Tutorial for A Hitchhiker Guide to Empirical Macro Models[1]

Filippo Ferroni (Chicago Fed)
Fabio Canova (Norwegian Business School, CAMP and CEPR)

January 8, 2021

[1]The views in this paper are solely the responsibility of the authors and should not be interpreted as reflecting the views of the Federal Reserve Bank of Chicago or any other person associated with the Federal Reserve System.

Outline

Introduction

- The Empirical Macro Toolbox (EMT) is a package which uses MATLAB (Octave) functions and routines to estimate a number of reduced form and structural macro models with either Classical or Bayesian methods and to forecast.

Introduction

- The Empirical Macro Toolbox (EMT) is a package which uses MATLAB (Octave) functions and routines to estimate a number of reduced form and structural macro models with either Classical or Bayesian methods and to forecast.
- It also allows the computation of trend and cycle decompositions, to date turning points and compute statistics of cyclical phases.

Introduction

- The Empirical Macro Toolbox (EMT) is a package which uses MATLAB (Octave) functions and routines to estimate a number of reduced form and structural macro models with either Classical or Bayesian methods and to forecast.
- It also allows the computation of trend and cycle decompositions, to date turning points and compute statistics of cyclical phases.
- Inference on parameters; point/density forecasts; dynamic propagation of *identified* shocks is possible; also allows for nowcasts and mixed-frequency estimation, system reduction, and large datasets.

Introduction

- The Empirical Macro Toolbox (EMT) is a package which uses MATLAB (Octave) functions and routines to estimate a number of reduced form and structural macro models with either Classical or Bayesian methods and to forecast.
- It also allows the computation of trend and cycle decompositions, to date turning points and compute statistics of cyclical phases.
- Inference on parameters; point/density forecasts; dynamic propagation of *identified* shocks is possible; also allows for nowcasts and mixed-frequency estimation, system reduction, and large datasets.
- Source codes with examples can be forked/downloaded at https://github.com/naffe15/BVAR_ or https://sites.google.com/view/fabio-canova-homepage/home/empirical-macro-toolbox

Introduction

- The Empirical Macro Toolbox (EMT) is a package which uses MATLAB (Octave) functions and routines to estimate a number of reduced form and structural macro models with either Classical or Bayesian methods and to forecast.
- It also allows the computation of trend and cycle decompositions, to date turning points and compute statistics of cyclical phases.
- Inference on parameters; point/density forecasts; dynamic propagation of *identified* shocks is possible; also allows for nowcasts and mixed-frequency estimation, system reduction, and large datasets.
- Source codes with examples can be forked/downloaded at https://github.com/naffe15/BVAR_ or https://sites.google.com/view/fabio-canova-homepage/home/empirical-macro-toolbox
- The hitchhiker guide can be downloaded here https://www.filippoferroni.com/empiricalmacrotoolbox or https://sites.google.com/view/fabio-canova-homepage/home/empirical-macro-toolbox

Bird's eye view of the content

- VAR methods:
  Classic and Bayesian estimation, Point/density forecasts, many identification procedures for SVARs, computation of IRF,historical and variance decompositions, missing values/mixed frequency, VARXs, panels of VARs, time varying coefficient and heteroschedastic models.

Bird's eye view of the content

- VAR methods:
  Classic and Bayesian estimation, Point/density forecasts, many identification procedures for SVARs, computation of IRF,historical and variance decompositions, missing values/mixed frequency, VARXs, panels of VARs, time varying coefficient and heteroschedastic models.

- Compression methods:
  Static (PC) FAVARs.

Bird's eye view of the content

- VAR methods:
  Classic and Bayesian estimation, Point/density forecasts, many identification procedures for SVARs, computation of IRF,historical and variance decompositions, missing values/mixed frequency, VARXs, panels of VARs, time varying coefficient and heteroschedastic models.

- Compression methods:
  Static (PC) FAVARs.

- Local projection methods:
  Classic and Bayesian estimation, point and density forecasts, structural IRF (via Cholesky or IV).

Bird's eye view of the content

- VAR methods:
  Classic and Bayesian estimation, Point/density forecasts, many identification procedures for SVARs, computation of IRF,historical and variance decompositions, missing values/mixed frequency, VARXs, panels of VARs, time varying coefficient and heteroschedastic models.

- Compression methods:
  Static (PC) FAVARs.

- Local projection methods:
  Classic and Bayesian estimation, point and density forecasts, structural IRF (via Cholesky or IV).

- Filtering methods:
  Extracting trend/cycles and gaps, Dating BC, computing turning points.

Road map I

Topics (with examples in the tutorials as blueprints):

I Dating turning points. Cyclical statistics.
`USERLOCATION/BVAR/v4.2/'Trend-Cycle-Dating tutorial'/example_2_dating.m`

II Trend and cycle decompositions. Comparison financial and real cycle.
`USERLOCATION/BVAR/v4.2/'Trend-Cycle-Dating tutorial'/example_1_cycles.m`

III Classical VAR inference (Bayesian estimation with flat prior).
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_1_classical.m`

IV VARX: estimation and inference.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_6_VARX.m`

V Bayesian VAR inference: Minnesota and conjugate priors. Chosing Minnesota hyper-parameters.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_2_minn.m`

VI IRFs with various identification schemes. Historical and Variance Decomposition.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_3_irf.m`

Road map II

Topics (examples in the tutorial as blueprints):

VII PCVAR (FAVAR) estimation and inference.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_5_favar.m`

VIII Panels of VAR models.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_8_panels.m`

IX Nowcasts and Mixed-Frequency VAR estimation.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_4_mfvar.m`

X Point and Density Forecasts. Conditional Forecasts in VARs.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_9_prediction.m`

XI Local projections and direct forecasts.
`USERLOCATION/BVAR/v4.2/'BVAR tutorial'/example_7_LP.m`

Getting started

I Open MATLAB, and add the toolbox to the MATLAB path.
Assuming that EMT is stored in `C:\USERLOCATION`, and you are using version 4.2, type in the command window (in your script):

```
addpath C:/USERLOCATION/BVAR/v4.2
```

You need to this every time you start MATLAB. Alternatively, go to the set-path icon in the MATLAB HOME window, click on it, add with BVAR/v4.2 with subfolders, and save.

II Use the same procedure to install Chris Sims' optimization routines (used to optimally select the Minnesota prior hyper-parameters). In this case, type in the command window

```
addpath C:/USERLOCATION/BVAR/cmintools
```

III Load the data

- If data is matlab format, type in the command window:
  ```
  load data
  ```
  and check the content in the workspace window.

- Alternatively, if the data comes in a spreadsheet use (standard MATLAB command):
  ```
  y=xlsread('filename','sheetname')
  ```

where `filename` is the name of the file, and `sheetname` is the name of the sheet where you have your data (in case you have many sheets.

- For more information about loading data in other formats, type in the command window:
```
help load
```

Outline

Turning point computations I

- Follows Bry and Boschen algorithm as implemented by Harding and Pagan (2006).

- Activated using the command:

```
[dt_] = date_(y, time, freq, tstart, tend, options);
```

  where `y` is the data, `time` is the time span (assumed of same length as `y`), `freq` is a string with the frequency of the data; that is, `'m'` (`'q'`) for monthly (quarterly) data; `tstart` (`tend`) is a scalar with the year and the month/quarter with the start (end) of the exercise, e.g. `tstart = [1970 3]` and `tend = [2020 1]`. The final input, `options`, allows to customize the exercise in various ways:
  - `options.phase` sets the minimum duration of the estimated phase; default for quarterly (monthly) data is 2 (6).
  - `options.cycle` sets the minimum length of the estimated cycle; default for quarterly (monthly) data is 5 (15).
  - `options.thresh` bypasses phase and cycle restrictions if peak to trough is larger than a threshold; default is 10.4.
  - `options.complete`. When equals one, it uses only complete cycles; else incomplete cycles are used (excess still computed on complete cycles). Default value equals one.
  - `options.nrep` is used when multiple datasets (e.g. simulated from a model) are considered. When `options.nrep` is employed, the dating exercise is computed using `options.nrep` $\times$ `T` observations. Default is one.

Turning point computations II

- The output of the function is a field structure containing a number of objects:
    - `dt_.st` is a $T \times 1$ a binary array containing the phases, i.e. expansion (1) and contraction (0).
    - `dt_.trinary` is a $T \times 1$ trinary array which contains the cycle states, i.e. peak (1), trough (-1) or none of the two (0).
    - `dt_.dura` is a 2×1 array with the average duration of contractions and of expansions, respectively.
    - `dt_.ampl` is a 2×1 array with the average amplitude of contractions and of expansions, respectively.
    - `dt_.cumm` is a 2×1 array with the average cumulative change in contractions and expansions, respectively.
    - `dt_.excc` is a 2×1 array reporting the percent of excess movements of the triangle area for contraction and expansions, respectively.
    - `dt_.durcv` is a 2×1 array the coefficient of variation (the inverse of the standard deviation) of the duration of contractions and expansions.
    - `dt_.amplcv` is a 2×1 array the coefficient of variation of the amplitude of contractions and expansions.
    - `dt_.exccv` is a 2×1 array the coefficient of variation of the excess movements in contractions and expansions.
- The concordance of turning points (or of cyclical phases) is not automatically computed. See manual (or tutorial examples) on how to do it.
- If `y` includes more than one series, one needs to loop the `date_` command appropriately.

Outline

Trend and Cycle decompositions I

Many decompositions are allowed. Most are univariate; some are bivariate (multivariate).

- Polynomial trends. The decomposition is activated using:

  ```
  [dc,dt] = polydet(y,ord,gg);
  ```

  where y is the data, ord is the order of the polynomial (the upper limit is 4) and gg an indicator; if 1 the data, the trend, and the cycle are plotted. The outputs are the estimated cycle dc and the estimated trend dt.

- Polynomial breaking trends. The decomposition is activated using:

  ```
  [dc,dt] = polydet_break(y,ord,tt,gg);
  ```

  where y is the data, ord is the order of the polynomial (upper limit 4), tt is the break date and gg an indicator; if 1 the data, the trend, and the cycle are plotted. The outputs are the estimated cycle dc and the estimated trend dt. The break date must be selected in advance.

- Differencing. No option is set, but can be easily implemented with MATLAB commands (see manual or tutorial example for how to do it).

Trend and Cycle decompositions II

- Hamilton (local projection) approach. The decomposition is activated using:

```
[dc,dt] = hamfilter(y,h,lags,ff,gg);
```

  where `y` is the data, `h` is the horizon of the projection (upper limit $h = T - lags - 1$); `lags` is the number of lags in the projection equation; `ff` an indicator; if it is equal to 1 a constant is included in the projection; gg an indicator; if 1 the data, the trend, and the cycle are plotted. The outputs are the estimated cycle `dc` and the estimated trend `dt`. `y` could be a $T \times n$ matrix, where T is the sample size and n the number of series. Conditioning variables, other than the lags of y, are not currently allowed.

- Hodrick and Prescott filter. The decomposition is activated using:

```
dt = Hpfilter(y,lam);      dc = y-dt;
```

  where `y` is the data, `lam` is a smoothing parameter. The outputs are the estimated trend `dt`. `lam` must be inputted by the user (there is no default value). Typical choice are 1,600 for quarterly data, 6.25 for annual data, and 129,000 for monthly data.

- One-sided Hodrick and Prescott filter. The decomposition is activated using:

```
[dt,dc] = one_sided_hpfilter_serial(y,lam,disc);
```

  where `y` is the data, `lam` is the smoothing parameter, `disc` the number of unused observations (typically, `disc = 0`). The outputs are the estimated trend `dt` and the estimated cycle `dc`. The one-sided HP filter only looks at past and current data to construct trend estimates (rather than past, current and future data).

Trend and Cycle decompositions III

- Band Pass filter: Baxter and King implementation It can be activated using:

```
dc = bkfilter(y,bp1,bp2);      dt = y-dc;
```

  bp1 (bp2) is the upper (lower) limit of the frequency band over which the squared gain function is 1. No default values for bp1 and bp2 are set. Typical choices for quarterly data are bp1 = 8 and bp2 = 32. For monthly (annual) data typical choices are bp1 = 24 (2) and bp2 = 96 (8).

- Band Pass filter: Christiano and Fitzgerald implementation It can be activated using:

```
dc = cffilter(y,bp1,bp2,cf1,cf2,cf3);     dt = y-dc;
```

  where cf1 is an indicator function describing the integration properties of the input (cf1 = 0 no unit root); cf2 is an indicator function for whether there is a drift in the unit root or a deterministic trend (cf2 = 0 no drift); cf3 chooses the format of the filter:
  - cf3 = 0 uses the basic asymmetric filter;
  - cf3 = 1 uses the symmetric filter;
  - cf3 = 2 uses a fixed length symmetric filter;
  - cf3 = 3 uses a truncated, fixed length symmetric filter (the BK choice);
  - cf3 = 4 uses a trigonometric regression filter.

  If cf1, cf2 and cf3 are omitted, cf1 = 1, cf2 = 1 and cf3 = 0 are the defaults.

Trend and Cycle decompositions IV

- Wavelet filter. The decomposition activated using:

```
[dc1,dt1,dc2,dt2] = wavefilter(y,gg);
```

  where dc1 and dc2 are the cyclical components computed focusing on 8-32 ($D_{3t} + D_{4t}$) or 8-64 ($D_{3t} + D_{4t} + D_{5t}$) quarters cycles and dt1 = y-dc1, dt2 = y-dc2 are the corresponding trend estimates; y is the input series, and gg is an indicator function for whether the cyclical components are plotted (for gg = 1 plots are shown).

- Butterworth filter. No option set; implementable with MATLAB commands (see manual).

- Unobservable component (UC) decomposition 1. The decomposition with random walk trend and AR(2) cycle can be activated using:

```
[dt,dc] = uc1_(y,opts);
```

  where y is the data and opts is a field structure that allows customization:
  - opts.figg: when equals 1 it plots the series, the trend, and the cycle.
  - opts.nsims: sets the number of MCMC simulations (default 5000).
  - opts.burnin: sets the number of burn-in draws (default, 30000).
  - opts.rhoyes when equals 1, it allows correlation between the trend and the cycle (default, 0)
  - opts.mu0 (opts.Vmu): prior mean (variance) trend drift (default, 0.1 [(default, 1)]
  - opts.phi0: prior mean cycle AR1 and AR2 coefficients (default, [0.2 0.2])
  - opts.Vphi: prior variance AR coefficients (default, diagonal matrix with 1)
  - opts.sigc2 (opts.sigtau2:) prior mean cycle (trend) variance (default, 0.5) [(default, 1.5)]

  The approach uses the MCMC implementation of Grant and Chan (2017).

Trend and Cycle decompositions V

- Unobservable component (UC) decomposition 2. Here the trend is assumed to have a local linear specification, the cycle specification is a user choice. Implementation is by maximum likelihood.The decomposition can be activated using:

  ```
  [yt,yc,out] = uc2_(y,lags,opts);
  ```

  where `y` is the data, `lags` is the number of lags in the cycle. The parameters are ordered as: $[\varphi_1, ..., \varphi_p, \sigma_\epsilon, \sigma_{\eta,1}, \sigma_{\eta,2}]$ (AR, stdev of measurement error, stdev of latent variables); they can be estimated or calibrated by setting the appropriate options and are reported as additional subfields in `out`. `opts` is a field structure that allows customization:
  - `opts.phi` is a `lags` $\times$ 1 vector that sets the initial values for the φ's (default, 0.5^j for $j = 1, ..., d$).
  - `opts.sigma` is a 3×1 vector that sets the initial values for the σ's (default, 0.5).
  - `opts.index_est` is a row vector that selects the parameters to be optimized. By default,all parameters are optimized, `opts.index_est=1:lags+3`.
  - `opts.lb` and `opts.ub` set the lower and upper bounds for the optimization. Both are row array vectors of the same size of `opts.index_est`.
  - `opts.max_compute` is a scalar selecting the maximization routine. The settings are the same as those for the maximization of the Minnesota hyper-parameters (see manual and later).

Trend and Cycle decompositions VI

- Beveridge and Nelson (BN) univariate. The decomposition is activated using:

  ```
  [dc,dt] = BNuniv(y,lags,ff,gg, mm);
  ```

  where `y` is the time series, `lags` is the number of lags in the estimated AR; `ff` is an indicator function for the constant (`ff = 0` no constant); `gg` is an indicator function for whether the estimated cycle is plotted or not (`gg = 0` no plot); `mm` is an indicator function for whether the decomposition is computed using the model-based long run mean or the estimated mean of y_t (`mm = 1` use the estimated mean; `mm = 0` use long run mean).

- Bivariate Beveridge and Nelson and Blanchard and Quah decompositions. The two decompositions are jointly activated using:

  ```
  [dt1,dc1,dt2,dc2] = BNBQbiv(y,lags,ff,rhoyes,ssts,gg,mm);
  ```

  where `lags` is the number of lags in the VAR; `ff` is an indicator function for the constant (if `ff = 1` there is a constant); `rhoyes` is an indicator function for whether the two disturbances are correlated (if `rhoyes = 1` shocks are correlated; this option triggers the Cover et al. (2006) decomposition); `ssts` is an indicator function for whether the other VAR variables are also integrated (if `ssts = 1` it is $I(1)$); `gg` is an indicator function for the plots (if `gg = 1` the permanent and transitory component of the first variable are plotted); `mm` is an indicator function for spectral density plots (if `mm = 1` spectral densities plots appear on the screen). `y` is a $T \times 2$ matrix; `dt1`, `dc1` (`dt2`, `dc2`) are the permanent and the transitory components of the first series obtained with BN (with BQ).

Practice I

Sample codes are in `...\BVARtoolbox\v4.2\Trend-Cycle-dating tutorial`.

- Dating turning points: US and Euro area (example_2_dating.m)
- Linking the cyclical components of output and credit (example_1_cycles.m)

Run them, do the little exercises and acquaint yourself with the language and the syntax.

Outline

Classical VAR Estimation and Inference

- Estimation in EMT can be performed using Classical or Bayesian method. By default, Jeffrey (flat) prior is assumed, which results in classical estimates.
- Given a prior, draws from the posterior distribution of the parameters are produced using a Gibbs sampler algorithm; see the guide for more details.
- The baseline, all purpose, estimation function is

```
[BVAR] = bvar_(y, lags, options)
```

- Inputs are:
  - `y` a $(\mathtt{T} \times \mathtt{n})$ array of data (rows=time; columns=variables)
  - `lags` > 0 the number of lags.
  - `options` is a field structure that specifies various customized options (i.e. priors, horizons, identification schemes). It can be omitted. The default identification scheme is Cholesky.

Outputs

- `BVAR` a field with many sub-fields; the most important are (check manual for full list).

  - `BVAR.Phi_draws` is a $(\mathtt{n} \times \mathtt{lags} + 1) \times \mathtt{n} \times \mathtt{K}$ array containg `K` draws for the parameters. `BVAR.Phi_draws(:,:,k)` stacks vertically the AR matrix; last row of `BVAR.Phi_draws(:,:,k)` contains the constant, Φ_0, if it assumed to be present.
  - `BVAR.Sigma_draws` is a $\mathtt{n} \times \mathtt{n} \times \mathtt{K}$ matrix containing `K` draws for Σ.
  - `BVAR.e_draws` is a $(\mathtt{T} - \mathtt{lags}) \times \mathtt{n} \times \mathtt{K}$ matrix containing `K` draws for the innovations.

  - `BVAR.Phi_ols`, `BVAR.Sigma_ols` and `BVAR.e_ols` are the OLS point estimate analogs.

  - `BVAR.ir_draws` is a four dimensional object (i.e. $\mathtt{n} \times \mathtt{hor} \times \mathtt{n} \times \mathtt{K}$ matrix) that collects the impulse response functions with the chosen identification.

  - `BVAR.forecasts.no_shocks` (`with_shocks`) is a three dimensional object (i.e. $\mathtt{fhor} \times \mathtt{n} \times \mathtt{K}$ matrix) that collects the forecasts assuming zero (non-zero) shocks in the future.

  - `BVAR.logmlike` contains the log marginal likelihood – a measure of fit.
    `BVAR.InfoCrit`: contains the Akaike, `AIC`, the Hannan-Quinn `HQIC` and the Bayes `BIC`, information criteria,.

    Both options can used to select the lag length : log marginal likelihood measures one step ahead predictive ability; other criteria measure in-sample fir.

- By default, `K=5000` (change it with `options.K = number`) and a constant is assumed (change it with `options.noconstant = 1`).

Outline

Priors

Three types of priors are allowed:

- Uninformative or Jeffrey priors (default, no further instruction needed).

```
[BVAR] = bvar_(y, lags, options)
```

- Minnesota prior with default hyper-parameter values.

```
options.priors.name = ‘Minnesota’;
[BVAR] = bvar_(y, lags, options)
```

- Multivariate-Normal Inverse-Wishart Conjugate with default values.

```
options.priors.name = ‘Conjugate’;
[BVAR] = bvar_(y, lags, options)
```

- If no other options are specified, default parameterizations are used.

Minnesota prior

- 5 scalar hyper-parameters control the shrinkage of the Minnesota prior:
  - τ `options.minn_prior_tau`: overall tightness (default 3). The larger is τ, the tighter is prior.
  - d `options.minn_prior_decay`: tightness on the lags greater than one (default 0.5). The larger is d, the faster is the lag decay.
  - λ `options.minn_prior_lambda`: the sum-of-coefficient prior tightness (default 5)
  - μ `options.minn_prior_mu`: co-persistence prior tightness (default 2)
  - ω `options.minn_prior_omega`: the covariance matrix tightness (default 2)

- Three ways to activate the Minnesota prior by defining the appropriate option
  1. default values:
     ```
     options.priors.name = 'Minnesota'
     ```
  2. cherry-picking values:
     ```
     options.minn_prior_tau=5;
     options.minn_prior_lambda=0.01;
     ```
  3. optimized values:
     ```
     options.max_minn_hyper = 1; % default for maximization
     ```

Minnesota prior with optimal hyper-param

- Various options can be set for the maximization step:
  - `options.index_est` is a row vector that selects the parameters to be optimized (default, `options.index_est=1:5`)
  - `options.lb (.ub)` set the lower (upper) bound for the optimization. It has the same size of `options.index_est`.
  - `options.max_compute` is a scalar selecting the maximization routine to be employed:
    - `options.max_compute = 1` uses the MATLAB `fminunc.m`
    - `options.max_compute = 2` uses the MATLAB `fmincon.m`
    - `options.max_compute = 3` uses the Chris Sims's `cminwel.m` (default)
    - `options.max_compute = 7` uses the MATLAB `fminsearch.m`
- Once the maximum is found, the posterior distribution is computed using the optimal hyper-parameter values. If optimization is unsuccessful, the posterior distribution is computed with default values (and a warning is issued).
- Tip: it is a good idea to max one parameter at time, starting the optimization from the values obtained in the previous step (bvar_max_hyper can be used to shortens the computational time. It does hyperparameter optimization, rather than full Gibbs sampling computations).

```
[hyperp,~,~] = bvar_max_hyper(x0,y,lags,options);
```

Conjugate MN-IW

- Multivariate Normal-Inverse Wishart conjugate prior can be activated using:

  $$\texttt{options.priors.name} = \texttt{'Conjugate'}$$

  Default values: $\Phi \sim N(0, 10\ I_{np+1})$ and $\Sigma \sim IW(I_n, n+1)$

  Useful options:

- `options.priors.Phi.mean` is a $(\mathtt{n} \times \mathtt{lags} + 1) \times \mathtt{n}$ matrix containing the prior means for the autoregressive parameters.

- `options.priors.Phi.cov` is a $(\mathtt{n} \times \mathtt{lags} + 1) \times (\mathtt{n} \times \mathtt{lags} + 1)$ matrix containing the prior covariance for the autoregressive parameters.

- `options.priors.Sigma.scale` is a $(\mathtt{n} \times \mathtt{n})$ matrix containing the prior scale of the covariance of the residuals (use it to adjust if variables have different units).

- `options.priors.Sigma.df` is a scalar defining the prior degrees of freedom.

VARX

- Can estimate Classical or Bayesian VAR with exogenous variables (X). To allow exogeneous variables need to specify it as an option

```
options.controls = x;
```

  where x is a (T $\times$ q) matrix containing the exogenous variables

- The first dimension of x is time and must coincide with the time dimension of y (or with the sum of the time dimension of y and the out-of-sample forecast horizon.)

- Lags of exogenous controls can be used and specified as additional columns in x.

- A *VARX* model can be estimated with Jeffrey priors (classical) or conjugate priors (Bayesian); Minnesota priors are not currently supported.

- To launch the estimation use the standard command:

```
BVAR  = bvar_(y,lags,options);
```

- Draws with the responses of the endogenous variables to shocks to the exogenous variables are stored in BVAR.irx_draws (a four dimensional matrix). if there is more than one exogenous variable, shocks are computed using a Cholesky decomposition.

Practice II

Sample codes are in `...\BVARtoolbox\v4.2\BVAR tutorial`.

- Calculation of the optimal lag length; Cholesky responses and variance decomposition of monetary policy shock (example_1_classical.m)
- Estimation with Minnesota prior with hyper-parameter optimization (example_2_minn.m)
- Effects of US interest rate shocks on the UK (example_6_VARX.m)

Outline

IRF

- For each draw of the posterior distribution, the toolbox computes the IRF for a given identification scheme.
- IRFs are stored in a four dimensional arrays of dimension ($\texttt{n} \times \texttt{hor} \times \texttt{n} \times \texttt{K}$) (variable, horizon, shock, draw).
- Flat, Minnesota, Conjugate MN-IW priors can be used.
- Default value for the horizon is 24; it can be changed with `options.hor = hor;` .
- By default Cholesky decomposition is used: `BVAR.ir_draws` is the default matrix containing IRF (no further command needed).
- Other identification schemes need to be specified by the user in the `options`, before launching `bvar`.

Plotting IRFs

- Plotting command for IRF is :

```
plot_irfs_(irfs_to_plot, optnsplt)
```

- `irfs_to_plot` is a four dimensional array; e.g. `irfs_to_plot = BVAR.ir_draws` or `irfs_to_plot = BVAR.irproxy_draws(:,:,1,:);`

- `optnsplt` is optional and defines plotting options. The most relevant are:
    - `optnsplt.varnames (.shocksnames)` is a cell string with variable (shock) names.
    - `optnsplt.conf_sig (.conf_sig_2)` is a number between 0 and 1 indicating the size of (second) highest posterior density (HPD) set to be plotted; the default is 0.68 for the first (no default for the second).
    - `optnsplt.saveas_strng (.saveas_dir)` a string array with name of the directory where to save the plot(s).
    - `optnsplt.add_irfs` allows to add other IRFs to the plot when one wants to compare responses.
    - `optnsplt.nplots` is a $\mathtt{n} \times \mathtt{m}$ array indicating the structure of the subplots.

Outline

Long run restrictions

- Activated using:

```
options.long_run_irf = 1;
```

- Need the first variable in `y` to be specified in log difference
- By default, the first disturbance has long run effects on the first variable.
- Basic setup follows Gali (1999).
- If more than one shock affects the variables in the long run, as in Fisher(2006), use more than one variable in first difference.
- IRFs are stored in the matrix `BVAR.irlr_draws`

Sign and Magnitude restrictions

- Activated using:

  `options.signs{1} ='y(a,b,c)>0'`

  where the array `'y'` refers to the IRF,`'a'` to the variable, `'b'` to the horizon, `'c'` to the shock.

- The syntax means that the shock `c` has a positive impact on variable `a` at horizon `b`.

- The syntax is flexible and allows several variations (e.g. lower bound on elasticity or threshold on the cumulative impact)

  ```
  options.signs{2} = 'y(a_1,1,c)/y(a_2,1,c)  >m'

  options.signs{3} = 'max(cumsum(y(a,:,c),2))<M'
  ```

- Implementation follows: Rubio-Ramirez, Waggoner and Zha (2010).

- IRFs are stored in the matrix `BVAR.irsign_draws`.

Narrative restrictions

- Activated using:

```
options.narrative{1} ='v(tau,n)>0'
```

  where the array `'v'` refers to the structural innovation, `'tau'` to time, and `'n'` to the shock.

- The syntax means that shock `n` is positive on the time periods `tau`

- Again the syntax is flexible and many variations are allowed:

```
options.narrative{1} = 'sum(v(tau_0:tau_1),n)>0'
```

  Here the sum of the shock `n` between periods `tau_0` and `tau_1` is positive.

- Implementation follows Ben Zeev (2018) and Antolin-Diaz and Rubio-Ramirez (2018)

- IRFs are stored in the matrix `BVAR.irnarrsign_draws`.

IV approach

- External instrumental or proxy variable estimation is activated using :

  ```
  options.proxy = instrument;
  ```

- Restriction 1: instrument vector cannot be longer than `(T - lags)`
- Restriction 2: Instrument ends when the VAR ends (default). When this is not the case, use the option

  ```
  options.proxy_end = periods;
  ```

  to line up instruments and data. `periods` is the number of periods separating the last observation of the instrument and the last observation of the VAR innovations.
- Multiple proxy variables are allowed to identify one structural shocks, i.e. `instrument` can be a vector. If one is interested in identifying, say, two disturbances, she need to repeat the IV exercise twice, separately for each disturbance (currently, no option to identify multiple structural shocks with multiple proxy variables).
- Implementation follows Miranda-Agrippino and Ricco (2020)
- IRFs are stored in the matrix `BVAR.irproxy_draws`
- By convention, the structural shock of interest is ordered **first**; i.e. `BVAR.irproxy_draws(:,:,1,:);`

Mixed identification restrictions

A mix of zero (short and long run) and sign restrictions can be used. This option is activated using a combination of previous options:

- `options.zero_signs{1} ='y(j,k)=+1'`
  Here shock `k` has a positive effect on the `j`-th variable **on impact**
- `options.zero_signs{2} ='ys(j,k_1)=0'`
  Here shock `k_1` has a zero impact effect on the `j`-th variable.
- `options.zero_signs{3} ='yl(j,k_2)=0'`
  Here shock `k_2` has a zero long run effect on the `j`-th variable.
- **Pay attention: short and long run restrictions are distinguished by the addition of `r` or `s` to the y vector.**
- Implementation follows Arias, Rubio-Ramirez, Waggoner and Zha (2018) and Binning (2013).
- IRFs are stored in the matrix `BVAR.irzerosign_draws`

Outline

Other summary functions

- The forecast error variance decomposition is activated using:

```
FEVD = fevd(hor,Phi,Sigma,Omega);
```

  `FEVD` is a `n` $\times$ `n` array; the `(i,j)` element the share of variance of variable `i` explained by shock `j` at horizon `h`. `Omega` is the rotation matrix used (omit, if Cholesky decomposition is empoyed).

- The historical decomposition is activated using:

```
[yDecomp,v] = histdecomp(BVAR,opts);
```

  where `yDecomp` is a `T` $\times$ `n` $\times$ `n+1` array (time, variable, shock and initial condition $\overline{\Psi}_t$); `v` is the `T` $\times$ `n` array of structural innovations; `BVAR` is the output of the `bvar.m` function; `opts.Omega` declares a different rotation/identification (omit, if Cholesky decomposition is employed).

- To plot the historical decomposition use:

```
plot_shcks_dcmp_(yDecomp, BVAR, optnsplt)
```

  `optnsplt` is optional and controls various plotting options (see manual)

Additional identifications: Maximize FEVD at horizon h

- Given a draw (`Phi, Sigma`) of the reduced form VAR parameters, one can use the command:

```
Qbar  = max_fevd(i, hor, j, Phi, Sigma)
```

  to find the orthonormal rotation maximizing the forecast error variance decomposition (FEVD) of variable `i` explained by shock `j` at horizon `hor`.

- Given the selected `Qbar`, the IRF can be computed using

```
y_fevd = iresponse(Phi, Sigma, hor, Q_bar)
```

- Should be looped over draws of (`Phi,Sigma`)

- Useful functions:
  `Q = generateQ(n)` a generic rotation matrix generator.
  `y = iresponse(Phi, Sigma, hor, Q)` impulse responses for a generic rotation Q.

Outline

Time variations

- No direct command to estimate parametric time varying BVARs. Can however do non-parameteric (fixed rolling window) estimation to check for time variations in the statistics of interest. See example 16 in the manual (page 49), or `example_3_IRF.m` in the tutorials.

- No direct command to control for time varying volatility. One can however estimate the model correcting for abnormal volatility movements (like those produced by COVID19) by playing around with the available options. See example 17 in the manual (page 52) or `example_10_VAR_heterosked.m` in the tutorials.

Practice III

Sample codes are in `...\BVARtoolbox\v4.2\BVAR tutorial`.

- Calculation IRFs with various identification schemes. Computation of variance and historical decompositions (example_3_irfs.m).
- Estimation of a TVC BVAR (example_3_irfs.m).
- Estimation of a BVAR with volatility changes (example_10_VAR_heterosked.m) .

Outline

Larger scale models

- For large scale VARs, the best option is a Minnesota prior with larger value of τ (overall tightness) than the default, to a priori set many coefficients close to zero.

- The factor structure of Canova and Ciccarelli (2009)-(2013) is not implemented here. See the BEAR toolbox, if you want to use it.

- To compress vector time series into static principal components, use

  ```
  [E, W, Lambda, EigenVal, STD] = pc_T(y_1, n_w, transf);
  ```

  `y_1` contains the data to be compressed,a Txn vector, `n_w` is a scalar, indicating the number of factors to be extracted, and `transf` indicates the data transformation: 0 means no transformation, 1 means demean, and 2 means demean and standardize. The ouptuts `E, W, Lambda, EIgenVal` are, respectively, the idiosyncratic errors, the principal components, the loadings and the eigenvalues. When `transf=2`, `STD` is the $n_2 \times 1$ vector containing the standard deviation of `y_2`; otherwise, it is a vector of ones.

- Alternatively to compress time series (with similar units and similar standard deviations) compute a cross sectional mean at each `t`:

  `y_2=mean(y1,2,'omitnan')` *or* `y_2=trimmean(y_1, pct,'round',2)`

  where `pct` is the percentage of observations eliminated.

FAVAR models

- Once y_1 or y_2 are computed, they can be added to the VAR as factors using

```
FAVAR = bvar([W y_3], lags)        FAVAR = bvar([y_2 y_3], lags)
```

  where W are the principal components and y_3 variable excluded from y_1.

- Any prior specification and any identification restriction works.

- When demeaned standardized principal components are used statistics for the original variables can be computed using two commands jointly:

```
ReScale = rescaleFAVAR(STD,Lambda,n_1)
```

  where ReScale is a (n_2+n_1) $\times$ (n_w + n_1) matrix. By default, factors loadings are ordered first. This rescales the FAVAR coefficient estimates.
  Then for any statistics, take the output of FAVAR and transform it with ReScale. For example., assuming interest is in the mean forecast of y_2, type

```
mean_forecast_y = mean(fabvar.forecasts.no_shocks,3) * ReScale';
```

Outline

Panels of VARs

No specific command but many options are allowed with available technology

- Exact pooling of cross sectional data with fixed effects (see example 22, page 69 of the manual).
- Average time series estimator along the lines of Pesaran and Smith (1996) (see example 21, page 66 of the manual).
- Partial (Bayesian) pooling with a data driven (or used supplied) prior mean and variance (see example 23, page 71 of the manual).
- The implicit assumption is that units are treated as 'islands'. No contemporaneous or lagged spillovers across units are assumed.
- Can use `VARX` technology to account for common (areawide) fluctuations or can scale unit specific variables by aggregate variables.

Outline

Mixed Frequency VAR

- Mixed Frequency VAR (MF-VAR) can be used to nowcasts, backcast or to retrieve variables that have an arbitrary pattern of missing observations.
- Examples: Monthly GDP, Weekly Unemployment.
- Estimation is performed with the Gibbs sampler:
  1. Generate a draw from the posterior distribution of the reduced form VAR parameters, conditional on observables variables and the states.
  2. With this draw run the Kalman filter and the Kalman smoother and get estimate the unobserved states (monthly GDP).
  3. Repeat 1. and 2 for each draw.
- It allows flat, Minnesota (except maximization), Conjugate MN-IW priors. It allows all identification schemes for IRFs. It allows all forecasting options.
- MF-VAR estimation is automatically triggered in bvar.m whenever there are `NaN` in the data, `y`. (A warning is printed on the screen).

Example: VAR with Monthly-Quarterly data

- Suppose the time unit is a month.
- Pool together the data in $y = [y^m, y^q]$; here the y^q variable is observed only in the last month of the quarter q, elsewhere is `NaN`. Define the unobserved states as x_t.
- State space representation where the transition equation is a monthly VAR for the x_t and measurement equation maps x_t into y_t.
    - For bf Monthly variables, the observation equation is: $y^m_t = x_{m,t}$
    - For **Stock** quarterly variables, the observation equation is : $y^q_t = x_{q,t}$. [no instruction is needed in this case]
    - For **Flow** quarterly variables (e.g. GDP), observation equation is: $y^q_t = \frac{1}{3}(x_{q,t} + x_{q,t-1} + x_{q,t-2})$. In this case one needs to specify the option:

      ```
      options.mf_varindex     = num;
      ```

      where `num` is a scalar indicating the position of the q flow variable (column number in `y`)
- Additional outputs produced by mixed frequency estimation: `BVAR.yfill` (smoothed), `BVAR.yfilt` (filtered) $\mathtt{T} \times \mathtt{n} \times \mathtt{K}$ arrays containing etimates of the latent variables.

Practice IV

Sample codes are in `...\BVARtoolbox\v4.2\BVAR tutorial`.

- Estimation and IRFs in a PCVAR (example_5_favar.m) .
- Calculation of average IRFs in a panel of VARs (example_8_panels.m).
- Estimation and inference in a mixed frequency BVAR (example_4_mfvar.m).

Outline

Forecasts

Forecasts are automatically computed with BVAR. They are stored as follows:

- BVAR.forecasts.no_shocks is a (nfor $\times$ n $\times$ K) matrix (time, variable, draw). All future shocks are zeros (this are unconditional forecasts).
- BVAR.forecasts.with_shocks is a (nfor $\times$ n $\times$ K) matrix (time, variable, draw). All future shocks are drawn randomly from the posterior distribution.
- BVAR.forecasts.conditional is a (nfor $\times$ n $\times$ K) matrix containing the conditional forecasts. Need to set a bunch of options:
  - options.endo_index is a row array containing the index of the variable constrained to a specified path.
  - options.endo_path is a matrix containing the path for each variable (rows horizon, column variables). size(options.endo_path,1) = options.fhor.
  - options.exo_index [optional] specifies the shocks of the VAR used to generate the assumed paths of the endogenous variables. exo_index could be one or more shocks. If no structural identification is performed, by default, the command uses a Cholesky factorization.
- BVAR.forecasts.EPS contains the shocks used to generate the conditional forecasts.
- Can use options.fhor to change the forecasting horizon

Plotting the Forecasts

- The command to plot forecast is:

```
plot_frcst_(frcst_to_plot,y,T,optnsplt)
```

- `frcst_to_plot` is a three dimensional array (time,variable,draw) containing the forecast; e.g. `frcst_to_plot = BVAR.forecasts.no_shocks`.

- `y` is the (`T` $\times$ `n`) array of data

- `T` is the (`T` $\times$ `1`) array with the in-sample time span; e.g. for quarterly data, `T= 1990 : 0.25 : 2010.25` is the sample 1990Q1-2010Q2.

- `optnsplt` can be omitted and defines a number of options. Many options similar to the ones of `plot_irf_`. A new useful one is:

- `options.order_transform` is a 1 $\times$ `n` array with values
  - =0 $\rightarrow$ no transformation performed.
  - =1 $\rightarrow$ period-by-period change
  - =12 $\rightarrow$ 12 periods change (monthly data) multiplied by 100, i.e. $100(y_{t+12} - y_t)$.
  - =4 $\rightarrow$ 4 periods change (quarterly data) multiplied by 100, i.e. $100(y_{t+3} - y_t)$.
  - =100 $\rightarrow$ one period change (annual data) multiplied by 100.

Forecasting in times of pandemics

- Are VAR (time series) models useful to forecast during the pandemic?

- Schorfheide and Song (2020): MF-VAR can be used to generate real-time macroeconomic forecasts for the U.S. during the COVID-19 pandemic. Quite pessimistic outlook: long-lasting recession.

- Primiceri and Lenza (2020): Heteroskedasticity adjusted VAR. Scale down the observables during the peak of the COVID-19 pandemic. Seems a sensible idea.

$$y_t = \mathbf{x}_t' \Phi + w_t u_t$$

  w_t is a vector of weights. Weights can be picked by the investigator:

- `options.heterosked_weights` is the (`T - lags` $\times$ 1) array of weights; e.g. if we are interested in scaling sample 2000m2 to 2020m7, use:

```
options.heterosked_weights =  [1 .... 1 20 30 30 20 20 10 1 .....1];
```

- They can also be optimized (as Minnesota hyper-parameters) by defining

```
options.objective_function = 'bvar_opt_heterosked';
options.tstar              = find(time==2020) + 2;  %picks March 2020
[postmode,logmlike,HH]     = bvar_max_hyper(hyperpara,y,lags,options);
```

Practice V

Sample codes are in `...\BVARtoolbox\v4.2\BVAR tutorial`.

- Forecast in a BVAR (example_9_prediction.m).
- Forecast in time of a pandemic (example_10_VAR_heterosked.m).

Outline

Classical Local projections I

- The baseline estimation function for local projections is:

  `[DM] = directmethods(y, lags, options)`

  The first input, `y`, is the data (no missing values are allowed); the second , `lags`, is the number of lags of `y` used in the projection; the third, `options`, specifies the options; it can be omitted, if default options are used. Avalable options are:

  - `options.hor` is a scalar indicating the horizon of the responses (default 24).
  - `options.conf_sig` is a number between 0 and 1, with the size of the confidence interval (default 0.9).
  - `options.controls` is a $(\mathtt{T} \times \mathtt{n\_c})$ array of `n_c` exogenous controls and the first dimension of controls must coincide with the first dimension of `y`.
  - `options.proxy` is a $(\mathtt{T} \times \mathtt{n\_p})$ array containing `n_p` variables that proxy for the shock of interest. The first dimension must coincide with the first dimension of the endogenous variables, `y`.
  - `options.robust_se_` a scalar indicating the type of standard errors to be computed; if 0, no adjustment ismade; if 1 standard errors are HAC adjusted as in Hamilton (1994), Ch 10, pag 282, (default); if 5, the MATLAB `hac.m` function is used (the Matlab Econ Toolbox required).
  - `options.Q` is a $(\mathtt{n} \times \mathtt{n})$ orthonormal matrix, $Q'Q = QQ' = I$, that can be used to rotate the Cholesky impact matrix. The toolbox assumes that $\Omega = chol(\Sigma_{(0)})Q$, and the default is $Q = I$.

Classical Local projections II

The output of the function, `DM`, is a structure with several fields and sub-fields.The two most important ones are:

- `DM.ir_lp` is a $(\mathtt{n} \times \mathtt{hor} \times \mathtt{n} \times 3)$ array containing the impulse response function computed with a recursive identification scheme. The first dimension corresponds the endogenous variable, the second to the horizon, the third to the shock. The first and third elements in the fourth dimension are to the upper and lower limits of the confidence interval, whereas the second dimension is the mean.
- `DM.irproxy_lp` a $(\mathtt{n} \times \mathtt{hor} \times 1 \times 3)$ array containing the impulse response function computed with a proxy shock. The dimensions are the same as above.

Direct forecasts are also stored in `DM`; in particular:

```
DM.forecasts
```

is a $(\mathtt{n} \times \mathtt{hor} \times 3)$ array containing the forecasts. The first dimension corresponds the endogenous variable, the second to the horizon. The first and third elements in the third dimension correspond to the upper and lower limits of the confidence interval, whereas the second element of the third dimension is the mean.

Bayesian Local projections I

The toolkit allows Bayesian local projections with a Normal-Inverted Wishart prior. To activate it use

- `options.priors.name = 'Conjugate'`

  With this setting, the prior for AR parameters has zero mean zero and diagonal covariance matrix of 10 and $\tau_h = 1$; the prior for the covariance matrix of the residual is inverse Wishart with a unitary diagonal matrix as scale and `n+1` degrees of freedom. Customization of prior parameters can be done as follows:

  - `options.priors.Phi.mean` is a $(\mathtt{n} \times \mathtt{lags} + 1) \times \mathtt{n}$ matrix containing the prior means for the autoregressive parameters.
  - `options.priors.Phi.cov` is a $(\mathtt{n} \times \mathtt{lags} + 1) \times (\mathtt{n} \times \mathtt{lags} + 1)$ matrix containing the prior covariance for the autoregressive parameters.
  - `options.priors.Sigma.scale` is a $(\mathtt{n} \times \mathtt{n})$ matrix containing the prior scale for the covariance of the residuals.
  - `options.priors.Sigma.df` is a scalar defining the prior degrees of freedom.
  - `options.priors.tau` is a $(\mathtt{hor} \times 1)$ vector controlling shrinkage at various horizons; the default is $\tau = 1$. The larger is τ, the larger is the shrinkage around the prior mean.

Bayesian Local projections II

The option `options.priors.tau`, can also be chosen to maximize the marginal data density using:

```
options.priors.max_tau = 1;
bdm_opt = directmethods(y, lags, options);
```

By default, optimization is performed unconstrained and `cminwel.m` is used (this is `options.max_compute = 3`). The following options can be set in the maximization step:

- `options.lb` and `options.ub` set the lower and upper bounds for the optimization. Both are row array vectors of the same size of `options.priors.tau`.
- `options.max_compute` is a scalar, selecting the maximization routine to be employed.

  The options are:
  - `options.max_compute = 1` uses the MATLAB `fminunc.m` (unconstrained)
  - `options.max_compute = 2` uses the MATLAB `fmincon.m` (constrained)
  - `options.max_compute = 3` uses the Chris Sims's `cminwel.m`
  - `options.max_compute = 7` uses the MATLAB `fminsearch.m`

  The first three are derivative-based algorithms; the latter is a direct search (simplex) method based on function comparisons. While typically slower, it is useful in situations where derivatives are not well behaved.

Bayesian Local projections III

The output of the DM function is a field with a number of sub-fields. The most relevant ones are

- dm.ir_blp is a (n $\times$ hor $\times$ n $\times$ K) matrix containing Bayesian local projections impulse responses obtained obatiend with recursive identification. The first dimension corresponds to the variables, the second to the horizon, the third to the disturbances, and the fourth to the responses obtained with particular draw from the posterior distribution of the local projection parameters.

- dm.irproxy_blp is a (n $\times$ hor $\times$ 1 $\times$ K) matrix containing Bayesian local projection impulse responses obtained with IV identification. The first dimension corresponds to the variables, the second to the horizon, the third to the shock, and the fourth to the responses obtained with particular draw from the posterior distribution of the local projection parameters.

- dm.bforecasts.no_shocks is a (hor $\times$ n $\times$ K) matrix containing unconditional forecasts. The first dimension corresponds the horizon, the second to the variable, and the third to the draw from the posterior distribution of the parameters. These trajectories are constructed assuming that all future shocks are zeros (forecasts only account for parameter uncertainty).

- dm.bforecasts.with_shocks is a (hor $\times$ n $\times$ K) matrix containing unconditional forecasts with shocks. The first dimension corresponds the horizon, the second to the variable, and the third to the draw from the posterior distribution of the parameters. These trajectories account for uncertainty in the parameter estimates and in the shocks realization.

Practice VI

Sample codes are in `...\BVARtoolbox\v4.2\BVAR tutorial`.

- Bayesian and Classical local projection (example_7_LP.m) .
- Forecast using local projections (example_7_LP.m).

Outline

Some notation I

- *VAR(p)*: A vector of autoregression with p lags:

$$y_t = \Phi_1 y_{t-1} + ... + \Phi_p y_{t-p} + \Phi_0 + u_t \quad u_t \sim N(0, \Sigma) \tag{1}$$

  where y_t is $n \times 1$ vector, Φ_j suitable matrices

- SURE: A VAR(p) be rewritten as a SURE

$$Y = X\Phi + E$$

- Forecasts and IRF: Any VAR(p) can be expressed as a VAR(1) (companion form)

$$x_t = Fx_{t-1} + F_0 + Gu_t$$

- Decompositions: Under some restrictions a VAR(p) can be written as a Vector MA (VMA(∞))

$$y_t = u_t + \Psi_1 u_{t-1} + ... + \Psi_t u_1 + \overline{\Psi}_t$$

  where Ψ_j for $j = 1, ..., t$ are functions of $(\Phi_1, ..., \Phi_p)$ and $\overline{\Psi}_t$ is deterministic.

Some notation II

- $LP(h, d, q)$: Local projection at horizon h is given by

$$y_{it+h} = \Phi_1 y_{it-1} + ... + \Phi_p y_{it-d} + \Lambda_1 y_{jt-1} + ... + \Lambda_q y_{jt-q} + \Phi_0 + \zeta v_t + u_{it+h} \quad (2)$$

  where y_{it} is scalar $i \neq j$, Φ_j, Λ_j are suitable matrices, v_t is a shock of interest, $u_{it+h} \sim iid(0, \Sigma)$
  Under certain conditions: $\Psi_j = \zeta \Phi_j$.

- Trend and Cycle decompositions: Any observable scalar (vector) y_t be decomposed into two latent components

$$y_t = \tau_t + c_t$$

  with suitable restrictions for τ_t, c_t or both.

Empirical and structural innovations

- Reduced-form VAR innovations u_t and structural disturbances ν_t

$$u_t = \Omega \ \nu_t = \Omega_0 \ Q \ \nu_t$$

- $E(\nu_t \nu_t') = I$ and $\Omega\Omega' = \Sigma$,
- Ω_0 is the Cholesky decompostion of Σ and Q is an orthonormal rotation such that $Q'Q = QQ' = I_n$.
- To recover ν_t, we need to impose restrictions on Ω.
- This is because Σ only contains $n(n+1)/2$ estimated elements, while Ω has n^2 elements.