# Quantitative Macroeconomics

# Winter 2023/24

# Week 10

Willi Mutschler
willi@mutschler.eu

Version: 1.0
Latest version available on: GitHub

# Contents

**1. Bootstrapping Standard Deviations of Structural IRFs** **1**

**2. How Well Does the IS-LM Model Fit Postwar US Data** **2**

**A. Solutions** **4**

# 1. Bootstrapping Standard Deviations of Structural IRFs

Consider an exactly-identified structural VAR model subject to short- and/or long-run restrictions, where the structural impulse response of variable $i$ to shock $j$ at horizon $h$ are simply denoted as $\theta \equiv \Theta_{ij,h}$. As an exact closed-form solution for the asymptotic standard errors of $\theta$ are only available under restrictive assumptions, we will rely on a numerical approximation using a bootstrap approach.

1. Reconsider an exercise (of your choice) from the lecture on SVAR models identified with exclusion restrictions and re-estimate the structural impulse response function.

2. Compute $\widehat{std}(\hat{\theta}^*)$ via a bootstrap approximation by following these steps:
   - Write a function `bootstrapDGP(VAR,opt)` which implements a standard residual-based bootstrap approach using sampling with replacement techniques on the residuals. Furthermore, the initial values should be drawn randomly in blocks. Hint: Use the companion form to do the simulations.
   - Set bootstrap repetitions $B$ equal to 1000 (or higher) and initialize a $K \times K \times H \times B$ array `THETAstar`, where the first dimension corresponds to variable $i = 1, \ldots, K$, the second dimension to shock $j = 1, /ldots, K$, the third dimension to the horizon of the IRFs $h = 0, \ldots, H$ and the fourth dimension to the bootstrap repetition $b = 1., \ldots, B$.
   - For $b = 1, \ldots, B$ do the following (you may also try `parfor` instead of `for` in order to make use of Matlab's parallel computing toolbox – if installed):
     - Compute a bootstrap DGP $y_t^b$ using the function `bootstrapDGP(VAR,opt)`.
     - Estimate the reduced-form and structural impulse response function on this artificial dataset with the same methodology, settings and identification restrictions as in the estimation of the original dataset.
     - Store the structural IRFs in `THETAstar` at position `(:,:,:,b)`.
   - Compute the standard deviation of the bootstrap structural IRFs using `std(THETAstar,0,4)`.

3. Plot approximate 68% and 95% confidence intervals for the structural impulse response functions according to the Delta method:

$$\hat{\theta} \pm z_{\gamma/2} \widehat{std}(\hat{\theta}^*)$$

   where $z_{\gamma/2}$ is the $\gamma/2$ quantile of the standard normal distribution.

### Readings

- Kilian and Lütkepohl (2017, Ch. 12.1-12.5, 12.9)

## 2. How Well Does the IS-LM Model Fit Postwar US Data?

Consider a quarterly model for $y_t = (\Delta gnp_t, \Delta i_t, i_t - \Delta p_t, \Delta m_t - \Delta p_t)'$, where $gnp_t$ denotes the log of GNP, $i_t$ the nominal yield on three-month Treasury Bills, $\Delta m_t$ the growth in M1 and $\Delta p_t$ the inflation rate in the CPI. There are four shocks in the system: an aggregate supply (AS), a money supply (MS), a money demand (MD) and an aggregate demand (IS) shock. Ignoring the lagged dependent variables for **expository** purposes ($B_1 = \cdots = B_p = 0$), the unrestricted structural VAR model can be simply written as $B_0 y_t = \varepsilon_t$. That is:

$$\Delta gnp_t = -b_{12}\Delta i_t - b_{13}(i_t - \Delta p_t) - b_{14}(\Delta m_t - \Delta p_t) + \varepsilon_t^{AS} \tag{1}$$

$$\Delta i_t = -b_{21}\Delta gnp_t - b_{23}(i_t - \Delta p_t) - b_{24}(\Delta m_t - \Delta p_t) + \varepsilon_t^{MS} \tag{2}$$

$$i_t - \Delta p_t = -b_{31}\Delta gnp_t - b_{32}\Delta i_t - b_{34}(\Delta m_t - \Delta p_t) + \varepsilon_t^{MD} \tag{3}$$

$$\Delta m_t - \Delta p_t = -b_{41}\Delta gnp_t - b_{42}\Delta i_t - b_{43}(i_t - \Delta p_t) + \varepsilon_t^{IS} \tag{4}$$

where $b_{ij}$ denotes the $ij$th element of $B_0$. Consider the following identification restrictions:

- Money supply shocks do not have contemporaneous effects on output growth, i.e.
$$\frac{\partial \Delta gnp_t}{\partial \varepsilon_t^{MS}} = 0$$
- Money demand shocks do not have contemporaneous effects on output growth, i.e.
$$\frac{\partial \Delta gnp_t}{\partial \varepsilon_t^{MD}} = 0$$
- Monetary authority does not react contemporaneously to changes in the price level.
Hint: compute from equation (2):
$$\frac{\partial \Delta i_t}{\partial \Delta p_t} = 0$$
- Money supply shocks, money demand shocks and aggregate demand shocks do not have long-run effects on the log of real GNP:
$$\frac{\partial gnp_t}{\partial \varepsilon_t^{MS}} = 0, \qquad \frac{\partial gnp_t}{\partial \varepsilon_t^{MD}} = 0, \qquad \frac{\partial gnp_t}{\partial \varepsilon_t^{IS}} = 0$$
- The structural shocks are uncorrelated with covariance matrix $E(\varepsilon_t \varepsilon_t') = \Sigma_\varepsilon$. In other words, the variances are **not** normalized.

Solve the following exercises:

1. Derive the implied exclusion restrictions on the matrices $B_0$, $B_0^{-1}$ and $\Theta(1)$.
2. Consider data given in the csv file `gali1992.csv`. Estimate a VAR(4) model with a constant.
3. Estimate the structural impact matrix using a nonlinear equation solver, i.e. the objective is to find the unknown elements of $B_0^{-1}$ and the diagonal elements of $\Sigma_\varepsilon$ such that
$$\begin{bmatrix} vech(B_0^{-1}\Sigma_\varepsilon B_0^{-1\prime} - \hat{\Sigma}_u) \\ \text{short-run restrictions on } B_0 \text{ and } B_0^{-1} \\ \text{long-run restrictions on } \Theta(1) \end{bmatrix}$$
is minimized. Normalize the shocks such that the diagonal elements of $B_0^{-1}$ are positive.
4. Use the implied estimates of $B_0^{-1}$ and $\Sigma_\varepsilon$ to plot the structural impulse responses functions for (i) real GNP, (ii) the yield on Treasury Bills, (iii) the real interest rate and (iv) real money growth. Add 68% and 95% confidence intervals using a bootstrap approach.

### Readings

- Gali (1992)

## References


Gali, J. (May 1992). "How Well Does The IS-LM Model Fit Postwar U. S. Data?" In: *The Quarterly Journal of Economics* 107.2, pp. 709–738. DOI: 10.2307/2118487.

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.

Rubio-Ramírez, Juan F., Daniel F. Waggoner, and Tao Zha (Apr. 2010). "Structural Vector Autoregressions: Theory of Identification and Algorithms for Inference". In: *Review of Economic Studies* 77.2, pp. 665–696. DOI: 10.1111/j.1467-937X.2009.00578.x.

# A. Solutions

## 1 Solution to Bootstrapping Standard Deviations of Structural IRFs

1. We will re-consider the Rubio-Ramírez, Waggoner, and Zha (2010) example.

2./3. The helper function to generate bootstrap DGPs might look like this:

progs/matlab/bootstrapDGP.m

```matlab
function yb = bootstrapDGP(VAR)
% yb = bootstrapDGP(VAR)
% -----------------------------------------------------------------------
% Computes a standard residual-based bootstrap data-generating process (DGP)
% for VAR models with iid sampling with replacement for errors and
% random sampling of blocks for initial values.
% This function uses the companion VAR(1) form to do the simulations
% Model:
%   y_t = c + A_1*y_{t-1} + ... + A_p*y_{t-p} + u_t
% Companion form:
%   Y_t = C + A*Y_{t-1} + U_t
% where
%   Y_t = [y_t;y_{t-1};...;y_{t-p-1}]
%   C   = [c;0;...;0]
%   A   = [A_1 A_2 ... A_{p-1} A_p;
%          I_K 0_K ... 0_K     0_K;
%          0_K I_K ... 0_K     0_K:
%          ... ... ... ...     ...;
%          0_K 0_K ... I_K     0_K]
%   U_t = [u_t; 0; ...;0]
% -----------------------------------------------------------------------
% INPUTS
%       - VAR  : structure of reduced-form estimation function VARReducedForm.m
% -----------------------------------------------------------------------
% OUTPUTS
%   - yb   : Data matrix. [number of periods x number of variables]
% -----------------------------------------------------------------------
% Willi Mutschler, January 23, 2024
% willi@mutschler.eu
% -----------------------------------------------------------------------

y     = transpose(VAR.ENDO);
p     = VAR.nlag;
const = VAR.opt.const;
[K,T] = size(y);
Acomp = VAR.Acomp;
U = [VAR.residuals; zeros(K*(p-1),T-p)]; % residuals for companion form
% create coefficient vectors for deterministic constant term in companion form
C = zeros(K*p,1);
if const == 1
    C(1:K,1) = VAR.A(:,1);
end

% create Y of companion form
% note that Y_t = [y_t;y_{t-1};...;y_{t-p+1}] is [Kp x 1]
```

```matlab
% such that Y = [Y_p Y_{p+1} ... Y_T] is [Kp x (T-p+1)]
Y = y(:,p:T);
for i = 1:p-1
        Y = [Y; y(:,p-i:T-i)];
end

% initalize bootstrap DGP in companion form
Yb = nan(K*p,T-p+1);
Ub = nan(K*p,T-p);

% iid blockwise resampling of initial values of VAR(p) model is equivalent
% to randomly choosing a column of the companion VAR(1) model
indexY = fix(rand(1,1)*(T-p+1))+1; % this generates a randomly chosen integer between 1 and T-p+1
Yb(:,1) = Y(:,indexY);

% iid resampling for error terms
indexU = fix(rand(1,T-p)*(T-p))+1; % this generates T-p randomly chosen integers between 1 and T-p
Ub(:,2:T-p+1) = U(:,indexU);

for t = 2:T-p+1
    Yb(:,t) = C + Acomp*Yb(:,t-1) + Ub(:,t);
end

% reformat the bootstrapped data from its companion form back into the original VAR model form
yb = Yb(1:K,:); % initial assignment
for i = 2:p % loop to concatenate lagged values
    % use indices that correspond to the rows of Yb that contain the ith lag of each variable
    % and extract first column of these rows which contains the ith lagged
    % values of all variables at the first time period
    yb = [Yb((i-1)*K+1:i*K,1) yb];
end
yb = yb'; % match original VAR data format

end
```

The main script might look like this:

progs/matlab/bootstrappingStdIRFs_RWZSRLR.m

```matlab
% Compute the standard deviation of structural IRFs via bootstrap
% using the example by Rubio-Ramirez, Waggoner, Zha (2010)'s model which
% uses both short-run and long-run restrictions for identification
% -------------------------------------------------------------------------
% Willi Mutschler, January 24, 2024
% willi@mutschler.eu
% -------------------------------------------------------------------------
clearvars

% point estimate for structural IRFs in exactly identified model
RWZSRLR; % simply run the RWZSRLR example
```

```
disp(IRFpoint);
close all
clearvars -except VAR IRFpoint opt f B0inv nsteps optim_opt IRFcumsum varnames epsnames

B = 1000; % number of bootstrap replications
nvar = size(VAR.ENDO,2); % number of variables
THETAstar = zeros(nvar,nvar,nsteps+1,B); % storage for bootstrapped IRFs
opt.dispestim = false; % don't display results in VARReducedForm
optim_opt.Display = 'off'; % don't display results of fsolve
parfor b = 1:B
    ystar = bootstrapDGP(VAR);
    VARstar = VARReducedForm(ystar,VAR.nlag,opt);
    % redo identification steps exactly as in RWZSRLR
    A1starinv_big = inv(eye(size(VARstar.Acomp,1))-VARstar.Acomp); % from the companion
    LRMatstar = A1starinv_big(1:nvar,1:nvar); % total impact matrix inv(eye(nvars)-A1hat-A2hat-...-Aphat)
    % Call optimization routine fsolve, note that we use point estimate B0inv as initial value
    [B0invstar,fval,exitflag,output] = fsolve(f,B0inv,optim_opt,VARstar.SigmaOLS,LRMatstar);
    % use same normalization rules
    if sign(B0invstar(2,1)) == -1
        B0invstar(:,1) = -B0invstar(:,1);
    end
    if sign(B0invstar(1,2)) == -1 && sign(B0invstar(3,2)) == -1
        B0invstar(:,2) = -B0invstar(:,2);
    end
    if sign(B0invstar(1,3)) == -1 && sign(B0invstar(3,3)) == 1
        B0invstar(:,3) = -B0invstar(:,3);
    end
    % store results
    THETAstar(:,:,:,b) = irfPlots(VARstar.Acomp,B0invstar,nsteps,IRFcumsum,varnames,epsnames,1); % the 1 at end disables plots
end
% compute standard deviation across b
IRFse   = std(THETAstar,0,4);
% set up confidence intervals
IRF95LO = IRFpoint - 1.96*IRFse;
IRF95UP = IRFpoint + 1.96*IRFse;
IRF68LO = IRFpoint - 1*IRFse;
IRF68UP = IRFpoint + 1*IRFse;

figure('Name','Inference');
countplots = 1;
x_axis = zeros(1,nsteps+1);
for ishock = 1:nvar
    for ivar = 1:nvar
        subplot(nvar,nvar,countplots);
        irfpoint = squeeze(IRFpoint(ivar,ishock,:));
        irf95up  = squeeze(IRF95UP(ivar,ishock,:));
```

```
        irf95lo  = squeeze(IRF95LO(ivar,ishock,:));
        irf68up  = squeeze(IRF68UP(ivar,ishock,:));
        irf68lo  = squeeze(IRF68LO(ivar,ishock,:));
        plot(0:1:nsteps,irfpoint,'b','LineWidth',2);
        hold on;
        plot(0:1:nsteps, [irf68lo irf68up] ,'--b');
        plot(0:1:nsteps, [irf95lo irf95up] ,'--r');
        plot(0:1:nsteps,x_axis,'k','LineWidth',2);
        grid;
        xlim([0 nsteps]);
        title(varnames{ivar})
        ylabel([epsnames{ishock}, 'Shock'])
        countplots = countplots + 1;
    end
end
```

## 2 Solution to How Well Does the IS-LM Model Fit Postwar US Data

1. First, let's rewrite the equations in matrix form:

$$\underbrace{\begin{bmatrix} 1 & b_{12} & b_{13} & b_{14} \\ b_{21} & 1 & b_{23} & b_{24} \\ b_{31} & b_{32} & 1 & b_{34} \\ b_{41} & b_{42} & b_{43} & 1 \end{bmatrix}}_{B_0} \begin{bmatrix} \Delta gnp_t \\ \Delta i_t \\ i_t - \Delta p_t \\ \Delta m_t - \Delta p_t \end{bmatrix} = \begin{bmatrix} \varepsilon_t^{AS} \\ \varepsilon_t^{MS} \\ \varepsilon_t^{MD} \\ \varepsilon_t^{IS} \end{bmatrix}$$

$$\begin{bmatrix} \Delta gnp_t \\ \Delta i_t \\ i_t - \Delta p_t \\ \Delta m_t - \Delta p_t \end{bmatrix} = \underbrace{\begin{bmatrix} b_{11}^* & b_{12}^* & b_{13}^* & b_{14}^* \\ b_{21}^* & b_{22}^* & b_{23}^* & b_{24}^* \\ b_{31}^* & b_{32}^* & b_{33}^* & b_{34}^* \\ b_{41}^* & b_{42}^* & b_{43}^* & b_{44}^* \end{bmatrix}}_{B_0^{-1}} \begin{bmatrix} \varepsilon_t^{AS} \\ \varepsilon_t^{MS} \\ \varepsilon_t^{MD} \\ \varepsilon_t^{IS} \end{bmatrix}$$

The long-run multiplier matrix is given by:

$$\Theta(1) = A(1)^{-1} B_0^{-1}$$

Now let's derive the restrictions on the impact matrix $B_0^{-1}$:

- Money supply shocks do not have contemporaneous effects on output growth, i.e.

$$\frac{\partial \Delta gnp_t}{\partial \varepsilon_t^{MS}} = b_{12}^* = 0$$

- Money demand shocks do not have contemporaneous effects on output growth, i.e.

$$\frac{\partial \Delta gnp_t}{\partial \varepsilon_t^{MD}} = b_{13}^* = 0$$

Summarizing this yields two restrictions:

$$B_0^{-1} = \begin{pmatrix} * & 0 & 0 & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix}$$

Next, let's derive the restrictions on the structural matrix $B_0$:

- Monetary authority does not react contemporaneously to changes in the price level. This can be computed directly from equation (2):

$$\frac{\partial \Delta i_t}{\partial \Delta p_t} = b_{23} + b_{24} = 0 \Leftrightarrow b_{23} = -b_{24}$$

Summarizing this yields one restriction:

$$B_0 = \begin{pmatrix} 1 & * & * & * \\ * & 1 & -b_{24} & b_{24} \\ * & * & 1 & * \\ * & * & * & 1 \end{pmatrix}$$

- Money supply shocks, money demand shocks and aggregate demand shocks do not have long-run effects on the log of real GNP.

The restrictions on the long-run multiplier matrix are thus:

$$\Theta(1) = \begin{pmatrix} * & 0 & 0 & 0 \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix}$$

This yields three restrictions.

In total we have $2+1+3=6$ restrictions, which is equal to the required number of $K(K-1)/2=6$ of an exactly identified SVAR model.

Lastly, we need to keep in mind that the variance of the structural shocks is not normalized:

$$\Sigma_\varepsilon = \begin{pmatrix} \sigma_{11} & 0 & 0 & 0 \\ 0 & \sigma_{22} & 0 & 0 \\ 0 & 0 & \sigma_{33} & 0 \\ 0 & 0 & 0 & \sigma_{44} \end{pmatrix}$$

2/3/4 Here is the helper function to impose the restrictions:

progs/matlab/gali1992_f.m

```
function f = gali1992_f(B0inv_diageps,SIGMAUHAT,LRMat)
% f = gali1992_f(B0inv_diageps,SIGMAUHAT,LRMat)
% -----------------------------------------------------------------------
% Evaluates the system of nonlinear equations vech(SIGMAUHAT) = vech(B0inv*SIGeps*B0inv')
% where SIGeps has only values on the diagonal
% subject to the short- and long-run restrictions in the Gali (1992) model.
% -----------------------------------------------------------------------
% INPUTS
%   - B0inv_diageps [var_nbr x (var_nbr+1)]  candidate matrix for both short-run impact matrix [var_nbr x var_nbr]
%                                            and diagonal elements of SIGeps [var_nbr x 1]
%   - SIGMAUHAT     [var_nbr x var_nbr]      covariance matrix of reduced-form residuals
%   - LRMat         [var_nbr x var_nbr]      total long-run impact matrix for VAR model A(1) = inv(eye(var_nbr)-A1hat-A2hat-...-Aphat)
% -----------------------------------------------------------------------
% OUTPUTS
%   - f : function value, see below
% -----------------------------------------------------------------------
% Willi Mutschler, February 14, 2024
% willi@mutschler.eu
% -----------------------------------------------------------------------

var_nbr = size(SIGMAUHAT,1);                % number of (endogenous) VAR variables
B0inv = B0inv_diageps(:,1:var_nbr);         % get short-run impact matrix from candidate matrix
SIGeps = diag(B0inv_diageps(:,var_nbr+1)); % get diagonal elements of SIGeps from candidate matrix and make it a full diagonal matrix
THETA = LRMat*B0inv;                        % compute long-run impact matrix
B0 = inv(B0inv);                            % compute B0

```

```
% impose restrictions
f = [vech(B0inv*SIGeps*B0inv' - SIGMAUHAT);
     B0inv(1,2) - 0;
     B0inv(1,3) - 0;
     B0(1,1) - 1;
     B0(2,2) - 1;
     B0(3,3) - 1;
     B0(4,4) - 1;
     B0(2,3) + B0(2,4) - 0;
     THETA(1,2) - 0;
     THETA(1,3) - 0;
     THETA(1,4) - 0;
    ];
```

The main code might look like this:

progs/matlab/gali1992.m

```
% -------------------------------------------------------------------------
% Replicates the Gali (1992) model using both short-run and long-run
% restrictions to identify the structural shocks using a numerical solver
% -------------------------------------------------------------------------
% Willi Mutschler, February 14, 2024
% willi@mutschler.eu
% -------------------------------------------------------------------------

clearvars; close all; clc;

% data handling
ENDO = importdata('../../data/gali1992.csv');
ENDO = ENDO.data;

% settings and options
varnames = ["Real GNP growth", "3M TBills yield growth", "Real interest rate", "Real money growth"];
varnames_IRFs = ["Real GNP" "3M yield on TBills" "Real interest rate" "Real money growth"];
epsnames = ["AS", "MS", "MD", "IS"] + " shock";
IRFcumsum = [1 1 0 0];
nsteps = 28;
nlag = 4;
[obs_nbr,var_nbr] = size(ENDO);

% Plotting the data
figure('name','Gali 1992: data')
for j = 1:var_nbr
    subplot(2,2,j);
    plot(datetime('1955Q1','InputFormat','yyyyQQQ'):calquarters(1):datetime('1988Q3','InputFormat','yyyyQQQ'),...
         ENDO(:,j),...
         'Color','black','LineWidth',2);
    title(varnames{j});
end

```

```
% estimate reduced-form
opt.const = 1;
VAR = VARReducedForm(ENDO,nlag,opt);
A1inv_big = inv(eye(size(VAR.Acomp,1))-VAR.Acomp); % from the companion form
LRMat = A1inv_big(1:var_nbr,1:var_nbr); % total long-run impact matrix inv(eye(nvars)-A1hat-A2hat-...-Aphat)

% options for fsolve
TolX = 1e-4;          % termination tolerance on the current point
TolFun = 1e-9;        % termination tolerance on the function value
MaxFunEvals = 50000; % maximum number of function evaluations allowed
MaxIter = 1000;       % maximum numberof iterations allowed
OptimAlgorithm = 'trust-region-dogleg'; % algorithm used in fsolve
optim_options = optimset('TolX',TolX,'TolFun',TolFun,'MaxFunEvals',MaxFunEvals,'MaxIter',MaxIter,'Algorithm',OptimAlgorithm);

% initital guess
B0inv_diageps = [chol(VAR.SigmaOLS,'lower') ones(var_nbr,1)]; % Use Cholesky matrix as starting value for B0inv and ones for variance

% structural identification
f = str2func('gali1992_f');
f(B0inv_diageps,VAR.SigmaOLS,LRMat)' % test whether function works at initial value (should give you no error)

% call optimization routine fsolve
[B0inv_diageps,fval,exitflag,output] = fsolve(f,B0inv_diageps,optim_options,VAR.SigmaOLS,LRMat);
B0inv = B0inv_diageps(:,1:var_nbr);
SIGeps = diag(B0inv_diageps(:,var_nbr+1));

% normalization rules
for j = 1:var_nbr
    if sign(B0inv(j,j)) == -1
        B0inv(:,j) = -B0inv(:,j);
    end
end
B0 = inv(B0inv);

% display results
fprintf('B0inv:\n')
disp(B0inv);
fprintf('B0:\n')
disp(B0);
fprintf('SIGeps:\n')
disp(SIGeps);

% some checks
if norm(B0inv*SIGeps*B0inv' - VAR.SigmaOLS) > 1e-13
    error('result is incorrect: B0inv*SIGeps*B0inv''-VAR.SigmaOLS should be close to a zero matrix')
end
```

```matlab
% check that structural innovations are orthogonal to one another (result should be identity matrix for correlations)
epsi = B0*VAR.residuals;
for i = 1:var_nbr
    for j = (i+1):var_nbr
        cor_ij = corrcoef(epsi(i,:),epsi(j,:));
        if norm(cor_ij-eye(2)) > 1e-15
            error('structural innovations should be orthogonal to one another')
        end
    end
end

% compute and plot structural impulse response function
IRFpoint = irfPlots(VAR.Acomp,B0inv,nsteps,IRFcumsum,varnames_IRFs,epsnames);

%% Bootstrap confidence bands
B = 1000; % number of bootstrap replications
nvar = size(VAR.ENDO,2); % number of variables
THETAstar = zeros(nvar,nvar,nsteps+1,B); % storage for bootstrapped IRFs
opt.dispestim = false; % don't display results in VARReducedForm
optim_opt.Display = 'off'; % don't display results of fsolve
parfor b = 1:B
    ystar = bootstrapDGP(VAR);
    VARstar = VARReducedForm(ystar,VAR.nlag,opt);
    % redo identification steps exactly as in RWZSRLR
    A1starinv_big = inv(eye(size(VARstar.Acomp,1))-VARstar.Acomp); % from the companion
    LRMatstar = A1starinv_big(1:nvar,1:nvar); % total long-run impact matrix inv(eye(nvars)-A1hat-A2hat-...-Aphat)
    % Call optimization routine fsolve, note that we use point estimate B0inv_diageps as initial value
    [B0inv_diagepsstar,fval,exitflag,output] = fsolve(f,B0inv_diageps,optim_opt,VARstar.SigmaOLS,LRMatstar);
    B0invstar = B0inv_diagepsstar(:,1:var_nbr);
    SIGepsstar = diag(B0inv_diagepsstar(:,var_nbr+1));
    % use same normalization rules
    % normalization rules
    for j = 1:var_nbr
        if sign(B0invstar(j,j)) == -1
            B0invstar(:,j) = -B0invstar(:,j);
        end
    end
    % store results
    THETAstar(:,:,:,b) = irfPlots(VARstar.Acomp,B0invstar,nsteps,IRFcumsum,varnames_IRFs,epsnames,1); % the 1 at end disables plots
end
% compute standard deviation across b
IRFse   = std(THETAstar,0,4);
% set up confidence intervals
IRF95LO = IRFpoint - 1.96*IRFse;
IRF95UP = IRFpoint + 1.96*IRFse;
IRF68LO = IRFpoint - 1*IRFse;
```

```matlab
IRF68UP = IRFpoint + 1*IRFse;

figure('Name','Inference','units','normalized','outerposition',[0 0.1 1 0.9]);
countplots = 1;
x = 0:1:nsteps;
x_axis = zeros(1,nsteps+1);
for ishock = 1:nvar
    for ivar = 1:nvar
        subplot(nvar,nvar,countplots);
        irfpoint = squeeze(IRFpoint(ivar,ishock,:));
        irf95up  = squeeze(IRF95UP(ivar,ishock,:));
        irf95lo  = squeeze(IRF95LO(ivar,ishock,:));
        irf68up  = squeeze(IRF68UP(ivar,ishock,:));
        irf68lo  = squeeze(IRF68LO(ivar,ishock,:));

        % Plot 95% confidence interval as a shaded area
        fill([x fliplr(x)], [irf95lo' fliplr(irf95up')], 'r', 'FaceAlpha', 0.2, '
            EdgeColor', 'none');
        hold on;
        % Plot 68% confidence interval as a shaded area
        fill([x fliplr(x)], [irf68lo' fliplr(irf68up')], 'b', 'FaceAlpha', 0.3, '
            EdgeColor', 'none');

        % Plot the IRF point estimate
        plot(x, irfpoint, 'b', 'LineWidth', 2);

        % Plot x-axis line
        plot(x, x_axis, 'k', 'LineWidth', 2);

        grid on;
        xlim([0 nsteps]);
        title(varnames{ivar});
        ylabel([epsnames{ishock}, ' Shock']);
        countplots = countplots + 1;
    end
end
```