**README FOR FOR THE GUERRIERI-IACOVIELLO OCCBIN TOOLKIT**
JUNE 3, 2013

CITE AS: Guerrieri, Luca, and Matteo Iacoviello (2013) "Occbin: A Toolkit to Solve Models with Occasionally Binding Constraints Easily", working paper, Federal Reserve Board

# Overview

1. Modify and run the file `setpathdynare4.m` so as to point to the local dynare installation directory and to the folder containing the `toolkit_files` directory.

   We have used successfully Dynare versions 4.3.1 on Windows and 4.2.1 on Mac.

2. LIST OF THE EXAMPLE FILES (in increasing order of complexity and decreasing order of documentation)

   (a) `runsim_irrcap.m.` An RBC model with non-negativity constraint on investment. The two relevant mod files are dynrbc.mod and dynrbcirr_i.mod.

   This folder shows how one can use the codes to declare the parameter values only once in an outside file (named paramfile_irrcap.m, called from dynrbc_steadystate.m).

   If the example file runs correctly, it will generate a figure like the one in the file figure_example.pdf

   (b) `runsim_borrcon.m` In this example, `borrcon.mod` is a model with borrowing constraint that binds occasionally.

   (c) `runsim_irrcap_twoconstraints.m.` An RBC model with non-negativity constraint on investment and an asymmetric capital adjustment cost. This file shows how one can use the codes to solve a model with two occasionally binding constraints using the function `solve_two_constraints`

   `runsim_irrcap_twoconstraints_computepolicy.m.` For the same model as above, but uses the code and the initcon option to show one can use the toolkit to derive the model's nonlinear policy functions.

   Additional examples

   - `runsim_smetswouters.m`. Solves the Smets-Wouters (AER, 2007) model allowing for the ZLB on nominal interest rates (see equation for `rnot`, the notional interest rate). The codes for the model were downloaded from the online Appendix on the AEA webpage. We affected minimal changes to the code for compatibility with Dynare 4. To avoid the estimation step, some parameter values were set at their initial level prior to estimation.

- `runsim_dnk.m.` Solve a new-keynesian model with ZLB and government spending. This folder shows how one can use the codes to declare the parameter values only once in an outside file (named paramfile_dnk.m). Shows how one can use separate sets of functions to solve model disregarding nonlinearities, or to compute impulse responses conditional on different baseline paths for the variables.

  – `dnk.mod` contains a standard new-keynesian model specified away from the zlb constraint.

  – `dnk_zlb.mod` is an exact replica of dnk.mod file with the model specified at the constraint.

  Except for the interest rate equation, the models in the two .mod files are identical.

- `runsim_nakata.m.` Solve a new-keynesian model with ZLB and Rotemberg pricing.

The only restriction for the .mod files is that they can accommodate at most one lag and one lead of the endogenous variables, and that the constraint only involve contemporaneous variables. With appropriate redefinitions, this restriction comes at no loss of generality.

3. The main program assumes that the endogenous, exogenous variables, and parameters are declared in exactly the same order in both .mod files.

   In general, to create the second file, it pays to simply make a replica of the main .mod file that disregards the constraint and amend the relevant equations.

4. The function that solves the model is:

   ```
   solve_one_constraint

   [zdatalinear zdatapiecewise zdatass oo_base M_base] =

   solve_one_constraint(modnam, modnamstar,

   constraint, constraint_relax,

   shockssequence, irfshock, nperiods, maxiter, init);
   ```

   Inputs:

   `modnam`: name of reference model (a Dynare .mod file applicable when the constraint does not bind)

   `modnamstar`: name of alternative model (a Dynare .mod file applicable when the constraint binds)

   `constraint`: the constraint (see note 1 below).

   `constraint_relax`: the violation of the `constraint` in the alternative model that triggers switch to reference model (often, just constraint with sign inverted; see example files)

`shockssequence`: a sequence of unforeseen shocks under which one wants to solve the model (size T×nshocks)

`irfshock`: label for innovation for IRFs, from Dynare mod file (one or more of the 'varexo')

`nperiods`: simulation horizon (can be longer than the sequence of shocks defined in `shockssequence`; in some cases it needs to be long enough to ensure convergence back to the reference model at the end of the simulation horizon)

`maxiter`: maximum number of iterations to solve model until convergence (20 if not specified)

`init:` the initial position for the vector of state variables, in deviation from steady state. (If not specified, the default is steady state)

Outputs:

`zdatalinear:` array containing variables ignoring the occasionally binding constraint (the linear solution), in deviation from steady state. Each column is a variable, the order is the definition order in the mod file.

`zdatapiecewise:` array containing variables satisfying the occasionally binding constraint (the occbin/piecewise solution), in deviation from steady state.

`zdatass:` steady state value of the endogenous variables (a vector, ordered as in the mod file)

`oobase_, Mbase_`: structures produced by Dynare for the reference model.

5. The function that solves the model with two constraints is

```
solve_two_constraints(modnam_00,modnam_10,modnam_01,modnam_11,...
constraint1, constraint2,...
constraint_relax1, constraint_relax2,...
shockssequence,irfshock,nperiods,curb_retrench,maxiter,init);
```

Inputs:

`modnam_00`: reference model

`modnam_10`: the first constraint switches to its alternative model but second does not

`modnam_10`: the second constraint switches to its alternative model but first one does not

`modnam_11`: both constraints switch to their alternative models

`constraint1`: the first constraint (see note 1 below).

`constraint_relax1`: the violation of `constraint1` in the alternative model that triggers switch to reference model (often, just constraint with sign inverted; see example files)

`constraint2`: the second constraint (see note 1 below).

`constraint_relax2`: the violation of `constraint2` in the alternative model that triggers switch to reference model (often, just constraint with sign inverted; see example files)

`shockssequence`: a sequence of unforeseen shocks under which one wants to solve the model

`irfshock`: label for innovation for IRFs, from Dynare mod file (one or more of the 'varexo')

`nperiods`: simulation horizon (can be longer than the sequence of shocks defined in `shockssequence`; in some cases it needs to be long enough to ensure convergence back to the reference model at the end of the simulation horizon)

`curb_retrench:` a scalar equal to 0 or 1. Default is 0. When set to 0, it updates the guess based on the full information gained at each iteration. When set to 1, it updates in a manner similar to a Gauss-Jacobi scheme, slowing the iterations down.

`maxiter`: maximum number of iterations to solve model until convergence (20 if not specified)

`init:` the initial position for the vector of state variables, in deviation from steady state. (If not specified, the default is steady state)

# Additional Notes

1. Suppose the original occasionally binding constraint in the model is

$$i_t \geq \log\left(\phi \cdot I_{ss}\right)$$

where $i_t$ is natural logarithm of investment (see mod file), $I_{ss}$ is steady state investment in levels, and $\phi$ is a parameter.

For the constraint to be violated, in the candidate solution calculated under the assumption that the constraint does not bind the following must be true

$$i_t < \log\left(\phi \cdot I_{ss}\right)$$

Rewrite each variable as

$$x_t \equiv \widetilde{x}_t + x_{ss}$$

In the runsim_irrcap.m code, the constraint will have to be expressed in linearized form as

$$\widetilde{i}_t + i_{ss} < \log\left(\phi \cdot I_{ss}\right) \Longleftrightarrow \widetilde{i}_t < \log\left(\phi\right)$$

and the string will be

$$i < log(PHII)$$

Therefore note that in the mod file, $i$ will denote the variable, whereas in the `constraint` the same $i$ will denote the variable minus its steady state level.

2. In all runsim_*.m files, we declare M_ and oo_ to be global variables (for use by Dynare)

3. In the toolbox, `constraint` and `constraint_relax` only admit contemporaneous endogenous variables. Note that this is not without loss of generality since appropriate redefinitions can accommodate a general lead and lag structure.

4. One does not need to specify the steady state of any alternative model, since all models are approximated around the steady state of the reference model (which needs to satisfy the Blanchard-Kahn conditions)

5. Values for the parameters (whether in the mod or in the steady state file) are specified only in the reference model. The parameter values specified in the alternative model are ignored by the code (but not the list of parameters). If a parameter only enters the alternative model, it needs to be declared as parameter and assigned a value in the reference model.