READ ME FILE

Title: Boundedly Rational Expectations and the Optimality of Flexible Average Inflation Targeting

Authors: Anthony Brassil, Christopher G Gibbs and Callum Ryan

Description

This 'read me' file details the replication files for RDP 2025-02. All data used to plot figures appearing in the RDP are available to the public and can be found in the spreadsheet 'rdp-2025-02-graph-data'.

The code is split into three parts:

1. Figures 1-6
2. ZLB - Figures 7-8
3. Robust optimal policy - Figures 9 D1 and Tables 3-4 C1-D2

We used Matlab 2023a. Figures 1-6 and the robust optimal policy figures/tables also require Dynare (we used Dynare 6.1).

Figures 1-6

The main script is called 'main_figures.m'. To produce figures 1-6, just run this script.

The folder also contains functions and Dynare mod files.

Functions:

- compare_models.m – this function loops over different models and/or parameter values, solves the models, and calculates IRFs. It is used for Figures 1 to 2 and 4 to 5.
- run_model.m – this function loops over different parameter values for a given model and calculates the solution matrices for each parameterisation. It is called within 'compare_models.m'.
- dyn_to_QG.m – this function takes Dynare output for a solved model and returns the matrices in the equilibrium law of motion. It is called within 'run_model.m'.
- compare_makeup_coef.m – this function loops over different models and/or parameter values and calculates the weights on current and lagged inflation and output gap outcomes under optimal policy. It is used for Figures 3 and 6.
- get_opt_coef.m – this function loops over different parameter values for a given model and calculates the coefficients on inflation, the output gap, and learners' expectations under various formulations of optimal policy. It is called within 'compare_makeup_coef.m'.
- get_zetas_Omega.m – this function factorises a lag polynomial in the derivation of the optimal target criteria. It is called within the Dynare mod files when specifying optimal policy.
- calc_irf.m – this function loops over different solution matrices and shocks, and calculates IRFs. It is called within 'compare_models.m'.

Dynare mod files:

- cost_push.mod – model in the unconstrained case with only cost-push shocks.
- imp_info_inf.mod – model in the constrained case with imperfect central bank information.
- imp_info_inf_part.mod – contains a subset of the imperfect information model. It is used only within get_opt_coef.m, in order to solve for a law of motion for the output gap in terms of lagged Lagrange multipliers as part of the calculation of optimal makeup coefficients.

ZLB - Figures 7-8

The main script is called 'plot_results.m'. To produce Figures 7 and 8, just run this script. It will load saved model solutions from the 'Model solutions' folder.

The other script in this folder is 'solve_models_loop.m'. This script loops over a set of parameter values and solves the model using policy function iteration for each parameterisation.

Functions:

- full_proj_upd.m – this function updates the policy function guess. It is called within each iteration of the policy function iteration algorithm. Specifically, it takes a guess for the policy variables on a state grid, solves for all other variables based on this guess, and returns a new guess from the Euler equations on which the solution algorithm is iterating.
- full_sim.m – this function simulates a solved model for any sequence(s) of shocks. It is called within 'plot_results.m' to calculate IRFs and stochastic steady states.
- eq_noZLB.m – this function solves the intratemporal equilibrium conditions given values for the state variables and policy variables under the assumption that the ZLB is not binding. It is called within 'full_proj_upd.m' and 'full_sim.m'.
- eq_ZLB.m – this function solves the intratemporal equilibrium conditions given values for the state variables and policy variables under the assumption that the ZLB is binding. It is called within 'full_proj_upd.m' and 'full_sim.m'.
- The subfolder 'Other functions' contains several auxiliary functions that support the policy function iteration algorithm. These functions are taken or slightly adapted from Sijmen Duineveld's PROMES toolbox (available at https://www.promestoolbox.com/) (Duineveld 2021). The main change to the original toolbox is that the 'solve_proj.m' function has been modified so that the 'tmi' option actually uses fixed point iteration, not time iteration.

The subfolder 'Model solutions' contains .mat files with the solution to versions of the model. Each .mat file contains two structures: 'GRID', which contains details of the state variable grid on which the model is approximated; 'POL', which contains the solution for the policy variables and details of the solution algorithm; and 'par', which contains parameter values.

Robust optimal policy - Figures 9 D1 and Tables 3-4 C1-D2

There are three main scripts:

1. osr_tables_simple.m – this script produces Tables 3, C1 to C2 and D1 (i.e. performance of simple target criteria).
2. osr_tables_robust.m – this script produces Tables 4, C3 to C4 and D2 (i.e. policy under parameter uncertainty).
3. osr_plot_loss.m – this script produces Figures 9 and D1 (i.e. loss under different parameter values).

Each of these scripts loads results saved in the 'Results' subfolder. These results are generated with the other three scripts:

1. osr_loop_simple.m – this script calculates the optimal coefficients for the simple target criteria under the baseline parameterisation.[1] It produces the results for Tables 3, C1 to C2 and D1.

1 The 'osr_loop_simple.m' script and 'osr_loop_robust.m' script are almost identical. The difference is just that for the 'simple' script, only one parameterisation is set in the 'Set parameters for loops' section.

2. osr_loop_robust.m – this script calculates the coefficients for the simple target criteria that minimise average loss over a set of different parameterisations. It produces the results for Tables 4, C3 to C4 and D2.
3. osr_loop_loss.m – this script calculates the loss under the robustly optimal policy rules for a range of different parameter values. It produces the results plotted in Figures 9 and D1.

In each script, set the 'constraints' variable to choose which model to use.

The folder also contains functions and Dynare mod files.

Functions:

- osr1_dist.m – this function is a modified version of the Dynare function 'osr1.m'. The input 'dist' contains a cell array of parameter names and a set of potential values for those parameters. The function then calculates the policy rule coefficients that minimise the average loss across this set of parameterisations. It is called within 'osr_loop_dist.m'.
- osr_obj_dist.m – this function is a modified version of the Dynare function 'osr_obj.m'. It is the objective function for 'osr1_dist.m'. For a given set of policy rule coefficients, it calculates the average loss across the set of parameterisations in 'dist'.
- osr_loop_dist.m – this function loops over a set of loss function weights, parameterisations, and starting values (for the optimisation routine). In each iteration, it calls 'osr1_dist.m' to calculate the policy rule coefficients that minimise the average loss across the set of parameterisations in 'dist'. It is called in 'osr_loop_robust.m' and 'osr_loop_simple.m'.[2]
- osr_loop.m – this function loops over a set of loss function weights, parameterisations, and starting values (for the optimisation routine). In each iteration, it calls the Dynare function 'osr.run.m' to calculate the policy rule coefficients that minimise the loss. It is equivalent to calling 'osr_loop_dist.m' with a 'dist' argument that contains only the baseline parameterisation. In practice, we use it only to calculate the loss for given policy rule coefficients (by letting it optimise a dummy coefficient that has no effect on equilibrium outcomes). It is called in 'osr_loop_simple.m', 'osr_loop_robust.m' and 'osr_loop_loss.m'.
- make_grid.m – this function constructs a grid of parameter values. It is called in the functions 'osr_loop_dist.m' and 'osr_loop.m', and in the scripts 'osr_loop_simple.m', 'osr_loop_robust.m' and 'osr_loop_loss.m'.

Dynare mod files:

- perf_info_osr.mod – simple target criteria in unconstrained case (Tables D1 to D2 and Figure D1).
- perf_info_opt.mod – optimal policy in unconstrained case (Tables D1 to D2 and Figure D1).
- imp_info_osr.mod – simple target criteria in imperfect information case (Tables 3 to 4 and Figure 9).
- imp_info_opt.mod – optimal policy in imperfect information case (Tables 3 to 4 and Figure 9).
- perf_info_inf_osr.mod – simple target criteria in unconstrained case with infinite-horizon Phillips curve (Tables C1 and C3).
- perf_info_inf_opt.mod – optimal policy in unconstrained case with infinite-horizon Phillips curve (Tables C1 and C3).
- imp_info_inf_osr.mod – simple target criteria in imperfect information case with infinite-horizon Phillips curve (Tables C2 and C4).
- imp_info_inf_opt.mod – optimal policy in imperfect information case with infinite-horizon Phillips curve (Tables C2 and C4).

2 In 'osr_loop_simple.m', 'dist' only contains one parameterisation. So there is no particular reason to use 'osr_loop_dist.m'; 'osr_loop.m' could be used instead.

Reference

Duineveld S (2021), 'Standardized Projection Algorithms to Solve Dynamic Economic Models', Unpublished manuscript, 3 December.

8 April 2025