Replicating package*: Should monetary policy care about redistribution? Optimal monetary and fiscal policy with heterogeneous agents.*

François Le Grand, Alaïs Martin-Baillon, Xavier Ragot

June, 27 2024

This set of files replicates the graphs and data of *Should monetary policy care about redistribution? Optimal monetary and fiscal policy with heterogeneous agents*. Be careful of not mixing up these files with other replicating files and they should be located in an independent folder. Requires Matlab / Julia / Dynare. Tested on Matlab 2018b and Julia release v1.10.0.

1. All the `.ipynb` files are `Julia` notebooks.
2. All the `.m` files can be runned in Matlab or `Octave`. Most of them also require `Dynare`.

Quantitative assessment of the sticky-price model.

How to run?

1. Start with the computation of the steady state allocation:

- Open `Sticky_Prices/steady_state/Main_SP.ipynb` and execute all cells. This computes the steady state.

2. Then, simulate the dynamics of the model (IRFs and second-order moments) in one of the following cases.

- **Baseline calibration and the uniform truncation**
  - Run `Sticky_Prices/dynamics/main_SP.m`. This simulates the dynamics of the model (IRFs and second-order moments) for the baseline calibration and the uniform truncation.
- Run `Sticky_Prices/dynamics/main_SP.m`. **Comment the line 3 "calib = `baseline`" and uncomment the line 4 "calib = refined"**. This simulates the dynamics of the model for the baseline calibration and the refined truncation.

- Run `Sticky_Prices/dynamics/main_taylor_SP.m`. This simulates the dynamics of the model for the baseline calibration and the refined truncation.
- Run `Sticky_Prices/dynamics/unequal.m`. This simulates the dynamics of the model in the case of the unequal profit distribution.

Output:

1. Figure 1: Run `Sticky_Prices/dynamics/Do_IRFs_SP_baseline.m`.
2. Tables 7/8: Run `Sticky_Prices/steady_state/Tables_SP.ipynb`.
3. Figure 7: Run `Sticky_Prices/dynamics/Do_IRFs_SP_unequal.m`.

The details

- The `Julia` files takes care of computing the steady state, while the .m file simulates the model in the presence of aggregate shocks.

The output of the `Julia` files is:

1. a file `todynare_SP_baseline.mat`
2. a file `todynare_SP_refined.mat`
3. a file `To_IRFs_SP_unequal.mat`

Those files will be used by main.m, main_taylor.m, unequal.m

- The outputs of the `Octave` / `Matlab` files are:
  - `To_IRFs_SP_baseline.mat`
  - `To_IRFs_SP_taylor.mat`
  - `To_IRFs_SP_unequal.mat`

Then:

- `Do_IRFs_SP_baseline.m` generate the Figure 1: `IRFs_SP_Eco_1\_2_taylor.png`
- `Do_IRFs_SP_unequal.m` generate the Figure 7: `IRFs_SP_uneq_Eco_1\_2.png`

.The output of the `Julia` file is also:

1. 6 files: moments_eco1/2_baseline, `moments_taylor`, moments_eco1/2_refined

Those files will be used by the Julia file `Tables.ipynb` to display the tables for First- and second-order moments for key variables.

The steady state computation

The steady state is computed thanks to nine `Julia` notebooks. Each of above files are commented and self-explained.

- `Main.ipynb`: Solves the steady-state model and returns the truncated model (as `steady_state_dynare.mat` for `Dynare`, saved in the current folder);
- `Structures.ipynb`: Structures and parameter calibration from targets;
- `Utils.ipynb`: Contains some useful functions;
- `SolveAiyagari.ipynb`: Solves the Aiyagari model;
- `Projection.ipynb`: Computes the steady-state truncated model;
- `Projection_ref.ipynb`: Computes the steady-state redined truncated model;
- `Ramsey.ipynb`: Computes the steady-state Lagrange multipliers.
- `Simulation.ipynb`: : Contains a function used to display tables of first and second order moments of key variables;
- `Tables.ipynb`: Display tables of first and second order moments of key variables;

Simulating the model with aggregate shocks,

- The file `main.m` simulates the model for the first two economies of the paper. For the baseline truncation or the refined truncation depending on the line 3 and 4 of the file.
  - Economy 1: optimal inflation
  - Economy 2: constant inflation
- The file `main_taylor.m` simulates the model for third economy of the paper:
  - Economy 3: Taylor Rule

The outcomes of the program can be parametrized as follows:

- The `Octave` / `Matlab` actually writes different `Dynare` codes
  - `code_dynare_baseline1.mod`, `code_dynare_baseline2.mod`, `code_dynare_taylor.mod` which correspond to the three economies of the baseline specifiation.
  - `code_dynare_refined1.mod`, `code_dynare_refined2.mod`.
  - `code_dynare_unequal1.mod`, `code_dynare_unequal2.mod`

Each of this code is then solved in `Dynare`. These files, as interim Dynare files are created in the current folder.

Comparisons with the Reiter method

How to run?

In this order:

1. Open `Reiter/steady_state/Main_Reiter_Comp.ipynb` and execute all cells.
2. Run `Reiter/dynamics/reiter_comp.m`.
3. Run `Reiter/dynamics/refined_comp.m`.
4. Run `Reiter/dynamics/code_difference.m` .

Output:

- Figure 8: run `Reiter/dynamics/fig_Comp_Reiter_Trunc.m`
- Table 11: run `Reiter/steady_state/Tables_Reiter_Comp.ipynb` cells 1 to 4
- Table 12 : run `Reiter/steady_state/Tables_Reiter_Comp.ipynb` cell 5

The details

The output of the `Julia` files is:

- a file `todynare_Comp_Reiter.mat`
- a file `todynare_Comp_Refined.mat`

Those files will be used by `reiter_comp.m`,refined_comp.m‘

- The outputs of the `Octave` / `Matlab` files are:
  - `todiff_Reiter IRFs.mat` save the IRFs of the Reiter simulation
  - `todiff_Comp_Refined.mat` save the IRFs of the refined truncation
  - `moments_Comp_Reiter.mat` save the moments of the Reiter simulation
  - `moments_Comp_Refined.mat` save the moments of model with the refined truncation
  - `to_code_difference_reiter.mat` save the result of the Reiter simulation
  - `to_code_difference_refined.mat` save the result of the refined truncation simulation
  - Then those files will be used by `code_difference.m` to generate `diff_Reiter.mat`: save the result for table 12.
- Finally:

- `Reiter/steady_dynamics/fig_Comp_Reiter_Trunc.m` generates the Figure 8: `Comp_Reiter_Trunc.png`
- `Reiter/steady_state/Tables_Reiter_Comp.m` displays Table 11 and

  12.

Quantitative assessment of the sticky-wage model.

How to run?

In this order:

1. Open `Sticky_Wages/steady_state/Main_SW.ipynb` and execute all cells.
2. Run `Sticky_Wages/dynamics/main_SW.m`.
3. Run `Sticky_Wages/dynamics/main_taylor_SW.m`.

Output

1. Graph : Run `Sticky_Wages/dynamics/Do_IRFs_SW_baseline.m`

The details

- The `Julia` files takes care of computing the steady state, while the .m file simulates the model in the presence of aggregate shocks.
  - The output of the `Julia` file is a file `todynare_SW.mat` that will be used by `Octave` / `Matlab`.
- The outputs of the .m are:
  - `To_IRFs_SW.mat`
  - `To_IRFs_SP_taylor.mat`
- `Do_IRFs_SP_baseline.m` generate IRFs_SW_Eco_1_4_taylor.png

The steady state computation

The steady state is computed thanks to seven `Julia` notebooks. Each of above files are commented and self-explained.

- `Main.ipynb`: Solves the steady-state model and returns the truncated model (as `steady_state_dynare.mat` for `Dynare`, saved in the current folder);
- `Structures.ipynb`: Structures and parameter calibration from targets;
- `Utils.ipynb`: Contains some useful functions;
- `SolveAiyagari.ipynb`: Solves the Aiyagari model;

- `Projection.ipynb`: Computes the steady-state truncated model;
- `Ramsey_SP.ipynb`: Computes the steady-state Lagrange multipliers.
- `ToDynare_SW`: Generate the output of the Julia file: `todynare_SW.mat` that will be used by `Octave` / `Matlab`.

Simulating the model with aggregate shocks,

- The file `main_SW.m` simulates the model for the first two economies of the section:
  - Economy 1: optimal inflation
  - Economy 2: constant wage inflation
- The file `main_SW_taylor.m` simulates the model for third economy of the paper:
  - Economy 3: Taylor Rule

The outcomes of the program can be parametrized as follows:

- The `Octave` / `Matlab` actually writes three `Dynare` codes `code_dynare_SW1.mod`, `code_dynare_SW4.mod`, `code_dynare_SW_taylor.mod` which correspond to the three economies. Each of this code is then solved in `Dynare`. These files, as interim Dynare files are created in the current folder.