to Bayesian OLS, you technically only need a single class, i.e. a Bayesian regression class. The classes constructed do in fact repeat themselves to a certain extent, which does impact runtime slightly. However, the time (and errors) saved in writing the code is a lot less.

2 SVBlock

The `SVBlock` class constructs objects that sample h_t, h_0, σ_h^2 from the simple model given by:

$$y_t = e^{\frac{1}{2}h_t} e_t, e_t \sim \mathcal{N}[0,1]$$
$$h_t = h_{t-1} + u_t, u_t \sim \mathcal{N}[0,\sigma_h^2],$$

for $t = 1, \ldots, T$. Priors have to take the following form:

$$h_0 \sim \mathcal{N}[a_0, b_0]$$
$$\sigma_h^2 \sim \mathcal{IG}[\nu_h, S_h].$$

It is possible to specify a prior in the non-centred paramterisation for σ_h, i.e.:

$$\sigma_h \sim \mathcal{N}[0, V_s].$$

If the analysis is carried out in the non-centred parameterisation, the following change of variable formula is used: $h_t = \sigma_h \tilde{h}_t + h_0$. Thus, model is changed to:

$$y_t = e^{\frac{1}{2}(\sigma_h \tilde{h}_t + h_0)} \tilde{e}_t, \tilde{e}_t \sim \mathcal{N}[0,1]$$
$$\tilde{h}_t = \tilde{h}_{t-1} + \tilde{u}_t, \tilde{u}_t \sim \mathcal{N}[0,1],$$

where $\tilde{h}_0 = 0$ (since $h_0 = \sigma_h \tilde{h}_0 + h_0 \iff \tilde{h}_0 = 0$). The user does not have to do anything except specify `ncp = true` (see below).

The derivations are the ones in "Bayesian Econometric Methods Second Edition" by Chan, Koop, Poirier, and Tobias Exercise 19.1, and are not repeated here for conciseness.

2.1 Usage

2.1.1 Construction

```
sv_Block = SVBlock(dep_init, opt)
```

- Inputs
  - `dep_init`: $T \times 1$ vector containing an initialisation for the observed variable y
  - `opt`: struct containing

        * `nu`: Scalar containing ν_h
        * `S`: Scalar containing S_h
        * `Vs`: Scalar containing V_s
        * `ic0`: Scalar containing a_0
        * `icV`: Scalar containing b_0
        * `ncp`: Boolean indicating whether the model should be estimated in the noncentred parameterisation

- Output
    - `sv_block`: struct containing (think of them as 'attributes' of an object)
        * `dep`: $T \times 1$ vector containing the observed variable y
        * `svsv`: $T \times 1$ vector of h
        * `ic`: Scalar value for h_0
        * `var`: Scalar value for σ_h^2
        * `T`: Scalar value containing T
        * `opt`: struct passed in at construction
        * `aux`: struct containing sparse difference matrix HH
        * If `opt.ncp` is set to `true`, the following fields are added:
            · `svsv_ncp`: $T \times 1$ vector containing $\tilde{h}_t$
            · `sd`: Scalar containing σ_h

2.1.2 Methods

Each method returns an updated `sv_block`.

- `sv_dep_update(sv_block, new_dep)`
    - `sv_block`: `SVBlock` object
    - `new_dep`: $T \times 1$ vector to replace previous y

This method replaces y_t with the new dependent variable supplied. It affords substantial flexibility: y_t need not stay constant across every step in the Gibbs sampler, and in fact it is often a residual from some equation, e.g. $y_t = \tilde{y}_t - \tau_t$, where τ_t is an unobserved trend variable and $\tilde{y}_t$ is the actual data.

- `sv_gibbs_update(sv_block)`
  - `sv_block`: an `SVBlock` object

  This steps samples h, h_0, and σ_h^2. This means that the attributes `sv_block.svsv`, `sv_block.ic`, and `sv_block.sig2` will be updated as per the parameterisation supplied when the object was constructed.

3 SBlock

The `SBlock` class constructs objects that sample s_t, and $\gamma = [s_0, s_{-1}, \ldots, s_{-p+1}]'$ from the simple model given by

$$\begin{aligned} y_t &= s_t + \varepsilon_t, \varepsilon \sim \mathcal{N}[0, \Sigma] \\ s_t &= \mu_t + \phi_1 s_{t-1} + \phi_2 s_{t-2} + \cdots + \phi_p s_{t-p} + v_t, v \sim \mathcal{N}[0, \Omega], \end{aligned}$$

for $t = 1, \ldots, T$, known μ_t, ϕ, Σ and Ω (note that ε and v are $T \times 1$, so that Σ and Ω are $T \times T$). This encompasses the popular case of a random walk (where $p = 1$ and $\phi_1 = 1$). μ_t is treated as an exogenous variable (e.g., a constant or a state variable sampled in another block) Priors for γ of the following type are supplied:

$$\gamma \sim \mathcal{N}[\gamma_0, V_\gamma].$$

No derivation for the above setup works 'out of the box', so the derivations for the conditional posterior are derived.

The state equation can be re-written by stacking observations in the usual way in matrix form as

$$H_\phi s = \alpha + \mu + v, \tag{1}$$

where H_ϕ is the $T \times T$ sparse matrix given by

$$H_\phi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ -\phi_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ -\phi_2 & -\phi_1 & 1 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ \vdots & -\phi_2 & -\phi_1 & 1 & 0 & 0 & 0 & 0 & \ldots & 0 \\ -\phi_p & \vdots & -\phi_2 & -\phi_1 & 1 & 0 & 0 & 0 & \ldots & 0 \\ 0 & -\phi_p & \vdots & -\phi_2 & -\phi_1 & 1 & 0 & 0 & \ldots & 0 \\ \vdots & 0 & -\phi_p & \vdots & -\phi_2 & -\phi_1 & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & 0 & -\phi_p & \vdots & -\phi_2 & -\phi_1 & 1 & \ddots & 0 \\ \vdots & \vdots & \vdots & 0 & -\phi_p & \vdots & -\phi_2 & -\phi_1 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & -\phi_p & \vdots & -\phi_2 & -\phi_1 & 1 \end{bmatrix},$$

and α is the $T \times 1$ vector given by

$$\alpha = \begin{bmatrix} \phi_1 s_0 + \phi_2 s_{-1} + \cdots + \phi_p s_{-p+1} \\ \phi_2 s_0 + \phi_3 s_{-1} + \cdots + \phi_p s_{-p+2} \\ \vdots \\ \phi_p s_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Hence the model can be written in matrix form as

$$\begin{aligned} y|s, \Sigma &\sim \mathcal{N}[s, \Sigma] \\ s|\mu, \phi, \Omega &\sim \mathcal{N}[H_\phi^{-1}(\alpha + \mu), (H_\phi' \Omega H_\phi)^{-1}], \end{aligned}$$

so that s can be sampled by the posterior given by

$$s|y, \mu, \phi, \Sigma, \Omega \sim \mathcal{N}[\hat{s}, K_s^{-1}],$$

where

$$\begin{aligned} K_s &= H_\phi' \Omega^{-1} H_\phi + \Sigma^{-1} \\ \hat{s} &= K_s^{-1}(H_\phi' \Omega^{-1} H_\phi H_\phi^{-1}(\alpha + \mu) + \Sigma^{-1} y). \end{aligned}$$

The initial states γ are sampled as follows. First notice that

$$\alpha = \tilde{X}\gamma,$$

where $\tilde{X}$ is the $T \times p$ matrix given by

$$\tilde{X} = \begin{bmatrix} \phi_1 & \phi_2 & \dots & \phi_p \\ \phi_2 & \vdots & & 0 \\ \vdots & \phi_p & & \vdots \\ \phi_p & 0 & \dots & \vdots \\ 0 & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}.$$

Thus, letting $X = H_\phi^{-1}\tilde{X}$, Equation (2) can be re-written as

$$\tilde{s} = X\gamma + H_\phi^{-1}v,$$

where $\tilde{s} = s - H_\phi^{-1}\mu$.

This means that γ can be treated as the coefficient vector of a linear regression model with covariance matrix $H_\phi^{-1}\Omega H_\phi^{-1\prime}$, so that the conditional posterior is given by

$$\gamma|y, \mu, \phi, \Omega \sim \mathcal{N}[\hat{\gamma}, K_\gamma^{-1}],$$

where

$$\begin{aligned} K_\gamma &= X' H_\phi' \Omega^{-1} H_\phi X + V_\gamma^{-1} \\ \hat{\gamma} &= K_\gamma^{-1}(X' H_\phi' \Omega^{-1} H_\phi \tilde{s} + V_\gamma^{-1}\gamma_0). \end{aligned}$$

3.1 Usage

3.1.1 Construction

```
SBlock(dep_init, exvar_init, coeff_init, obsVar_init, transVar_init,
    opt)
```

- Input
  - `dep_init`: $T \times 1$ vector containing an initialisation for y
  - `exvar_init`: $T \times 1$ vector containing an initialisation for μ
  - `coeff_init`: $p \times 1$ vector containing an initialisation for ϕ
  - `obsVar_init`: $T \times T$ matrix containing an initialisation for Σ
  - `transVar_init`: $T \times T$ matrix containing an initialisation for Ω
  - `opt`: struct containing
    * `ic0`: $p \times 1$ vector containing γ_0
    * `icV`: $p \times p$ matrix containing V_γ
- Output
  - `s_block`: struct containing (think of them as 'attributes' of an object)

      * `dep`: $T \times 1$ vector containing the observed variable y
      * `s`: $T \times 1$ vector of s
      * `exvar`: $T \times 1$ vector of μ
      * `coeff`: $p \times 1$ vector of ϕ
      * `res`: $T \times 1$ vector of residuals of the transition equation
      * `obsVar`: $T \times T$ matrix of Σ
      * `transVar`: $T \times T$ matrix of Ω
      * `H_coeff`: $T \times T$ matrix for representing autoregressive process
      * `ic`: $p \times 1$ vector containing γ
      * `alpha`: $T \times 1$ vector containing γ in matrix notation
      * `X_tilde`: $T \times p$ auxiliary matrix for sampling γ
      * `X`: $T \times p$ auxiliary matrix for sampling γ
      * `p`: Scalar indicating number of autoregressive lags
      * `T`: Scalar value containing T
      * `opt`: struct passed in at construction

3.1.2 Methods

- `s_dep_update(s_block, new_dep)`
  - `s_block`: `SBlock` object
  - `new_dep`: $T \times 1$ vector to replace previous y

  This method replaces y with the new dependent variable supplied. It affords substantial flexibility: y_t need not stay constant across every step in the Gibbs sampler, e.g. $y_t = \tilde{y}_t - \lambda f_t$, where f_t and λ are an unobserved factor and its loading, respectively, and $\tilde{y}_t$ is the actual data.

- `s_ex_coeff_var_update(s_block, new_exvar, new_coeff, new_obsVar, new_transVar)`
  - `s_block`: `SBlock` object

- `new_exvar`: (new) $T \times 1$ vector for μ
- `new_coeff`: (new) $p \times 1$ vector for ϕ
- `new_obsVar`: (new) $T \times T$ matrix for Σ
- `new_transVar`: (new) $T \times T$ matrix for Ω

This method updates μ, ϕ, Σ and Ω, and updates some auxiliary matrices for the class (which makes use of the methods `s_make_Xtilde`). The variables sampled in other blocks should be passed to this state block in this way.

- `s_gibbs_update(s_block)`
  - `s_block`: `SBlock` object

This step samples s and γ. This means that the attributes `s_block.s`, `s_block.ic` will be updated as per the parameterisation supplied when the object was constructed.

4 `RSBlock`

4.1 Motivation

`RSBlock` constructs objects that are used to sample the initial conditions of '*residual* state variables' of the type in `SBlock`. Consider the following model:

$$
\begin{aligned}
y_t &= s_{1t} + s_{2t} \\
s_{1t} &= \mu_{1t} + \phi_{11} s_{1,t-1} + \cdots + \phi_{1p_1} s_{1,t-p_1} + u_{1,t} \\
s_{2t} &= \mu_{2t} + \phi_{21} s_{2,t-1} + \cdots + \phi_{2p_2} s_{2,t-p_2} + u_{2,t}.
\end{aligned}
$$

where as before, μ_t is some known variable, s_{2t} is a residual state variable in the sense that at any step in the Gibbs sampler, s_{2t} is perfectly determined by y_t and s_{2t} (i.e., it does not have to be sampled).

These models are usually solved by substituting the residual state variable in the observation equation, subtracting the term $\mu_{2t} + \phi_{21} s_{2,t-1} + \cdots + \phi_{2p_2} s_{2,t-p_2} + u_{2,t}$ from the dependent variable in that block (this is legitimate since you are considering the distribution of s_1 *conditional* on s_2, so that s_2 can be treated as known), and sampling the non-residual state variable through an `SBlock`. To spell this out (since you will probably forget), the model to be estimated via an `SBlock` object is given by

$$
\begin{aligned}
\tilde{y}_t &= s_{1t} + u_{2,t} \\
s_{1t} &= \mu_{1t} + \phi_{11} s_{1,t-1} + \cdots + \phi_{1p_1} s_{1,t-p_1} + u_{1,t},
\end{aligned}
$$

where $\tilde{y}_t = y_t - (\mu_{2t} + \phi_{21} s_{2,t-1} + \cdots + \phi_{2p_2} s_{2,t-p_2})$. This setup fits exactly into the `SBlock` discussed above, so that the non-residual state variable (s_1), the initial conditions of the non-residual state variable (s_{10}) can be sampled. The residual state variable (s_2) is then simply given by $s_2 = y - s_1$.

However, this still requires sampling the initial conditions for the residual state variable. The class `RSBlock` does precisely this.

4.2 `RSBlock`

The `RSBlock` class constructs objects that sample $\gamma = [s_0, s_{-1}, \ldots, s_{-p+1}]'$ from the simple model given by

$$
s_t = \mu_t + \phi_1 s_{t-1} + \cdots + \phi_p s_{t-p} + u_t, u \sim \mathcal{N}[0, \Omega]
$$

where everything is specified outside of the class (i.e. known), except for the initial conditions $\gamma = [s_0, \cdots s_{-p+1}]'$ (notice that u is $T \times 1$ so that Ω is $T \times 1$).

Note: s_t need not necessarily be a state variable, but can also be an observed variable, e.g. if you want to estimate an AR process for some data. In this case you should use `RSBlock` to sample the initial conditions, and `ARBlock` to sample the coefficients and the variance.

4.3 Usage

4.3.1 Construction

```
RSBlock(exvar_init, coeff_init, Var_init, opt)
```

- Input
  - `exvar_init`: $T \times 1$ vector containing an initialisation for μ
  - `coeff_init`: $p \times 1$ vector containing an initialisation for ϕ
  - `Var_init`: $T \times T$ matrix containing an initialisation for Ω
  - `opt`: struct containing
    - `ic0`: $p \times 1$ vector containing γ_0
    - `icV`: $p \times p$ matrix containing V_γ
- Output
  - `rs_block`: struct containing (think of them as ‘attributes’ of an object)
    - `s`: $T \times 1$ vector of s
      - **Note**: this should be updated based on the draws for the non-residual state variable through the `rs_s_update` function
    - `exvar`: $T \times 1$ vector of μ
    - `coeff`: $p \times 1$ vector of ϕ
    - `res`: $T \times 1$ vector of residuals of the transition equation
    - `Var`: $T \times T$ matrix of Ω
    - `H_coeff`: $T \times T$ matrix for representing autoregressive process

* `ic`: $p \times 1$ vector containing γ
* `alpha`: $T \times 1$ vector containing γ in matrix notation
* `X_tilde`: $T \times p$ auxiliary matrix for sampling γ
* `X`: $T \times p$ auxiliary matrix for sampling γ
* `p`: Scalar indicating number of autoregressive lags
* `T`: Scalar value containing T
* `opt`: struct passed in at construction

4.3.2 Methods

- `rs_s_update(rs_block, new_s)`
  - `rs_block`: `RSBlock` object
  - `new_s`: (new) $T \times 1$ vector for s

  This method updates the residual state viarable s, e.g. based on the draws of the non-residual state variables in other parts of the Gibbs sampler.

- `rs_ex_coeff_var_update(rs_block, new_exvar, new_coeff, new_Var)`
  - `rs_block`: `RSBlock` object
  - `new_exvar`: (new) $T \times 1$ vector for μ
  - `new_coeff`: (new) $p \times 1$ vector for ϕ
  - `new_Var`: (new) $T \times T$ matrix for Ω

  This method updates μ, ϕ and Ω, and updates some auxiliary matrices for the class (which makes use of the methods `rs_make_Xtilde`). The variables sampled in other parts of the Gibbs sampler should be passed to this state block in this way.

- `rs_gibbs_update(rs_block)`
  - `rs_block`: `RSBlock` object

  This step samples γ. This means that the attribute `rs_block.ic` will be updated as per the parameterisation supplied when the object was constructed.

5 RegBlock

`RegBlock` constructs objects that are used to sample from a standard Bayesian regression:

$$y = X\beta + e, e \sim \mathcal{N}[0, \Sigma],$$

where y, e are $T \times 1$ vectors X is a $T \times k$, and β is $k \times 1$ vector. It always samples β, $\Sigma = \sigma^2 I_T$ based on the priors

$$\beta \sim \mathcal{N}[\beta_0, V_\beta]$$
$$\sigma^2 \sim \mathcal{IG}[\nu, S].$$

However, the class contains a method that allows to manually update the covariance matrix Σ, e.g. to include a covariance matrix estimated via stochastic volatility.

5.1 Usage

5.1.1 Construction

```
RegBlock(dep_init, indie_init, opt)
```

- Input
  - `dep_init`: $T \times 1$ vector containing an initialisation for y
  - `indie_init`: $T \times k$ matrix containing an initialisation for X
  - `opt`: struct containing
    - `pmean`: $k \times 1$ vector containing β_0
    - `pVar`: $k \times k$ matrix containing V_β
    - `nu`: Scalar containing ν
    - `S`: Scalar containing S
- Output
  - `reg_block`: struct containing (think of them as 'attributes' of an object)
    - `dep`: $T \times 1$ vector containing the observed variable y
    - `indie`: $T \times k$ matrix containing X

    * `coeff`: $k \times 1$ vector β
    * `res`: $T \times 1$ vector of residuals
    * `Var`: $T \times T$ matrix Σ
    * `k`: Scalar indicating number of independent variables
    * `T`: Scalar value containing T
    * `opt`: struct passed in at construction

5.1.2 Methods

- `reg_dep_indie_update(reg_block, new_dep, new_indie)`
  - `reg_block`: `RegBlock` object
  - `new_dep`: $T \times 1$ vector to replace previous y
  - `new_indie`: $T \times k$ vector to replace previous X

  This method replaces y and X with the new dependent and independent variable supplied.

- `reg_var_update(reg_block, new_Var)`
  - `reg_block`: `RegBlock` object
  - `new_Var`: (new) $T \times T$ matrix for Σ

  This method updates Σ. This is very useful in case the covariance matrix is sampled via stochastic volaility in another part of the Gibbs sampler.

- `reg_gibbs_update(reg_block)`
  - `reg_block`: `RegBlock` object

  This step samples β and σ^2 (and hence $\Sigma = \sigma^2 I_T$). This means that the attributes `reg_block.coeff`, `reg_block.Var` will be updated as per the parameterisation supplied when the object was constructed.

6 `ARBlock`

Note: very similar to `RegBlock`.

`ARBlock` constructs objects that are used to sample from a standard Bayesian autoregressive process:

$$y_t = \mu_t + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + e_t, e \sim \mathcal{N}[0, \Sigma],$$

where μ_t is some known variable (notice that e is $T \times 1$ so that Σ is $T \times 1$). It always samples ϕ, $\Sigma = \sigma^2 I_T$ based on the priors

$$\phi \sim \mathcal{N}[\phi_0, V_\phi] 1_{\phi \in R}$$
$$\sigma^2 \sim \mathcal{IG}[\nu, S],$$

where R indicates the stationarity region for the AR process. However, the class contains a method that allows to manually update the covariance matrix Σ, e.g. to include a covariance matrix estimated via stochastic volatility.

Initial conditions are not sampled, and they should be sampled through an `SBlock`.

6.1 Usage

6.1.1 Construction

```
ARBlock(dep_init , exvar_init , ic_init , opt)
```

- Input
  - `dep_init`: $T \times 1$ vector containing an initialisation for y
  - `exvar_init`: $T \times 1$ vector containing μ
  - `ic_init`: $p \times 1$ vector containing initialisation for initial conditions
  - `opt`: struct containing
    - `pmean`: $p \times 1$ vector containing ϕ_0
    - `pVar`: $p \times p$ matrix containing V_ϕ
    - `nu`: Scalar containing ν
    - `S`: Scalar containing S

- Output
  - `ar_block`: struct containing (think of them as 'attributes' of an object)
    * `dep`: $T \times 1$ vector containing the observed variable y
    * `coeff`: $p \times 1$ vector ϕ
    * `Var`: $T \times T$ matrix Σ
    * `X_phi`: $T \times p$ auxiliary matrix for sampling p
    * `p`: Scalar indicating number of independent variables
    * `T`: Scalar value containing T
    * `opt`: struct passed in at construction

6.1.2 Methods

- `ar_dep_ic_update(reg_block, new_dep, new_ic)`
  - `ar_block`: `ARBlock` object
  - `new_dep`: $T \times 1$ vector to replace previous y
  - `new_ic`: $T \times p$ vector to replace previous $[y_0, \ldots, y_{-p+1}]'$

  This method replaces y and $[y_0, \ldots, y_{-p+1}]'$ with the new dependent variable and initial conditions supplied.

- `ar_ex_var_update(ar_block, new_exvar, new_Var)`
  - `ar_block`: `ARBlock` object
  - `new_exvar`: (new) $T \times 1$ vector for μ
  - `new_Var`: (new) $T \times T$ matrix for Σ

  This method updates μ. This is very useful in case where μ contains state variables sampled in other steps of the Gibbs sampler. It is also very useful in case the covariance matrix is sampled via stochastic volaility in another part of the Gibbs sampler.

- `ar_gibbs_update(reg_block)`
  - `ar_block`: `ARBlock` object

This step samples ϕ and σ^2 (and hence $\Sigma = \sigma^2 I_T$). This means that the attributes `ar_block.coeff`, `ar_block.Var` will be updated as per the parameterisation supplied when the object was constructed.

7 DFBlock

`DFBlock` constructs objects that are used to sample from a dynamic factor model:

$$\begin{aligned} y_t &= \lambda f_t + \epsilon_t, \epsilon_t \sim \mathcal{N}[0, \Sigma_t] \\ f_t &= \mu_t + \Phi_1 f_{t-1} + \cdots + \Phi_p f_{t-p} + \varepsilon_t^f, \varepsilon_t^f \sim \mathcal{N}[0, \Omega_t], \end{aligned}$$

where y_t is a $n \times 1$ vector, f_t is a $q \times 1$ vector, μ_t is a $q \times 1$ vector of exogenous variables, λ is a $n \times q$ matrix, Σ_t is $n \times n$, Φ_j is $q \times q$, and Ω_t is $q \times q$.

This block samples f_t, $\gamma = [f_0', f_{-1}', f_{-p+1}']'$, λ, $\Sigma_t = \sigma^2 I_n$ based on the priors

$$\begin{aligned} \gamma &\sim \mathcal{N}[\gamma_0, V_\gamma] \\ \lambda &\sim \mathcal{N}[\mu_\lambda, V_\lambda] \\ \sigma^2 &\sim \mathcal{IG}[\nu, S] \end{aligned}$$

However, the class contains a method that allows to manually update the covariance matrices, e.g. to include a covariance matrix estimate via stochastic volatility methods.

Note: the `SBlock` did not sample any potential coefficient in the observation equation. This was because in most cases it does not exist, and because in case it does (e.g. a constant) it can be easily sampled via a `RegBlock`. I chose to include the sampling of λ directly in the class of `DFBlock` because factor loadings are sufficiently 'special' in the sense that some restrictions have to be imposed that can be a pain to figure out every single time one wants to estimate a quick factor model. Since the coefficients of the observation equation are sampled, for conceptual coherence and completeness I also make the class sample homoscedastic variances of the observation equation. As always, these be overwritten with the method described below. Since the coefficients Φ are likely best sampled via a `ARBlock` or `VARBlock` (which sample homoscedastic variances already), the homoscedastic variances of the transition equation are not sampled.

7.1 Sampling the Factor and the Initial Conditions

Stack y_t over t, to give the $Tn \times 1$ vector $y = [y_1', y_2', \ldots, y_T']'$, stack ϵ_t over t to give the $Tn \times 1$ vector $\epsilon = [\epsilon_1', \epsilon_2', \ldots, \epsilon_T']'$, stack f_t over t to give the $Tq \times 1$ vector $[f_1', f_2', \ldots, f_T']'$, define $\Lambda = I_T \otimes \lambda$, and let $\breve{\Sigma}$ be the $Tn \times Tn$ matrix given by $\breve{\Sigma} = \text{diag}(\Sigma_1, \Sigma_2, \ldots, \Sigma_T)$. Then the observation equation can be written as

$$y = \Lambda f + \epsilon, \epsilon \sim \mathcal{N}[0, \breve{\Sigma}].$$

The transition equation can be re-written as

$$H_\Phi f = \alpha + \mu + \varepsilon^f, \varepsilon \sim \mathcal{N}[0, \breve{\Omega}] \tag{2}$$

where H_Φ is the $Tq \times Tq$ sparse matrix given by

$$H_\Phi = \begin{bmatrix} I_q & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -\Phi_1 & I_q & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ -\Phi_2 & -\Phi_1 & I_q & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & -\Phi_2 & -\Phi_1 & I_q & 0 & 0 & 0 & 0 & \dots & 0 \\ -\Phi_p & \vdots & -\Phi_2 & -\Phi_1 & I_q & 0 & 0 & 0 & \dots & 0 \\ 0 & -\Phi_p & \vdots & -\Phi_2 & -\Phi_1 & I_q & 0 & 0 & \dots & 0 \\ \vdots & 0 & -\Phi_p & \vdots & -\Phi_2 & -\Phi_1 & I_q & 0 & \dots & 0 \\ \vdots & \vdots & 0 & -\Phi_p & \vdots & -\Phi_2 & -\Phi_1 & I_q & \ddots & 0 \\ \vdots & \vdots & \vdots & 0 & -\Phi_p & \vdots & -\Phi_2 & -\Phi_1 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & -\Phi_p & \vdots & -\Phi_2 & -\Phi_1 & I_q \end{bmatrix},$$

α is the $Tq \times 1$ vector given by

$$\alpha = \begin{bmatrix} \Phi_1 f_0 + \Phi_2 f_{-1} + \cdots + \Phi_p f_{-p+1} \\ \Phi_2 f_0 + \Phi_3 f_{-1} + \cdots + \Phi_p f_{-p+2} \\ \vdots \\ \Phi_p f_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$\mu = [\mu_1', \mu_2', \dots, \mu_T']'$, $\varepsilon^f = [\varepsilon_1^{f\prime}, \varepsilon_2^{f\prime}, \dots, \varepsilon_T^{f\prime}]'$, and $\breve{\Omega} = \text{diag}(\Omega_1, \Omega_2, \dots, \Omega_T)$.

Thus, the system can be written as

$$\begin{aligned} y &= \Lambda f + \epsilon, \epsilon \sim \mathcal{N}[0, \breve{\Sigma}] \\ f &= H_\phi^{-1}(\alpha + \mu) + H_\Phi^{-1}\varepsilon^f, \varepsilon \sim \mathcal{N}[0, \breve{\Omega}], \end{aligned}$$

i.e.,

$$y|\Lambda, f, \breve{\Sigma} \sim \mathcal{N}[\Lambda f, \breve{\Sigma}]$$
$$f|\Phi_j, \gamma, \mu, \breve{\Omega} \sim \mathcal{N}[H_\phi^{-1}(\alpha + \mu), H_\phi^{-1}\breve{\Omega}H_\phi^{-1\prime}].$$

By the central linear regression result,

$$f|y, \Phi_j, \gamma, \mu, \breve{\Omega} \sim \mathcal{N}[\hat{f}, K_f],$$

where

$$\begin{aligned} K_f &= H_\phi'\breve{\Omega}^{-1}H_\phi + \Lambda'\breve{\Sigma}^{-1}\Lambda \\ \hat{f} &= K_f^{-1}\left(H_\phi'\breve{\Omega}^{-1}H_\phi H_\phi^{-1}(\alpha+\mu) + \Lambda'\breve{\Sigma}^{-1}y\right). \end{aligned}$$

The initial states γ are sampled as follows. First notice that

$$\alpha = \tilde{X}\gamma,$$

where $\tilde{X}$ is the $Tq \times pq$ matrix given by

$$\tilde{X} = \begin{bmatrix} \Phi_1 & \Phi_2 & \dots & \Phi_p \\ \Phi_2 & \vdots & & 0 \\ \vdots & \Phi_p & & \vdots \\ \Phi_p & 0 & \dots & \vdots \\ 0 & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}.$$

Thus, letting $X = H_\Phi^{-1}\tilde{X}$, the transition equation can be written as

$$\tilde{f} = X\gamma + H_\Phi^{-1}\varepsilon^f,$$

where $\tilde{f} = f - H_\Phi^{-1}\mu$.

This means that γ can be treated as the coefficient vector of a linear regression model with covariance matrix $H_\Phi^{-1}\breve{\Omega}H_\Phi^{-1\prime}$, so that the conditional posterior is given by

$$\gamma|y, \mu, \Phi, \breve{\Omega} \sim \mathcal{N}[\hat{\gamma}, K_\gamma^{-1}],$$

where

$$K_\gamma = X' H'_\Phi \breve{\Omega}^{-1} H_\phi X + V_\gamma^{-1}$$
$$\hat{\gamma} = K_\gamma^{-1}(X' H'_\Phi \breve{\Omega}^{-1} H_\phi \tilde{f} + V_\gamma^{-1}\gamma_0).$$

7.2 Sampling the Factor Loadings

Factors loadings are sampled equation-by-equation for all $i = 1, \ldots, n$, i.e. in the equation

$$Y_i = F\lambda_i + \epsilon_j, \epsilon_j \sim \mathcal{N}[0, \Sigma_j],$$

where Y_i is the $T \times 1$ vector containing the i^{th} column of Y, where $y = vec(Y')$, F is the $T \times q$ matrix containing the factors ordered in a similar way, and λ_i is the i^{th} row of λ of dimension $q \times 1$. Note that you have to extract Σ_i from $\breve{\Sigma}$ which is a bit awkward but can be done in a single line of code by extracting every $(t-1) * n + i$ element from $\breve{\Sigma}$.

The rotational indeterminacy problem requires restricting the matrix λ to be lower triangular with ones on the diagonal. Define the restricted factor-loadings-matrix

$$\lambda = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ \lambda_{21} & 1 & 0 & \ldots & 0 \\ \lambda_{31} & \lambda_{32} & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \lambda_{n1} & \lambda_{n2} & \lambda_{n3} & \ldots & 1 \end{bmatrix}$$

1. For $i = 1$, the restrictions imply $\lambda_1 = [1, 0, \ldots, 0]'$ (leave unchanged at every iteration)

2. For $i = 2$, the restrictions imply $\lambda_2 = [\lambda_{21}, 1, 0, \ldots, 0]'$; to sample λ_2, have to consider the modified equation

$$\tilde{Y}_2 = \lambda_{21} F_2$$

   where $\tilde{Y}_2 = Y_2 - F_1$, and F_j indicates the j^{th} column of F. Sampling is now standard.

3. For $i = 3$, the restrictions imply $\lambda_3 = [\lambda_{31}, \lambda 32, 1, 0, \ldots, 0]'$; to sample λ_3, have to consider the modified equation

$$\tilde{Y}_3 = F_{1:2}\tilde{\lambda}_3$$

   where $\tilde{Y}_3 = Y_3 - F_3$, $F_{j:k}$ indicates the columns j to k of F, and $\tilde{\lambda}_3 = [\lambda_{31}, \lambda_{32}]'$. Sampling is now standard.

4. ...

7.3 Usage

7.3.1 Construction

```
DFBlock(dep_init, exvar_init, loadings_init, coeff_init,
    transVar_init, opt)
```

- Input
  - `dep_init`: $T \times n$ matrix containing an initialisation for Y
    OR (class is smart enough to figure out which one you are passing in)
    $Tn \times 1$ vector containing an initialisation for y
  - `exvar_init`: $T \times q$ matrixr containing an initialisation for μ
    OR (class is smart enough to figure out which one you are passing in)
    $Tq \times 1$ vector containing an initialisation for μ (in vector form)
  - `loadings_init`: $n \times q$ matrix of factor loadings
  - `coeff_init`: $q \times q \times p$ array containing an initialisation for Φs
  - `transVar_init`: $T \times T$ matrix containing an initialisation for Ω_t
    OR (class is smart enough to figure out which one you are passing in)
    $Tn \times Tn$ matrix containing an initialisation for $\breve{\Omega}$
  - `opt`: struct containing
    * `ic0`: $pq \times 1$ vector containing γ_0
    * `icV`: $pq \times pq$ matrix containing V_γ
    * `nu`: Scalar for ν
    * `S`: Scalar for S
    * `T`: Scalar for T (needed to help code figure out whether you are passing in flattened data / covariance matrices)
- Output
  - `df_block`: struct containing (think of them as 'attributes' of an object)
    * `dep`: $Tn \times 1$ vector containing the observed variable y
    * `Dep`: $T \times n$ matrix containing the observed variable Y

* `f`: $Tq \times 1$ vector of f
* `F`: $T \times q$ matrix of F
* `exvar`: $Tq \times 1$ vector of μ
* `Exvar`: $T \times q$ matrix of reshaped μ
* `loadings`: $Tn \times Tq$ matrix of Λ
* `loadings_s`: $n \times q$ matrix of λ
* `coeff`: $p \times p \times q$ array of Φs
* `res`: $Tq \times 1$ vector of residuals of the transition equation
* `Res`: $T \times q$ matrix of reshaped `res` residuals of the transition equation
* `Res_obs`: $T \times n$ matrix of reshaped residuals of the observation equation
* `obsVar`: $Tn \times Tn$ matrix of $\breve{\Sigma}$
* `transVar`: $Tq \times Tq$ matrix of $\breve{\Omega}$
* `H_coeff`: $Tq \times Tq$ matrix for representing autoregressive process
* `ic`: $pq \times 1$ vector containing γ
* `alpha`: $Tq \times 1$ vector containing γ in matrix notation
* `X_tilde`: $Tq \times pq$ auxiliary matrix for sampling γ
* `X`: $Tq \times pq$ auxiliary matrix for sampling γ
* `p`: Scalar indicating number of autoregressive lags
* `n`: Scalar indicating number of columns in Y lags
* `q`: Scalar indicating number of factors
* `T`: Scalar value containing T
* `opt`: struct passed in at construction

7.3.2 Methods

- `df_dep_update(df_block, new_dep)`
  - `df_block`: `DFBlock` object
  - `new_dep`: (new) $T \times n$ matrix containing a new value for Y
    OR (class is smart enough to figure out which one you are passing in)
    $Tn \times 1$ vector containing a new value for y

  This method replaces y with the new dependent variable supplied.

- `df_ex_coeff_var_update(df_block, new_exvar, new_coeff, new_obsVar, new_transVar)`
  - `df_block`: `SBlock` object
  - `new_exvar`: (new) $T \times q$ matrix containing a new value for μ
    OR (class is smart enough to figure out which one you are passing in)
    $T \times q$ matrix containing reshaped μ
  - `new_coeff`: (new) $q \times q \times p$ array for Φs
  - `new_obsVar`: $Tn \times Tn$ matrix containing $\breve{\Sigma}$
    OR (class is smart enough to figure out which one you are passing in)
    (new) $T \times T$ matrix containing a new value for $\Sigma_1 = \Sigma_t$
  - `new_transVar`: $Tq \times Tq$ matrix containing $\breve{\Omega}$
    OR (class is smart enough to figure out which one you are passing in)
    (new) $T \times T$ matrix containing a new value for $\Omega_1 = \Omega_t$

  This method updates μ, $\Phi, \breve{\Sigma}$ and $\breve{\Omega}$, and updates some auxiliary matrices for the class (which makes use of the methods `s_make_Xtilde`). The variables sampled in other blocks should be passed to this state block in this way.

- `df_gibbs_update(df_block)`
  - `df_block`: `DFBlock` object

  This step samples f and γ. This means that the attributes `df_block.f`, `df_block.ic` will be updated as per the parameterisation supplied when the object was constructed.

8 SLBlock

This class generalises the `SBlock` class to the case where there are leads and lags in both the observation equation and the transition equation,

$$y_t = cs_t + a_1 s_{t-1} + a_2 s_{t-2} + \cdots + a_{p_1} s_{t-p_1} + b_1 s_{t+1} + b_2 s_{t+2} + \cdots + b_{p_2} s_{t+p_2} + \varepsilon_t, \varepsilon \sim \mathcal{N}[0, \Sigma]$$
$$s_t = \mu_t + \phi_1 s_{t-1} + \phi_2 s_{t-2} + \cdots + \phi_{q_1} s_{t-q_1} + \theta_1 s_{t+1} + \theta_2 s_{t+2} + \cdots + \theta_{q_2} s_{t+q_2} + v_t, v_t \sim \mathcal{N}[0, \Omega]$$

Define the $(q_1+q_2)\times 1$ vector of initial and final conditions as $\gamma = [s_0, s_{-1}, \ldots, s_{-q_1+1}, s_{T+1}, s_{T+2}, \ldots, S_{T+q_2}]'$; the following prior has to be supplied

$$\gamma \sim \mathcal{N}[\gamma_0, V_\gamma]$$

Importantly, $p_1 \leq q_1$, $p_2 \leq q_2$.

The strategy is the same as before: we want to express the likelihood of y as a function of s, and derive a function of s in terms of the initial (and future) conditions, and finally apply standard Bayesian regression results.

8.1 Manipulating Observation Equation

(the matrix is a bit ugly, but you get the idea; the 'bs' on the upper diagonal, and the 'as' on the lower diagonal)

$$y = \underbrace{\begin{bmatrix} c & b_1 & b_2 & \ldots & b_{p_2} & 0 & \ldots & 0 \\ a_1 & c & b_1 & b_2 & \ldots & b_{p_2} & 0 & 0 \\ a_2 & a_1 & c & b_1 & b_2 & \ldots & b_{p_2} & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ a_{p_1} & a_{p_1-1} & a_{p_1-2} & \ldots & \ddots & b_1 & b_2 & \ldots \\ 0 & a_{p_1} & a_{p_1-1} & \ddots & \ddots & \ddots & b_1 & b_2 \\ 0 & 0 & a_{p_1} & \ddots & \ddots & & \ddots & b_1 \\ 0 & 0 & 0 & a_{p_1} & \ddots & \ddots & & c \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_T \end{bmatrix}}_{s} + \underbrace{\begin{bmatrix} a_{p_1} s_{-p_1+1} + a_{p_1-1} s_{-p_1+2} + \cdots + a_1 s_0 \\ a_{p_1} s_{-p_1+2} + a_{p_1-1} s_{-p_1+3} + \cdots + a_1 s_1 \\ \vdots \\ a_{-p+1} s_0 \\ 0 \\ \vdots \\ b_{p_2} s_{T+1} \\ b_{p_2} s_{T+2} + b_{p_2-1} s_{T+1} \\ \vdots \\ b_{p_2} s_{T+p_2} + b_{p_2-1} s_{T+p_2-1} + \cdots + b_1 s_{T+1} \end{bmatrix}}_{\xi} + \varepsilon$$

$$= Gs + \xi + \varepsilon$$

where G is $T \times T$, ξ is $T \times 1$.

8.2 Manipulating Transition Equation

Notice that

$$\underbrace{\begin{bmatrix} 1 & -\theta_1 & -\theta_2 & \dots & -\theta_{q_2} & 0 & \dots & 0 \\ -\phi_1 & 1 & -\theta_1 & -\theta_2 & \dots & -\theta_{q_2} & 0 & 0 \\ -\phi_2 & -\phi_1 & 1 & -\theta_1 & -\theta_2 & \dots & -\theta_{q_2} & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ -\phi_{q_1} & -\phi_{q_1-1} & -\phi_{q_1-2} & \dots & \ddots & -\theta_1 & -\theta_2 & \dots \\ 0 & -\phi_{q_1} & -\phi_{q_1-1} & \ddots & \ddots & \ddots & -\theta_1 & -\theta_2 \\ 0 & 0 & -\phi_{q_1} & \ddots & \ddots & & \ddots & \theta_1 \\ 0 & 0 & 0 & -\phi_{q_1} & \ddots & \ddots & & 1 \end{bmatrix}}_{\tilde{H}} \underbrace{\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_T \end{bmatrix}}_{s} = \begin{bmatrix} s_1 - \theta_1 s_2 - \dots - \theta_{p_2} s_{1+q_2} \\ -\phi_1 s_1 + s_2 - \theta_1 s_3 - \dots - \theta_{q_2} s_{2+p_2} \\ -\phi_2 s_1 - \phi_1 s_2 + s_3 - \theta_1 s_4 - \dots - \theta_{q_2} s_{3+q_2} \\ \vdots \\ -\phi_{q_1} s_{-q_1+T} + \dots + \phi_1 s_{T-1} + s_T \end{bmatrix}$$

Thus,

$$\tilde{H} s = \tilde{\alpha} + \mu + v,$$

where

$$\tilde{\alpha} = \begin{bmatrix} \phi_1 s_0 + \phi_2 s_{-1} + \dots + \phi_{q_1} s_{-q_1+1} \\ \phi_1 s_1 + \dots + \phi_{q_1} s_{-q_1} \\ \vdots \\ \phi_{q_1} s_0 \\ 0 \\ \vdots \\ \theta_{q_2} s_{T+1} \\ \theta_{q_2} s_{T+2} + \theta_{q_2-1} s_{T+1} \\ \vdots \\ \theta_1 s_{T+1} + \dots + \theta_{q_2} T + q_2 \end{bmatrix}.$$

where H is $T \times T$, $\tilde{\alpha}$ is $T \times 1$.

We have thus arrived at our favourite dual expression:

$$\begin{aligned} \tilde{y}|a,b,c,\Sigma,\xi &\sim \mathcal{N}[Gs,\Sigma] \\ s|\phi,\theta,\mu,\tilde{\alpha} &\sim \mathcal{N}[\tilde{H}^{-1}(\tilde{\alpha}+\mu),\tilde{H}^{-1}\Omega\tilde{H}^{-1\prime}], \end{aligned}$$

where $\tilde{y} = y - \xi$. Thus, the conditional posterior is given by

$$s|\tilde{y},a,b,c,\Sigma,\xi,\phi,\theta,\mu,\tilde{\alpha} \sim \mathcal{N}[\hat{s},K_s^{-1}],$$

where

$$\begin{aligned} K_s &= \tilde{H}'\Omega^{-1}\tilde{H} + G'\Sigma^{-1}G \\ \hat{s} &= K_s^{-1}(\tilde{H}'\Omega^{-1}\tilde{H}\tilde{H}^{-1}(\tilde{\alpha}+\mu) + G'\Sigma^{-1}\tilde{y}). \end{aligned}$$

8.3 Getting initial and final conditions

Just as before, write $\tilde{\alpha}$ as

$$\tilde{\alpha} = \begin{bmatrix} \phi_1 s_0 + \phi_2 s_{-1} + \cdots + \phi_{q_1} s_{-q_1+1} \\ \phi_1 s_1 + \cdots + \phi_{q_1} s_{-q_1} \\ \vdots \\ \phi_{q_1} s_0 \\ 0 \\ \vdots \\ \theta_{q_2} s_{T+1} \\ \theta_{q_2} s_{T+2} + \theta_{q_2-1} s_{T+1} \\ \vdots \\ \theta_1 s_{T+1} + \cdots + \theta_{q_2} T + q_2 \end{bmatrix} = \underbrace{\begin{bmatrix} \phi_1 & \phi_2 & \dots & \phi_{q_1} & 0 & 0 & \dots & 0 \\ \phi_2 & \dots & \phi_{q_1} & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{q_1} & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & \theta_{q_2} & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & \theta_{q_2-1} & \theta_{q_2} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & \theta_1 & \theta_2 & \dots & \theta_{q_2} \end{bmatrix}}_{\tilde{X}} \underbrace{\begin{bmatrix} s_0 \\ s_{-1} \\ \vdots \\ s_{-q_1+1} \\ s_{T+1} \\ s_{T+2} \\ \vdots \\ S_{T+q_2} \end{bmatrix}}_{\gamma}$$

where $\tilde{X}$ is a $T \times (q_1 + q_2)$ matrix, and γ is a $(q_1 + q_2) \times 1$ vector.

Thus, letting $X = \tilde{H}^{-1}\tilde{X}$,

$$\tilde{s} = X\gamma + \tilde{H}^{-1}v,$$

where $\tilde{s} = s - \tilde{H}^{-1}\mu$. Thus, the vector γ can be sampled by standard linear regression results (given the prior supplied),

$$\gamma|y,\mu,\phi,\theta,\Omega \sim \mathcal{N}[\hat{\gamma},K_\gamma^{-1}],$$

where

$$K_\gamma = X'\tilde{H}'\Omega^{-1}\tilde{H}X + V_\gamma^{-1}$$
$$\hat{\gamma} = K_\gamma^{-1}(X'\tilde{H}'\Omega^{-1}\tilde{H}\tilde{s} + V_\gamma^{-1}\gamma_0).$$

8.4 Usage

8.4.1 Construction

```
SLBlock(dep_init, exvar_init, coeff_init, obsVar_init,
    transVar_init, opt)
```

- Input
  - `dep_init`: $T \times 1$ vector containing an initialisation for y
  - `exvar_init`: $T \times 1$ vector containing an initialisation for μ
  - `coeff_init`: struct containing
    - `zero`: $T \times 1$ vector containing an initialisation for c
    - `obs_lag`: $p_1 \times 1$ coefficient vector a
    - `obs_lead`: $p_2 \times 1$ coefficient vector b
    - `trans_lag`: $q_1 \times 1$ vector of coefficients for ϕ
    - `trans_lead`: $q_2 \times 1$ vector of coefficients for θ
  - `obsVar_init`: $T \times T$ matrix containing an initialisation for Σ
  - `transVar_init`: $T \times T$ matrix containing an initialisation for Ω
  - `opt`: struct containing
    - `ic0`: $(q_1 + q_2) \times 1$ vector containing γ_0
    - `icV`: $(q_1 + q_2) \times (q_1 + q_2)$ matrix containing V_γ
- Output
  - `s_block`: struct containing (think of them as 'attributes' of an object)
    - `dep`: $T \times 1$ vector containing the observed variable y
    - `s`: $T \times 1$ vector of s

* `exvar`: $T \times 1$ vector of μ
* `coeff_init`: struct containing
  * `zero`: $T \times 1$ vector containing an initialisation for c
  * `obs_lag`: $p_1 \times 1$ coefficient vector a
  * `obs_lead`: $p_2 \times 1$ coefficient vector b
  * `trans_lag`: $q_1 \times 1$ vector of coefficients for ϕ
  * `trans_lead`: $q_2 \times 1$ vector of coefficients for θ
* `res`: $T \times 1$ vector of residuals of the transition equation
* `obsVar`: $T \times T$ matrix of Σ
* `transVar`: $T \times T$ matrix of Ω
* `H_coeff`: $T \times T$ matrix for representing autoregressive process
* `ic`: $(q_1 + q_2) \times 1$ vector containing γ
* `ic_obs`: $(p_1 + p_2) \times 1$ vector of initial conditions that feature in the observation equation (since $p_1 \leq q_1, p_2 \leq q_2$ this will always be a subset of `ic`).
* `alpha`: $T \times 1$ vector containing γ in matrix notation
* `xi`: $T \times 1$ vector containing ξ (see notation above)
* `X_tilde`: $T \times p$ auxiliary matrix for sampling γ
* `X`: $T \times p$ auxiliary matrix for sampling γ
* `T`: Scalar value containing T
* `opt`: struct passed in at construction

8.4.2 Methods

* `s_dep_update(s_block, new_dep)`
  - `s_block`: `SBlock` object
  - `new_dep`: $T \times 1$ vector to replace previous y

- `s_ex_coeff_var_update(s_block, new_exvar, new_coeff, new_obsVar, new_transVar)`
  - `s_block`: `SBlock` object
  - `new_exvar`: (new) $T \times 1$ vector for μ
  - `new_coeff`: struct containing
    * `zero`: $T \times 1$ vector containing an initialisation for c
    * `obs_lag`: $p_1 \times 1$ coefficient vector a
    * `obs_lead`: $p_2 \times 1$ coefficient vector b
    * `trans_lag`: $q_1 \times 1$ vector of coefficients for ϕ
    * `trans_lead`: $q_2 \times 1$ vector of coefficients for θ
  - `new_obsVar`: (new) $T \times T$ matrix for Σ
  - `new_transVar`: (new) $T \times T$ matrix for Ω
- `s_gibbs_update(s_block)`
  - `s_block`: `SBlock` object

  This step samples s and γ.

9 Example 1: Stochastic Volatility in Mean

$$\begin{aligned} y_t &= \mu + e^{\frac{1}{2}h_t}\epsilon_t, \epsilon_t \sim \mathcal{N}[0,1] \\ h_t &= h_{t-1} + u_t, u_t \sim \mathcal{N}[0,\sigma_h^2], \end{aligned} \tag{3}$$

where μ is a constant. Priors are given by

$$\begin{aligned} \mu &\sim \mathcal{N}[0, V_\mu] \\ h_0 &\sim \mathcal{N}[a_0, b_0] \\ \sigma_h^2 &\sim \mathcal{N}[\nu, S]. \end{aligned}$$

The following variables need to be sampled: μ, h_t, h_0, σ_h^2.

9.1 Coefficient Block

Conditional on h_t, it is possible to re-write the observation equation as

$$y = \mu\mathbf{1} + \epsilon, \epsilon \sim \mathcal{N}[0, \Sigma],$$

where $\Sigma = \text{Diag}(e^{h_1}, \ldots, e^{h_T})$.

It is hence possible to draw a new draw for μ through a `RegBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. Σ has to be updated to $\text{Diag}(e^{h_1}, \ldots, e^{h_T})$ (to overwrite the default values of σ_ϵ^2 generated by `RegBlock`)
    - When `reg_gibbs_update` is called, a homoscedastic variance estimator will be generated, which has to be overwritten by the stochastic variance draws before the next Gibbs step. This is somewhat inefficient, but hardly has an impact on runtimes.

In pseudo-code:

while $i < num\_rep$ **do**
  with *reg_block_obs*
    Update $\Sigma = \text{Diag}(e^{h_1}, \ldots, e^{h_T})$;
    Sample μ (and 'redundant' homoscedastic Σ, overwritten at next iteration anyway);
end

Algorithm 1: Block for drawing μ.

9.2 Stochastic Volatility Block

Conditional on μ (i.e. it is treated as known), it can be written as

$$\tilde{y}_t = e^{\frac{1}{2}h_t}\epsilon_t, \epsilon_t \sim \mathcal{N}[0,1]$$
$$h_t = h_{t-1} + u_t, u_t \sim \mathcal{N}[0,\sigma_h^2]$$

for $\tilde{y}_t = y_t - \mu$ (i.e., the residual).

It is hence possible to draw a new draw for h_t and h_0 through a `SVBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. $\tilde{y}$ has to be updated with the new value for μ

In pseudo-code:

while $i < num\_rep$ **do**
  with *sv_block*
    Update $\tilde{y}$ with new value of μ;
    Sample h, h_0;
end

Algorithm 2: Block for drawing h, h_0.

9.3 Pseudo-Code Summary

```
Init: reg_block_obs, sv_block;
while i < num_rep do
    with reg_block_obs
        Update Σ = Diag(e^{h_1}, ..., e^{h_T});
        Sample μ (and 'redundant' homoscedastic Σ, overwritten at next iteration anyway);
    with sv_block
        Update ỹ with new value of μ;
        Sample h, h_0;
end
```

Algorithm 3: MCMC algorithm for the Model in Equation (3), drawing μ, h, h_0, σ_h^2.

See examples directory `Example_1.m` for the implementation in Matlab.

Note: this example was taken from Exercise 19.1 in "Bayesian Econometric Methods Second Edition" by Chan, Koop, Poirier, and Tobias. The code was compared to the solution code, and it yields identical results.

10 Example 2: Unobserved Components with Stochastic Volatility (I)

$$
\begin{aligned}
y_t &= \tau_t + \varepsilon_t^y \\
\tau_t &= \tau_{t-1} + \varepsilon_t^\tau \\
h_t &= h_{t-1} + \varepsilon_t^h \\
g_t &= g_{t-1} + \varepsilon_t^g,
\end{aligned}
\tag{4}
$$

where $\varepsilon_t^y \sim \mathcal{N}[0, e^{h_t}]$, $\varepsilon_t^\tau \sim \mathcal{N}[0, e^{g_t}]$, $\varepsilon_t^h \sim \mathcal{N}[0, \sigma_h^2]$, $\varepsilon_t^g \sim \mathcal{N}[0, \sigma_g^2]$. Priors for initial conditions and variances are standard.

The following variables need to be sampled: τ_t, τ_0, h_t, h_0, σ_h^2, g_t, g_0, σ_g^2.

10.1 State Variable Block

Conditional on h, g, h_0, g_0, σ_h^2, σ_g^2, the model can be written as

$$
\begin{aligned}
y_t &= \tau_t + \varepsilon_t^y, \varepsilon^y \sim \mathcal{N}[0, \Sigma] \\
\tau_t &= \tau_{t-1} + \varepsilon_t^\tau, \varepsilon^\tau \sim \mathcal{N}[0, \Omega],
\end{aligned}
$$

where $\Sigma = \text{Diag}(e^{h_1}, \ldots, e^{h_T})$, $\Omega = \text{Diag}(e^{g_1}, \ldots, e^{g_T})$.

It is hence possible to draw a new draw for τ and τ_0 through a `SBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. Σ has to be updated to $\text{Diag}(e^{h_1}, \ldots, e^{h_T})$
2. Ω has to be updated to $\text{Diag}(e^{g_1}, \ldots, e^{g_T})$

In pseudo-code:

while $i < num\_rep$ **do**
    with *s_block_tau*
        Update Σ with $\text{Diag}(e^{h_1}, \ldots, e^{h_T})$;
        Update Ω with $\text{Diag}(e^{g_1}, \ldots, e^{g_T})$;
        Sample τ, τ_0;
end

Algorithm 4: Block for drawing τ, τ_0.

10.2 Stochastic Volatility Blocks

10.2.1 Drawing h_t, h_0, σ_h^2

Conditional on τ (i.e., it is treated as known), the observation equation can be written as

$$\begin{aligned}\tilde{y}_t &= e^{\frac{1}{2}h_t}\epsilon_t, \epsilon_t \sim \mathcal{N}[0,1] \\ h_t &= h_{t-1} + \varepsilon_t^h, \varepsilon_t^h \sim \mathcal{N}[0, \sigma_h^2],\end{aligned}$$

for $\tilde{y}_t = y_t - \tau$ (i.e., the residual).

It is hence possible to draw a new draw h_t, h_0, and σ_h^2 through a `SVBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. $\tilde{y}$ has to be updated with new values of τ

In pseudo-code:

while $i < num\_rep$ **do**
    with *sv_block_h*
        Update $\tilde{y}$ with new values of τ;
        Sample h, h_0, σ_h^2;
end

Algorithm 5: Block for drawing h, h_0, σ_h^2.

10.2.2 Drawing g_t, g_0, σ_g^2

Conditional on τ and τ_0 (i.e., both τ_t and τ_{t-1} are known), the transition equation can be written as

$$\tilde{\tau}_t = e^{\frac{1}{2}g_t}\epsilon_t, \epsilon_t \sim \mathcal{N}[0,1]$$
$$g_t = g_{t-1} + \varepsilon_t^g, \varepsilon_t^g \sim \mathcal{N}[0,\sigma_g^2],$$

for $\tilde{\tau}_t = \tau_t - \tau_{t-1}$ (i.e., the residual).

It is hence possible to draw a new draw g_t, g_0, and σ_g^2 through a `SVBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. $\tilde{\tau}$ has to be updated with new values of τ, τ_0

In pseudo-code:

while $i < num\_rep$ **do**
    with *sv_block_g*
        Update $\tilde{\tau}$ with new values of τ, τ_0;
        Sample g, g_0, σ_g^2;
end

Algorithm 6: Block for drawing g, g_0, σ_g^2.

10.3 Pseudo-Code Summary

```
Init: s_block_tau, sv_block_h, sv_block_g;
while i < num_rep do
    with s_block_tau
        Update Σ with Diag(e^{h_1}, ..., e^{h_T}) ;
        Update Ω with Diag(e^{g_1}, ..., e^{g_T}) ;
        Sample τ, τ_0;
    with sv_block_h
        Update ỹ with new values of τ;
        Sample h, h_0, σ_h^2;
    with sv_block_g
        Update τ̃ with new values of τ, τ_0;
        Sample g, g_0, σ_g^2;
end
```

Algorithm 7: MCMC algorithm for the Model in Equation (4), drawing τ, τ_0, h_t, h_0, σ_h^2, g_t, g_0, σ_g^2.

See examples directory `Example_2.m` for the implementation in Matlab.

Note: this example was taken from Exercise 19.2 in "Bayesian Econometric Methods Second Edition" by Chan, Koop, Poirier, and Tobias. The code was compared to the solution code, and it yields identical results.

11 Example 3: Unobserved Components with Stochastic Volatility (II)

$$\begin{aligned} y_t &= \tau_t + c_t \\ c_t &= \phi_1 c_{t-1} + \cdots + \phi_p c_{t-p} + u_t^c \\ \tau_t &= \mu + \tau_{t-1} + u_t^\tau \\ h_t &= h_{t-1} + \varepsilon_t^h \\ g_t &= g_{t-1} + \varepsilon_t^g, \end{aligned} \tag{5}$$

where $u_t^c \sim \mathcal{N}[0, e^{g_t}]$, $u_t^\tau \sim \mathcal{N}[0, e^{h_t}]$, $\varepsilon_t^h \sim \mathcal{N}[0, \sigma_h^2]$, $\varepsilon_t^g \sim \mathcal{N}[0, \sigma_g^2]$. Priors for initial conditions and variances are standard. Define $\gamma = [c_0, \ldots, c_{-p+1}]'$. Note that μ can be interpreted as the average growth rate of trend output.

One of the state variables τ_t, c_t is a residual state variable in the sense that conditional on a draw of the other, it is perfectly determined. This means that at every iteration of the Gibbs sampler, the draw from this 'residual state variable' is simply y minus the other state variable. To apply the code (and other approaches) **substitute the residual state variable into the observation equation** to give

$$y_t = \mu + \tau_{t-1} + c_t + u_t^\tau. \tag{6}$$

The following variables need to be sampled: c, γ, τ_0, μ, ϕ, h_t, h_0, σ_h^2, g_t, g_0, σ_g^2.

11.1 State Variable Block

Conditional on everything but c, the modified observation equation in Equation (6) and the transition equation for c_t can be written as

$$\begin{aligned} \tilde{y}_t &= c_t + u_t^\tau, u^\tau \sim \mathcal{N}[0, \Sigma] \\ c_t &= \phi_1 c_{t-1} + \cdots + \phi_p c_{t-p} + u_t^c, u^c \sim \mathcal{N}[0, \Omega], \end{aligned}$$

where $\tilde{y}_t = y_t - \mu - \tau_{t-1}$, $\Sigma = \text{Diag}(e^{h_1}, \ldots, e^{h_T})$, $\Omega = \text{Diag}(e^{g_1}, \ldots, e^{g_T})$.

It is hence possible to draw a new draw for c, γ through a `SBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. Σ has to be updated to $\text{Diag}(e^{h_1}, \ldots, e^{h_T})$

2. Ω has to be updated to $\text{Diag}(e^{g_1}, \ldots, e^{g_T})$

3. ϕ has to be updated to new draws of AR coefficients

4. $\tilde{y}$ has to be updated with new values of μ, τ_{t-1}

In pseudo-code:

while $i < \textit{num\_rep}$ **do**
  with *s_block_c*
    Update Σ with $\text{Diag}(e^{h_1}, \ldots, e^{h_T})$;
    Update Ω with $\text{Diag}(e^{g_1}, \ldots, e^{g_T})$;
    Update ϕ with new draws;
    Update $\tilde{y}$ with new values of μ, τ_{-1};
    Sample c, γ;
end

Algorithm 8: Block for drawing c, γ.

11.2 Residual State Variable Block

τ is treated as the 'residual' state variable so that $\tau = y - c$. Hence the only thing that is not determined by the preceding block are the initial condition τ_0, i.e. we only have to sample τ_0 from the model

$$\tau = \mu + \tau_{t-1} + u_t^{\tau}, u \sim \mathcal{N}[0, \Sigma]$$

where $\Sigma = \text{Diag}(e^{h_1}, \ldots, e^{h_T})$. In this block, everything is known except for τ_0.

It is hence possible to draw a new draw for τ_0 an `RSBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. τ_t has to be updated to $y_t - s_t$

2. μ has to be updated to new draws of μ

3. Σ has to be updated to $\text{Diag}(e^{h_1}, \ldots, e^{h_T})$ (to overwrite the homoscedastic covariance matrix generated by `RegBlock`)

In pseudo-code:

```
while i < num_rep do
    with s_block_tau
        Update τ with new values of c;
        Update μ;
        Update Σ with Diag(e^{h_1}, ..., e^{h_T}) ;
        Sample τ_0 (and create τ_{-1});
end
```

Algorithm 9: Block for drawing τ_0.

11.3 Coefficient Blocks

11.3.1 Block for μ

It is possible to re-write the observation equation in Equation (6) as:

$$\breve{y}_t = \mu + u_t^\tau, u^\tau \sim \mathcal{N}[0, \Sigma]$$

where $\breve{y}_t = y_t - \tau_{t-1} - c_t$, $\Sigma = \text{Diag}(e^{h_1}, \ldots, e^{h_T})$. In this block, everything is known except for μ.

It is hence possible to draw a new draw for μ through an `RegBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. $\breve{y}_t$ has to be updated to $y_t - \tau_{t-1} - c_t$
2. Σ has to be updated to $\text{Diag}(e^{h_1}, \ldots, e^{h_T})$ (to overwrite the homoscedastic covariance matrix generated by `RegBlock`)
    - When `reg_gibbs_update` is called, a homoscedastic variance estimator will be generated, which has to be overwritten by the stochastic variance draws before the next Gibbs step. This is somewhat inefficient, but hardly has an impact on runtimes.

In pseudo-code:

while $i < num\_rep$ **do**

  with *reg_block_obs*

    Update $\breve{y}$ with new values of τ_{-1}, c;

    Update Σ with $\text{Diag}(e^{h_1}, \dots, e^{h_T})$;

    Sample μ (and 'redundant' homoscedastic Σ, overwritten at next iteration anyway);

end

Algorithm 10: Block for drawing μ.

11.3.2 Block for ϕ

The transition equation is given by

$$c_t = \phi_1 c_{t-1} + \cdots + \phi_p c_{t-p} + u_t^c, u \sim \mathcal{N}[0, \Omega]$$

where $\Omega = \text{Diag}(e^{g_1}, \dots, e^{g_T})$. In this block, everything is known except for ϕ.

It is hence possible to draw a new draw for ϕ through an `ARBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. c has to be updated to new draws of c
2. γ has to be updated to new draw of γ
3. Ω has to be updated to $\text{Diag}(e^{g_1}, \dots, e^{g_T})$ (to overwrite the homoscedastic covariance matrix generated by `RegBlock`)
    - When `ar_gibbs_update` is called, a homoscedastic variance estimator will be generated, which has to be overwritten by the stochastic variance draws before the next Gibbs step. This is somewhat inefficient, but hardly has an impact on runtimes.

In pseudo-code:

while $i <$ *num_rep* **do**
  with *ar_block_trans*
    Update c_t;
    Update γ;
    Update Ω with $\mathrm{Diag}(e^{g_1}, \dots, e^{g_T})$;
    Sample ϕ (and 'redundant' homoscedastic Ω, overwritten at next iteration anyway);
end

Algorithm 11: Block for drawing ϕ.

11.4 Stochastic Volatility Blocks

11.4.1 Drawing h_t, h_0, σ_h^2

Conditional on everything except h_t, h_0, and σ_h^2, the transition equation of the residual state variable can be combined with the transition equation for h_t:

$$\tilde{\tau}_t = e^{\frac{1}{2}h_t}\epsilon_t, \epsilon_t \sim \mathcal{N}[0,1]$$
$$h_t = h_{t-1} + \varepsilon_t^h, \varepsilon_t^h \sim \mathcal{N}[0, \sigma_h^2],$$

where $\tilde{\tau}_t = \tau_t - \tau_{t-1} - \mu$ (i.e., the residual).

It is hence possible to draw a new draw h_t, h_0, and σ_h^2 through a `SVBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. $\tilde{\tau}$ has to be updated with new values of τ, τ_{-1}, μ

In pseudo-code:

while $i <$ *num_rep* **do**
  with *sv_block_h*
    Update $\tilde{\tau}$ with new values of τ, τ_{-1}, μ;
    Sample h, h_0, σ_h^2;
end

Algorithm 12: Block for drawing h, h_0, σ_h^2.

11.4.2 Drawing g_t, g_0, σ_g^2

Conditional on everything except g_t, g_0, and σ_g^2, the transition equation for c_t can be combined with the transition equation for g_t

$$\tilde{c}_t = e^{\frac{1}{2}g_t}\epsilon_t, \epsilon_t \sim \mathcal{N}[0,1]$$
$$g_t = g_{t-1} + \varepsilon_t^g, \varepsilon_t^g \sim \mathcal{N}[0,\sigma_g^2],$$

for $\tilde{c}_t = c_t - (\phi_1 c_{t-1} + \cdots + \phi_p c_{t-p})$ (i.e., the residuals).

It is hence possible to draw a new draw g_t, g_0, and σ_g^2 through a `SVBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. $\tilde{c}$ has to be updated with new values of c, γ

In pseudo-code:

while $i <$ *num_rep* **do**
  with *sv_block_g*
    Update $\tilde{c}$ with new values of c, γ;
    Sample g, g_0, σ_g^2;
end

Algorithm 13: Block for drawing g, g_0, σ_g^2.

11.5 Pseudo-Code Summary

Init: s_block_c, s_block_tau, reg_block_obs, ar_block_trans, sv_block_h, sv_block_g;

while $i < num\_rep$ **do**

- **with** *s_block_c*
  - Update Σ with $\text{Diag}(e^{h_1}, \ldots, e^{h_T})$;
  - Update Ω with $\text{Diag}(e^{g_1}, \ldots, e^{g_T})$;
  - Update ϕ with new draws;
  - Update $\tilde{y}$ with new values of μ, τ_{-1};
  - Sample c, γ;
- **with** *s_block_tau*
  - Update τ with new values of c;
  - Update μ;
  - Update Σ with $\text{Diag}(e^{h_1}, \ldots, e^{h_T})$;
  - Sample τ_0 (and create τ_{-1});
- **with** *reg_block_obs*
  - Update $\breve{y}$ with new values of τ_{-1}, c;
  - Update Σ with $\text{Diag}(e^{h_1}, \ldots, e^{h_T})$;
  - Sample μ (and 'redundant' homoscedastic Σ, overwritten at next iteration anyway);
- **with** *ar_block_trans*
  - Update c_t;
  - Update γ;
  - Update Ω with $\text{Diag}(e^{g_1}, \ldots, e^{g_T})$;
  - Sample ϕ (and 'redundant' homoscedastic Ω, overwritten at next iteration anyway);
- **with** *sv_block_h*
  - Update $\tilde{\tau}$ with new values of τ, τ_{-1}, μ;
  - Sample h, h_0, σ_h^2;
- **with** *sv_block_g*
  - Update $\tilde{c}$ with new values of c, γ;
  - Sample g, g_0, σ_g^2;

end

Algorithm 14: MCMC algorithm for the Model in Equation (5), drawing c, γ, τ_0, μ, ϕ, h_t, h_0, σ_h^2, g_t, g_0, σ_g^2.

See examples directory `Example_3.m` for the implementation in Matlab.

Note: this exercise in an extension of an exercise in"Bayesian Econometric Methods Second Edition" by Chan, Koop, Poirier, and Tobias, so it cannot be directly compared to the solution code.

12 Example 4: Factor Models with Stochastic Volaility

$$\begin{aligned} y_t &= \lambda f_t + \epsilon_t \\ f_t &= \phi f_{t-1} + \epsilon_t^f \\ g_t &= g_{t-1} + \varepsilon_t^g \\ h_{tj} &= h_{t-1,j} + \varepsilon_{tj}^h, \end{aligned} \tag{7}$$

where y_t is an $n \times 1$ vector, $\epsilon_t^f \sim \mathcal{N}[0, e^{g_t}]$, $\epsilon_{tj} \sim \mathcal{N}[0, e^{h_{tj}}]$, $\varepsilon_{tj}^h \sim \mathcal{N}[0, \sigma_{hj}^2]$, $\varepsilon_t^g \sim \mathcal{N}[0, \sigma_g^2]$. Priors for initial conditions and variances are standard.

The following variables need to be sampled: f, f_0, ϕ, h_{tj}, h_{0j}, σ_{hj}^2, g_t, g_0, σ_g^2.

12.1 Dynamic Factor Block

Conditional on everything but f, f_0, and λ

$$\begin{aligned} y_t &= \lambda f_t + \epsilon_t, \epsilon_t \sim \mathcal{N}[0, \Sigma_t] \\ f_t &= \phi_1 f_{t-1} + \epsilon_t^f, \epsilon_t^f \sim \mathcal{N}[0, \Omega_t]. \end{aligned}$$

where $\Sigma_t = \text{diag}(e^{h_{t1}}, e^{h_{t2}}, \ldots, e^{h_{tn}})$ and $\Omega_t = e^{g_t}$.

It is hence possible to draw a new draw for f, f_0, and λ through a `DFBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. Σ_t has to be updated to $\text{diag}(e^{h_{t1}}, e^{h_{t2}}, \ldots, e^{h_{tn}})$ for all $t = 1, \ldots, T$
2. Ω_t has to be updated to e^{g_t} for all $t = 1, \ldots, T$
3. ϕ_1 has to be updated to new draws of the AR coefficient

In pseudo-code:

while $i < num\_rep$ **do**
  with *df_block_f*
    Update Σ_t with $\mathrm{diag}(e^{h_{t1}}, e^{h_{t2}}, \ldots, e^{h_{tn}})$ (and construct $\breve{\Sigma}$) ;
    Update Ω_t with e^{g_t} (and construct $\breve{\Omega}$) ;
    Update ϕ with new draws;
    Sample f, f_0, λ;
end

Algorithm 15: Block for drawing f, f_0, λ.

12.2 AR Block

Conditional on everything but ϕ

$$f_t = \phi_1 f_{t-1} + \epsilon_t^f, \epsilon^f \sim \mathcal{N}[0, \breve{\Omega}].$$

where $\breve{\Omega} = \mathrm{diag}(e^{g_1}, \ldots, e^{g_T})$.

It is hence possible to draw a new draw for ϕ through a `ARBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. f_t and f_0 have to be updated
2. $\breve{\Omega}$ has to be updated to $\mathrm{diag}(e^{g_1}, \ldots, e^{g_T})$
    - When `ar_gibbs_update` is called, a homoscedastic variance estimator will be generated, which has to be overwritten by the stochastic variance draws before the next Gibbs step. This is somewhat inefficient, but hardly has an impact on runtimes.

In pseudo-code:

while $i < num\_rep$ **do**
  with *s_block_phi*
    Update f, f_0 ;
    Update $\breve{\Omega}$ with $\mathrm{diag}(e^{g_1}, \ldots, e^{g_T})$;
    Sample ϕ;
end

Algorithm 16: Block for drawing ϕ.

12.3 Stochastic Volatility Blocks

12.3.1 Drawing h_{tj}, h_{0j}, σ^2_{hj}

Conditional on f and λ, for every $j = 1, \ldots, n$

$$\tilde{y}_{tj} = e^{\frac{1}{2}h_{tj}}\epsilon_{tj}, \epsilon_{tj} \sim \mathcal{N}[0, 1]$$
$$h_{tj} = h_{t-1,j} + \varepsilon^h_{tj},$$

for $\tilde{y}_{tj} = y_{tj} - f\lambda_j$ (i.e., the residual).

It is hence possible to draw a new draw h_{tj}, h_{0j}, σ^2_{hj} through n `SVBlock` objects, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. $\tilde{y}_j$ has to be updated with new values of f and λ_j for each $j = 1, \ldots, n$

In pseudo-code:

```
while i < num_rep do
    for j = 1:n do
        with sv_block_h_j
            Update ỹ_j with new values of f, λ_j;
            Sample h_tj, h_0j, σ²_hj;
    end
end
```

Algorithm 17: Block for drawing h_{tj}, h_{0j}, σ^2_{hj}.

12.3.2 Drawing g_t, g_0, σ^2_g

Conditional on f and f_0 (i.e., both f_t and f_{t-1} are known), the transition equation can be written as

$$\tilde{f}_t = e^{\frac{1}{2}g_t}\epsilon_t, \epsilon_t \sim \mathcal{N}[0, 1]$$
$$g_t = g_{t-1} + \varepsilon^g_t, \varepsilon^g_t \sim \mathcal{N}[0, \sigma^2_g],$$

for $\tilde{f}_t = f_t - \phi f_{t-1}$ (i.e., the residual).

It is hence possible to draw a new draw g_t, g_0, and σ^2_g through a `SVBlock` object, as discussed in the code documentation.

It is immediately clear what parts of this block have to be updated at every iteration:

1. $\tilde{f}$ has to be updated with new values of f, f_0

In pseudo-code:

while $i <$ *num_rep* **do**
    with *sv_block_g*
        Update $\tilde{f}$ with new values of f, f_0;
        Sample g, g_0, σ_g^2;
end

Algorithm 18: Block for drawing g, g_0, σ_g^2.

12.4 Pseudo-Code Summary

Init: df_block_f, sv_block_phi, sv_block_h_j, sv_block_g;
while $i <$ *num_rep* **do**
    with *df_block_f*
        Update Σ_t with $\mathrm{diag}(e^{h_{t1}}, e^{h_{t2}}, \ldots, e^{h_{tn}})$ (and construct $\breve{\Sigma}$) ;
        Update Ω_t with e^{g_t} (and construct $\breve{\Omega}$) ;
        Update ϕ with new draws;
        Sample f, f_0, λ;
    with *s_block_phi*
        Update f, f_0 ;
        Update $\breve{\Omega}$ with $\mathrm{diag}(e^{g_1}, \ldots, e^{g_T})$;
        Sample ϕ;
    for j = *1:n* **do**
        with *sv_block_h_j*
            Update $\tilde{y}_j$ with new values of f, λ_j;
            Sample h_{tj}, h_{0j}, σ_{hj}^2;
    end
    with *sv_block_g*
        Update $\tilde{f}$ with new values of f, f_0;
        Sample g, g_0, σ_g^2;
end

Algorithm 19: MCMC algorithm for the Model in Equation (7), drawing f, f_0, ϕ, h_{tj}, h_{0j}, σ_{hj}^2, g_t, g_0, σ_g^2.

See examples directory `Example_4.m` for the implementation in Matlab.

Note: this exercise in an extension of an exercise in"Bayesian Econometric Methods Second Edition" by Chan, Koop, Poirier, and Tobias, so it cannot be directly compared to the solution code. However, the model without the stochastic volatility blocks yields exactly

References

Chan, Joshua, Gary Koop, Dale Poirier, and Justin Tobias (2019). *Bayesian Econometric Methods*. Ed. by Karim Abadir, Jan Magnus, and Peter Phillips. Second Edition. Cambridge University Press.