Toolkit Documentation of "Loose Commitment in Medium-Scale Macroeconomic Models: Theory and an Application"

Davide Debortoli — UC San Diego
Junior Maih — Norges Bank
Ricardo Nunes — Federal Reserve Board

This version: November 2010

This note documents the toolkit to solve loose commitment settings easily in medium- and large-scale linear-quadratic models. If you use or modify these codes, please cite the paper "Loose Commitment in Medium-Scale Macroeconomic Models: Theory and an Application".

The first section describes the installation and a simple example. The second section describes the example used in the paper. The paper replication codes allow the user to explore the possibilities and options in the toolkit quite extensively. The third section discusses the specific files in the toolkit.

The toolkit codes can be downloaded at dss.ucsd.edu/~ddebortoli/ or ricardonunes.net. These codes are written in Matlab and have been tested in version 7.7. The toolkit integrates with Dynare and has been tested with version 4.1.1. Do not add the directory to the Matlab path.

The files needed for the toolkit are contained in the main folder, while the two subfolders contains the files for the two specific examples described below.

1 Getting started

The first step consists in writing your `.dyn` file, where the model of interest is specified. At the beginning of your file, specify the location of the toolkit files, adding the line:

```
addpath('[destination folder]','-begin');
```

Here we provide two examples.

1.1 Example: a standard New-Keynesian model

The folder `NK_example.m` contains the files related to a simple New-Keynesian model, whose only structural equation is a standard New-Keynesian Phillips curve, and dynamics are driven by an AR(1) cost-push shock.

The model is declared as follows:

```
varexo E_Y;
var PAI OUT;
parameters ProbabilityOfCommitment;
model(linear);
   PAI = 0.99*PAI(+1)+0.1*OUT+E_Y;
end;
shocks;
   var E_Y   ; stderr .01;
end;
```

and the policymaker's objective function is declared with the command lines:

```
planner_objective -.5*(1*PAI^2+0.048*OUT^2);
options_.planner_discount = 0.99;
```

Models with more lags and leads can be specified in their original formulation, since the code automatically transforms them into the compact formulation used in the paper.

The following commands initiate the loose commitment toolkit:

```
Compare=0;
if Compare;
   ramsey_policy(nograph,nomoments); //,nograph
else
   options_.loosecommit= 1;
   ProbabilityOfCommitment=.5;
   stoch_simul(nograph,nomoments,noprint);
end
```

Setting `options_.loosecommitment = 1` tells the program to use the toolkit.[1]

The command `stoch_simul` permits to solve the model. The solution of the model, as well other information, is stored in the structured variable `oo_`. In particular, the law of motion is summarized by the matrices `oo_.dr.ghx` and `oo_.dr.ghu`.

Once the solution has been obtained, the user can produce simulations and statistics according to her needs. For convenience, the toolkit already provides additional files to generate impulse responses, second moments, and welfare calculations. In particular:

1. Impulse response functions are executed with the commands:

   ```
   Periods2=20;   Sim_nbr=1;   InitShocks=[];   InitVals=[];
   CommitmentHistory=ones(Periods2,1);
   IRF=LooseCommitmentIrf(Periods2,Sim_nbr,InitShocks,InitVals,...
   CommitmentHistory);
   figure('name','Never reoptimization');
   plot(IRF')
   legend(M_.endo_names)
   ```

   The sample codes contains several additional examples for IRFs – where the history of commitment shocks, initial values, and scope of the IRFs are changed.

2. Moment calculations are executed with the commands:

   ```
   Periods2=500; Sim_nbr=1500; Burn=100; InitVals = [];
   CommitmentHistory=[];
   MOM=LooseCommitmentMoments(Periods2,Sim_nbr,Burn,InitVals,...
   CommitmentHistory);
   disp 'Simulated Moments'
   disp(M_.endo_names)
   disp(MOM)
   ```

3. Welfare calculations are executed with the commands:

[1]The line referring to the `ramsey_policy` even if it is not read is necessary to execute several intermediate steps.

```
[UncondWelf10,CondWelf10]=LooseCommitmentWelfare;
v=0; Y_Mu_0=rand(M_.endo_nbr,1);
[UncondWelf11,CondWelf11]=LooseCommitmentWelfare(Y_Mu_0,v);
```

The first line considers the initial conditions and steady-state to be at zero. The second command computes welfare for different initial conditions.

2 The Smets and Wouters (2007) model

The folder `SW_example.m` contains the files used to generate the main results in the paper. The main file is `SW_main.dyn`, which starts by calling the following files:

- `SW_model.dyn`, defining the model equations and calibration;
- `SW_objective_planner_benchmark.dyn` and `SW_objective_planner_alternative.dyn`, setting the two specifications of the central bank loss function analyzed in the paper.

These two files are the only files that need to be adapted when solving a different model.

The user is then required to specify some options. In particular, the user can choose to solve the model for multiple degrees of commitment (including full-commitment and discretion). Accordingly, the core of the program iterates on the possible degrees of commitment, as follows

```
if Compare;
    ramsey_policy(nograph,nomoments) pinf y yf r; //,nograph
    disp('Problem solved using Ramsey Policy')
else
    for j = [iter:-1:1];
        ProbabilityOfCommitment=probCOM_grid(j);
        stoch_simul(nograph,nomoments,noprint);
        [...]
    end
end
```

For all the degrees of commitment – contained in the vector `probCOM_grid` – the model is solved using the file `stoch_simul.m` and, if desired, welfare, impulse response functions and second moments are computed by calling the functions `LooseCommitmentWelfare.m`, `LooseCommitmentIrf.m` and `LooseCommitmentMoments.m`, respectively. Finally, the output is reorganized and displayed on the screen using the file `SW_showresults.m`, which makes use of the toolkit files `plot_IRFs.m` and `show_tables.m`.[2]

The policy frontiers and the Monte-Carlo simulations reported in the paper are produced using two similar files – `SW_frontier.m` and `SW_regression.m`, respectively.

3 File documentation of the main toolkit files

- `stoch_simul.m`: When the option `options_.loosecommit=1` in the `.dyn` file, the toolkit will be used. `stoch_simul.m` is an intermediate file that integrates the toolkit with Dynare. This file temporarily substitutes the original stoch_simul.m dynare file, and later versions will incorporate this toolkit directly.

- `LooseCommitment.m`: This file transforms the equations of the model according to the formulation in the paper, and then solve the problem (using SolveLooseCommitment.m). Options to be set:

  - `qz_criteriumLC:` determines the cutoff point to characterize an eigenvalue to be explosive (default is 1.000001);
  - `MaxIterLC:` determines the maximum number of iterations for convergence (default is 3000);
  - `critLC:` determines the the convergence criteria (default is 1e-7);
  - `noprint:` determines whether the code prints an output in the command window (default is 0 for printing).

  Variables of interest: The variable `Hold` is conveniently used to store the latest solution. This is useful when solving the model for different degrees of commitment, so that an homothopy method can be exploited.

[2]The policy frontiers and the Monte-Carlo simulations reported in the paper are produced using two additional files, and are available upon request.

`M_.endo_names` and `M_.exo_names` are character vectors with the names of the endogenous and exogenous variables, respectively.

- `SolveLooseCommitment.m`: This file is the main engine for the solution procedure. It executes the iteration loop described in the paper, and exits successfuly if

  $$max(H - Hold) < critLC,$$

  and produces an error if the maximum number of iterations is reached, the model is unstable, or if an unspecified error occurs.

- `GetDynareLooseCommitmentResults.m`: This file conveniently reorganizes the output of Dynare (stored in `oo_`). After the solution is computed, this file recovers all the necessary information and passes them to other subcodes to compute moments, IRFs, etc. This file shows where each variable is kept in memory.

- `LooseCommitmentIrf.m`: This file computes the impulse response functions. If the inputs to the file are not passed, the file resets those but the order of inputs skipped needs to be in the correct order (see code for the specific details). Inputs to the file are:

  - `Periods`: number of periods in the simulation (default is 40).
  - `Sim_nbr`: number of simulations (default is 1000).
  - `InitShocks`: vector of initial conditions for the shocks. If this input is empty, then it is drawn stochastically. The initial condition for the specific IRF shock is always reset to one positive standard deviation.
  - `InitVals`: vector of initial conditions for the variables. If this input is empty, then it is set to a vector of zeros (assumed to be the steady-state).
  - `CommitmentHistory`: vector of re-optimization shocks at each period (0 for reoptimization, 1 otherwise). The code uses the commitment history for each simulation. If this option is empty, the commitment history is re-sampled stochastically (default is resampling stochastically).

The code takes the difference between the series where the initial IRF shock is standardized to one positive standard deviation, and the series where it is set to zero. The seed of the random number generator is not set inside this function and needs to be set in the `.dyn` file (see example files). The file produces the IRF for all shocks and all series. To identify specific series by name we provide the file `find_var.m`. To plot the series, we provide the file `plot_IRFs.m`.

- `LooseCommitmentMoments.m`: This file computes the variance of the series and its structure is similar to the impulse response functions file.[3] However, the series computed are not in deviations to a benchmark. Inputs to the file are:
  - `Periods, InitVals, Sim_nbr`, and `CommitmentHistory`: options equal to file `LooseCommitmentIrf.m`.
  - `Burn`: number of periods to discard in the simulation (default is 100). Total number of periods in the simulation is given by input Periods (not Periods minus Burn).
  - `shocks_sel`: allows to turn off some shocks in case the corresponding element is set to zero (default is vector of ones). This option is useful to compute conditional moments and variance decompositions.

  Option InitShocks is not used, and shocks are always stochastic.

- `LooseCommitmentWelfare.m`: This file computes conditional and unconditional (on the initial shocks) welfare. The corrections discussed in the paper when $\lambda_{t-1} \neq 0$ are incorporated. Inputs to the file are:
  - `Y_Mu_0`: vector of initial conditions for the variables. If this input is empty, then it is set to a vector of zeros.
  - `v`: vector of initial conditions for the shocks. If this input is empty, then it is set to a vector of zeros.

[3]This file can be easily adapted to produce averages or other moments.