Code Documentation for ”A Macroeconomic Model with Financially Constrained Intermediaries and Producers”

Vadim Elenev, Tim Landvoigt, Stijn Van Nieuwerburgh*

October 21, 2020

Contents

*Elenev: Johns Hopkins University, Carey Business School; email: velenev@jhu.edu. Landvoigt: University of Pennsylvania Wharton School, NBER, and CEPR; email: timland@wharton.upenn.edu. Van Nieuwerburgh: Columbia University Graduate School of Business, NBER, and CEPR; email: svnieuwe@gsb.columbia.edu.

This folder contains all the code to replicate the results of Elenev, Landvoigt, and Van Nieuwerburgh "A Macroeconomic Model with Financially Constrained Intermediaries and Producers", forthcoming at Econometrica. This code is also available online at:
https://github.com/elenev/MacroModelProducersIntermediaries.

1 Fast Steps to Reproduce Benchmark Model

1. In Matlab, execute the script `main_create_env`.
   - If you have the Matlab Symbolic Math Toolbox and use version 2018b or earlier, leave the flag `useJacobian` in line 43 on. Otherwise set to false to use numerical differentiation.
   - Note: generating the analytic Jacobian for the benchmark model takes approximately 5 minutes with version 2018b, and can take longer for the other experiments.
   - `main_create_env` will create a file `env_bench_ini0` that contains the experiment definition for the benchmark economy on the initial coarse grid.
2. Execute script `main_run_exper` to run the benchmark on the coarse grid for up to 100 iterations.
   - You can set the number of parallel workers to be started in line 12.
   - Set to zero if you want to run it with a single process.
   - On a computer with sixteen cores (and 16 parallel workers) this should take about 30 minutes.
   - `main_run_exper` creates a results file named `res_[current_date_time]` that contains the converged policy functions.
   - Rename this file to `res_20200910_bench_i100.mat`.
3. Execute `main_create_env` again using results file from previous step as input.
   - Configure `main_create_env` as follows (leave other variables as is):
     * `xpername = 'econ';`

    * `guess_mode = 'guess';`
    * `guess_path = 'res_20200910_bench_i100';`
  - `main_create_env` will create a file `env_bench` that contains the experiment definition for the benchmark economy on the fine grid, using the resulting policy functions from the coarse grid iterations in `res_20200910_bench_i100.mat` as initial guess.

4. Execute script `main_run_exper` to run the benchmark on the fine grid for up to 30 iterations.
  - Configure `main_run_exper` as follows (leave other variables as is):
    * `exper_path = 'env_bench.mat';`
    * `maxit = 30;`
    * `price_zns = true;`
  - Set to zero if you want to run it with a single process.
  - On a computer with sixteen cores (and 16 parallel workers) this should take about 45 minutes.
  - `main_run_exper` creates a results file named "`res_[current_date_time]`" that contains the converged policy functions.
  - Rename this file to `res_20200910_bench_s130.mat`.

5. Simulate the model using `sim_stationary` and `sim_trans_cluster`.
  - `sim_stationary` simulates the model policies contained in `res_20200910_bench_s130.mat` for 10,000 periods and writes out the resulting time-series and several statistics. The main output is a file named `sim_res_20200910_bench_s130.mat`.
  - `sim_trans_cluster` reads both `res_20200910_bench_s130.mat` and `sim_res_20200910_bench_s130.` and simulates generalized IRFs.
  - To plot IRFs, run `plot_trans`.

2 Overall Structure

The overall approach of the computational procedure is to represent the economy as one large system of functional equations, with the unknown functions being the choice variables of the household problems and the asset prices. We then use an algorithm that we call "transition function iteration", which is an extension of Judd (1998)'s "policy function iteration", to compute the policy and price functions that depend on the state variables.

The key steps are solving of a system on nonlinear equations at each point in the state space given a guess for future policy and price functions, and then updating these approximated functions based on the newly found solution. Iterate on this until convergence. For a detailed description of the algorithm see Appendix C of the paper.

2.1 Algorithm Implementation

At a high level, we can describe the necessary steps for the algorithm as follows. First, define bounds for the endogenous state variables. We are approximating the system in a hypercube, hoping that it is stationary within these bounds. Choose the kind of approximation, e.g. how many spline knots in each dimension. For the implementation in this paper, we use multivariate linear interpolation. Then create an initial guess of the policy and price functions (and other functions required to compute expectations in Euler equations). Using this guess, the algorithm proceeds as follows:

1. Loop over all individual points in the state space, and solve a nonlinear system of N equations (FOCs and constraints) in N unknowns (choices, prices, and Lagrange multipliers), using last guess to compute expectations. Save the solution vector at each point.

2. Update policy and price functions using new solution. This becomes the next guess.

3. If distance between this new guess and last guess is below a certain threshold, stop. Otherwise go back to step 1.

2.2 Transition Functions for Endogenous State Variables

Depending on the specifics of the problem, it may not be possible to compute next period's state variables in closed-form form only based today's state variables, choices, and prices. As an example in this specific paper, intermediary wealth next period depends on next period's prices, which in turn are a function of next period's intermediary wealth. Since the problem has a first-order Markov structure in its state variables (see Kubler and Schmedders (2003)), we can define a mapping from today's aggregate state variables into tomorrow's states, and approximate these *transition functions* numerically. These functions are updated between iterations in step 2. of the algorithm above along with policy functions. We developed this variant of policy iteration, which we call "transition function iteration", in Elenev, Landvoigt, and Van Nieuwerburgh (2016) and this paper.

3 Preparing the Computation

3.1 Setting Parameters and Defining Numerical Experiments

The file `experdef_20200910.m` sets the parameters of the model for all numerical experiments in the paper. The baseline calibration described section III of the paper is the fixed-rate-only benchmark model. All other experiments contained in the paper, for instance aggregate or local indexation, are variations on the parameters of the benchmark model. These are defined in the MATLAB struct `expers_macropru` in line 327 of `experdef_20200910.m`. Other experiments can defined by adding lines to this struct. Note that state variable grids are part of each experiment definition. Each experiment has a coarse initial grid (suffix `_ini0`) and a fine grid. If no custom grid is specified for a given experiment, it defaults to the `benchgrid`.

3.1.1 Creating an Experiment Definition File

The script `main_create_env` reads the experiment definitions from `experdef_20200910.m` and creates a MATLAB data file that serves as input for a run of the computational algorithm. The most important inputs to set at the top of the file are name of the experiment definition script, and name of the experiment for which the definition file should be created, e.g. `experdef_20200910.m` and `bench_ini0` to create a definition file for the benchmark economy in the paper on the coarse grid.

`main_create_env` computes an approximate non-stochastic steady state of the economy (making assumptions on the bindingness of constraints and the wealth distribution). This steady-state serves as initial guess for policies and prices of the dynamic stochastic model. The script performs several other steps needed to initialize the code and then writes the result into a file that serves as input for the actual computation script.

3.1.2 Running the Computation

Script `main_run_exper` reads the contents of the file created by `main_create_env`, and starts the computation. It either runs until convergence or until it hits the maximum number of iterations. The key inputs that should be set at the top of the file are:

- name of the experiment definition file,
- maximum number of iterations,
- convergence tolerance,
- number of parallel workers to start (set to 0 for no parallel workers),
- whether to execute pricing of assets in zero net supply for welfare calculations at the end (flag `price_zns`).

The actual model computation is in a class file (using MATLAB's object-oriented features), which is called `ProductionModel.m`. The two key methods are `calcStateTransition` and `calcEquations`. The first method uses budget constraints and market clearing conditions, and evaluates the transition functions to calculate the state variables for the next period. It is useful to separate this part of the code, since it is also needed to simulate the model after the solution has been calculated. The second method uses current choices/prices, and the values of next-period state variables as inputs to compute the equilibrium equations listed in Appendix C.1. Note that these functions are called at each point in the state space by MATLAB's nonlinear equation solver 'fsolve' many times.

Once the computation is done, either by convergence or because it reached the maximum number of iterations, it will write final policy and transition functions in a MATLAB data file.

Note that `main_run_exper` executes additional iterations after the main loop of the algorithm has converged if the control parameter `price_zns` is activated. These additional iterations do not update policy functions, but only compute equilibrium prices of non-traded assets in zero net supply using stochastic discount factors from the converged model. In particular, the code computes the prices of borrowers and saver consumption streams, which are needed as inputs for the CEV welfare calculations in the paper.

4 Processing the Results

4.1 Running a Long Simulation

`sim_stationary` can be used to simulate the economy for many periods and compute moments of the variables. The script also calculates Euler equation errors along the simulated path. The script calls another method in the model-specific class `ProductionModel.m` that is called `computeSimulationMoments`. Input for `sim_stationary` is the results file produced by `main_run_exper`. It creates another file that contains the simulated data for the model; the code archive contains the output of the simulation for the paper's benchmark economy in `sim_res_20200910_bench_s130.mat`. It can also create Excel files with model moments.

4.2 Simulating IRFs and Transition Paths

`sim_trans_cluster` can be used to simulate paths of the economy after it was hit by a specific shock. The paths are initiated at the ergodic steady state of the stochastic model that is calculated by applying a clustering algorithm to the long time-series of state variables created by `sim_stationary`. To plot IRFs, use the scripts `plot_trans` or `plot_trans_finrec`. The first script was used to create Figures 2 and 3 in the paper and is for comparison of different shocks within in the same model parameterization. The second script was used to create Figures 4(a,b) in the paper and is for comparison of the same shock across different model parameterizations.

4.3 Computing Welfare Comparisons

To compare compensating-variation welfare measures reported as CEV-welfare in the paper, we first need to compute prices for the consumption streams of agents. This is done in the script `priceZNS.m` based on the model's solution file. After `priceZNS.m` has finished, `welfare.m` creates an Excel sheet that contains the welfare comparison of the different economies for which solution files are present in the main code folder to the benchmark.

5 Step-by-Step Detailed Replication Instructions

This section lists all steps required to reproduce all results in the paper. Note that this requires computing the model solution for all parameter combinations specified in `experdef_20200910.m`. In practice, it is impractical to compute all of these experiments locally; rather, we run all experiments as a batch job on a high-performance computing cluster (HPCC). The code repository contains sample scripts for batch execution on a Linux-based system with a job scheduler (`slurmcombined0910.sh`). Depending on the specific HPCC environment, the implementation of batch execution will of course have to be adapted. However, `experdef_20200910.m` fully specifies all aspects of experiment definitions required for the Matlab code, irrespective of the HPCC environment. Below is the content of file `readme_replication.txt` also in the repository.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Part 1: Solve and Simulate the Model
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

This part can be done locally or on the cluster. There are separate instructions
for each method below. Recomputing all 93 economies required for the paper
is much faster on the cluster.
```

Part 1a: Computing locally

Repeat the steps below for each economy in experdef_20200910.txt. The definitions of these economies are in experdef_20200910.m. Let the placeholder "econ" represent the name of the economy from this text file.

First, run the economy for up to 100 iterations on the coarse grid.

1. Configure main_create_env.m as follows (leave other variables as is):

- expername = 'econ_ini0';
- guess_mode = 'no_guess';

2. Run main_create_env.m and it will produce a file called env_econ_ini0.mat

3. Configure main_run_exper.m as follows (leave other variables as is):

- no_par_processes = XX; % where XX is the number of processors on the machine you're running it on
- exper_path = 'env_econ_ini0.mat';
- maxit = 100;
- price_zns = false;

4. Run main_run_exper.m. On a machine with 16 cores, this should take about 30 min. It will create a file named res_TIMESTAMP.mat, where TIMESTAMP is the time at which the calcuation is finished expressed as YYYY_MM_DD_hh_mm

5. Rename the res_TIMESTAMP.mat file to res_20200910_econ_i100.mat.

Next, run the economy for up to 100 iterations on the fine grid

6. Configure main_create_env.m as follows (leave other variables as is):

- expername = 'econ';
- guess_mode = 'guess';
- guess_path = 'res_20200910_econ_i100';

7. Run main_create_env.m and it will produce a file called env_econ.mat.

8. Configure main_run_exper.m as follows (leave other variables as is):

- no_par_processes = XX; % where XX is the number of processors on the machine you're running it on
- exper_path = 'env_econ.mat';
- maxit = 30;
- price_zns = true;

9. Run main_run_exper.m. On a machine with 16 cores, this should take about 40 min. It will create a file named res_TIMESTAMP.mat, where TIMESTAMP is the time at which the calcuation is finished expressed as YYYY_MM_DD_hh_mm

10. Rename the res_TIMESTAMP.mat file to res_20200910_econ_s130.mat.

Next, simulate. The model must be solved and res* file must exist.

11. Configure sim_stationary.m as follows (leave other variables as is):

- resfile = 'res_20200910_econ_s130';

12. Run sim_stationary.m. If you are running sim_stationary having run it before during the current MATLAB session, run "clear" beforehand. This operation will create the following files:

- sim_res_20200910_econ_s130.mat: all simulation results incl. full series, statistics, errors, and parameters
- Results/statsexog_res_20200910_econ_s130.xls: statistics using exogenous subsampling to define crises (one sample per worksheet)
- Results/statsendog_res_20200910_econ_s130.xls: statistics using endogenous subsampling to define crises (one sample per worksheet)
- Results/errstats_res_20200910_econ_s130.xls: statistics of EE, VF, and TF errors

Next, compute impulse response functions. Model must be solved and simulated. Both res* and sim_res* files must exist.

13. Configure sim_trans_cluster.m as follows (leave other variables as is):

- resfile = 'res_20200910_econ_s130';

14. Run sim_trans_cluster.m. On a machine with 16 cores, this should take about 15 min. It will create the following files:

- GR_res_20200910_econ_s130.mat: mean, median, and sd of IRF paths for each of 4 shocks (no shock, non-fin rec, fin rec, and pure uncertainty)
- statsirf_res_20200910_econ_s130.mat: means of IRF paths (one sheet per shock)

Part 1b: Computing on the cluster

The following instructions work for a cluster that uses the SLURM job manager, has an environment variable $SCRATCH defined and pointed to a writeable folder. They are an ALTERNATIVE to Part 1a.

1. Configure the cluster job file slurmcombined0312.sh processor, memory, and time requests. Current config of 2 hrs, 49 min is generous with 20 cores, but better to be safe. At least 2 gigs of memory per core.
2. Copy the folder ecma to the cluster into $SCRATCH/code/InterProd (e.g. using WinSCP)
3. If it doesn't exist, create $SCRATCH/InterProd folder.
3. Connect to the cluster's terminal (e.g. using PuTTY) and cd into that folder.
4. Run "sbatch -a 2-94 slurmcombined0910.sh." This will submit jobs to solve, simulate, and compute IRFs for each economy used in the paper. This will create for each economy in experdef_20200910.txt (let "econ" denote the economy name)
- res_20200910_econ_i100.mat: results file after just the coarse grid iterations
- res_20200910_econ_s130.mat: results file
- sim_res_20200910_econ_s130.mat: all simulation results incl. full series, statistics, errors, and parameters
- GR_res_20200910_econ_s130.mat: mean, median, and sd of IRF paths for each of 4 shocks (no shock, non-fin rec, fin rec, and pure uncertainty)
- Results/statsexog_res_20200910_econ_s130.xls: statistics using exogenous

subsampling to define crises (one sample per worksheet)

- Results/statsendog_res_20200910_econ_s130.xls: statistics using endogenous subsampling to define crises (one sample per worksheet)

- Results/errstats_res_20200910_econ_s130.xls: statistics of EE, VF, and TF errors

- statsirf_res_20200910_econ_s130.mat: means of IRF paths (one sheet per shock)

%%%

Part 2: Compute and Save Results

%%%

This part pre-supposes that the res*, sim_res*, GR_res* and Excel files for each economy created by Part 1 exist.

The steps below re-create all the results used in the paper.

1. Compute CEV welfare by running "welfare(10)" and "welfare_appd5(10)" in MATLAB. "10" is referring to the number of data clusters to create for evaluating transitions from benchmark to one of the other economies. welfare.m computes CEV welfare relative to benchmark. welfare_appd5.m computes CEV welfare at xi=95 relative to xi=91 for several different parameter combinations, creating the welfare numbers in Table D.2.

Note: this requires res* files for each economy, which are very large (~350MB). If they're stored on the cluster and not locally, log into the cluster and issue the following commands to run the welfare function on the cluster as a interactive job:

```
cd $SCRATCH/code/InterProd/version20200320
```

```
srun -c8 -t2:00:00 --mem=32000 --pty /bin/bash
module load matlab/2018b
matlab -nodesktop -nodisplay -r "welfare(10);"
matlab -nodesktop -nodisplay -r "welfare_appd5(10);"
```

This will create Results/welfare_20200910_bench_s130.xls, containing the CEV welfare changes from going to bench to altrnative economies for each computed economy in the folder.

2. Run writeStats.m. This will create Results/simres.mat, which contains a variable called "mapDict," a hash-map of every simulation and IRF statistic and model parameter. They can be retrieved by running mapDict("key"), where "key" has the following format:
[command] - [econ] - [subsmaple] - [variable] - [statistic]

command:
- sim: simulation statistic
- comp: simulation statistic relative to bench (either level or percent change)
- irf: IRF value at t=1

econ: name of the economy (as in experdef_20200910.txt)

subsample: if command is "sim" or "comp"
- u: unconditional
- e: expansions (not available if command = "sim" or "comp")
- r: recessions
- c: crises
- l: low-uncertainty (not available if command = "sim" or "comp")

variable: all variables reported by sim_stationary as well as "cvwelfare."

Different variables are reported differently e.g. in percentage terms. See the code of writeStats.m for details.

statistic: for command = "sim" and "comp", all statistics reported by sim_stationary e.g. "mean," "std," "p50". For a few ratios, there is "nmean" which is ratio of means (whereas "mean" is mean of ratios).

For "command" = irf, choices are "mean" (level at t=1) or "change" (deviation of t=1 from t=0 value, either in levels or percent)

After simres.mat has been created, the build.m LaTeX parser uses these values to fill in placeholders in the *.tex file (see below).

3. Create benchmark economy IRFs (Figures 2 and 3). Configure plot_trans.m as follows:

- outfile = ['GR_',resfile];
- plot_shocks = 1:3;

Run plot_trans.m. This will create figures in Results/GR_res_20200910_bench_s130_IRF#.(pdf|eps)with # = {1,2,3,4}.

4. Create IRFs comparing financial crises in various economies (Figure 4). Configure plot_trans_finrec.m as follows:

- econ{1} = 'FLbench';
- econ{2} = 'FLwithdef';
- econ{3} = 'bench';

Comment out econ{4} if needed and run. This will create figures in Results/GR_res_20200910_finrec_FLbenchFLwithdefbench_s130_IRF1.(pdf|eps)

5. Create IRFs comparing financial crises in benchmark vs counter-cyclical cap reqs economies (Figure 7). Configure plot_trans_finrec.m as follows:

- econ{1} = 'bench';

- econ{2} = 'xi9195';

Comment out econ{3} and econ{4} if needed and run. This will create figures in Results/GR_res_20200910_finrec_benchxi9195_s130_IRF1.(pdf|eps)

6. Create graphs comparing equilibrium quantities and welfare across macroprudential experiments. run makeMacropruPlots.m. This will create figures in Figures/macropru*.(pdf|eps).

7. Compute transitions from benchmark to other capital requirements (Figure 8). For each of econ = xi85 and econ = xi95benchgrid,

a. Configure sim_trans_policy.m as follows:

- resfile = 'res_20200910_econ_s130';

b. Run sim_trans_policy.m. This will create PT_res_20200910_econ_s130.mat.

c. Configure plot_trans_policy.m as follows:

- resfiles = {'res_20200910_econ_s130'};

- labels = {'\xi=XX'}; % where XX is the value of xi in that economy

This will create figures in

Results/PT_res_20200910_econ_s130.(pdf|eps).

8. Create policy function plots (Figure B.1) by running plotPolicyFunctions.m. This will create figures in Results/polPlot#.(eps|pdf), where # = {,1,2}.

9. Create state space histograms with grid point lines (Figure B.2) by running compareStateSpaces.m. This will create figures in Results/stateSpaceHistograms.(pdf|eps).

10. Create Gentskow-Shapiro sensitivity analysis bar. This will create figures in Figures/GS_1.(pdf|eps).

11. Create benchmark economy IRFs comparing financial recession to pure

uncertainty (Figures D.1 and D.2). Configure plot_trans.m as follows:

- outfile = ['GR_unc_',resfile];
- plot_shocks = [1,4,3];

Run plot_trans.m. This will create figures in
Results/GR_unc_res_20200910_bench_s130_IRF#.(pdf|eps)with # = {1,2,3,4}.

12. Create a figure showing how the credit spread and expected excess return changes with intermediary wealth (Figure D.3) by running makeEERhist.m. This will create figures in Results/WI_EERhistres_20200910_bench_s130.(pdf|eps).

13. Create a figure showing the IRFs after an unanticipated shock to intermediary wealth ("mortgage" shock, Figure D.4). For each of econ = bench and econ = benchriskysafesigI0,

a. Configure sim_trans_mit.m as follows:

- resfile = 'res_20200910_econ_s130';

b. Run sim_trans_mit.m. This will create MIT_res_20200910_econ_s130.mat.

Run plot_trans_mit.m. This will create figures in
Results/GR_res_20200910_mit_benchbenchriskysafesigI0s130_IRF#.(pdf|eps),
where # = {1,2}.

14. Create a table showing the robustness of macropru results to alternative parameters (Table D.2).

a. Copy every Excel file containing *xi91* and *xi95* from Results into Results_D5. Also copy welfareappd5_20200910_s130.xls.

b. Open table_d2.xlsm, enabling macros.

c. In the "Overview" sheet, click "Update from sim_stationary." The macro will run for a minute or so.

d. When done, go to "Drivers of Macropru." Use the Excel2Latex marcro to convert the two shaded tables into Panel A (top) and Panel B (bottom)

References

ELENEV, V., T. LANDVOIGT, AND S. VAN NIEUWERBURGH (2016): "Phasing Out the GSEs," *Journal of Monetary Economics*, 81, 111–132.

JUDD, K. L. (1998): *Numerical Methods in Economics*, The MIT Press.

KUBLER, F. AND K. SCHMEDDERS (2003): "Stationary Equilibria in Asset-Pricing Models with Incomplete Markets and Collateral," *Econometrica*, 71, 1767–1795.