```
% SetDSGE
%
% This file sets the DSGE environment
%
% Mandatory structure:
%
% 1. Set file name for the output
%    FileName.Output (string)
%
% 2. Set variable Actions
%    Possible Actions:
%      Actions = {'All'};
%      Actions = {'Setup','MaxPost','MCMC'};
%        Can also use a subset of this, assuming that the previous stages are
%        completed.
%
% The following are required for the Setup stage:
%
% 3. Set data input
%      FileName.Data (string)
%
% 4. Set parameters list and priors: Params
%    lists all parameters in a vertical array, with 4 or 5 elements per column:
%      1st: (string) name of parameter
%      2nd: (string) name of distribution
%           Valid names: 'N','B','G','IG1','IG2'
%      3rd: (double) mean of prior
%      4th: (double) SE of freedom of prior. If prior dist is igam then
%           either specify the se or set it to inf, in which case the
%           codes will assume degrees of freedom equal to 2 and extract
%           the other parameter from the mean.
%      5th: (string, optional) Pretty representation of variable name, to
%           be used in plot titles and tables. If included, it needs to be
%           included in all. If ommited, the original names are used.
%
% 5. Set list of observation variables: Obs
%    This must be a cell array of strings.
%
% 6. Set list of State space variables: States
%    This must be a cell array of strings.
%
% 7. Set list of iid shocks: Shocks
%    This must be a cell array of strings.
%    NOTE: Shocks need to be iid and with unit variance.
%
% 8. Generate symbolic variables: GenSymVars
%    Script to automate all required transformations.
%    NOTE: this must be called prior to specify any auxiliary calculations
%          or the equations in the model.
%    NOTE: in the equations below if constants show up then they should be
%          multiplied by 'one', which is defined as a symbolic variable to
%          be used later to identify those constants.
```

```
%    Convention: x_t refers to x(t)
%                x_tF refers to x(t+1)
%                x_tL refers to x(t-1)
%                x_ss refers to steady state of x(t)
%
% 9. Construct any necessary auxiliary definitions [OPTIONAL]
%    In this section introduce any calibrated parameters and/or parameter
%    transformations to be called on the equations.
%    NOTE: This section should be called after generating symbolic
%          variables and before setting the equations.
%
% 10. Set observation equations: ObsEq
%     Symbolic column array.
%     NOTE: Cannot contain any lags or leads. If needed augment state space
%           representation.
%
% 11. Set state Equations: StateEq
%     Symbolic column array.
%     Each state equation equations can contain lags and leads, but not both
%     simultaneous in the same equation. If leads and lags in the same
%     equation, then create artificial variables for the lagged variables.
%
% The above 11 steps will set up the model. At this point either list the
% other scripts to estimate and manipulate the DSGE or simply call them
% separately. Refer to section "See also" for a list of all scripts and
% functions that can be called.
%
% Optional Settings:
%
%   UseParallel (logical)
%   If set to true, it uses parallel computing in several stages of the
%   estimation, most remarkably on the posterior maximization and on the
%   MCMC part.
%
%   DataVarName
%   Name of the variable in the datafile. Needs to be specified if
%   datafile contains multiple variables. Otherwise it does not have to
%   be specified.
%
% See also:
% GenSymVars, DataAnalysis, PriorAnalysis, GenPost, MaxPost,
% MaxPostFcn, MakeTableMaxPost, MCMC, MCMCFcn, MCMCSearchScaleFactor,
% MakePlotsMCMCConv, MCMCInference, MakeTableMCMCInference,
% MakePlotsMCMCTrace, MakePlotsMCMCPriorPost, MCMCConv, MakeTableMCMCConv
%
% ..........................................................................
%
% Created: March 17, 2008 by Vasco Curdia
% Updated: February 3, 2015 by Vasco Curdia
%
% Copyright (C) 2008-2015 Vasco Curdia
```

Preamble

```
clear all
tic
ttic = toc();
```

Settings

```
FileName.Output = 'Baseline';
```

Parallel options

```
UseParallel = 0;
nMaxWorkers = 4;
if UseParallel,matlabpool('open',nMaxWorkers),end
```

Allow for running only some of the blocks of actions

Possible Actions: Actions = {'All'}; Actions = {'Setup','MaxPost','MCMC'};

```
Actions = {'All'};
```

Load framework if already set

```
if ~any(ismember({'All','Setup'},Actions))
  save NewSettings ttic ListPathDependencies Actions UseParallel nMaxWorkers
  load(FileName.Output)
  load NewSettings
  delete NewSettings.mat
else
```

Setup

Data

```
FileName.Data = 'data_greenspan_bernanke_20091204';
nPreSample = 0;
DateLabels.Start = '1987q3';
DateLabels.End = '2009q3';
TimeIdx = TimeIdxCreate(DateLabels.Start,DateLabels.End);
```

```
DateLabels.XTick = find(ismember(TimeIdx,{'1990q1','1995q1','2000q1','2005q1'}));
DateLabels.XTickLabels = {'1990','1995','2000','2005'};
```

Estimated parameters

```
Params = {...
    'omega', 'G', 1, 0.2,'\omega';
    'xi', 'G', 0.1, 0.05,'\xi';
    'eta', 'B', 0.6, 0.2,'\eta';
    'zeta', 'B', 0.6, 0.2,'\zeta';
    'rho', 'B', 0.7, 0.15,'\rho';
    'phipi', 'N', 1.5, 0.25,'\phi_\pi';
    'phix', 'N', 0.5, 0.2,'\phi_x';
    'pistar', 'N', 2, 1,'\pi^*';
    'ra', 'N', 2, 1,'r^a';
    'gammaa', 'N', 3, .35,'\gamma^a';
    'rhodelta', 'B', 0.5, 0.2,'\rho_\delta';
    'rhogamma', 'B', 0.5, 0.2,'\rho_\gamma';
    'rhou', 'B', 0.5, 0.2,'\rho_u';
    'sigmadelta', 'IG1', 0.5, 2,'\sigma_\delta';
    'sigmagamma', 'IG1', 0.5, 2,'\sigma_\gamma';
    'sigmau', 'IG1', 0.5, 2,'\sigma_u';
    'sigmai', 'IG1', 0.5, 2,'\sigma_i';
    };
zeta=[];
```

Observation variables

```
ObsVar = {'gYa';'pia';'ira'};
```

State space variables

```
StateVar = {...
    % Regular variables
    'xtil';'YA';'pitil';'pi';'ir';'r';
    'xe';'re';'YAe';
    'delta';'gamma';'u';
    % a couple of artificial variables
    'YAL';'YAeL';
    };
```

Shocks

```
ShockVar = {'edelta';'egamma';'eu';'ei'};
```

create symbolic variables

```
GenSymVars
```

Auxiliary definitions

```
beta = 0.99;
gamma = gammaa/400;
r = ra/400;
phigammatil = exp(gamma)/(exp(gamma)-beta*eta);
etagammatil = exp(gamma)/(exp(gamma)-eta);
phigamma = phigammatil*etagammatil;
etagamma = eta/exp(gamma);
```

Observational Equations

```
ObsEq = [...
    gammaa*one+400*(YA_t-YAL_t+gamma_t) - gYa_t;
    pistar*one+400*pi_t - pia_t;
    (ra+pistar)*one+400*ir_t - ira_t];
```

State Equations

```
StateEq = [...
    % IS Block
    xtil_tF-phigamma^(-1)*(ir_t-pi_tF-re_t)-xtil_t;
    (xe_t-etagamma*(YAL_t-YAeL_t))-beta*etagamma*(xe_tF-etagamma*xe_t)-xtil_t;
    ir_t-pi_tF - r_t;
    % Efficient Rates
    YA_t-YAe_t-xe_t;
    YAe_tF-omega^(-1)*(gamma_tF-re_t+delta_tF)-YAe_t;
    -phigamma*(YAe_t-etagamma*(YAeL_t-gamma_t)-...
        beta*etagamma*(YAe_tF+gamma_tF-etagamma*YAe_t))+...
        beta*etagamma/(1-beta*etagamma)*delta_tF-omega*YAe_t;
    % PC Block
    beta*pitil_tF+xi*(omega*xe_t+phigamma*xtil_t)+u_t-pitil_t;
    pi_t-zeta*pi_tL-pitil_t;
    % Policy Rule
    rho*ir_tL+(1-rho)*(phipi*pi_t+phix/4*xe_t)+sigmai/400*ei_t-ir_t;
    % Shocks
    rhodelta*delta_tL+sigmadelta/400*edelta_t-delta_t;
    rhogamma*gamma_tL+sigmagamma/400*egamma_t-gamma_t;
    rhou*u_tL+sigmau/400*eu_t-u_t;
    % Auxiliary equations
    YAL_t-YA_tL;
    YAeL_t-YAe_tL;
    ];
```

Data

```
DataAnalysis
```

Make REE mats

```
MakeMats
```

Priors

```
PriorAnalysis
```

Generate posterior function

```
GenPost
```

end Setup Action if

```
TimeElapsed.Setup = toc();
fprintf('\n%s\n\n',vctoc([],TimeElapsed.Setup))
save(FileName.Output)
end
```

MaxPost

```
if any(ismember({'All','MaxPost'},Actions))
    nMax = 20;
    MinParams.H0 = diag([Params(:).priorse].^2);
    MinParams.crit = 1e-8;
    MinParams.nit = 1000;
    MinParams.Ritmax = 30;
    MinParams.Ritmin = 10;
    MinParams.RH0 = 1;
    MinParams.Rscaledown = 0.5;
    MaxPost
    save(FileName.Output)
    save([FileName.Output,'MaxPost'])
end
```

MCMC

```
if any(ismember({'All','MCMC'},Actions))
    nChains = 4;
    nDrawsSearch = 1000;
    dscale = [0.2,0.05,0.01];
    BurnIn = 0.25;
    nThinning = 1;
    nDraws = 200000;
    for nUpdate=1
        fprintf('\n*****************')
        fprintf('\n* MCMC Update %.0f *',nUpdate)
        fprintf('\n*****************')
        MCMCOptions.ScaleJumpFactor = 2.4;
        MCMCSearchScaleFactor
        save(sprintf('%sMCMCUpdate%.0f_SSF',FileName.Output,nUpdate))
        MCMC
        save(FileName.Output)
        save(sprintf('%sMCMCUpdate%.0f',FileName.Output,nUpdate))
        delete(sprintf('%sMCMCUpdate%.0f_SSF.mat',FileName.Output,nUpdate))
        MCMCAnalysis
        save(FileName.Output)
        save(sprintf('%sMCMCUpdate%.0f',FileName.Output,nUpdate))
    end
end
```

Close matlabpool

```
if UseParallel,matlabpool close,end
```

elapsed time

```
fprintf('\n%s\n\n',vctoc(ttic))
```

Save environment

```
save(FileName.Output)

%%-------------------------------------------------------------------------
```