

JMCB 11-557 REPLICATION FILES

''MANAGING BELIEFS ABOUT MONETARY POLICY UNDER DISCRETION''

Elmar Mertens, em@elarmertens.com

Overview

The replication files attached to the paper ''Managing beliefs about monetary policy under discretion'' solve for the equilibrium policies of the different models discussed in the main paper and the online appendix (henceforth: ''the appendix''). The code is written in MATLAB and runs both under the Windows and Linux versions of MATLAB 2012a and 2013a. (I expect the code to function also under other MATLAB versions.)

The code solely consists of Matlab scripts and functions stored in m-files (text files named with a .m suffix). All results are obtained from simulated model economies, so there are no files containing empirical data in this set of replication files. In particular, this replication set consists of five types of m-files:

- goFigure*.m: are separate scripts that can be run to recreate the corresponding figures of the main paper. Each of these scripts clears the workspace, sets up the model, solves it and generates the appropriate graphs.
- goAppendixFigure*.m are scripts that generate the corresponding figures of the web-appendix.
- setup*.m scripts contain model definitions, mapping specific models into the matrices defined in Section 2 of the main paper.
 - setup201static.m for the simple New Keynesian with a static information problem as described in Section 3 of the main paper.
 - setup201.m for the simple New Keynesian model with a dynamic information problem as described in Section 3 of the main paper.
 - The model version with information shocks, described in

Section 4, is encoded directly in `goFigure3.m` and `goFigure4.m`.

- Model solver functions:
 - `burgundypolicyi.m` solves for the Markov-perfect optimal discretion policy under hidden information model via policy iteration.
 - `markovfullinfopolicyi.m` and `markovfullinfoVFI.m` solve for the Markov-Perfect discretion policy under full information using policy iteration algorithm (`policyi`) and value function iteration (`VFI`), respectively.
 - Notice that no separate solution routine is provided for the static information problem. The case of the static information problem is handled in `goFigure1.m` by using a model definition (given in `setup201static.m`) that includes lagged expectations of all backward-looking variables in the public's information set. (This generic solution can also be verified by comparison against the manual solution for this particular model implemented in `goAppendixFigure1.m`)
- Various toolbox files and helper routines are provided with the remaining `*.m` files.

(As usual, the asterisk `*` used in the file names above represents a placeholder for different strings. For example, `goFigure*.m` is meant to comprise both `goFigure1.m` and `goFigure3.m`.)

Installation and use

Simply copy all `m`-files provided by this replication set into one directory and set this directory as work directory in MATLAB. To recreate any of the figures shown in the main paper and its appendix, call the appropriate script on the MATLAB prompt. For example, `>> goFigure1` solves the simple model described in Section 3 of the paper and recreates Figure 1.

Each of the `go*.m` scripts works independently of the others; every script begins by clearing the workspace (via `initscript.m`). The header of each script contains an assigned to

a variable called `wrap`; by default `wrap` is set to empty. In this case, figures are only displayed on screen but not stored in a file. To store the figures, please set `wrap` to a string that contains the name of the target directory in which the figure shall be stored; for example `wrap = pwd` will store the figures in the present working directory. By default, figures are stored in eps format. To change this setting, please edit the function `wrapcf.m`.