**First version: November 2016; updated October 2017.**

The Matlab codes replicate the results in Qu (2017): "**A Composite Likelihood Framework for Analyzing Singular DSGE Models**”. All folders and subfolders must be added to Matlab path for all the scripts to work. They have been tested on Matlab version 2015a.

This readme file is structured as follows.

First, I give some general information on how the replication files are organized.

Second, I use an example to illustrate how to carry out estimation and inference for a four shocks model under a given composite likelihood specification.

**************************

## ***GENERAL INFORMATION

**************************

Most importantly, the directory "Replication_scripts" contains scripts reproducing and displaying table output labeled by its number and panel, e.g., Table1_part_1.m reproduces the first part (i.e., panel (a)) of Table 1 in the main paper. Similarly, TableS2_part_1 reproduces the first part of Table S2 of the supplementary appendix. Inside these subfolders, there are further readme files that explain how the codes can be executed.

The codes for carrying out the identification analysis are also included under the "Replication_scripts" folder, see the subfolder /Identification/

The directory 'General' contains Sims' gensys routines as well as the gensys code modified to allow indeterminacy following the development in Lubik and Schorfheide (2003). You should not have to edit them.

The rest of subroutines are organized by model folder (i.e., Medium_scale_models and Small_scale_models). They are needed for running the codes in the “Replication_scripts” folder. You should not have to modify them.

********************************
*****AN EXAMPLE**
********************************

Here an example of replicating part 1 of Table 1 is provided to demonstrate the structure of replication scripts found in the directory "Replication_scripts".

There are five scripts in this subfolder. They will need to be run sequentially in order to obtain the results corresponding to the four shocks model in Table 1.

The approximating computational time is indicated at the beginning of each file.

The five files are:

F01_mode4.m – locates the quasi posterior mode.

F02_imat4.m – computes the variance of the score at the mode.

F03_mcmc4.m – obtains the mcmc draws.

F04_interval4.m – computes the mcmc and asymptotic intervals.

F05_table1_part1.m – uses the output from the above to produce the first four columns in Table 1.

I now explain these four individual files in some detail.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% F01_mode4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This procedure finds the quasi posterior mode using the GA algorithm followed by multistart.
%% The total runtime is about 3.3 hours on a Xeon E5-2665 8-core 2.4Ghz processor, using Matlab R2015a.


%% First, set up objective function to maximize and parameter bounds
ObjectiveFunction = @post_sw_4m; %the minus quasi posterior
lb=[0.010;0.010;0.010;0.010;0.010;2;0.250;0.0010;1;0.010;0.30;0.010;0.50;0.250;1.00;0.0010;0.0010;0.50;0.010;0.010;0.010;0.010;0.010;0.010;0.0010;0.010;0.0250;0.010;0.010;0.010;0.010;0.010;0.10;0.010;0.10;-10];
ub=[2;0.990;0.990;1;1;15;3;0.99000;3;0.99000;0.95000;0.99000;0.95000;10;3;0.50000;0.50000;0.97500;0.990;0.990;0.990;0.990;0.990;0.990;0.990;3;5;3;3;3;3;3;0.80000;2;2;10];
```

```
lb(13)=0.1;

%% Now pick out the parameters that appear in the four shocks model,
and the corresponding bounds
ind=[1,4:19,21:23,26,28:30,33:36];
lb=lb(ind)';
ub=ub(ind)';
numpar=length(lb);


%% Set up parallel computation
numcore=8; %specify the number of cores
% %for versions before R2014a
% matlabpool open local numcore
% %for versions R2014a and above
ppool=parpool('local',numcore); %create parallel pool object


%% Run optimization
warning('off','all')
%the algorithm used random starting values; suppress warnings for
cleaning output.
timega=tic;%time keeping
GA_optim % call optimization routine GA+Multistart
timelga=toc(timega); %time taken by GA/Multistart
postmod4=xestga2;
save postmod4 postmod4; %save the maximizer

%% Close parallel pool
% %for versions before R2014a
% matlabpool close
% %for versions R2014a and above
delete(ppool) %delete the parallel pool object
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% F02_imat4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The program computes the covariance matrix of the score at the quasi
posterior mode
%It is used later as the scaling matrix to generate MCMC draws.
%% The total runtime is about 2 minutes on a Xeon E5-2665 8-core
2.4Ghz processor, using Matlab R2015a.


%%load the quasi posterior model produced by the F01_mode4
load('postmod4.mat');
```

```
para=postmod4;
npara=length(para);

%% call subroutine to compute the variance of the score
score = score_4(para);
Imat4=score;
save Imat4 Imat4


% Create Inverse by Singular Value Decomposition
HHm   = score;
rankHHM=npara;
[u , s, v] = svd(HHm);
invHHMdet = 1;
for i=1:npara
   if i > rankHHM
      s(i,i) = 0;
   else
      s(i,i)    = 1/s(i,i);
      invHHMdet = invHHMdet*s(i,i);
   end
end
invHHM  = u*s*u';
fhmult4 = u*sqrt(s);


%%save the inverse
save fhmult4 fhmult4;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% F03_mcmc4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The program generates values from the quasoi posterior distribution
via the random walk Metropolis Hastings Algorithm
% To utilize multiple cores, several independent chains are produced
and then merged.
%% The total runtime (for 200000 draws) is about 3.3 hours on a Xeon
E5-2665 single-core 2.4Ghz processor, using Matlab R2015a.

%%time keeping
timega=tic;
```

```
%% Load parameter bounds
lb=[0.010;0.010;0.010;0.010;0.010;2;0.250;0.0010;1;0.010;0.30;0.010;0
.50;0.250;1.00;0.0010;0.0010;0.50;0.010;0.010;0.010;0.010;0.010;0.010
;0.0010;0.010;0.0250;0.010;0.010;0.010;0.010;0.010;0.10;0.010;0.10;-
10];
ub=[2;0.9999;0.9999;1;1;15;3;0.99000;3;0.99000;0.95000;0.99000;0.9500
0;10;3;0.50000;0.50000;0.97500;0.9999;0.9999;0.9999;0.9999;0.9999;0.9
999;0.9999;3;5;3;3;3;3;3;0.80000;2;2;10];
lb(13)=0.1;
compo=[1,4:19,21:23,26,28:30,33:36];
lb=lb(compo);
ub=ub(compo);



%% Specify input parameters
cc=0.1; %scaling parameter when generating proposal mcmc draws
cc0=1;  %scaling parameter for generating the initial mcmc draw

nsim    = 200000; %total number of draws

nchain=8; %number of independent chains

npara=28;%number of parameters


%% load the scaling matrices for generating the proposal mcmc draws
load fhmult4;
load postmod4;
para=postmod4;
SIGSCALE=fhmult4;

%%specify the number of draws by each chain
simj=nsim/nchain;


%% set up parallel computation
numcore=8; %specify the number of cores
%
% %for versions before R2014a
% matlabpool open local numcore
%
% %for versions R2014a and above
ppool=parpool('local',numcore); %create parallel pool object


%% specify the criterion function, i.e., log composite likelihood
plus prior
ObjFunction = @post_sw_4;
```

```
%% running the chains
parfor j=1:nchain
     mchain(ObjFunction,para,cc,cc0,SIGSCALE,lb,ub,simj,j);
     %the individual chains are saved automatically within mchain
end


%% Close parallel pool
% %for versions before R2014a
% matlabpool close
%
% %for versions R2014a and above
delete(ppool) %delete the parallel pool object

%Now load individual files, merge them and keep only the merged file
load parasim1;
pdraws=parasim;
clear parasim;
delete parasim1.mat;
for j=2:nchain
    filename=sprintf('parasim%d.mat',j);
    load (filename);
    pdraws=[pdraws;parasim];
    clear parasim;
    delete (filename);
end


%%save the output
parasim4=pdraws;
save parasim4 parasim4;
timelga=toc(timega);
```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% **F04_interval4**
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```
%%% Computes the mcmc and asymptotic intervals using the draws
%% The computational time is about 2 minutes.


%load the mcmc draws
load parasim4;
parasim=parasim4;
load postmod4;
```

```
%specify parameter bounds
lb=[0.010;0.010;0.010;0.010;0.010;2;0.250;0.0010;1;0.010;0.30;0.010;0
.50;0.250;1.00;0.0010;0.0010;0.50;0.010;0.010;0.010;0.010;0.010;0.010
;0.0010;0.010;0.0250;0.010;0.010;0.010;0.010;0.010;0.10;0.010;0.10;-
10];
ub=[2;0.9999;0.9999;1;1;15;3;0.99000;3;0.99000;0.95000;0.99000;0.9500
0;10;3;0.50000;0.50000;0.97500;0.9999;0.9999;0.9999;0.9999;0.9999;0.9
999;0.9999;3;5;3;3;3;3;3;0.80000;2;2;10];
lb(13)=0.1;
compo=[1,4:19,21:23,26,28:30,33:36];
lb=lb(compo);
ub=ub(compo);


ndraw=rows(parasim);
mpara=mean(parasim,1);

% specify the coverage
clevel=0.95;
cv1=floor(ndraw*(1-clevel));
cv2=floor(ndraw*clevel);
spara=parasim;


%sort parameter values to obtain mcmc intervals
for j=1:28
   tempv=sort(parasim(:,j));
   spara(:,j)=tempv;
end
display('-- 90% mcmc intervals:-----');
mcmcind=[spara(cv1,:);spara(cv2,:)];
display(mcmcind');

%save the results;
save mcmcind mcmcind;


%obtain asymptotic intervals
dpara=spara;
for j=1:rows(parasim)
dpara(j,:)=parasim(j,:)-mpara;
end
score = score_4(postmod4);
invhess=dpara'*dpara/rows(dpara);
[v,d] = svd(invhess);
d=inv(d);
H12 = v*sqrt(d)*v';
[v,d] = svd(score);
S12 = v*sqrt(d)*v';
newpara=invhess*S12*H12*dpara';
newpara=newpara';
center=mpara(ones(ndraw,1),:);
```

```
draw1=newpara+center;
for j=1:28
   tempv=sort(newpara(:,j));
   spara(:,j)=tempv;
end
display('-- 90% asymptotic intervals:-----');
asyind=[max(spara(cv1,:)+mpara,lb');min(spara(cv2,:)+mpara,ub')];
display(asyind');
save asyind asyind;
draw4=draw1;

%save the results
save draw4 draw4;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% F05_table1_part1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% this procedure reorganizes the output from the previous procedure, to generate a display that matches with the table
% takes 1 minute to run

% load output, re-order parameters, and display
load postmod4;
theta=postmod4;
theta0=[theta(1),0,0,theta(2:17),0,theta(18:20),0,0,theta(21),0,theta(22:24),0,0,theta(25:28)];
ord=[4:24,3,25,2,1,26:36];
display('-- Mode:-----');
theta0(ord)

% load output, re-order parameters, and display
load parasim4;
theta=mean(parasim4);
theta0=[theta(1),0,0,theta(2:17),0,theta(18:20),0,0,theta(21),0,theta(22:24),0,0,theta(25:28)];
ord=[4:24,3,25,2,1,26:36];
display('-- Mean:-----');
theta0(ord)

% load output, re-order parameters, and display
load mcmcind;
theta=mcmcind(1,:);
thetal0=[theta(1),0,0,theta(2:17),0,theta(18:20),0,0,theta(21),0,theta(22:24),0,0,theta(25:28)];
theta=mcmcind(2,:);
thetau0=[theta(1),0,0,theta(2:17),0,theta(18:20),0,0,theta(21),0,theta(22:24),0,0,theta(25:28)];
display('-- MCMC Interval:-----');
```

```
[thetal0(ord);thetau0(ord)]


% load output, re-order parameters, and display
load asyind;
theta=asyind(1,:);
thetal0=[theta(1),0,0,theta(2:17),0,theta(18:20),0,0,theta(21),0,thet
a(22:24),0,0,theta(25:28)];
theta=mcmcind(2,:);
thetau0=[theta(1),0,0,theta(2:17),0,theta(18:20),0,0,theta(21),0,thet
a(22:24),0,0,theta(25:28)];
display('-- Asymptotic Interval:-----');
[thetal0(ord);thetau0(ord)]

%THE END!
```