Bayesian VAR modeling in MATLAB

Table of Contents

New features in Econometrics - MATLAB

Interactive workflows for multivariate models

The **Econometric Modeler** app enables you to test multiple time series for cointegration among the variables by performing the Engle-Granger cointegration test or the Johansen cointegration test.

After assessing the cointegrating relationships among the series, you can fit vector autoregression (VAR) models, including optional exogenous variables (VARX) or vector error-correction (VEC) models. As with univariate models, after working in the app, you can export variables to the MATLAB® workspace and generate code or a report summarizing your session. For more details, Econometric Modeler App Overview.

Bayesian VAR models

Econometrics Toolbox™ provides Bayesian vector autoregression (VAR) models for analyzing multivariate time series data from a Bayesian point of view. The `bayesvarm` function creates a Bayesian VAR model object that characterizes the joint prior distribution on the coefficients and innovations covariance matrix. The framework enables you to regularize coefficients by using the Minnesota prior [124]. This table lists supported prior models, ordered by increasing flexibility and complexity.

`normalbvarm`

- Normal model
- Normal coefficients; fixed innovations covariance matrix

`conjugatebvarm`

- Conjugate matrix-normal-inverse-Wishart model
- Dependent coefficients and innovations covariance matrix

`semiconjugatebvarm`

- Semiconjugate normal-inverse-Wishart model
- Independent coefficients and innovations covariance matrix

`diffusebvarm`

- Diffuse model
- Joint prior distribution is inversely proportional to the determinant of the innovations covariance matrix

Bayesian State Spate Models

A new Bayesian state-space models for analyzing multivariate time series data from a Bayesian point of view. The `bssm` function creates a Bayesian state-space model (`bssm` object) that characterizes the joint prior distribution on the state and observation equation parameters, including coefficient, state-disturbance-loading, and observation-innovation matrices.

The framework enables you to create a Bayesian state-space model in two ways:

- Convert a standard state-space model (`ssm` object) to a Bayesian state-space model by using the `ssm2bssm` function.
- For maximum flexibility, pass the following inputs to `bssm`: (1) A parameter-to-matrix mapping function that specifies the structure of the state-space model and determines the likelihood computed by the Kalman filter (2) A prior distribution of the parameters, a function representing the log prior or matrix of random draws from the distribution

After creating a Bayesian state-space model, you can combine the prior model with data to form and analyze the posterior distribution. The data likelihood is formed by assuming that the state disturbances and observation innovations are multivariate Gaussian random variables with a mean of 0. The posterior distribution does not have a closed form; the framework uses Metropolis-Hastings to sample from the posterior. Posterior distribution formation and analysis functions include:

- `estimate` – Estimate moments of the posterior distribution given the state-space model structure, data, and initial values for optimization.
- `simulate` – Draw random state-space model parameters for custom Monte Carlo estimation.
- `tune` – Compute proposal distributions for the Metropolis-Hasting sampler to facilitate Monte Carlo simulation.

BVAR models

Bayesian VAR (BVAR) models have the same mathematical form as any other **VAR model**, i.e.

$$y_t = c + \sum_{l=i}^{p} A_i y_{t-i} + \epsilon_t$$

where y_t is a $K \times 1$ vector of endogenous variables in period t, A_i is the cofficient matrix corresponding to the i^{th} lag of y_t, c is a constant deterministic term and ϵ is an error term with zero mean and variance-covariance Σ.

The only difference between usual VAR models and BVAR models is the way parameter estimates are obtained and interpreted. VAR models are usually estimated by OLS (ordinary least squares), which is a simple and computationally fast estimator. By contrast, Bayesian estimators are slightly more complicated and more burdensome in terms of algebra and calculation power. The coefficients obtained by so-called *frequentist* estimators like OLS are interpreted based on the concept of the **sampling distribution**. In Bayesian inference, the coefficients are assumed to have their own distribution. A more detailed treatment of the difference between frequentist and Bayesian inference can be found in Kennedy (2008, ch. 14), which provides a short introduction to the Bayesian approach and a series of references for interested readers. Koop and Korobilis (2010) provide a very good introduction to Bayesian VAR estimators.

Prior distributrion

$\pi(\mathbf{A})$: distribution of the parameters of the VAR model based on the observed frequency of a sample.

Posterior distribution

$\pi(\mathbf{A}|\mathbf{y})$: distribution of the parameters of the VAR model based on the observed frequency of the sample

Bayes theorem

$$\pi(A|\mathbf{y}) = \frac{\pi(\mathbf{y}|\mathbf{A})\pi(\mathbf{y})}{\pi(\mathbf{A})}$$

As already mentioned, Bayesian inference can be algebraically demanding. However, Bayesian estimators for linear VAR models can be implemented in a straightforward manner. A standard implementation is a so-called **Gibbs sampler**, which belongs to the family of Markov-Chain-Monte-Carlo (MCMC) methods. A detailed treatment of this method is beyond the scope of this document, but **Wikipedia** might be a good start to become familiar with it. Personally, I like to think of the Gibbs sampler as throwing a bunch of random numbers at a model and see what sticks. The remainer of this text provides the code to set up and estimate a basic BVAR model with MATLAB.

Monetary Policy in a Trivariate VAR

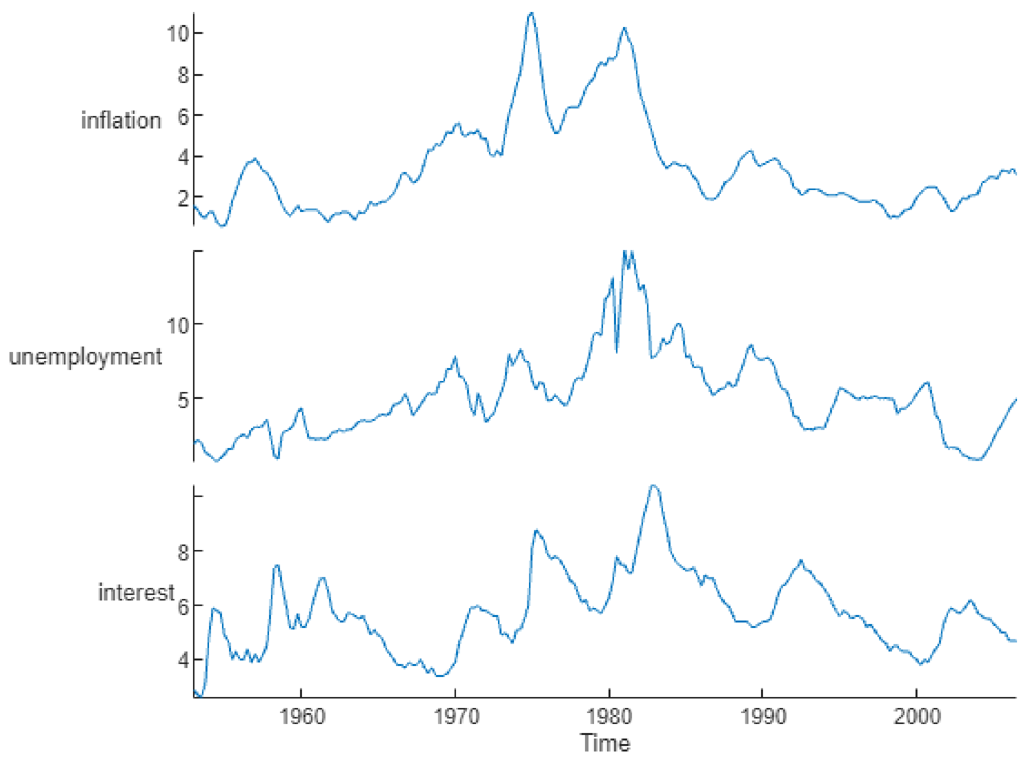
Let's consider a model for studying monetary policies involving three macroeconomic variables: the inflation rate, the unemployment rate and the interest rate, which are assumed to follow a vector autoregression (VAR) process. We could interpret the unexpected change of the interest rate as the monetary policy shock. This model is studied by Cogley and Sargent (2005), Primiceri (2005) and Koop and Korobilis (2009), among others.

Our bash tasks closely follow the empirical illustration of Bayesian VAR in Koop and Korobilis (2010).

To illustrate Bayesian VAR methods using some of the priors and methods described above, we use a quarterly US data set on the inflation rate $\Delta\pi_t$ (the annual percentage change in a chain-weighted GDP price index), the unemployment rate u_t (seasonally adjusted civilian unemployment rate, all workers over age 16) and the interest rate r_t (yield on the three month Treasury bill rate). Thus $y_t = (\Delta\pi_t, u_t, r_t)'$. The sample runs from 1953Q1 to 2006Q3. These three variables are commonly used in New Keynesian VARs.

```
clear;

c = fred('https://research.stlouisfed.org/fred2');
c.DatetimeType = 'datetime';
c.DataReturnFormat = 'timetable';
```

```
startdate = datetime(1953,1,1);
enddate = datetime(2006, 7, 1);

inflation    = c.fetch('B191RO1Q156NBEA', startdate , enddate );
inflation = inflation.Data{1};
unemployment = c.fetch('UNRATE', startdate , enddate ); unemployment =
unemployment.Data{1};
unemployment = retime(unemployment,"regular","fillwithmissing",...
    "TimeStep",calquarters(1));
interest     = c.fetch('TB3MS', startdate , enddate ); interest = interest.Data{1};
interest = retime(interest,"regular","fillwithmissing","TimeStep",calquarters(1));

% Synchronize timetables
Table = synchronize(inflation,interest,unemployment,"union",...
    "fillwithmissing");
Table.Properties.VariableNames = {'inflation', 'unemployment', 'interest'};

Y = Table{:,:};
figure;
stackedplot(Table)
```



To illustrate Bayesian VAR analysis using this data, we work with an unrestricted VAR with an intercept and four lags of all variables included in every equation and consider the following priors:

- **Noninformative**: The diffuse model is the limiting case of the natural conjugate when $M \to 0, V^{-1} \to 0, \Omega \to 0,$ and $\nu \to -k$
- **Natural conjugate**: The *dependent, matrix-normal-inverse-Wishart conjugate model* is an *m*-D Bayesian VAR(*p*) model in which the conditional prior distribution of $\Lambda|\Sigma$ is matrix normal with mean matrix M and scale matrices Σ and V. The prior distribution of Σ is inverse Wishart with scale matrix Ω and degrees of freedom ν.
- **Minnesota**: Fixed Innovations Matrix Σ, while the coefficient vector follows a normal distribution.
- **Independent Normal-Wishart**: The *independent, normal-inverse-Wishart semiconjugate model* is an *m*-D Bayesian VAR model in which the conditional prior distribution of $\lambda|\Sigma$ is multivariate normal with mean vector μ and covariance matrix V. The prior distribution of Σ is inverse Wishart with scale matrix Ω and degrees of freedom v.

For the first three priors, analytical posterior and predictive results are available. For last one, posterior and predictive simulation is required.

```
numSeries = 3;
numLags = 4;
priorType = "Minnesota";
seriesNames = ["pi", "ut", "rt"];

MdlF = varm(numSeries,numLags);
EstMdlF = estimate(MdlF,Y);
```

```
Unrecognized function or variable 'Y'.
```

```

[Mdl, EstMdl] = estimateBayesVARM(numSeries, numLags, priorType, seriesNames, Y,
EstMdlF);
disp("The Prior model is of type: " + class(Mdl))
disp("The Posterior model is of type: " + class(EstMdl))
disp("The following is the model summary")
summarize(EstMdl)
```

Impulse response functions

(Deeper dive here)

In empirical studies, the VAR coefficients *per se* are rarely of direct interest since it hard to interpret them. Instead, the main interest lies in some nonlinear transformations of those coefficients such as the impulse response analysis. Currently, we have obtained the reduced-form VAR, which needs identification restrictions to recover the structural-form VAR. In this model, the identification conditions can be a specific variable order: inflation, unemployment and interest rate. That is, the interest rate shock does not have a contemporaneous effect on the other two variables, which reflects the time lag of the responses to the monetary policy shock.

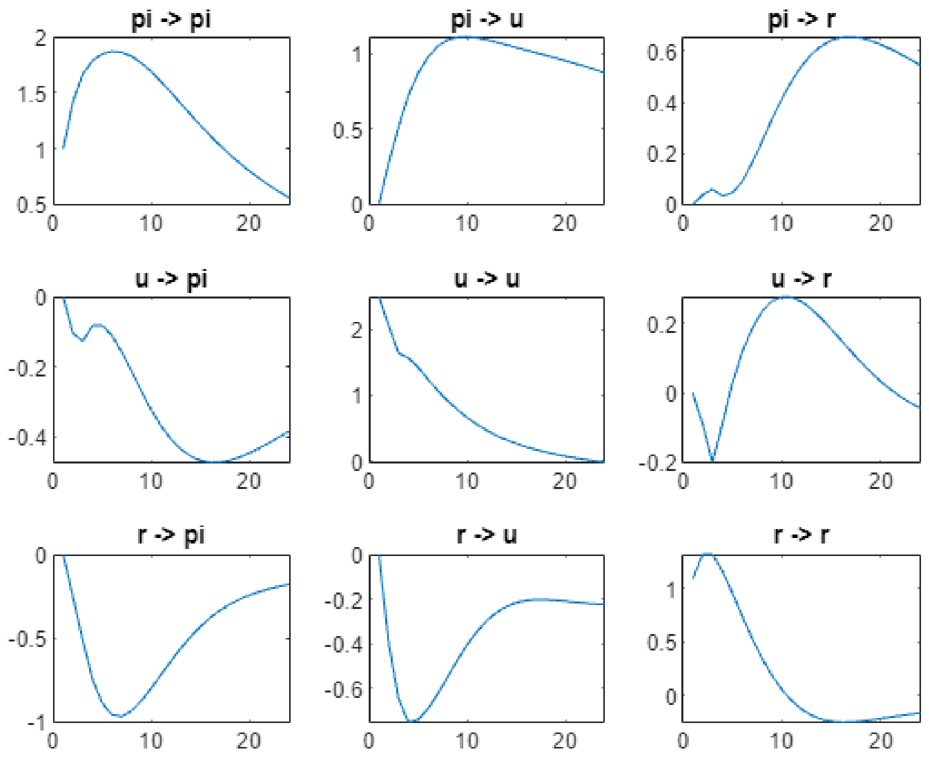
Now we extract the posterior mean of the estimated model and construct VARM for the impulse response analysis.

```
mdl = varm('Constant',EstMdl.Constant,'AR',EstMdl.AR,
'Covariance',EstMdl.Covariance);
Response = irf(mdl,'NumObs',24);
```

With the structural shocks identified by the lower triangular constraints, we could interpret the unexpected change of interest rate as a monetary policy shock. As the interest rate rises, a contractionary monetary policy will reduce the inflation rate, lower the outputs and increase the unemployment rate, if the money is not a veil on the real economy.

Note: we will plot the responses to one unit of shock, instead of one standard deviation of shock.

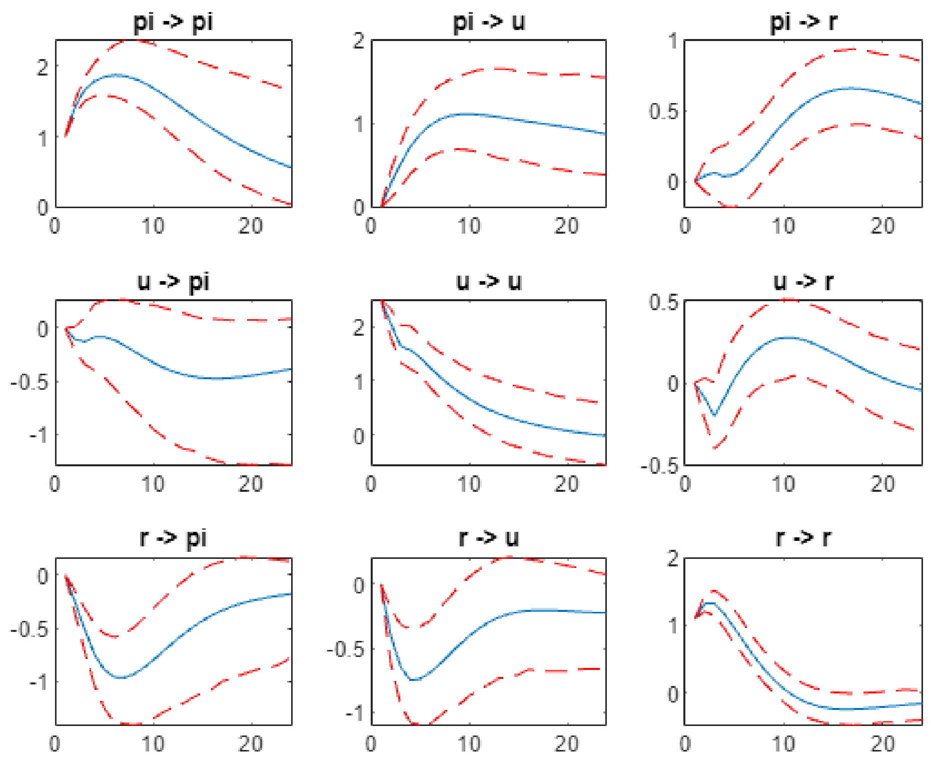
```
t = tiledlayout(3, 3);
names = ["pi", "u", "r"];
s2 = diag(EstMdlF.Covariance);
for i = 1 : 3
    for j = 1 : 3

        ax = nexttile(t);
        plot(ax, Response(:,i,j) ./ sqrt(s2(1)));
        title(names(i) + " -> " + names(j))

    end
end
```



Computing IRF intervals by Monte Carlo

First, we can simulate many posterior draws of VAR coefficients and the covariance matrix. For each draw, we construct a VARM model and compute the impulse response functions. Repeat the process for all draws, we obtain a collection of impulse response functions. After sorting them we can compute the quantiles (say 5% and 95%), which constitute a band that characterizes the uncertainties on the impulse response functions. Example code:

```
numDraws = 500;
if isa(EstMdl, 'empiricalbvarm')
    draws = randsample(10000,numDraws);
    Coeff = EstMdl.CoeffDraws(:, draws);
    Sigma = EstMdl.SigmaDraws(:, :, draws);
else
    [Coeff,Sigma] = simulate(EstMdl,'NumDraws',numDraws);
end
Response4D = zeros(24,3,3,numDraws);
for m = 1:numDraws
    A = reshape(Coeff(:,m),13,3)';
    mdl = varm('Constant',A(:,13),'AR',
{A(:,1:3),A(:,4:6),A(:,7:9),A(:,10:12)},'Covariance',Sigma(:,:,m));
    Response4D(:,:,:,m) = irf(mdl,'NumObs',24);
end
Response05 = quantile(Response4D,0.05,4);
Response95 = quantile(Response4D,0.95,4);
t = tiledlayout(3, 3);
names = ["pi", "u", "r"];
for i = 1 : 3
    for j = 1 : 3

        ax = nexttile();
        plot(ax, Response(:,i,j) ./ sqrt(s2(1)));
        title(names(i) + " -> " + names(j))
        hold on
        plot(ax, Response05(:,i,j) ./ sqrt(s2(1)), 'r--')
        plot(ax, Response95(:,i,j) ./ sqrt(s2(1)), 'r--')
        hold off

    end
end
```



Other resources

BEAR toolbox

The Bayesian Estimation, Analysis and Regression toolbox (BEAR) is a comprehensive (Bayesian Panel) VAR toolbox for forecasting and policy analysis.

BEAR is a MATLAB based toolbox which is easy for non-technical users to understand, augment and adapt. In particular, BEAR includes a user-friendly graphical interface which allows the tool to be used by country desk economists.

Furthermore, BEAR is well documented, both within the code as well as including a detailed theoretical and user's guide. BEAR includes state-of-the art applications such as FAVARs, stochastic volatility, time-varying parameters, mixed-frequency, sign and magnitude restrictions, conditional forecasts, Bayesian forecast evaluation measures, Bayesian Panel VAR using different prior distributions (for example hierarchical priors).

BEAR is specifically developed for transparently supplying a tool for state-of-the-art research and is planned to be further developed to always be at the frontier of economic research.

Use of BEAR implies acceptance of the End User Licence Agreement (EULA) for the Use of the Software "the Bayesian Estimation, Analysis and Regression (BEAR) toolbox".

```
BEARapp; % Run this command if you have the BEAR toolbox installed.
```

Dynare

Dynare is a software platform for handling a wide class of economic models, in particular dynamic stochastic general equilibrium (DSGE) and overlapping generations (OLG) models. The models solved by Dynare include those relying on the rational expectations hypothesis, wherein agents form their expectations about the future in a way consistent with the model. But Dynare is also able to handle models where expectations are formed differently: on one extreme, models where agents perfectly anticipate the future; on the other extreme, models where agents have limited rationality or imperfect knowledge of the state of the economy and, hence, form their expectations through a learning process. In terms of types of agents, models solved by Dynare can incorporate consumers, productive firms, governments, monetary authorities, investors and financial intermediaries. Some degree of heterogeneity can be achieved by including several distinct classes of agents in each of the aforementioned agent categories.

Dynare offers a user-friendly and intuitive way of describing these models. It is able to perform simulations of the model given a calibration of the model parameters and is also able to estimate these parameters given a dataset. In practice, the user will write a text file containing the list of model variables, the dynamic equations linking these variables together, the computing tasks to be performed and the desired graphical or numerical outputs.

A large panel of applied mathematics and computer science techniques are internally employed by Dynare: multivariate nonlinear solving and optimization, matrix factorizations, local functional approximation, Kalman filters and smoothers, MCMC techniques for Bayesian estimation, graph algorithms, optimal control, etc. References to the literature can be found here.

```
launchDynare % Run this command if you have the Dynare editor installed.
```

SAMNet is a repository of models for modelling and forecasting the economy

```
web('https://samnet.org.za/model-databases/','browser')
```

```
edit SamNet.mod
```

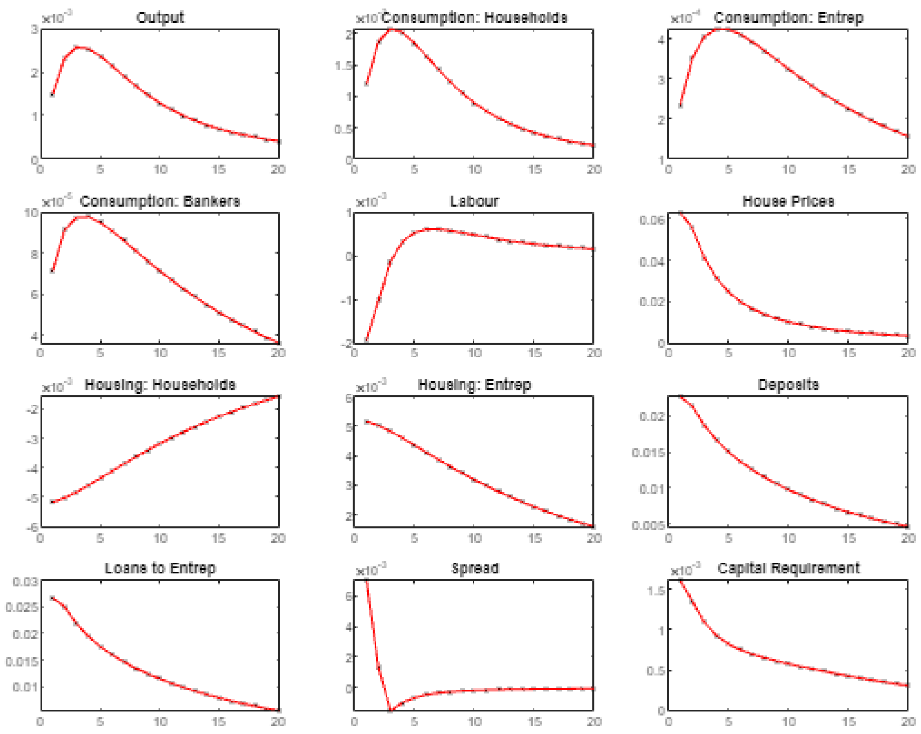
```
dynare SamNet.mod
```

```
Starting Dynare (version 5.3).
Calling Dynare with arguments: none
Starting preprocessing of the model file ...
Found 19 equation(s).
Evaluating expressions...done
Computing static model derivatives (order 1).
Computing dynamic model derivatives (order 1).
Processing outputs ...
done
Preprocessing completed.


STEADY-STATE RESULTS:

C               0.31994
C_h             0.2941
C_e             0.0216553
C_b             0.00418431
H_h             0.748768
H_e             0.251232
```

```
N               0.32865
W               0.876146
q               4.36421
D               0.677659
L_e             0.757161
R_d             1.00908
R_l             1.01365
Sprd            0.00457273
Y               0.31994
kappa           0.105
A               1
Z               1
vareps          0

EIGENVALUES:
         Modulus             Real        Imaginary

       1.903e-09       -1.903e-09                0
       1.903e-09        1.903e-09                0
       1.559e-08        1.559e-08                0
       1.559e-08       -1.559e-08                0
          0.5877           0.5877                0
          0.6759           0.6759                0
          0.6979           0.6979                0
             0.8              0.8                0
             0.8              0.8                0
             0.9              0.9                0
           0.931            0.931                0
           1.011            1.011                0
           1.049            1.049                0
           1.062            1.062                0
           1.652           -1.652                0
        9.19e+17        -9.19e+17                0
             Inf              Inf                0


There are 6 eigenvalue(s) larger than 1 in modulus
for 6 forward-looking variable(s)

The rank condition is verified.


MODEL SUMMARY

  Number of variables:         19
  Number of stochastic shocks: 3
  Number of state variables:   11
  Number of jumpers:           6
  Number of static variables:  5


MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS
Variables           ups_a       ups_z  ups_vareps
ups_a            0.000000    0.000000    0.000000
ups_z            0.000000    0.000100    0.000000
ups_vareps       0.000000    0.000000    0.000000

POLICY AND TRANSITION FUNCTIONS
                                   C             C_h             C_e             C_b             H_h             H_e
Constant                    0.319940        0.294100        0.021655        0.004184        0.748768        0.25123
H_h(-1)                     0.796128        0.644542        0.104758        0.046828       -2.647356        2.64735
H_e(-1)                     0.933362        0.677875        0.183908        0.071578       -5.303482        5.30348
D(-1)                      -0.018000        0.003252       -0.016196       -0.005055        0.564827       -0.56482
L_e(-1)                    -0.001428        0.000258       -0.000902       -0.000784       -0.002205        0.00220
```

```
R_d(-1)                    -0.012088        0.002184       -0.010877       -0.003395        0.379315       -0.379315
A(-1)                       0.000549       -0.000099        0.000522        0.000126       -0.009798        0.009798
Z(-1)                       0.119672        0.095202        0.018758        0.005712       -0.413517        0.413517
vareps(-1)                 -0.005939        0.001073       -0.013588        0.006576        0.465540       -0.465540
C_h(-1)                     0.572596        0.596566       -0.017151       -0.006818        0.373666       -0.373666
C_e(-1)                     0.535169       -0.096674        0.642606       -0.010763        0.717473       -0.717473
C_b(-1)                     0.538170       -0.097216       -0.029119        0.664505        0.962821       -0.962821
ups_a                       0.000686       -0.000124        0.000653        0.000157       -0.012248        0.012248
ups_z                       0.149590        0.119002        0.023447        0.007141       -0.516896        0.516896
ups_vareps                 -0.006599        0.001192       -0.015098        0.007307        0.517266       -0.517266


THEORETICAL MOMENTS
VARIABLE         MEAN  STD. DEV.   VARIANCE
C              0.3199     0.0069     0.0000
C_h            0.2941     0.0052     0.0000
C_e            0.0217     0.0014     0.0000
C_b            0.0042     0.0003     0.0000
H_h            0.7488     0.0158     0.0003
H_e            0.2512     0.0158     0.0003
N              0.3287     0.0028     0.0000
W              0.8761     0.0188     0.0004
q              4.3642     0.1088     0.0118
D              0.6777     0.0558     0.0031
L_e            0.7572     0.0652     0.0043
R_d            1.0091     0.0048     0.0000
R_l            1.0137     0.0078     0.0001
Sprd           0.0046     0.0075     0.0001
Y              0.3199     0.0069     0.0000
kappa          0.1050     0.0034     0.0000
A              1.0000     0.0000     0.0000
Z              1.0000     0.0167     0.0003
vareps         0.0000     0.0000     0.0000



VARIANCE DECOMPOSITION (in percent)
               ups_a       ups_z  ups_vareps
C               0.00      100.00        0.00
C_h             0.00      100.00        0.00
C_e             0.00      100.00        0.00
C_b             0.00      100.00        0.00
H_h             0.00      100.00        0.00
H_e             0.00      100.00        0.00
N               0.00      100.00        0.00
W               0.00      100.00        0.00
q               0.00      100.00        0.00
D               0.00      100.00        0.00
L_e             0.00      100.00        0.00
R_d             0.00      100.00        0.00
R_l             0.00      100.00        0.00
Sprd            0.00      100.00        0.00
Y               0.00      100.00        0.00
kappa           0.00      100.00        0.00
A             100.00        0.00        0.00
Z               0.00      100.00        0.00
vareps          0.00        0.00      100.00



MATRIX OF CORRELATIONS
Variables          C      C_h      C_e      C_b      H_h      H_e        N        W        q        D      L_e
C             1.0000   0.9958   0.9609   0.9671  -0.9692   0.9692   0.2061   0.9212   0.8605   0.9662   0.9651  -0.8
C_h           0.9958   1.0000   0.9315   0.9402  -0.9482   0.9482   0.1601   0.9353   0.8785   0.9561   0.9550  -0.8
```

```
C_e          0.9609   0.9315   1.0000   0.9975  -0.9803   0.9803   0.3442   0.8271   0.7564   0.9435   0.9427  -0.6
C_b          0.9671   0.9402   0.9975   1.0000  -0.9914   0.9914   0.2783   0.8596   0.7969   0.9634   0.9628  -0.7
H_h         -0.9692  -0.9482  -0.9803  -0.9914   1.0000  -1.0000  -0.1533  -0.9113  -0.8639  -0.9891  -0.9889   0.7
H_e          0.9692   0.9482   0.9803   0.9914  -1.0000   1.0000   0.1533   0.9113   0.8639   0.9891   0.9889  -0.7
N            0.2061   0.1601   0.3442   0.2783  -0.1533   0.1533   1.0000  -0.1908  -0.3205   0.0206   0.0177   0.2
W            0.9212   0.9353   0.8271   0.8596  -0.9113   0.9113  -0.1908   1.0000   0.9907   0.9610   0.9612  -0.9
q            0.8605   0.8785   0.7564   0.7969  -0.8639   0.8639  -0.3205   0.9907   1.0000   0.9286   0.9291  -0.9
D            0.9662   0.9561   0.9435   0.9634  -0.9891   0.9891   0.0206   0.9610   0.9286   1.0000   1.0000  -0.8
L_e          0.9651   0.9550   0.9427   0.9628  -0.9889   0.9889   0.0177   0.9612   0.9291   1.0000   1.0000  -0.8
R_d         -0.8240  -0.8552  -0.6831  -0.7206   0.7756  -0.7756   0.2899  -0.9419  -0.9436  -0.8479  -0.8478   1.0
R_l         -0.4815  -0.4898  -0.4384  -0.4080   0.3300  -0.3300  -0.5902  -0.2484  -0.1478  -0.2890  -0.2858   0.3
Sprd         0.0282   0.0397  -0.0176   0.0384  -0.1554   0.1554  -0.8041   0.3480   0.4543   0.2449   0.2482  -0.2
Y            1.0000   0.9958   0.9609   0.9671  -0.9692   0.9692   0.2061   0.9212   0.8605   0.9662   0.9651  -0.8
kappa        0.9390   0.9270   0.9224   0.9468  -0.9804   0.9804  -0.0446   0.9597   0.9376   0.9959   0.9963  -0.8
Z            0.8410   0.8670   0.7139   0.7550  -0.8267   0.8267  -0.3492   0.9825   0.9941   0.8996   0.9001  -0.9


COEFFICIENTS OF AUTOCORRELATION
Order            1        2        3        4        5
C           0.9648   0.8920   0.8059   0.7191   0.6378
C_h         0.9595   0.8746   0.7748   0.6755   0.5839
C_e         0.9812   0.9428   0.8947   0.8423   0.7890
C_b         0.9734   0.9298   0.8798   0.8280   0.7767
H_h         0.9450   0.8899   0.8359   0.7836   0.7336
H_e         0.9450   0.8899   0.8359   0.7836   0.7336
N           0.6381   0.3305   0.1606   0.0694   0.0226
W           0.8679   0.7228   0.6075   0.5148   0.4396
q           0.8139   0.6336   0.5074   0.4163   0.3486
D           0.9130   0.8282   0.7567   0.6947   0.6399
L_e         0.9118   0.8265   0.7551   0.6935   0.6392
R_d         0.8380   0.5816   0.4153   0.3050   0.2299
R_l         0.2597  -0.0192   0.0076   0.0213   0.0276
Sprd        0.1834  -0.1749  -0.1107  -0.0723  -0.0491
Y           0.9648   0.8920   0.8059   0.7191   0.6378
kappa       0.8825   0.7862   0.7168   0.6630   0.6183
Z           0.8000   0.6400   0.5120   0.4096   0.3277
```



```
Total computing time : 0h00m05s
```

IRIS toolbox

IRIS is a free, open-source toolbox for macroeconomic modeling and forecasting in Matlab®, originally developed by the IRIS Solutions Team and currently maintained and supported by the Global Projection Model Network.

In a user-friendly command-oriented environment, IRIS integrates core modeling functions (including a flexible model file language with a powerful preparser, a variety of tools for simulation, estimation, forecasting and model diagnostics, practical techniques for judgmental adjustments, and more) with supporting infrastructure (such as data management, time series analysis, and reporting).

References

[1] Cogley, T. and Sargent, T. (2005). **Drifts and volatilities: Monetary policies and outcomes in the post WWII** U.S, Review of Economic Dynamics, 8, 262-302.

[2] Primiceri. G., (2005). **Time varying structural vector autoregressions and monetary policy**, Review of Economic Studies, 72, 821-852.

[3] Koop, G. and Korobilis, D. (2009). **Bayesian Multivariate Time Series Methods for Empirical Macroeconomics**, Working Paper series 47-09, Rimini Centre for Economic Analysis.