RegGae: How to Use It

Eduardo C. Castro*

Central Bank of Brazil

April 27, 2020

Abstract

This note includes the instructions to implement on Matlab/Dynare the formulas in the article ¨RegGae: A Toolkit for Macroprudential Policy with DSGEs.¨

Contents

*Prudential and Foreign Exchange Regulation Department, ecastro@bcb.gov.br.

1 What you need

To use RegGae with Matlab/Dynare, place the following files in the same folder:

1. Your Dynare model.mod file or files. There should be one .mod file for each system of equations. If the difference between regimes is only on the value of parameters, only one .mod file suffices. If the equations of the regimes also differ, there should be one .mod file for each system of equations. For example, the DSGE equations where a constraint binds (holds with equality) is distinct and should be in a separate .mod file from the DSGE where the constraint is slack and some other equilibrium condition is in its place.
2. The RegGae.m file
3. Five functions:
   (a) RegGae_matrix.m
   (b) RegGae_jacobian.m
   (c) RegGae_steadystate_(model).m
   (d) RegGae_infinite.m
   (e) RegGae_finite.m
4. If necessary, make available also the model_steadystate.m file or files to compute the steady-states for Dynare.

2 Setting up the Dynare .mod file

In setting up each of your Dynare model.mod file (or files), here are a few requirements to observe:

1. The variables should be declared in this order:
   (a) Static variables first
   (b) Pure pre-determined variables
   (c) Mixed (lead-lag) variables
   (d) Pure forward-looking variables

If you don't know which one is which, run your .mod file with Dynare once and then type: M_.lead_lag_incidence' . The rows with a non-zero number in the middle column only correspond to the static variables. The rows with non-zero number in the left and middle columns only are the pure pre-determined variables. The rows with non-zero numbers in all three columns are the mixed variables. The rows with with non-zero numbers in the middle and right column only are the pure forward-looking variables.

2. The parameter (or parameters) whose value with vary according to the regime ("variable parameters") should be entered with the Dynare command: set_param_value('(parameter)',Gamma(row,regime)); where Gamma $\Gamma(s)$ is the matrix of parameters where each column corresponds to a regime s.

3. Ensure there is a shocks section with at least one shock for Dynare to compute the steady-state and linearize.

4. At the end, only the commands steady; and check; should be entered. Don't write stoch_simul or it will slow RegGae.

5. Run the model.mod file with Dynare once just to check it is running properly. Remember that not satisfying the BK conditions is permitted only to finite regimes.

3 Setting up RegGae's main .m file

The main file is RegGae.m. It calls all the functions that compute the transition matrices for each history type. It runs the process forward so that you can see the trajectory of the variables. With that and a regime probability function, the user can plot the distribution of trajectories.

Section I of RegGae.m initializes Matlab variables and structures to be used later on. In particular, the structure R will store the regime-specific matrices and steady-states. The structure TM will store the history-specific transition matrices.

3.1 Regimes

In Section II, the user must indicate the parameter values in the matrix $\Gamma(s)$ where each column associated a regime s. Use Matlab resources and functions to fill in $\Gamma(s)$ as you see fit (you will also need to specify the expectation formation protocol for each history type, and the regime probability function later on). In the codes made available, for simplicity columns 1 to 10 contain regime values for the SKANK model (forthcoming) in its normal-times version ("tight", when policy rules bind). Regimes numbered 101 and

above refer to SKANK under a slack situation (crisis) with the release of policy commands. The declaration order of parameters in Dynare files must match the ordering of rows of $\Gamma(s)$. The values of endogenous parameters (those who are functions of the other parameters, such a output and structural interest rate) can be set at any arbitrary value and their correct values be left to be computed with the steady-state.

3.2 Regime-wise linearization

Section III of the RegGae file performs the regime-wise linearization. It is made by the Matlab functions RegGae_matrix.m, RegGae_jacobian.m and RegGae_steadystate.m where this last function is used as an argument to the jacobian function.

3.2.1 The RegGae_matrix.m function

This RegGae_matrix.m function is called by the RagGae main file to execute the Dynare model file once for the first time for regime $s = 1$. It stores the regime specific matrices in the first page of the R structure. Each page of the R structure corresponds to a regime. If necessary, your Dynare model.mod file may be associated with its own model_steadystate.m file. After executing the RegGae_matrix.m function, you need to attribute the endogenous parameters calculated by Dynare to the rows of the $\Gamma(s)$ matrix.

3.2.2 The RegGae_jacobian.m and RegGae_steadystate.m functions

With the execution RegGae_matrix.m function in the previous step, Dynare has been executed once and created the Matlab model.m file. Then, it suffices to provide each desired steady-state to extract the regime-specific Jacobians. This is only to sped up the calculation but not strictly needed; you can simple run the previous step as many regimes as you have. To speed it up:

1. The function RegGae_steadystate needs to be set up and ran to compute the steady state for each set of parameter values. It is a mirror image of the model_steadystate.m file needed for Dynare. Its argument is the value of the parameters (a column of Γ) and its output is the steady-state values in Dynare's declaration order and the endogenous parameters (the parameters which are functions of the others). If you are unable to set up the RegGae_steadystate.m on your own (because you need Dynare's algorithms to compute the steadystate), simply use the RegGae_matrix.m function to all and each one of your regimes.

2. Then the function RegGae_jacobian.m must be executed. Its arguments are the .mod file name, the steady state values (in declaration order computed by Reg-

Gae_steadystate.m function) and the parameter values (again). It fills in the pages of the R structure with the regime specific matrices.

If your model has k different sets of equations, the functions must be ran the corresponding number of times.

3.3 Histories

Think about the regime tree of your model. Each node ι in the regime history tree is associated with a sequence of expected future regimes as per the expectation formation protocol of your choice. But all expectations must culminate in an s-infinite regime. To compute the the transition matrices TM at the s-infinite node ι , use function RegGae_infinite.m. Its arguments are the components of page s of structure R. To compute the transition matrices for finite regimes, move backwards from the infinite regime. Plug in the function RegGae_finite.m the regime-specific matrices in R with and the expectations of variables in the following period, vectors $V_{\iota'}$ and $Y_{\iota'}$.

After completing these steps you will have on hand the structure TM with all the ι-specific transition matrices. Then simply use them to simulate the variables, applying the regime probability function and the expectation formation protocol of your choice.