

README

The codes in the package are used to replicate the results in “*Lumpy Investment, Lumpy Inventories*” by Bachmann and Ma. The work-flow of the code is:

1. Compile the FORTRAN codes.
2. Call `run.m` to run the FORTRAN code with the desired specification.
3. Call `busi_stat.m` to extract the results from the FORTRAN codes.
4. Call various routines in `matlab` folder to generate tables and graphs after repeating the previous two steps for all the required specifications.

The codes here are organized into four folders:

- `fortran` contains the FORTRAN codes used to solve and simulate the model.
- `initial_conditions` contains the required initial conditions to run the FORTRAN codes.
- `matlab` contains the MATLAB codes used to generate tables and graphs, after calling the two files, `run.m` and `busi_stat.m` in the main folder.
- `data` contains the data files and codes to generate the data moments used in the paper.

Two important notes for replication:

1. For the ease of disposition we group the files into different folders. In order to run the codes properly, the reader needs to first pool all the files in different folders into a single folder before compiling the codes.
2. All the codes are written for UNIX system. Look for the `unix()` function calls in the `*.m` files and change them accordingly before calling them in non-UNIX systems. Consult `Makefile` to for the compilation requirements for the FORTRAN codes and adapt accordingly for compilation on non-UNIX systems.

The rest of the file will explain each step in detail.

1 The FORTRAN Code

1.1 Compilation

The FORTRAN code is compiled with Intel FORTRAN Compiler version 16.0 in Ubuntu 14.04. In addition to the files in the package, it also requires LAPACK, BLAS, and the core MKL libraries. If you want to compile with other compilers, make sure to set the size for “real” variables to 64 (so “real” is essentially treated as “double precision”). Consult “Makefile” for the details of compiling the code.

On most Linux distributions, directly type:

```
./make
```

in the folder that contains all the files shall compile the code.

1.2 Running the Code

Once the code is compiled, `main` will be generated. Calling the program requires three input parameters:

```
./main SET_NAME E_UPPER ZETA_UPPER
```

1. `SET_NAME`. It specifies the set of parameters to use. It must be a 5-character string that takes the following values:
 - `r95bn`: the benchmark parameters for the main results.
 - `r95kt`: the robustness check that uses the production function in Khan and Thomas (2008).
 - `r98bn`: the robustness check that sets $\rho_z = 0.98$.
 - `ftech`: the robustness check that the aggregate shock only shows up in the production of the final goods, not in that of the intermediate goods.
 - `heter`: the robustness check that allows for persistent idiosyncratic productivity shocks in the intermediate goods sector.
 - `r95p2`: the benchmark parameters, partial equilibrium with both the pricing kernel and the relative price fixed.
 - `r95p3`: the benchmark parameters, partial equilibrium with the pricing kernel fixed.
 - `r95p4`: the benchmark parameters, partial equilibrium with the relative price fixed.
2. `E_UPPER` and `ZETA_UPPER` are two 7-character strings that specify the value of $\bar{\epsilon}$ and $\bar{\zeta}$ respectively according to a modified floating-point rule. Suppose `E_UPPER` takes the following form:

XXXXZEE.

Here **YYYY** is the significand, **EE** is the exponent on base 10, and **Z** is the sign of the exponent. The value of **E_UPPER** is then defined as:

$$\bar{\epsilon} = \begin{cases} X.YYY * 10^{EE} & , \text{if } Z = 1 \\ X.YYY * 10^{-EE} & , \text{if } Z = 0 \end{cases}$$

For example, calling the program with

```
./main r95bn 4580001 1950001
```

solves the benchmark model with $\bar{\epsilon} = 0.4580$ and $\bar{\zeta} = 0.1950$, which is the Model I1 in the paper. The sample outputs of running the above line of code are included in the package.

Running the code also requires the following input files:

- **para_SET_NAME.txt**.
Required.
The file that includes all the parameters in addition to $\bar{\epsilon}$ and $\bar{\zeta}$ for a given **SET_NAME**. The parameter files for all the **SET_NAME** listed above are included in the package.
- **ks_SET_NAME_E_UPPERZETA_UPPER.txt**.
Required.
The file that includes the forecasting rules for the distributions and prices in Equation 8-11 in the paper. The files for all the relevant cases used in the paper are included in the package. All the partial equilibrium cases will use the forecast rules in **r95bn**. See **run.m** line 308 for more details.
- **dist_bench_inter** and **dist_bench_final**.
Required.
The initial distribution of intermediate and final goods producers over capital stocks and inventory holdings, respectively.
- **v0inter_SET_NAME_E_UPPERZETA_UPPER.txt** and **v1final_SET_NAME_E_UPPERZETA_UPPER.txt**.
Optional.
The initial guess of the value functions for intermediate and final goods producers. If these files are absent, the program will initialize with a zero matrix. All the partial equilibrium cases will use the value functions in **r95bnas** initial inputs. See **run.m** line 308 for more details. The files for the **heter** case are large, and thus they are available upon request from the authors. All the other files are provided in this package.
- **price_SET_NAME_E_UPPERZETA_UPPER.txt**
Required if **SET_NAME==r95p2, r95p3 or r95p4**.
The sequence of prices from simulating **r95bn**. Used as inputs to the partial equilibrium cases.

`main` is meant to be called within `run.m`. The readers shall consult `run.m` for examples of calling this program. We include all the required files for the cases listed in `set_names.m` in the package. If the user needs the input files for other input files, please directly contact the authors.

Table 1 lists all the files written by the authors.

Name	Purpose
<code>acc.f90</code>	auxiliary functions
<code>data_modules.f90</code>	modules that pass data between subroutines
<code>den_haan.f90</code>	computes the Den Haan measure of accuracy
<code>equidist.f90</code>	the core subroutine to compute equilibrium prices in each period in simulations.
<code>FinalFirm.f90</code>	solves the value function for final good producers.
<code>FinalFirm_PE.f90</code>	solves the value function for final good producers in partial equilibrium.
<code>init_dist.f90</code>	initializes the distributions of all firms before the simulation.
<code>InterFirm.f90</code>	solves the value function for intermediate good producers.
<code>InterFirm_PE.f90</code>	solves the value function for intermediate good producers in partial equilibrium.
<code>irf.f90</code>	runs simulations for impulse-response functions.
<code>ks_update.f90</code>	update the parameters for forecasting rules in the Krusell-Smith algorithm
<code>main.f90</code>	the main code that deals with inputs and outputs
<code>sim.f90</code>	simulates the model conditional on value functions
<code>sim_splines.f90</code>	creates the splines used for simulations
<code>solve.f90</code>	calls <code>equidist.f90</code> to solve for equilibrium prices in simulations
<code>util.f90</code>	a simple numerical library written by the authors

Table 1: FORTRAN File List, Written by the Authors

Our program also requires the following codes listed in Table 2 from the public domain to work. These files are also included in the package for the ease of use.

Name	Purpose	Source
fmm.f90	A collection of FORTRAN procedures for mathematical computation	“Computer Methods for Mathematical Computations”, by George E. Forsythe, Michael A. Malcolm, and Cleve B. Moler. Prentice-Hall, 1977.
refsor.f90 uniinv.f90 unista.f90	Quick-sorts and finds unique values in vectors	ORDERPACK from FORTRAN-2000.com
cubspl.f interv.f ppvalue.f	Creates and evaluates cubic-splines	PPPACK from netlib.org

Table 2: FORTRAN File List, Public Domain

2 Calling run.m

The file used to call `main` is `run.m`.

1. `run.m`.

- Purpose: the main code to call the FORTRAN program. It takes the inputs (`data_var` on line 11) of the following format:

`SET_NAME_XY`

where `SET_NAME` is the same as used in calling `main`. `X` is a 1-character string that controls two parameters in the FORTRAN codes:

- If `X` is set to be `A`, then the switch `after_sim` will be set to `true` in `para_*.txt`. This will force `main` to simulate the distribution of intermediate producers with a finite number of firms. This will slow down the code, but it produces more accurate estimates on the fraction of intermediate good producers with lumpy adjustments. This is only meant to be called after we have solved the model.
- If `X` is set to be `I`, then the switch `irf` will be set to `true` in `para_*.txt`. This will force `main` to run impulse-response simulations, instead of solving the model. This is only meant to be called after we have solved the model.
- If `X` is set to anything else, then both `after_sim` and `irf` will be set to `false`. This is the intended setup in order to solve the model.

`Y` can take any character to indicate the batch of parameters `E_UPPER` and `ZETA_UPPER` to run. The pre-defined batches can be found inside `run.m`.

With a given input string, `run.m` will create a working folder named `SET_NAME_XY`, copy all the source codes and the relevant files to the working folder, compile the codes in the worker folder, runs the codes in batch, and pushes the running codes

to the background of the system. The logs and the outputs of the codes will be stored in the working folder, later to be extracted by the other MATLAB codes.

For example, calling `run.m` with input string `r95bn_bn` will run `main` with `SET_NAME=r95bn` for the values of $\bar{\epsilon}$ and $\bar{\zeta}$ defined in line 96 of `run.m`.

- Inputs:
 - `*.f90, *.f`: the FORTRAN source files.
 - `Makefile`: the configuration file to compile the code.
 - All the other required and optional files listed in the previous section to run `main`.
- Outputs:
 - `log*.txt`: the log of running the file.
 - `output_*.txt`: the main output files that contains the results of the simulation.
 - `den_*.txt`: the output files used to estimate the Den Haan measures of accuracy.
 - `v1final_*.txt, v0inter_*.txt`: the value functions.
 - `ks_*.txt`: the equilibrium forecast rules.

3 Calling `busi_stat.m`

After calling `run.m`, you need to call `busi_stat.m` with the same `date_var` to process the results of the FORTRAN code:

1. `busi_stat.m`.

- Purpose: the main code used to extract the outputs of the FORTRAN code from the working folder, to compute the business cycle statistics and to produce input files for the codes that generates tables. It calls `irf_stat.m` and `non_irf_stat.m` inside.
- Inputs:
 - `output_*.txt`: the output from `main`.
 - `para_*.txt`: the parameter file.
- Outputs:
 - `price_*.txt`, from `non_irf_stat.m`: the equilibrium prices in simulation, later used in partial equilibrium simulations.
 - `result_*.mat`, from `non_irf_stat.m`: the file that contains business cycle statistics and later used to generate tables.
 - `irf_*.mat`, from `irf_stat.m`: the file that contains business cycle statistics for impulse response functions and later used to generate graphs.

4 All the Other Files

All the other files deal with the outputs of `run.m` and `busi_stat.m` to generate tables and graphs. A file shall only be called when all the required specifications defined in that file have all been solved by calling `run.m` and `busi_stat.m`.

1. `data.m` in folder `data`.

- Purpose: Computes the data moments that are used in the paper.
- Inputs:
 - `ORG_*.csv` in folder `data`: the original data files downloaded from the source. These files are only used for reference and not called inside the code.
 - `INPUT_*.csv` in folder `data`: the cleaned data files based on `ORG_*.csv`. These files are fed into `data.m`.
- Outputs:
 - `data_moments_*.mat`: the data moments later used in the tables.

2. `graph_den.m`.

- Purpose: generate the graphs on the Den Haan measures of numerical accuracy.
- Inputs:
 - `den_*.txt`: the output from `main`.
 - `para_*.txt`: the parameter file.
- Outputs:
 - `den_*.eps`: the graphs on the Den Haan measures of numerical accuracy.

3. `irf_impact_plot.m`.

- Purpose: Draw the impact response plots (the first period of the IRF) for the benchmark case (`r95bn`).
- Inputs:
 - `para_*.txt`: the parameter file.
 - `irf_*.mat`: the outputs of calling `busi_stat.m`.
- Outputs:
 - `irf_*_ac.eps`: the impact response plots.

4. `irf_impact_plot_robust.m`.

- Purpose: Draw the impact response plots (the first period of the IRF) for the robustness checks, where we have fewer data points computed. It calls `set_specs.m`.
- Inputs:
 - `para_*.txt`: the parameter file.

- `irf_*.mat`: the outputs of calling `busi_stat.m`.
- Outputs:
 - `irf_*_ac_*.eps`: the impact response plots.

5. `irf_plot.m`.

- Purpose: Draw the impulse response functions for all cases. It calls `set_specs.m`.
- Inputs:
 - `para_*.txt`: the parameter file.
 - `irf_*.mat`: the outputs of calling `busi_stat.m`.
- Outputs:
 - `irf_*.eps`: the impulse response plots.

6. `irf_stat.m`.

- Purpose: subroutine called inside `busi_stat.m` to compute impulse-response functions. It calls `load_irf_content.m`.

7. `load_irf_content.m`.

- Purpose: subroutine called inside `irf_stat.m` to transform the IRF data into readable formats.

8. `maketable.m`

- Purpose: Creates the tables used in the paper. It calls `set_specs.m` and `maketable_para.m`. Tables not generated by this file are created manually.
- Inputs:
 - `para_*.txt`: the parameter file.
 - `result_*.mat`: the outputs of calling `busi_stat.m`.
 - `data_moments_*.mat`: the data moments generated from `data.m`.
- Outputs:
 - `std_*.tex`: the standard deviation tables.
 - `atc_*.tex`: the persistence tables.
 - `para_table_bench.tex`: the parameter table for benchmark case.

9. `maketable_para.m`

- Purpose: Create the parameter tables. Called inside `maketable.m`

10. `multi_plot.m`

- Purpose: Draws various graphs used in the paper for the benchmark case.
- Inputs:

- `para_*.txt`: the parameter file.
- `result_*.mat`: the outputs of calling `busi_stat.m`.
- Outputs:
 - `std_*.eps`: the standard deviation graphs.
 - `per_*.eps`: the persistence graphs.

11. `multi_plot_robust.m`

- Purpose: Similar to `multi_plot.m`, it draws various graphs used in the paper for the robustness checks, where we have fewer data points computed. Calls `set_specs.m`.
- Inputs:
 - `para_*.txt`: the parameter file.
 - `result_*.mat`: the outputs of calling `busi_stat.m`.
- Outputs:
 - `std_*.eps`: the standard deviation graphs.
 - `per_*.eps`: the persistence graphs.

12. `non_irf_stat.m`

- Purpose: subroutine called inside `busi_stat.m` to compute business cycle statistics for non-IRF cases.

13. `percent_y.m`

- Purpose: a simple subroutine to add the percentage sign (%) to the graphs. Called in all the codes that generate graphs.

14. `set_specs.m`

- Purpose: a subroutine to store the calibrated `E_UPPER` and `ZETA_UPPER` for each `SET_NAME`. Called in various codes.

In various codes that deal with tables, we call a modified version of `latexable.m`, which we rename as `latexableown.m`. The original file was copy-righted by Andrew E. Slaughter in 2009, and available for download via MATLAB central. The original code was covered under the BSD license, which allows re-distribution of the source code conditional on the acknowledgement of copyright. We include the modified version in the package for the ease of replication.