OPDSGE (formerly lq_solution): A DYNARE-based Matlab-code for Optimal Policy in DSGE models

Giovanni Lombardo

Bank for International Settlements

First Version 2007 – Update 2017

Abstract

This note explains how to use the OPDSGE toolbox (formerly lq_solution but essentially unchanged). This toolbox derives optimal policies for DSGE models in closed and open economies. For the latter it allows to derive cooperative and non-cooperative policies. It also generate the quadratic welfare function following the algorithm suggested by Benigno and Woodford (2011). Starting from a DSGE model written in DYNARE-compatible syntax, this toolbox computes the non-linear optimal (cooperative or non-cooperative) policy and rewrites the result in a new DYNARE-compatible (.mod) file. Optimal and sub-optimal policies can then be compared using the LQ solution technique. This toolbox is meant to be used in conjunction with DYNARE.

1 The toolbox

The toolbox consists of two parts: The *OPDSGE* folder and the *utilities* sub-folder. The latter contains auxiliary functions mainly needed to manipulate strings. Furthermore in the *example* folder you find a simple (Rotemberg-type of) model used to test this toolbox (The test is not very thorough!). The *example* folder also contains a two-country model used to compute the non-cooperative and cooperative equilibria.

1.1 Core of the toolbox

OPDSGE.m The $OPDSGE.m$ is the main code. This code rewrites (rather inefficiently) the original model, as read out of the .mod file, into a new model that conforms with the notation used in Benigno and Woodford (2011) (BW henceforth).[1] The new model is written in a new .mod file. The name of this file is <name of the original file><suffix>.mod, where <suffix> is "_lq" if not otherwise specified.

For **example**, the "suffix" coul be "optimal", when the optimal policy is computed, and "suboptimal" when the model is only parsed but no optimal policy is computed. The two models can then be used for welfare comparison as discussed below.

If requested, the new .mod file will also include the FOC of the Lagrange problem obtained by combining the objective function with all the structural equations excluding exogenous shocks.

In order to activate the **derivation of FOCs**, must append a tag to the policy rules (e.g. append + poly to the policy rule, where poly is a declared parameter with value=0.)

This new file can used directly to run DYNARE.

Example: add the policy tag to the policy equation and run $OPDSGE[options]$. This will produce an optimal policy model. Then remove the tag and run $OPDSGE[options]$ giving a different extension (see below how to run $OPDSGE$).

Now have a reference model and a sub-optimal-policy model. Both can be solved using DYNARE.

1.2 lq_eval_welfare

The OPDSGE program can also be used to compute the LQ solution of Benigno and Woodford (2011)

This file must be run after DYNARE is run on your preferred .mod file that you have previously generated with $OPDSGE.m$.

In the first few lines of $lq\_eval\_welfare$ have some input you have to give. In particular the "namemod" variable must be assigned to your reference model (e.g. the optimal Ramsey model); The rest should be general.

Proceed as follows

1. Run DYNARE on your reference model

2. Run **lq_eval_welfare**

3. Agree to save the requested information

4. Run DYNARE on the alternative model (e.g. suboptimal policy)

---

[1]Notice that the solution technique has also been discussed by Levine et al. (2006)

5. Run **lq_eval_welfare**

6. Don't save

7. Read on screen the welfare comparison.

Note: if there are lagged variables **for which the parser creates dummies**, you have to extend the sub-optimal model to include the dummies generated in the optimal model. That is, generate the optimal model starting from a model without the dummies. Add the dummies to the original model.

lq_symbol_welf   If you run lq_symbol_welf you will get a partially symbolic expression for the welfare function: i.e. the variables will appear in symbolic form. This allows you to see which variables appear in the quadratic welfare function.

lq_soc   This code (called from lq_eval_welfare), computes the second order conditions of the optimal problem as described by BW.

1.3   Non-cooperative policy games

The OPDSGE toolbox can handel non-cooperative policy games.[2]

For this to work you have to specify the objective function as the weighted sum of the "home" and "foreign" welfare. I suggest to define it as $Util = uu\_h * Ut\_h + uu\_f * Ut\_f$, where Ut_h is the welfare measure for the "home" country and Ut_f is the one for the "foreign" country. uu_h and uu_f are $0 - 1$ parameters.

In this case a further parameter, called "nash_param" will appear in the policy problem.

After having generated the new .mod file with $OPDSGE.m$, you can evaluate the cooperative equilibrium setting the auxiliary variable $nash\_param = 1$; $nash\_param = 0$ will produce the "non-cooperative" equilibrium.

The cooperative equilibrium can be taken as the reference policy when running lq_eval_welfare

1.4   The .mod file

The $OPDSGE$ toolbox runs on models written in DYNARE notation in a .mod file.

There are only few syntax rules that must be followed in order to make the .mod file readable by this toolbox.

---

[2]Currently only one instrument per player.

1.4.1 Syntax rules

1. The list of variables must be separated by commas (e.g. var C,Y,L;)

2. For the sake of comparison I've tried to make the new .mod file compatible with the syntax required by *get_ramsey* notation (see Levin and López-Salido (2004)). As it is difficult to foresee everything, have a second look at the mod file (or wait for it to crash on *get_ramsey* and check why)

3. The equations of the exogenous shocks should enter last in the model block of the .mod file and preceded by the comment \\. Exogenous Shocks;

4. At present no **lags** larger than 1 or **leads** larger than 1 are allowed.[3]

5. If want to have have FOCs of the optimal rule you must add a tag (e.g. R+poly), where the tag must be a listed parameter (just assign a zero to it). This tag will identify the policy rule.

1.5 The _steadystate.m file

DYNARE allows the user to provide a MATLAB function that returns the steady state of the model. The name of this function must be <name of the mod file>_steadystate.

The OPDSGE toolbox assumes that this function exists. Furthermore it assumes that this function takes the following form

function [ys_,flag]=<name of the mod file>_steadystate(x)

$\vdots$

[solution of the steady state and assignment of values to variables (user specific)]

$\vdots$

% assign to lgy_ the name of the variables in the order used by DYNARE
lgy_=evalin('base','lgy_');

$\vdots$

[some way to assign the steady-state variables to the ys_ output of this function]
e.g. for kkk=1:size(lgy_,1); ys_(kkk)=eval(lgy_(kkk,:)); end;

There is a reason why this function must be written in this form. The toolbox will make a copy of this function for the new mod file and it will write the new auxiliary steady-state definition right above the "lgy_=evalin('base','lgy_');" command. This auxiliary definitions are

---

[3]Obviously, if the model need to have this extra leads and lags, the user must replace them with auxiliary variables that go back or forward in time period by period.

generated only in conjunction with new variables when this is necessary in order to conform with BW notation (i.e. when forward looking equations display backward-looking variables or leads of shocks appear in the equations).

Notice that you don't really have to use this steady-state function if you want to use DYNARE built-in solver. In this case you just need to rename the new _steadystate file (or delete it) while making sure that the steady-state of auxiliary variables (if exist) is somehow made known to DYNARE (e.g. add it to the "initval" block);

1.6 Running $OPDSGE.m$

The toolbox contains one code that must be run only once on a new model (i.e. every time you change the equations but not every time you change the parameter values)

To run this part of the toolbox simply type in the command window of MATLAB

$>>$ OPDSGE $<$list of 9 entries$>$ separated by space

1. name_of_original_mod_file (obvious)

2. Util (name of variable describing utility)

3. Welf (name of variable describing Welfare... not really used)

4. policy_id (tag to the policy equation: if the tag is not found... all equations are used as constraints)

5. lagr_name (Name assigned to the Lagrange multipliers (they are going to be numbered)

6. ext_ss (Possible extension used in the model to denote steady-state variables[4]

7. extension (suffix used to save the new mod file)

8. 0/1 value: 1= symbolic evaluation of matrices will be computed; 0= numeric evaluation of matrices.

9. 0/1 value: 1=non-cooperative solution will be generated; 0= Standard solution will be generated. Notice that for the optimal (cooperative or non-cooperative) solution to be computed you must add a tag to the policy rules (see above).

If one of the inputs is not given, you will be prompted to supply the missing one on the command line.

---

[4]You must be careful when using steady-state variables in the model as they might affect the optimal policy. In particular, if policy is not neutral in the steady-state, steady-state variables should not appear in the model.

2 Initial Condition and Timeless Perspective

The current version of OPDSGE does compute the penalty term in the BW LQ solution that takes into account the violation of the initial conditions.

This is done by saving information on the solution of the optimal model. This information is then retrieved when evaluating another model in order to compute the unconditional covariance between the optimal Lagrange multipliers (on the forward-looking block) with the endogenous variables under the sub-optimal model. (see *penalty_term.m*)

3 The NK-Rotemberg model

This code has been tested only on a very simple model for which an analytical solution of the quadratic welfare function can be easily derived.

The model is a forward looking model with sticky prices à la Rotemberg (The mod file is nk_model_rot.mod, parameters are in param_nk_model.m) and productivity shocks.

The household utility function is

$$U = shock\_pref \left(\frac{C^{1-\gamma}}{1-\gamma} - \frac{L^{1+\varsigma}}{1+\varsigma} \right) \qquad (1)$$

Assuming that $\gamma = 2$, $\varsigma = 1$, prices are flexible ($xip = 0$) and that there are subsidies to offset the mark-up distortion, I obtain after running **lq_symbol_welf**

$$\widetilde{W} = -\widetilde{C}^2 - 1/2\,\widetilde{L}^2 + \left(\widetilde{C} - \widetilde{L} \right) shock\_pref + \widetilde{C} shock\_prod \qquad (2)$$

The presence of subsidies imply that the welfare function is purely quadratic. Under flexible prices $L = C/shock\_prod$. One can see that the result is indeed the second order expansion of (1).

Please cite as Coenen et al. (2009)

References

Benigno, P. and Woodford, M. (2011). Linear-Quadratic Approximation of Optimal Policy Problems. *Journal of Economic Theory.*

Coenen, G., Lombardo, G., Smets, F., and Straub, R. (2009). International Transmission and Monetary Policy Coordination. In Galí, J. and Gertler, M., editors, *International Dimensions of Monetary Policy*. University Of Chicago Press, Chicago.

Levin, A. T. and López-Salido, J. D. (2004). Optimal Monetary Policy with Endogenous Capital Accumulation.

Levine, P., Pearlman, J., and Pierce, R. (2006). Linear-Quadratic Approximation, External Habit and Targeting Rules.