Matlab Code Documentation

Michele Fornino
Andrea Manera

November 6, 2019

1 Introduction

This documentation pertains to the Matlab code that is used to draw all the figures and make all the computations necessary for the paper "Automation and the Future of Work: Assessing the Role of Labor Flexibility."

2 Main Routine: `RunFigures.m`

Please make sure to have the latest edition of Matlab installed on the machine, R2019a or R2019b. The code was tested on a Linux workstation running Ubuntu with Kernel 5.0.0, as well as on a Macintosh machine running MacOS 10.14.

The user should open the script `RunFigures.m` to run the code on her machine. It is also advised to start a "parallel pool," either on the same machine or on a suitable server cluster. Some parts of the code take quite a long time to be executed, but they also can take advantage of parallelism quite effectively.

The first section "`Housekeeping & Setup`" contains a few settings parameters that can be tweaked to attain the desired functionality:

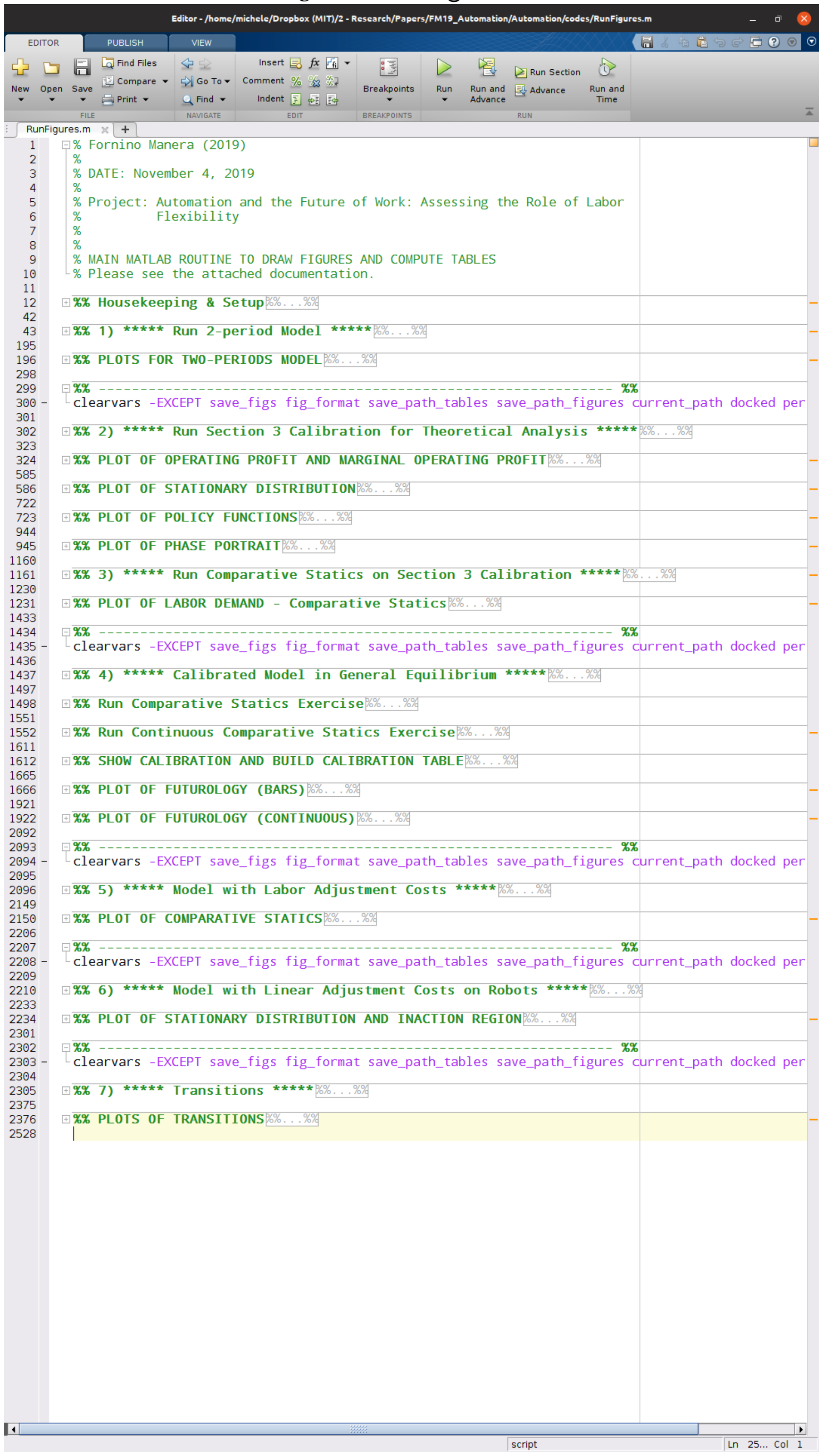
- `save_figs`, must be either "`true`" or "`false`" (Boolean values). Specifies whether or not the figures shall be written to disk
- `save_path_figures` and `save_path_tables`, must be a `string`. Specifies the path where to save figures and tables, *relative to the current working directory*.
- `current_path`, must be a `string`. Specifies the path where the script `RunFigures.m` is executed.
- `fig_format`, must be a string chosen in the set {"`png`", "`eps`", "`jpeg`"}. Specifies the format of the figures. Use EPS for best results, PNG or JPEG for better portability.
- `docked`, must be either "`true`" or "`false`" (Boolean values). It specifies whether or not the figures can be drawn to screen in floating windows, or should be drawn in a quadrant of the main Matlab IDE window.
- `persistent_mode`, must be either "`true`" or "`false`" (Boolean values). It specifies whether the figure window/panel should be closed immediately after it is drawn, to avoid clutter. This is useful if the user wants to run the code and save all pictures to disk without examining them in Matlab.

2.1 Structure of `RunFigures.m`

The main script contains 7 computationally intensive parts. Parts 2 and 3 must be run sequentially, as the latter depends on the former. The structure is shown in Figure 1. The parts are:

1. Run 2-period Model: uses numerical integration to compute expected marginal values of robots.
   - Figure 2: Simple Model Solution, with $\bar{R}(z)$ Distribution.
2. Run Section 3 Calibration for Theoretical Analysis
   - Figure 1: Operating Profit and Marginal Operating Profit.
   - Figure 3: Policy Functions
   - Figure 4: Stochastic Dynamics (MC simulation is in "PLOT OF PHASE PORTRAIT")

Figure 1: `RunFigures.m`



```
% Fornino Manera (2019)
%
% DATE: November 4, 2019
%
% Project: Automation and the Future of Work: Assessing the Role of Labor
%          Flexibility
%
%
% MAIN MATLAB ROUTINE TO DRAW FIGURES AND COMPUTE TABLES
% Please see the attached documentation.

%% Housekeeping & Setup%%...%%

%% 1) ***** Run 2-period Model *****%%...%%

%% PLOTS FOR TWO-PERIODS MODEL%%...%%

%% ----------------------------------------------------------------- %%
clearvars -EXCEPT save_figs fig_format save_path_tables save_path_figures current_path docked per

%% 2) ***** Run Section 3 Calibration for Theoretical Analysis *****%%...%%

%% PLOT OF OPERATING PROFIT AND MARGINAL OPERATING PROFIT%%...%%

%% PLOT OF STATIONARY DISTRIBUTION%%...%%

%% PLOT OF POLICY FUNCTIONS%%...%%

%% PLOT OF PHASE PORTRAIT%%...%%

%% 3) ***** Run Comparative Statics on Section 3 Calibration *****%%...%%

%% PLOT OF LABOR DEMAND - Comparative Statics%%...%%

%% ----------------------------------------------------------------- %%
clearvars -EXCEPT save_figs fig_format save_path_tables save_path_figures current_path docked per

%% 4) ***** Calibrated Model in General Equilibrium *****%%...%%

%% Run Comparative Statics Exercise%%...%%

%% Run Continuous Comparative Statics Exercise%%...%%

%% SHOW CALIBRATION AND BUILD CALIBRATION TABLE%%...%%

%% PLOT OF FUTUROLOGY (BARS)%%...%%

%% PLOT OF FUTUROLOGY (CONTINUOUS)%%...%%

%% ----------------------------------------------------------------- %%
clearvars -EXCEPT save_figs fig_format save_path_tables save_path_figures current_path docked per

%% 5) ***** Model with Labor Adjustment Costs *****%%...%%

%% PLOT OF COMPARATIVE STATICS%%...%%

%% ----------------------------------------------------------------- %%
clearvars -EXCEPT save_figs fig_format save_path_tables save_path_figures current_path docked per

%% 6) ***** Model with Linear Adjustment Costs on Robots *****%%...%%

%% PLOT OF STATIONARY DISTRIBUTION AND INACTION REGION%%...%%

%% ----------------------------------------------------------------- %%
clearvars -EXCEPT save_figs fig_format save_path_tables save_path_figures current_path docked per

%% 7) ***** Transitions *****%%...%%

%% PLOTS OF TRANSITIONS%%...%%

```

- Figure 5: Stationary Distribution and Marginals over z and R, with Automation Bounds

3. Run Comparative Statics on Section 3 Calibration
    - Figure 6: Comparative Statics on $L^d(w)$
4. Calibrated Model in General Equilibrium
    - Figure 7: Total Sectoral Employment
    - Figure 8: Comparative Statics on the Calibrated Model
5. Model with Labor Adjustment Costs
    - Figure 9: Comparative Statics on ψ_L - Model with Labor Market Rigidities
6. Model with Linear Adjustment Costs on Robots
    - Figure 11: Stationary Distribution for the Linear Adjustment Costs Model
7. Transitions
    - Figure 10: Transition to Steady State with $p_R = 0$

3 Model Parameters in `SetParameters.m` and `SetParametersGE.m`

At the beginning of sections 2, 4, 5, 6, and 7, a call is made to the routine `SetParameters.m`. This subroutine expects to find the file `Statistics.csv` in the same folder as `RunFigures.m`. This is needed in particular if the model has to be calibrated to match the moments of any one of the IFR sectors (the rows in the `Statistics.csv` file).

The output of the routine is a `struct` object, which we call `params`, or variations thereof, with a number of fields. The names of the fields were chosen to be reminiscent of the notation adopted in the main text of the paper, but differences and inconsistencies across routines and subroutines are unfortunately still present in the current version of the code.

We advise users to refrain from modifying the hard coded parameters in the `SetParameters.m` routine, and instead to change directly the values inside the output `params` structure once it is created within `RunFigures.m` or any other routines which call `SetParameters.m`.

4 Routine `LaborDemand_*.m`

This routine takes as input the wage w, the purchase price of robots p_R (which is frequently denoted by R in the code following an earlier iteration), and the `params struct` object described in the previous section. The baseline version of this routine is called `LaborDemand_trapz.m`, because the trapezoidal rule is used to deal with potentially unequally spaced Cartesian grids for the state space at hand. Other versions are self-explanatory, and deal with the two extensions to the main model, those with linear adjustment costs and with labor as a state variable.

Its outputs are the aggregate labor demand of the sector described by the parameters passed as inputs, as well as the `struct` object denoted as `out`. Depending on whether or not the parameter `ReducedOutput` is set to true or false, `out` contains detailed distributional information on the policy and value functions, as well as on the stationary distribution, state by state. In addition, `out` contains a host of aggregates that are useful for solving for equilibria, such as aggregate sectoral revenues, output, or adjustment cost outlays.

5 Discretization of Diffusion Process in `DiscretizeDiffusion.m`

A call is made within `LaborDemand_*.m` to the subroutine responsible for the discretization of the diffusion process. This subroutine is flexible, in that it can discretize any (stationary) Ito diffusion, provided the terms $\mu(x)$ and $\sigma(x)$ are provided as function handles. The output is a transition matrix, which discretizes the infinitesimal generator of the diffusion, and a stationary distribution. The subroutine is written so as to be able to deal with a suitably specified unequally spaced grid for the stochastic term z (which is often referred to as p in the code because of past iterations in the notation).

6 Routines `partialEquilibrium.m` and `generalEquilibrium.m`

These routines are called by `RunFigures.m`. The first one computes the aggregate quantities for the 13 IFR sectors for the particular vector of prices that is passed as an input. These prices are the prices of the intermediate goods, as well as the wage. The second one solves for equilibrium prices and quantities given a complete calibration of the model, attained by calling `SetParametersGE.m`, the routine that calibrates the model in General Equilibrium.

7 Routine `SolveTransition.m`

This routine computes the dynamics of the system for a specified path of the purchase price of robots. That is, given an initial and a final steady state, as well as as path of the parameter p_R, the code solves for the transition by iterating backwards on the HJB equation (to bring values back from the long run steady state) and forwards on the KFE equations (to compute the stationary distribution at any point in time). Note that this exercise *is not done in equilibrium*. That is, the wage is assumed constant throughout.

This routine calls the subroutines `GetGrids.m` and `SolvePolicy_rigidlabor.m`. First one is just a minor routine whose purpose is to allow the code to find a grid that is suitable for both the initial and the final steady states. The second one solves for the policy function at time t given the value function at time $t+\Delta$.

8 Other Routines and Notes

8.1 Intermediate Files

One of the ways to greatly speed up execution is to allow Matlab to start looking for an equilibrium in a neighborhood of a good initial guess. To implement this across comparative statics on equilibria, we have the code save to disk the initial values for the `fsolve` calls. These files are `x0.csv` and `x0_gamma.csv`. To be clear, these are not needed for execution and can be deleted if the user so wishes.

8.2 `table2latex.m`

Routine shared in the Matlab user repository by Victor Martinez Cagigal. Given a Matlab `table` object, it outputs a text file containing the code to typeset it in a LaTeXdocument.[1]

8.3 `num2hms.m`

Routine to create pretty timestamps.

8.4 `ind2sub_vec.m` and `sub2ind_vec.m`

Routines that vectorize the corresponding routines provided with Maltab. These are useful for the representation of the state space as MD-arrays vs flat vectors.

[1] Víctor Martínez-Cagigal (2019). MATLAB Table to LaTeX conversor (https://www.mathworks.com/matlabcentral/fileexchange/69063-matlab-table-to-latex-conversor), MATLAB Central File Exchange. Retrieved November 5, 2019.