∫ IRIS Macroeconomic Modeling Tutorials

SIMPLE SPBC MODEL: READ ME FIRST

by Jaromir Benes

January 18, 2016

Summary

This tutorial is a collection of example files for a simple sticky-price business cycle model (SPBC). The model file and m-files included in the tutorial describe a typical workflow in building and operating a small-scale DSGE models for practical policy analysis and forecasting.

CONTENTS

1 HOW TO BEST RUN THIS TUTORIAL?

Each m-file in this tutorial is split into what is called "code sections" in Matlab. A code cell is a shorter block of code performing a specific task, separated from other code cells by a double percent sign, `%%` (usually with a title and brief introduction added). By default, the cells are visually separated from each other by a horizontal rule in the Matlab editor.

Instead of running each m-file from the command window, or executing this `read_me_first` as a whole, do the following. Open one tutorial m-file in the Matlab editor. Arrange the editor window and the command window next to each other so that you can see both of them at the same time. Then run the m-file cell by cell. This will help you watch closely what exactly is going on.

To execute one particular cell, place the cursor in that cell (the respective block of code will get highlighted), and select "Run Current Section" from a contextual menu (upon a right click on the mouse), or pressing a keyboard shortcut (which differ on different systems and Matlab versions). To learn more on code sections, search Matlab documentation for "code section".

```
clc;
```

2 SIMPLE STICKY PRICE BUSINESS CYCLE MODEL FILE

This is a model file for a simple sticky-price business model. The model file describes variables, parameters and equations. Note that the model file does not specifies the tasks that will be performed with the model. The tasks will be set up in separate m-files, using standard Matlab functions and IRIS functions.

You cannot run the model file itself. Instead, load the file into Matlab using the function `model` function. See the m-file `read_model` for more details.

The IRIS model files can be syntax-highlighted in the Matlab editor; this makes the files easier to read. To this end, associate the model file extension(s) (which can be anything) with the editor. Open the menu File - Preferences, and click on the Editor/Debuger - Language tab. Use the Add button in the File extensions section to add new extensions, e.g. 'model'. Then restart the editor, and that's it.

```
edit simple_SPBC.model;
```

3 READ AND SOLVE MODEL

Create a model object by loading the model file `Simple_SPBC.model`, assign parameters to the model object, find its steady state, and compute the first-order accurate solution. The model object is then saved to a mat file, and ready for further experiments.

```
% edit read_model.m;
read_model.m;
```

4 GET INFORMATION ABOUT MODEL OBJECT

Use the function get (and a few others) to access various pieces of information about the model and its properties, such as variable names, parameter values, equations, lag structure, or the model eigenvalues. Two related topics are furthermore covered in separate files: assigning/changing parameters and steady-state values, and accessing model solution matrices.

```
% edit get_info_about_model.m;
get_info_about_model.m;
```

5 ASSIGN AND CHANGE PARAMETERS AND STEADY STATES

Assign or change the values of parameters and/or steady states of variables in a model object using a number of different ways. Under different circumstances, different methods of assigning parameters may be more convenient (but they, of course, all do the same job).

```
% edit change_parameters_and_sstates.m;
change_parameters_and_sstates.m;
```

6 MODEL SOLUTION MATRICES

Describe and retrieve the state-space form of a solved model. IRIS uses a state-space form with two modifications. First, the state-space system is transformed so that the transition matrix is upper triangular (quasi-triangular). Second, the effect of future anticipated shocks can be directly computed upon request, and added to the system stored in the model object.

```
% edit know_all_about_solution.m;
know_all_about_solution.m;
```

7 FIND AND DESCRIBE BALANCED GROWTH PATH

The SPBC.model is a BGP model: It does not have a stationary long run. Instead, it has two unit roots, introduced through the productivity process, and the general nominal price level. To deal with BGP models, there is absolutely no need to stationarise them. They can be worked with directly in their non-stationary forms.

```
% edit play_with_bgp.m;
play_with_bgp.m;
```

8 SIMULATE SIMPLE SHOCK RESPONSES

Simulate a simple shock both as deviations from control and in full levels, and report the simulation results.

```
% edit simulate_simple_shock.m;
simulate_simple_shock.m;
```

9 MORE COMPLEX SIMULATION EXPERIMENTS

Simulate the differences between anticipated and unanticipated future shocks, run experiments with temporarily exogenised variables, and show how easy it is to examine simulations with mutliple different parameterisations.

```
% edit simulate_complex_shocks.m;
simulate_complex_shocks.m;
```

10 SIMULATE PERMANENT CHANGE IN INFLATION TARGET

Simulate a permanent change in the inflation target, calculate the sacrifice ratio, and run a simple parameter sensitivity exercise using model objects with multiple parameterizations.

```
% edit simulate_disinflation.m;
simulate_disinflation.m;
```

11 Monte-Carlo Stochatic Simulations

Draw random time series from the model distribution, and compare their sample properties against the unconditional model-implied models. Keep in mind that this is a purely simulation exercise, and no observed data whatsoever are involved.

```
% edit resample_from_model.m;
resample_from_model.m;
```

12 Import CSV Data Files and Prepare Data

Load basic data from CSV data files into databases where each series is represented by a tseries (time series) object. Prepare the data to be used later with the model: seasonally adjust, convert to quaterly periodicity, and create model-consistent variable names.

```
% edit read_data.m;
read_data.m;
```

13 Simulate Fisher Info Matrix and Test Parameter Identification

Calculate estimates of the Fisher information matrix. The Fisher matrix is a property of the model itself, and is independent of any data. It represents the maximum amount of information one can hope for to find in the data in case the data are really generated by the model DGP.

Compare two approaches: a time-domain approach, and a frequency-domain approach. Use the singular value decomposition to learn more about which parameters (or combinations of them) are identified the best or the worst.

```
% edit fisher_information_matrix.m;
fisher_information_matrix.m;
```

14 Run Bayesian Parameter Estimation

Use bayesian methods to estimate some of the parameters. First, set up our priors about the individual parameters, and locate the posterior mode. Then, run a posterior simulator (adaptive random-walk Metropolis) to obtain the whole distributions of the parameters.

```
% edit estimate_params.m;
estimate_params.m;
```

15 POSTERIOR SIMULATOR WITH 'SAVEEVERY=' OPTION

In this file, we show two features of convenience when running larger posterior simulations. First, the posterior simulator can be run with the option `'saveEvery='` to split the simulated posterior chain into smaller bits and saving them each in a separate data file. This is a way to get around possible out-of-memory problems when simulating larger models and/or longer chains. Second, a large posterior simulation can executed incrementally in smaller chunks, with the final state of one simulation being used as the initial state for the next one.

```
% edit more_on_poster_simulator.m;
more_on_poster_simulator.m;
```

16 KALMAN FILTERING AND HISTORICAL SIMULATIONS

Run the Kalman filter on the historical data to back out unobservable variables (such as the productivity process) and shocks, and perform a number of analytical exercises that help understand the inner workings of the model.

```
% edit filter_hist_data.m;
filter_hist_data.m;
```

17 MORE ON KALMAN FILTER

Run more advanced Kalman filter exercises. Split the data sample into two sub-samples, and pass information from one to the other. Run the filter with time-varying std deviations of some shocks. Evaluate the likelihood function and the contributions of individual time periods to the overall likelihood.

```
% edit more_on_kalman_filter.m;
more_on_kalman_filter.m;
```

18 FORECASTS WITH JUDGMENTAL ADJUSTMENTS

Use the Kalman filtered data as the starting point for forecasts, both unconditional and conditional, i.e. with various types of judgmental adjustments.

```
% edit forecasts_with_judgment.m;
forecasts_with_judgment.m;
```

19 COMPARE SECOND MOMENT PROPERTIES IN MODEL AND DATA

Compute and compare several second-moment properties of the estimated model and the data. Describe the data using an estimated VAR; this also allows to evaluate sampling uncertainty of the empirical estimates using bootstrap methods.

```
% edit compare_model_and_data.m;
compare_model_and_data.m;
```