Readme for the "matlab" Folder for: "Automation and the Future of Work: Assessing the Role of Labor Flexibility"

Michele Fornino*   Andrea Manera†

January 24, 2021

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." The code was tested on a Linux workstation running Linux with Kernel 5.10.7, as well as on a Macintosh laptop running MacOS 11.1 (Big Sur). We advise to have the latest version of Matlab available (2020b).

2 Main Routine: `Run.m`

The user should open the script `Run.m` to run the code on their 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*. It is advised to keep the defaults to save the figures and tables in the subfolders `/graphs` and `/tables`
- `current_path`, must be a `string`. Specifies the path where the script `RunFigures.m` is executed. It is advised to run your instance of Matlab from the `/codes/matlab` folder where the code comes in.
- `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. Remember that you can tweak the DPI of saved PNG or JPEG figures by adding `-r{DPI}` as an option to the call to `print` (e.g., `'-r600'`). Note that figures with transparencies are saved as high-DPI PNG's because the EPS format does not handle transparency correctly.
- `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.
- `format long`, specifies that number be written with as many digits as possible to allow for easier rounding of parameter values.

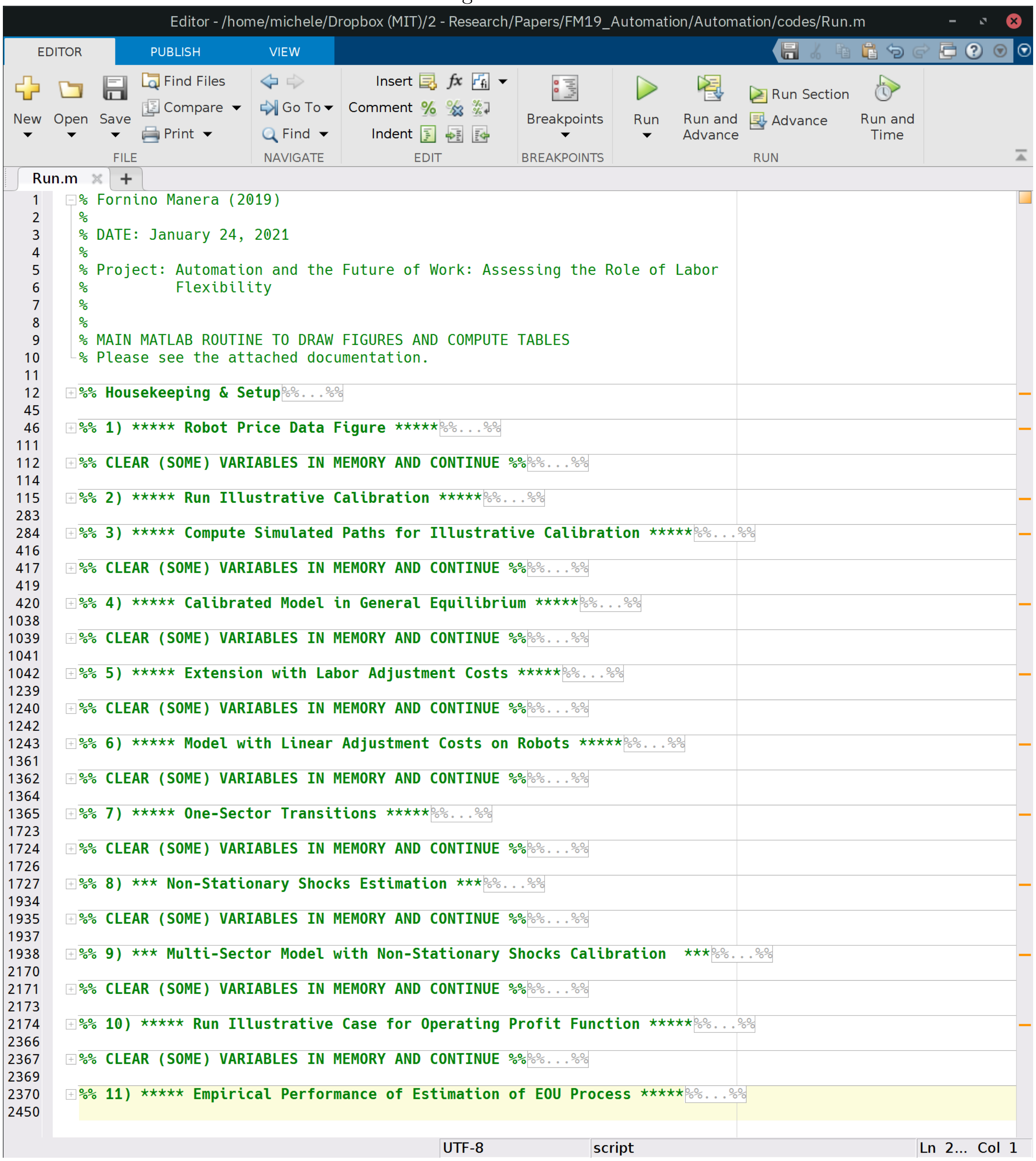
*MIT, email.
†MIT, email.

2.1 Structure of `Run.m`

The main script contains a few computationally intensive parts. The general rule is to run the code in blocks until `‘‘CLEAR (SOME) VARIABLES IN MEMORY AND CONTINUE’’`. It is up to the reader to choose the order with which they want to run the code, but in principle accepting all defaults and hitting “Run” should produce all graphs and tables and save them to the default folders. The structure of the file is shown in Figure 1. The parts are:

1. Robot Price Data Figure
   - Figure 1: Stacked bar chart of total system cost of typical spot-welding robot in U.S. Automotive industry from Sirkin et al. (2015).
2. Run Illustrative Calibration - MUST BE RUN BEFORE NUM 3 BELOW!
   - Figure 2: Stationary Distribution of Illustrative Calibration
3. Compute Simulated Paths for Illustrative Calibration - MUST BE RUN AFTER NUM 2 ABOVE!
   - Figure 3: Simulated Paths of firm-level variables.
4. Calibrated Model in General Equilibrium
   This section leverages the file `EOU_cs_workspace.mat` to avoid a very costly calibration of the parameters of the benchmark quantitative model. If the reader wishes to perform such re-computation, she may run `RunEOUCalibration.m` separately, which produces the `EOU_cs_workspace.mat` file used by `Run.m`.
   - Table 1: Parameters constant across sectors. Please use the output in the screen to read off the numbers
   - Table 2: saved as `calibrationtable.tex`
   - Table 3: Please use the output in the screen to read off the elasticities and their bounds.
   - Figure 4: Main comparative statics exercises in the multi-sector GE model
   - Figure 5: Continuous comparative statics exercises in the multi-sector GE model along with bands for uncertainty on robot price data.
5. Extension with Labor Adjustment Costs
   - Figure 9: Transition to steady state with $p_R = 0$
6. Model with Linear Adjustment Costs on Robots
   - Figure 10: Stationary Distribution for the Linear Adjustment Costs Model
7. One-Sector Transitions
   - Figure 6: Calibrated Transitional Dynamics in Equilibrium along calibrated path of price robots that converges to 0 asymptotically
   - Figure 7: Comparative Statics on steady states of the one-sector model with the parameters calibrated by matching the transition to zero asymptotic robot prices
8. Non-Stationary Shocks Estimation – DOES NOT PRODUCE FIGURES OR TABLES FOR THE PAPER. This section needs the file `RawResiduals.csv` to be available in the subfolder `/data`. This section must be run to produce `GBMStatistics.csv`, which is needed by the next section and is usually provided in the `/data` subfolder.
9. Multi-Sector Model with Non-Stationary Shocks Calibration
   This section leverages the file `GBM_cs_workspace.mat` to avoid a very costly calibration of the parameters of the quantitative model for the non-stationary case. If the reader wishes to perform such re-computation, she may run `RunGBMCalibration.m` separately, which produces the `GBM_cs_workspace.mat` file used by `Run.m`.
   - Table 4: saved as `calibrationtable_GBM.tex`
   - Figure 8: Continuous Comparative Statics on Steady States with Non-Stationary Shock Process (GBM with reset rates)

Figure 1: `Run.m`



```
% Fornino Manera (2019)
%
% DATE: January 24, 2021
%
% 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) ***** Robot Price Data Figure *****%%...%%
%% CLEAR (SOME) VARIABLES IN MEMORY AND CONTINUE %%%%...%%
%% 2) ***** Run Illustrative Calibration *****%%...%%
%% 3) ***** Compute Simulated Paths for Illustrative Calibration *****%%...%%
%% CLEAR (SOME) VARIABLES IN MEMORY AND CONTINUE %%%%...%%
%% 4) ***** Calibrated Model in General Equilibrium *****%%...%%
%% CLEAR (SOME) VARIABLES IN MEMORY AND CONTINUE %%%%...%%
%% 5) ***** Extension with Labor Adjustment Costs *****%%...%%
%% CLEAR (SOME) VARIABLES IN MEMORY AND CONTINUE %%%%...%%
%% 6) ***** Model with Linear Adjustment Costs on Robots *****%%...%%
%% CLEAR (SOME) VARIABLES IN MEMORY AND CONTINUE %%%%...%%
%% 7) ***** One-Sector Transitions *****%%...%%
%% CLEAR (SOME) VARIABLES IN MEMORY AND CONTINUE %%%%...%%
%% 8) *** Non-Stationary Shocks Estimation ***%%...%%
%% CLEAR (SOME) VARIABLES IN MEMORY AND CONTINUE %%%%...%%
%% 9) *** Multi-Sector Model with Non-Stationary Shocks Calibration  ***%%...%%
%% CLEAR (SOME) VARIABLES IN MEMORY AND CONTINUE %%%%...%%
%% 10) ***** Run Illustrative Case for Operating Profit Function *****%%...%%
%% CLEAR (SOME) VARIABLES IN MEMORY AND CONTINUE %%%%...%%
%% 11) ***** Empirical Performance of Estimation of EOU Process *****%%...%%
```

- Figure B.2: fit of distributions for the non-stationary shock case against kernel density estimates of raw TFP residuals

10. Run Illustrative Case for Operating Profit Function

    - Figure A.1: slices of Operating Profit Function Π and Marginal Operating Profit Function Π_R.

11. Empirical Performance of Estimation of EOU Process
    This section needs the file `HamiltonEstimationSample.csv` to be available within the subfolder `/data`.

    - Figure B.1: fit of distributions for the EOU stationary shock case against kernel density estimates of filtered TFP residuals

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

The routine `SetParameters.m` expects to find the files `OUStatistics.csv` `GBMStatistics.csv` in the subfolder `/data`. 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 for legacy considerations.

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 Routines `LaborDemand_*.m`

These routines take 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 Routines `Discretize*.m`

A call is made within `LaborDemand_*.m` to the subroutine responsible for the discretization of the diffusion process (can also be GBM for the non-stationary case). 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 `RunEOUCalibration.m` and `RunGBMCalibration.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 Routines `SolveTransition*.m`

These routines compute 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.

The exercise is done in equilibrium by using `SolveTransitionWrapper.m` in `Run.m`, section 7.

This routine calls the subroutines `GetGrids.m` and `SolvePolicy*.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 Routine `solvePathpR.m`

This routine calibrates an exponential function to the observed path of relative robot prices:

$$p_{R,t} \equiv p_{R,\infty} + (p_{R,2010} - p_{R,\infty}) \exp\{-\alpha (t - 2010)\}$$

Note that we set $p_{R,\infty} = 0$ because this is also the value that the routine produces when we let the solver decide on it. We plot this function along with the data points in Figure 6 in the main text of the paper.

9 Other Routines and Notes

9.1 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, in the subfolder `/guesses` the initial values for the `fsolve` calls. These files are `wpath.csv`, `x0.csv` and `Gamma.csv`. To be clear, these are not needed for execution and can be deleted if the user so wishes.

9.2 `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]

9.3 `num2hms.m`

Routine to create pretty timestamps.

9.4 `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.