build passing

Overview

This repository contains the complete code used to calibrate the model and to compute equilibrium steady-state and transition dynamics of the model in Equilibrium Technology Diffusion, Trade, and Growth by Perla, Tonetti, and Waugh (AER 2020); OpenICPSR project number openicpsr-119393.

The derivation document has the complete set of equations defining the model. All equation numbers in the code refer to this document. General code and derivations for upwind finite difference methods are in the SimpleDifferentialOperators.jl package with detailed derivations.

As in the derivation, the code has a "warmup" model without trade or monopolistic competition to help build an understanding of the transition dynamics in the full model in a related but simpler growth model. The warmup model is also helpful for experimenting with the DAE and finite-difference discretization methods.

Directory of Code

- **/src/full** contains all of the code used to compute the numerical results presented in the paper, given parameter values.
- **/src/calibration** contains all of the code used to create empirical moments and calibrate the model to match those moments. Parameter values are outputs. Readme file describes this part of the code.

Directory of Notebooks for Results in Paper

The following notebooks compute all of the quantitative results and figures presented in the paper. They are organized by Section in accordance with NBER working paper No. 20881 (April 2020 revision).

- **Section 7-1: Calibration—Compute Empirical Moments** jupyter (python) notebook which directly pulls data and constructs empirical moments for the calibration of the PTW model. The resulting output are moments saved as `.csv` files which are then imported into the simulated method of moments calibration routine. **NOTE** the firm-level moments (transition probabilities and firm entry rate) are from the restricted access SLBD. Instructions on how to access the SLBD and access our code are at `/src/calibration/SLBD_instructions/SLBD_instructions.md`.
- **Section 7-2: Calibration—SMM and Resulting Parameter Values** jupyter (python) notebook which calls the MATLAB code to implement the calibration procedure, which finds parameter values such that moments simulated from the model best fit moments in data. This procedure has parameter values as output, which are saved at `/parameters/calibration_params.csv`. Headers for each column identify the parameter associated with each value.
- **Section 7.3 The Sources of the Welfare Gains from Trade—A Quantitative Decomposition** jupyter (julia) notebook that performs the local welfare decomposition and associated results contined in the discussion surrounding the decomposition in the paper.
- **Section 7.4 The Welfare Effects of a Reduction in Trade Costs** jupyter (python) notebook which (i) calls Julia notebook TransitionDynamics.ipynb and computes the transition dynamics of the economy

and (ii) plots the results to create Figures 1-5 of the paper.

- **Section 7.4 The Welfare Effects of a Reduction in Trade Costs, Further Analysis** jupyter (python) notebook which (i) performs calibration routines by calling MATLAB routines and (ii) computes welfare gains (by calling julia notebook SteadyState.ipynb) for alternatives to the baseline model (ACR, Sampson, Atkenson and Burstein, No GBM).
- **Section 7.5. The Role of Firm Dynamics and Adoption Costs** jupyter (python) notebook (i) which calls MATLAB code to generate results for different GBM and delta shock parameter values and (ii) plots the results corresponding with Figure 6 and 7 of the paper. Also computes related results contained in the text of Section 7-5.

Installation and Use

1. Follow the instructions to install Julia and Jupyter

   1a. **Optional**: A Python installation is required for most of the figures.

   1b. **Optional**: Re-running the calibration requires Matlab.

2. Open the Julia REPL (see the documentation above) and then install the package (by entering package mode with `]`) with

   ```
   ] add https://github.com/jlperla/PerlaTonettiWaugh.jl.git
   ```

   2a. **Optional, but strongly encouraged**: To install the exact set of packages used here (as opposed to using existing compatible versions on your machine), first run the following in the REPL

   ```
   using PerlaTonettiWaugh # will be slow the first time, due to precompilation
   cd(pkgdir(PerlaTonettiWaugh))
   ```

   Then, enter the package manager in the REPL (**note** that `]` cannot be copy-pasted on OSX; you need to type it to enter the REPL mode.)

   ```
   ] activate .; instantiate
   ```

3. There are several ways you can run the notebooks after installation

   Using the built-in Jupyter is straightforward. In the Julia terminal

   ```
   using PerlaTonettiWaugh, IJulia
   jupyterlab(dir=pkgdir(PerlaTonettiWaugh))
   ```

Note: If this is the first time running a `jupyterlab()` command inside Julia, it may prompt you to install Julia via conda. Hit `yes`. Also, this will **hand over control of your Julia session to the notebook.** To get it back, hit `control+c` in the Julia REPL.

Alternatively, to use a separate Jupyter installation you may have installed with Anaconda,

```
using PerlaTonettiWaugh
cd(pkgdir(PerlaTonettiWaugh))
; jupyter lab
```

where the last step runs your `jupyter lab` in the shell. **The ; cannot be copy-and-pasted**; to access shell mode, you must type it manually (and you will see your prompt go red).

In either case, the first time running the `using` command will be very slow, as all dependencies need to be precompiled.

4. In addition to the notebooks mentioned above, there is also the `simple_transition_dynamics.ipynb` notebook to solve the simple warm-up variation of the model (which does not appear in the paper) as described in the notes.

NOTE: When using the notebooks for the first time, it will be very slow as the package and its dependencies are all compiled.