Quantitative Macroeconomics

Winter 2023/24

Week 7

Willi Mutschler
willi@mutschler.eu

Version: 1.0
Latest version available on: GitHub

Contents

1. Ordinary Least Squares Estimation of VAR(p)

Consider the VAR(p) model with a constant written in the compact form

$$y_t = [c, A_1, \dots, A_p]Z_{t-1} + u_t = AZ_{t-1} + u_t$$

where $Z_{t-1} = (1, y'_{t-1}, \dots, y'_{t-p})'$ and u_t is assumed to be iid white noise with non-singular covariance matrix Σ_u. Given a sample of size T, $y_1, \dots, y_T$, and p presample vectors, $y_{-p+1}, \dots, y_0$, ordinary least squares for each equation separately results in efficient estimators. The OLS estimator is

$$\hat{A} = \left[\hat{c}, \hat{A}_1, \dots, \hat{A}_p\right] = \left(\sum_{t=1}^{T} y_t Z'_{t-1}\right)\left(\sum_{t=1}^{T} Z_{t-1}Z'_{t-1}\right)^{-1} = YZ'(ZZ')^{-1}$$

where $Y = [y_1, \dots, y_T]$ and $Z = [Z_0, \dots, Z_{T-1}]$. More precisely, stacking the columns of $A = [c, A_1, \dots, A_p]$ in the vector $\alpha = vec(A)$,

$$\sqrt{T}\,(\hat{\alpha} - \alpha) \xrightarrow{d} \mathcal{N}(0, \Sigma_{\hat{\alpha}})$$

where $\Sigma_{\hat{\alpha}} = plim(\frac{1}{T}ZZ')^{-1} \otimes \Sigma_u$, if the process is stable. Under fairly general assumptions this estimator has an asymptotic normal distribution. A sufficient condition for the consistency and asymptotic normality of $\hat{A}$ would be that u_t is a continuous iid random variable with four finite moments. A consistent estimator of the innovation covariance matrix Σ_u is, for example,

$$\hat{\Sigma}_u = \frac{\hat{U}\hat{U}'}{T - Kp - 1}$$

where $\hat{U} = Y - \hat{A}Z$ are the OLS residuals. Thus, in large samples,

$$vec(\hat{A}) \overset{a}{\sim} \mathcal{N}(vec(A), (ZZ')^{-1} \otimes \hat{\Sigma}_u)$$

where $\overset{a}{\sim}$ denotes the approximate large-sample distribution. In other words, asymptotically the usual t-statistics can be used for testing restrictions on individual coefficients and for setting up confidence intervals.

1. What are the dimensions of y_t, Y, u_t, U, c, A_1, $\dots$, A_p, A, α, Z_{t-1}, Z, Σ_u and $\Sigma_{\hat{\alpha}}$.
2. Modify your `ARpOLS` function such that it is able to estimate VAR(p) models. Save the modified function as `VARReducedForm`.
3. Consider data given in `threeVariableVAR.csv` for $y_t = (\Delta gnp_t, i_t, \Delta p_t)'$, where gnp_t denotes the log of U.S. real GNP, p_t the corresponding GNP deflator in logs, and i_t the federal funds rate, averaged by quarter. The estimation period is restricted to 1954q4 to 2007q4.
   - Load the data and visualize it. Comment whether you think the data looks stationary.
   - Estimate a VAR(4) model using the `VARReducedForm` function. Examine the stability of the estimated process and the significance of the estimated parameters at a 95% level.

Readings

- Kilian and Lütkepohl (2017, Ch. 2.3)

2. Maximum Likelihood Estimation of VAR(p)

Consider the VAR(p) model with constant written in the more compact form

$$y_t = [c, A_1, \ldots, A_p]Z_{t-1} + u_t = AZ_{t-1} + u_t$$

where $Z_{t-1} = (1, y'_{t-1}, \ldots, y'_{t-p})'$. In VAR analysis, it is common to postulate that the innovations, u_t, are iid $\mathcal{N}(0, \Sigma_u)$ random variables. This assumption implies that the y_t's are also jointly normal and, for given initial values $y_{-p+1}, \ldots, y_0$,

$$f_t(y_t|y_{t-1}, \ldots, y_{-p+1}) = \left(\frac{1}{2\pi}\right)^{K/2} \det(\Sigma_u)^{-1/2} \exp\left\{-\frac{1}{2}u'_t\Sigma_u^{-1}u_t\right\}$$

Conditional on the first p observations, the conditional log-likelihood becomes:

$$\log l = -\frac{KT}{2}\log(2\pi) - \frac{T}{2}\log(\det(\Sigma_u)) - \frac{1}{2}\sum_{t=1}^{T} u'_t\Sigma_u^{-1}u_t$$

Maximizing this function with respect to the unknown parameters yields the Gaussian ML estimators $\widetilde{A}$ and $\widetilde{\Sigma_u}$.

1. Compare the Gaussian ML estimator $\widetilde{A}$ with the OLS estimator $\widehat{A}$ from the previous exercise. Comment on the asymptotic distribution.
2. Provide an expression for the ML estimator $\widetilde{\Sigma_u}$ of the innovation covariance matrix.
3. Consider data given in `threeVariableVAR.csv` for $y_t = (\Delta gnp_t, i_t, \Delta p_t)'$, where gnp_t denotes the log of U.S. real GNP, p_t the corresponding GNP deflator in logs, and i_t the federal funds rate, averaged by quarter. The estimation period is restricted to 1954q4 to 2007q4.
   - Estimate the parameters with Maximum Likelihood.
   - Compare your estimation results to an OLS estimation.

Readings

- Kilian and Lütkepohl (2017, Ch. 2.3)

3. Identification Problem in Structural Vector Autoregressive Models

Consider a simple 2-variable model:

$$\begin{aligned} i_t &= \beta \pi_t + \gamma_1 i_{t-1} + \gamma_2 \pi_{t-1} + \varepsilon_t^{MP} \\ \pi_t &= \delta i_t + \gamma_3 i_{t-1} + \gamma_4 \pi_{t-1} + \varepsilon_t^{\pi} \end{aligned}$$

where i_t denotes the interest rate set by the central bank and π_t the inflation rate. Assume for the **structural shocks**: $\varepsilon_t = (\varepsilon_t^{MP}, \varepsilon_t^{\pi})' \sim N(0, \Sigma_\varepsilon)$.

1. Rewrite the model in a compact matrix form $B_0 y_t = B_1 y_{t-1} + \varepsilon_t$. Note that this is a structural VAR(1) model.
2. Since the structural VAR model is not directly observable, derive the reduced-form representation: $y_t = A_1 y_{t-1} + u_t$. What is the relationship between structural shocks ε_t and reduced-form residuals u_t?
3. In your own words, explain the identification problem in SVAR models. Provide intuition behind the popular identification assumptions of short-run, long-run and sign restrictions.

Readings

- Kilian and Lütkepohl (2017, Ch. 7.6)

References

Kilian, Lutz and Helmut Lütkepohl (2017). *Structural Vector Autoregressive Analysis*. Themes in Modern Econometrics. Cambridge: Cambridge University Press. ISBN: 978-1-107-19657-5. URL: https://doi.org/10.1017/9781108164818.

A. Solutions

1 Solution to Ordinary Least Squares Estimation of VAR(p)

1. The dimensions are: y_t is $K \times 1$, Y is $K \times T$, u_t is $K \times 1$, U is $K \times T$, c is $K \times 1$, A_1 is $K \times K$, $\dots$, A_p is $K \times K$, A is $K \times (1 + pK)$, α is $(pK^2 + K) \times 1$, Z_{t-1} is $(1 + pK) \times 1$, Z is $(Kp + 1) \times T$, Σ_u is $K \times K$ and $\Sigma_{\hat{\alpha}}$ is $(pK^2 + K) \times (pK^2 + K)$.

progs/matlab/VARpDimensionsIllustration.m

```matlab
% -----------------------------------------------------------------------
% Illustrate dimensions of VAR(p) model using MATLAB's symbolic toolbox
% -----------------------------------------------------------------------
% Willi Mutschler, November 29, 2022
% willi@mutschler.eu
% -----------------------------------------------------------------------
clearvars; clc; close all;

T = 10; % time periods
K = 3;  % number of variables
p = 2;  % number of lags

data = transpose(sym('y', [K T],'real')) % e.g. y2_4 denotes the second variable at t=4

%% y_{t} is [Kx1]
y_1  = data(1,:)'
y_2  = data(2,:)'
y_3  = data(3,:)'
y_4  = data(4,:)'
y_5  = data(5,:)'
y_6  = data(6,:)'
y_7  = data(7,:)'
y_8  = data(8,:)'
y_9  = data(9,:)'
y_10 = data(10,:)'

%% Z_{t-1} is [(1+K*p)x1]; note that we start in t=p not in t=0
Z_2 = [1 y_2' y_1']'
Z_3 = [1 y_3' y_2']'
Z_4 = [1 y_4' y_3']'
Z_5 = [1 y_5' y_4']'
Z_6 = [1 y_6' y_5']'
Z_7 = [1 y_7' y_6']'
Z_8 = [1 y_8' y_7']'
Z_9 = [1 y_9' y_8']'


%% Y = [y_{p+1},..., y_{T}] is [Kx(T-p)]; note that we need to start in t=p+1 not in t=1
% manually by hand
Y_hand = [y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_10]
% more general
Y = transpose(data(p+1:T,:))
% check whether both are equal
```

```
isequal(Y,Y_hand)
isequal(size(Y),[K (T-p)])

%% Z = [Z_{p} Z_{p+1} ... Z_{T-1}] is [(1+K*p)x(T-p)]
% manually by hand
Z_hand = [Z_2 Z_3 Z_4 Z_5 Z_6 Z_7 Z_8 Z_9]
% more general
Z = sym(nan(p*K,T));
for ii=1:p
    Z( (K*(ii-1)+(1:K)) , (1+ii):T ) = data(1:T-ii,:)'; % this is basically what transpose(lagmatrix(data,[1:p])) does!
                                                          % note that lagmatrix.m does not work on symbolic variables
                                                          % unless you change line 85 of lagmatrix.m with
                                                          % "YLag = sym(missingValue(ones(numObs,numSeries*numLags)));
end
Z = [ones(1,T-p); Z(:,p+1:T)] % add deterministic term
% check whether both are equal
isequal(Z,Z_hand)
isequal(size(Z),[(1+K*p) (T-p)])
```

2. A modified version looks like this. Note that it also includes OLS estimation of each equation in turn.

progs/matlab/VARReducedForm.m

```
function VAR = VARReducedForm(ENDO,nlag,opt)
% VAR = VARReducedForm(ENDO,nlag,opt)
% -------------------------------------------------------------------------
% Perform vector estimation with OLS and Gaussian-ML of a VAR(p) model:
%   y_t = [c d A_1 ... A_p] [1 t y_{t-1}' ... y_{t-p}']' + u(t) = A Z_{t-1} + u_t
% -------------------------------------------------------------------------
% INPUT
%       - ENDO: [nobs x nvar] matrix of endogenous variables, nobs is number of observations and nvar is number of variables
%       - nlag: [integer]     lag length
%       - opt:  [structure]   optional, with possible fields
%      * const      [flag]     0 no constant; 1 constant; 2 constant and linear trend
%      * dispestim [boolean]  1: display estimation results, 0: do not
%      * eqOLS      [boolean]  1: perform additional estimation for each equation in turn, 0: do not
% -------------------------------------------------------------------------
% OUTPUT
%   VAR: structure including VAR estimation results with the following fields:
%   * ENDO:      [nobs x nvar] matrix of endogenous variables
%   * nlag:      [integer] lag length
%   * opt:       [structure] options used in estimation
%   * Z:         [(opt.const+nvar*nlag)x(nobs-nlag)] matrix of regressors
```

```
%   * Y:          [nvar x (nobs-nlag)] matrix of lagged endogenous variables actually used in estimation
%   * A:          [nvar x (opt.const+nvar*nlag] matrix of estimated coefficients
%   * residuals: [(nobs-nlag) x 1] vector of residuals
%   * SigmaOLS:  [nvar x nvar] OLS estimate of covariance matrix of innovations u
%   * SigmaML:   [nvar x nvar] ML estimate of covariance matrix of innovations u
%   * Acomp      [nvar*nlag x nvar*nlag] matrix of companion VAR(1) form
%   * maxEig     [double] maximum absolute Eigenvalue of Acomp
%
%   Moreover, equation by equation OLS estimation results can be accessed
%   by the substructures VAR.eqj where j=1,...,nvar, i.e. VAR.eq1, VAR.eq2,...
%   with the following fields for equation j
%   *  beta: [opt.const+nvar*nlag x 1] double vector of regression coefficients
%   *  yhat: [(nobs-nlag) x 1] predicted values of endogenous variable
%   * resid: [(nobs-nlag) x 1] residuals
%   *  sige: [double] estimated standard error of error term
%   *  bstd: [opt.const+nvar*nlag x 1] estimated standard error of regression coefficients
%   *  bint: [opt.const+nvar*nlag x 2] confidence intervall for regression coefficients
%   * tstat: [opt.const+nvar*nlag x 1] t-statistic of regression coefficients
%   *  rsqr: [double] determination coefficient
%   *  rbar: [double] adjusted determination coefficient
%   *    dw: [double] Durbin-Watson statistic
%   *     y: [(nobs-nlag) x 1] endogenous variable used in estimation
%   *     x: [(nobs-nlag) x (opt.const+nvar*nlag)] exogenous variables used in estimation
%   *  nobs: [double] effective sample size used in estimation
%   *  nvar: [double] number of exogenous variables
% -------------------------------------------------------------------------
% CALLS
%   - OLSmodel.m: builtin function (see below) to robustly estimate regression models with ols
% -------------------------------------------------------------------------
% Willi Mutschler, November 29, 2022, willi@mutschler.eu
% Codes are based on
% - vare.m function of James P. LeSage
% - VARmodel.m function of Ambrogio Cesa-Bianchi
% -------------------------------------------------------------------------


%% Get some parameters and set defaults
if nargin < 2
    error('You need to specify the number of lags ''nlag''.');
end
if nlag < 1
    error('nlag needs to be positive');
end

% set default options
if nargin < 3
    opt.const = 1;
```

```
    opt.dispestim = true;
    opt.eqOLS = true;
end

if ~isfield(opt,'const')
    opt.const = 1;
else
    if ~ismember(opt.const,[0,1,2])
        error('''opt.const'' can only take values 0, 1, or 2');
    end
end

if ~isfield(opt,'dispestim')
    opt.dispestim = 1;
end

if ~isfield(opt,'eqOLS')
    opt.eqOLS = 1;
end

[nobs, nvar]  = size(ENDO);
% feasability check
if nobs < nvar
    error('The number of observations is smaller than the number of variables, you probably need to transpose the ''ENDO'' input.')
end
nobs_eff = nobs - nlag; % effective sample size used in estimation

%% create independent vector and lagged dependent matrix
% Y = [y_{nlag+1},..., y_{nobs}] is [nvarx(nobs-nlag)] matrix of lagged endogenous variables; note that we need to start in t=nlag+1 not in t=1
Y = transpose(ENDO((nlag+1):nobs,:));

% Z = [Z_{nlag} Z_{nlag+1} ... Z_{nobs-1}] is [(opt.const+nvar*nlag)x(nobs-nlag)] matrix of regressors
Z = transpose(lagmatrix(ENDO,[1:nlag]));
Z = Z(:,nlag+1:nobs); % remove initial observations
% add deterministic terms if any
if opt.const == 1
    Z = [ones(1,nobs_eff); Z];
elseif opt.const == 2
    Z=[ones(1,nobs_eff); (nlag+1):nobs; Z];
end


%% compute the matrix of coefficients and covariance matrix
A = (Y*Z')/(Z*Z'); % OLS and Gaussian ML estimate
U = Y-A*Z; % OLS and Gaussian ML residuals
UUt = U*U'; % sum of squared residuals
SIGOLSu = (1/(nobs_eff-nvar*nlag-opt.const))*UUt; % OLS: adjusted for number of estimated coefficients
SIGMLu = (1/nobs_eff)*UUt; % Gaussian ML: not adjusted for number of estimated
```

```
coefficients

% compute maximum absolute Eigenvalue of companion VAR(1) matrix to check for stability
Acomp = [A(:,1+opt.const:nvar*nlag+opt.const);
         eye(nvar*(nlag-1)) zeros(nvar*(nlag-1),nvar)];
maxEig = max(abs(eig(Acomp)));


%% OLS estimation equation by equation
if opt.eqOLS == 1
    for j=1:nvar
        y = Y(j,:)';
        x = Z';
        % put into structure
        aux = sprintf('eq%d',j); % this creates strings 'eq1' 'eq2' 'eq3' which you can use below, i.e. VAR.(aux) is then VAR.eq1, VAR.eq2, etc.
        VAR.(aux) = OLSmodel(y,x); %uses built-in function (see below)
    end
end

%% display estimation results
if opt.dispestim
    if opt.const == 0
        estimtable = table([]);
    elseif opt.const == 1
        nuhat = A(:,1);
        estimtable = table(nuhat);
    elseif opt.const == 2
        nuhat = A(:,1);
        timehat = A(:,2);
        estimtable = table(nuhat,timehat);
    end
    ntrend = size(estimtable,2);
    Ai = reshape(A(:,(1+ntrend):end),[nvar,nvar,nlag]);
    for ii = 1:nlag
        estimtable = [estimtable table(Ai(:,:,ii),'VariableNames', {sprintf('Ahat%d',ii)})];
    end
    disp(estimtable);
    disp([table(SIGOLSu) table(SIGMLu)]);
end

%% save into structure
VAR.ENDO = ENDO;
VAR.nlag = nlag;
VAR.opt = opt;
VAR.Z = Z;
VAR.Y = Y;
VAR.A = A;
VAR.residuals = U;
VAR.SigmaOLS = SIGOLSu;
```

```
VAR.SigmaML = SIGMLu; % Maximum Likelihood COV Matrix is not adjusted for # of estimated coefficients
VAR.Acomp = Acomp;
VAR.maxEig = maxEig;

%% OLSmodel.m
function OLS = OLSmodel(y,x,meth)
    % OLS = OLSmodel(y,x)
    % -----------------------------------------------------------------------
    % INPUT
    %   - y: dependent variable vector    (nobs x 1)
    %   - x: independent variables matrix (nobs x nvar)
    % -----------------------------------------------------------------------
    % OUPUT
    %   - OLS: structure including OLS estimation results
    % -----------------------------------------------------------------------
    % Based on OLSmodel.m from Ambrogio Cesa Bianchi and olse.m from James P.
    % LeSage and fn_ols.m from Tao Tzha (Dynare implemenation).
    if nargin < 3
        meth = 0; % use SVD decomposition, it is not the fastest but most robust to compute the inverse
    end
    signifVal = 0.05;
    [T, K] = size(x);
    %% compute inv(X'X)
    if meth == 0 % use SVD decomposition
        [u d v] = svd(x,0);
        vd = v.*(ones(size(v,2),1)*diag(d)');
        dinv = 1./diag(d);
        vdinv = v.*(ones(size(v,2),1)*dinv');
        xtxinv = vdinv*vdinv';
        uy = u'*y;
        xty = vd*uy;
        beta = xtxinv*xty;
        yhat =  u*uy;
    else
        if T < 10000 % use QR decomposition
            [~, r] = qr(x,0);
            xtxinv = (r'*r)\eye(K);
        else % use built-in functions
            xtxinv = (x'*x)\eye(K);
        end
        beta = xtxinv*(x'*y);
        yhat = x*beta;
    end
    resid = y - yhat;
    sigu = resid'*resid;
    sige = sigu/(T-K);
    tmp = (sige)*(diag(xtxinv));
    sigb = sqrt(tmp);
    tcrit = -tinv(signifVal/2,T);
    bint = [beta-tcrit.*sigb, beta+tcrit.*sigb];
```

```
    tsta = beta./(sigb);

    ym = y - mean(y);
    rsqr1 = sigu;
    rsqr2 = ym'*ym;
    rsqr = 1.0 - rsqr1/rsqr2;
    rsqr1 = rsqr1/(T-K);
    rsqr2 = rsqr2/(T-1.0);
    if rsqr2 ~= 0
        rbar = 1 - (rsqr1/rsqr2);
    else
        rbar = rsqr;
    end
    ediff = resid(2:T) - resid(1:T-1);
    dw = (ediff'*ediff)/sigu; % durbin-watson

    % put into output structure
    OLS.beta = beta;
    OLS.yhat = yhat;
    OLS.resid = resid;
    OLS.sige = sige;
    OLS.bstd = sigb;
    OLS.bint=bint;
    OLS.tstat = tsta;
    OLS.rsqr = rsqr;
    OLS.rbar = rbar;
    OLS.dw = dw;
    OLS.y = y;
    OLS.x = x;
    OLS.nobs = T;
    OLS.nvar = K;

end % OLSmodel end


end % main Function end
```

3. The OLS estimation of the three variables VAR model might look like this:

progs/matlab/threeVariableVAROLS.m

```
% ----------------------------------------------------------------------
% Visualize and estimate 3-equation VAR(4) model with OLS
% ----------------------------------------------------------------------
% Willi Mutschler, November 29, 2022
% willi@mutschler.eu
% ----------------------------------------------------------------------

clearvars; close all;

%% load data
threeVariableVAR = importdata('../../data/threeVariableVAR.csv');
y = threeVariableVAR.data;
varnames = {'Real GNP Growth' 'Federal Funds Rate' 'GNP Deflator Inflation'};
```

```matlab
subsample_start = datetime('1954Q4','InputFormat','yyyyQQQ');
subsample_end   = datetime('2007Q4','InputFormat','yyyyQQQ');
subsample       = transpose(subsample_start:calquarters(1):subsample_end);

%% plot data
for j=1:size(y,2)
    subplot(3,1,j);
    plot(subsample,y(:,j),'linewidth',2);
    title(varnames{j});
end

%% VAR(4) estimation with OLS
nlag = 4;
opt.const = 1;
VAR4 = VARReducedForm(y,nlag,opt);

%% check stability via maximum eigenvalue
VAR4.maxEig

%% check significance of coefficients via confidence intervals
VAR4.eq1.bint
VAR4.eq2.bint
VAR4.eq3.bint
```

The data for Federal Funds Rate as well as the GNP Deflator Inflation do seem to have some trend in it, but nothing serious.

2 Solution to Maximum Likelihood Estimation of VAR(p)

1. From the univariate case (and undergraduate econometrics), we know that both estimators are identical; hence, the asymptotic normal distribution holds as well.

2. Taking the derivative of the conditional log-likelihood function with respect to Σ_u yields:

$$\widetilde{\Sigma}_u = \frac{\hat{U}\hat{U}'}{T}$$

where $\hat{U}$ are both the ML and OLS residuals (as $\widetilde{A} = \widehat{A}$). Note that from previous exercises in the univariate case we have already seen that the only difference to the OLS estimator of Σ_u is given in the fact that for ML we don't correct the degrees of freedom, but simply divide by the **effective** sample size used in the estimation T.

3. See the previous exercise, as the `VARReducedForm` function also outputs the ML estimate of Σ_u:

progs/matlab/threeVariableVARML.m

```
% ------------------------------------------------------------------------
% Estimate 3-equation VAR(4) model with ML
% ------------------------------------------------------------------------
% Willi Mutschler, November 17, 2021
% willi@mutschler.eu
% ------------------------------------------------------------------------
clearvars; close all;
threeVariableVAR = importdata('../../data/threeVariableVAR.csv');
y = threeVariableVAR.data;
nlag = 4;
opt.const = 1;
VAR4 = VARReducedForm(y,nlag,opt);
% note the only difference between OLS and ML is in the estimate for Sigma_u
% VARReducedForm computes both for convenience
```

3 Solution to Identification Problem in Structural Vector Autoregressive Models

1. Rewrite the equations:

$$
\begin{aligned}
i_t - \beta \pi_t &= \gamma_1 i_{t-1} + \gamma_2 \pi_{t-1} + \varepsilon_t^{MP} \\
\pi_t - \delta i_t &= \gamma_3 i_{t-1} + \gamma_4 \pi_{t-1} + \varepsilon_t^{\pi}
\end{aligned}
$$

   or in matrix notation:

$$
\underbrace{\begin{pmatrix} 1 & -\beta \\ -\delta & 1 \end{pmatrix}}_{B_0} \underbrace{\begin{pmatrix} i_t \\ \pi_t \end{pmatrix}}_{y_t} = \underbrace{\begin{pmatrix} \gamma_1 & \gamma_2 \\ \gamma_3 & \gamma_4 \end{pmatrix}}_{B_1} \underbrace{\begin{pmatrix} i_{t-1} \\ \pi_{t-1} \end{pmatrix}}_{y_{t-1}} + \underbrace{\begin{pmatrix} \varepsilon_t^{MP} \\ \varepsilon_t^{\pi} \end{pmatrix}}_{\varepsilon_t}
$$

2. Pre-multiply both sides by B_0^{-1}:

$$
y_t = \underbrace{B_0^{-1} B_1}_{A_1} y_{t-1} + \underbrace{B_0^{-1} \varepsilon_t}_{u_t}
$$

   Note that the reduced-form innovations u_t are a composite of the underlying structural shocks ε_t:

$$
u_t = B_0^{-1} \varepsilon_t
$$

   The covariance matrices are related by:

$$
E[u_t u_t'] = \Sigma_u = B_0^{-1} \Sigma_\varepsilon B_0^{-1'} = B_0^{-1} B_0^{-1'}
$$

   Above, we make use of a normalization rule for $\Sigma_\varepsilon = I$. For the example above:

$$
B_0 = \begin{pmatrix} 1 & -\beta \\ -\delta & 1 \end{pmatrix}
$$

$$
B_0^{-1} = \frac{1}{\det(B_0)} \begin{pmatrix} 1 & \beta \\ \delta & 1 \end{pmatrix} \equiv \begin{pmatrix} a & b \\ c & d \end{pmatrix}
$$

   So the system of equations that relates reduced-form innovations to structural shocks is given by:

$$
\begin{aligned}
u_t^i &= a\varepsilon_t^{MP} + b\varepsilon_t^{\pi} \\
u_t^{\pi} &= c\varepsilon_t^{MP} + d\varepsilon_t^{\pi}
\end{aligned}
$$

   Each reduced-form shock is a **weighted average** of structural shocks, where a, b, c, d represents the amounts by which a particular structural shock contributes to the variation in each residual.

3. There is not enough information to solve this system of equations, because in B_0 we have 4 unknowns, but due to symmetry from $\Sigma_u = B_0^{-1} B_0^{-1'}$ we only have 3 elements in Σ_u: two variances and one covariance. More generally, the covariance structure leaves $K(K-1)/2$ degrees of freedom in specifying B_0^{-1} and hence further restrictions are needed to achieve identification.

   Some popular strategies:

   a) Recursive ordering of variables (aka orthogonalization): In the above example, we would set $b = 0$ to get a lower triangular B_0^{-1}. The *economics* behind this choice is based on *delay* assumptions, i.e. how long it takes for a variable to react to a certain shock. We can think of the structural shock in terms of the effect it exerts **contemporaneously** on the variable of interest: $\partial y_t = u_t = B_0^{-1} \varepsilon_t$, so we could write:

$$
\begin{pmatrix} i_t \\ \pi_t \end{pmatrix} = \begin{pmatrix} a & 0 \\ c & d \end{pmatrix} \begin{pmatrix} \varepsilon_t^{MP} \\ \varepsilon_t^{\pi} \end{pmatrix}
$$

   This lower triangular structure can be obtained by e.g. a Cholesky decomposition of Σ_u and yields **exact identification**. The order of variables, however, matters!

b) Short-run restrictions: Exclusion restrictions on the impact matrix B_0^{-1}, more flexible than orthogonalization.

c) Separating transitory from permanent components by assuming long-run structural relationships, i.e. on the long-run multiplier matrix $(I - A(L))^{-1}B_0^{-1}$.

d) Combination of short-run and long-run relationships.

e) Sign restrictions: Take the Cholesky decomposition which yields exact identification $\Sigma_u = B_0^{-1}B_0^{-1'} = PP'$. In this special case: $B_0{}^{-1} = P$, but this is just **ONE** possible solution. It is also possible to decompose $\Sigma_u = \tilde{P}\tilde{P}'$, where $\tilde{P} = PQ'$ and Q is an orthogonal rotation matrix: $Q'Q = QQ' = I$; that is, $\tilde{P}$ and P are **observationally equivalent**, because they both reproduce Σ_u. Q is called a rotation matrix because it allows us to *rotate* the initial Cholesky (recursive) matrix while maintaining the property that shocks are uncorrelated. Put differently, it helps us generate new weights! This is the basic idea of sign restrictions: Examine a large number of candidate impact matrices by repeatedly drawing at random from the set of orthogonal matrices Q. For each B_0^{-1} check whether the candidate impact matrix is compatible with the sign restrictions that characterize a certain structural shock. Then we construct the set of admissable models based on accepted draws.