Winberry 2018 Code Replication

Brandon Kaplowitz *

January 25, 2021

1 Introduction

This document walks through the main approaches used in replicating Winberry 2018 and the current limitations/future extensions possible. The code is written almost entirely in Python, except for the perturbation component which is written using Dynare in contrast to the original code which was done purely in Matlab/Dynare. Thanks must be given to both Tom Winberry for the original code and the Dynare code library. The Python code uses several key packages including:

- Numpy
- Scipy
- Icecream (for debugging)
- Logging (for debugging)
- Base.IO
- Base.Time
- Matplotlib
- Numba (for speeding up portions)

and the code would not be possible without these libraries.

*Immense thanks to Tom Winberry for writing the original 2018 code which my code is heavily adopted from and the many helpful comments in understanding it.

2 Algorithms (Stationary)

Now I will give a brief description of the key aspects of the code in a verbal Pseudocode format (for succinctness). First, we go through and set the key parameters of the model and approximation in the `replication_winberry_2018.ipynb` main file. Then we run `create_grids.py`, which respectively construct the grids we need on the asset, and epsilon space, as well as scaled versions of the asset space between 0 and 1 to find the Chebyshev zeros. We then run `create_polynomials.py` which generates the Chebyshev polynomials over the asset space which we use to interpolate. Now, we get to the first of our three main algorithms, Young's 2010 method to find a histogram approximation to the steady state endogenous distribution and an estimate of steady state capital. It is implemented in `compute_MC_Residual_Histogram.py` with details left to the appendix of this Description. Next we run `compute_MC_Residual_poly.py` to compute the estimated parametric (exponential) form of the SS distribution and a new guess of K_{SS}. Again high-level descriptions of the implementation are moved to the appendix. Finally, we graph our extract policy functions for savings and consumption for employed $\epsilon = 1$ and unemployed $\epsilon = 0$ individuals as seen in figures 1 and 2. We also plot the endogenous distribution of asset holdings in steady state using our parametric form and Young's histogram method on our grid. We can see these exactly match the original output from Winberry (see Appendix for Graphs: 3 and 4)

3 Algorithms (Dynamics)

In the final portion we prepare our code to be loaded into Dynare as `.mat` data files. The Dynare program is executed through a series of custom `.mod` data files that describe the equations executed in the stationary component and then performs first (or higher order) perturbations. The Dynare file is written to and run using the Matlab package for Python that enables execution of Matlab code from within Python and so it is also executed in the IPython notebook. The `*_steadystate.mod` file loads the steady state values computed from our previous code to Dynare to use as the point around which the perturbation will occur. Due to the lack of novelty here, detailed descriptions of the .mod files are omitted, although they go through and repeat much of the procedure described around the steady state, just in terms of Dynare variables and parameters to define the relevant equations for perturbation. The perturbation method itself is a direct implementation of Schmitt-Grohe et al. 2004. We can see again, that we exactly match

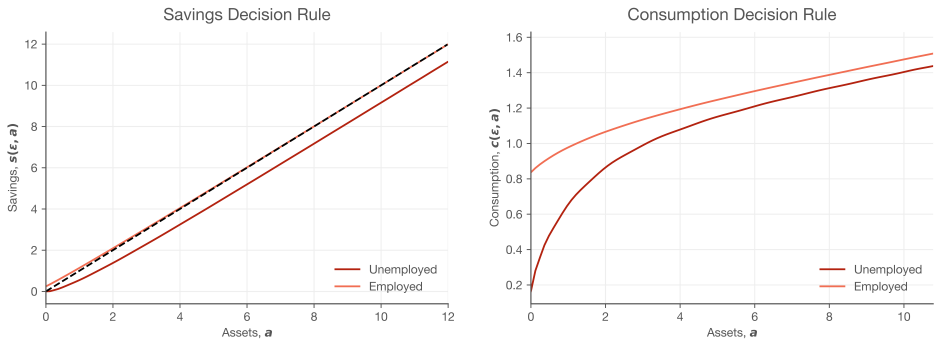

Figure 1: Replicated savings and consumption rules for employed and unemployed workers. Can be reproduced by running IPython notebook (.ipynb) file

Figure 2: Winberry's 2018 calculations of the implied policy rules for savings and consumption in the Aiyagari model, taken from his 2018 paper and User Guide from his website.

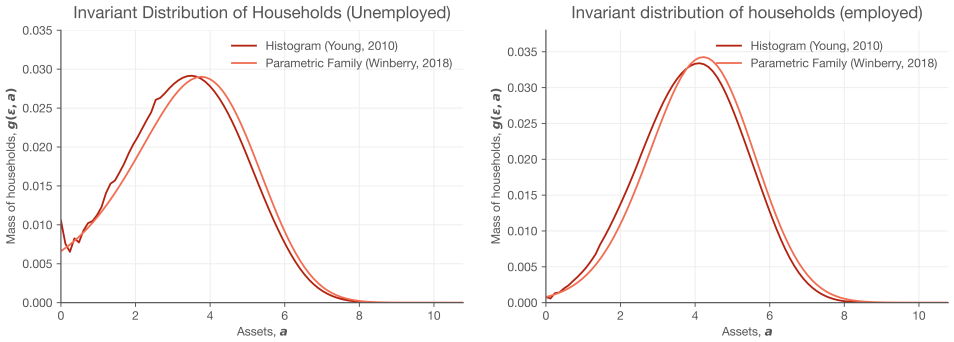

Figure 3: Replicated value for invariant distribution of unemployed and employed households

Figure 4: Winberry's 2018 calculations of the Young 2010 method implied endogenous distribution over assets and his endogenous distribution over assets in the Aiyagari model taken from his 2018 paper and User Guide from his website.

the first order approximation result he gets graphically, and the numerical output from Dynare confirms this as seen in figures 5 and 6. Finally, we note that as expected, in the Aiyagari model the difference between first and higher order perturbations is negligible.

Figure 5: Replicated impulse response functions around Python computed steady state of Order 1 and Order 2 (from left to right).

Figure 6: Original Figure by Winberry 2018 of order 1 perturbation of Aiyagari model around steady state with respect to TFP

4 Robustness

The algorithm's robustness centers on two key assumptions. First, the algorithm will only produce a good approximation to the density if the resulting density is close to the form of a distribution in the exponential family, with some initial mass at 0. If this is not the case, the algorithm will not produce a close fit (though it will produce the projection of the true density onto the exponential family), and may not be useful. Second, if there is a second kink not at some lower bound of the asset space in a policy function, such as from endogenous employment decisions, then the form of the distribution needs to be modified. The most restrictive aspect of the method is that the form of the density and mass points needs to be specified exogenously and in advance. This is a critique that, for example, Bayer and Luetticke 2020 raises with the method. However, it also raises the possibility that the distributional approximation form could be endogenized via a nonparametric method. In particular, a form selected along the lines of sieve MLE or similar nonparametric, but interpolating, procedure might enable a wide number of applications where this method could be both efficient and robust.

5 Future Extensions

Future extensions include extending the code to Winberry 2021 fully (it is only partially implemented at the moment, and adding discount rate hetero-

geneity. Additionally, replacing the parametric estimation of the distribution with a nonparametric technique, whether Sieve MLE or otherwise that captures essential features of the distribution in a low dimensional basis, is a crucial step to extending the technique. Finally, certain aspects of the code can be rewritten in lower level code for performance gains– the sparse operations which are run 54k times represent an opportunity for speeding up the code dramatically by leveraging some of the C sparse libraries. Additionally areas of the code can be JIT-compiled.

6 Appendix: Algorithm Details

6.1 Young 2010 Implementation

`compute_MC_Residual_Histogram.py` works as follows:

1. Solve for the optimal capital level as follows, given an initial guess K_0 of capital and $i = 0$:
   (a) Computed implied prices from K_i.
   (b) Given an initial guess of the next period assets via a rule of thumb decision rule, we loop through and update the EMUC by:
      i. On the first iteration only, projecting the current chebyshev polynomial approximation on the next period assets (effectively assuming $E(V'(a', \epsilon')) = a'^2/2$) and extracting coefficients of the EMUC-estimating chebyshev polynomial
      ii. Using the estimated chebyshev polynomial EMUC to extract next period assets
      iii. Create a new chebyshev polynomials with zeros on next period assets
      iv. Extract an implied future savings and future consumption
      v. Compute the empirical EMUC $\widetilde{EMUC}$ using estimated future consumption from FOC
      vi. Project the new chebyshev polynomial onto $\widetilde{EMUC}$ to extract new coefficients
      vii. Update coefficients as a mixture between the previous estimate and current estimate and loop back to (ii) until these coefficients converge below a given tolerance level.
      viii. Return an estimate of $EMUC$

(c) Using the EMUC estimate and FOC, extract implied consumption and savings, as well as endogenous empirical distribution over assets and epsilon $g'(a', \epsilon')$

(d) Aggregate up savings to get a new $\hat{K}_{i+1}$ guess using Gauss-Legendre quadrature (chosen because it performs exact integration for the polynomial integral here.)

(e) Create a true new estimate of K_{i+1} as a weighted sum of $\hat{K}_{i+1}$ and K_i

(f) Check if $\|K_{i+1} - K_i\|_\infty <$ tol. If so, return $K_{i+1} \equiv K_{hist}$, else set $i \leftarrow i + 1$ and loop back to (a).

2. Finally, we run the sequence above one more time up to part c, with the final K_{hist} estimated to extract the implied consumption, savings and distribution of savings (histogram)

6.2 Parametric Implementation

`compute_MC_Residual_poly.py` works as follows:

1. Compute the first `nmeasure` (here 8) centered moments of the empirical distribution over assets for each ϵ. `nmeasure` will represent the truncation order of our parametric exponential family approximation to the distribution.

2. Solve for the optimal capital level as follows, given an initial guess K_{hist} of capital and $i = 0$:

   (a) Computed implied prices from K_i.

   (b) Follow steps (i)-(viii) of `compute_MC_Residual_histogram.py` to compute the chebyshev projection of the EMUC

   (c) Extract the implied asset prime grid using the EMUC estimate

   (d) Project the exponential family form of the density onto the moments for each epsilon to fit coefficients of exponential family using moment-matching approach

   (e) Compute new moments and constrained fraction of population using grid over asset space, exponential form and Gauss-Legendre quadrature

   (f) Compute maximum error between new moments and old moments and new constrained and old constrained fractions. If larger than tol, loop back to (c).

(g) Else, compute K_{i+1} from using K_{i+1} = first moment * unconstrained fraction + constrained * $\bar{a}$. If $\|K_{i+1} - K_i\| \geq tol$ then loop back to (a) with $i \leftarrow i+1$. Else return estimated coefficients of the EMUC, parameters of the exponential family distribution, moments and new capital level.

References

Bayer, Christian, and Ralph Luetticke. 2020. "Solving discrete time heterogeneous agent models with aggregate risk and many idiosyncratic states by perturbation." *Quantitative Economics* 11, no. 4 (November): 1253–1288. ISSN: 1759-7323. https://doi.org/10.3982/qe1243. https://onlinelibrary.wiley.com/doi/full/10.3982/QE1243%20https://onlinelibrary.wiley.com/doi/abs/10.3982/QE1243%20https://onlinelibrary.wiley.com/doi/10.3982/QE1243.

Schmitt-Grohe, Stephanie, Martín Uribe, Stephanie Schmitt-Grohe, and Martín Uribe. 2004. "Solving dynamic general equilibrium models using a second-order approximation to the policy function." *Journal of Economic Dynamics and Control* 28 (4): 755–775. ISSN: 0165-1889. https://econpapers.repec.org/RePEc:eee:dyncon:v:28:y:2004:i:4:p:755-775.

Winberry, Thomas. 2018. "A method for solving and estimating heterogeneous agent macro models." *Quantitative Economics* 9 (3): 1123–1151. ISSN: 1759-7323. https://doi.org/10.3982/qe740.

———. 2021. *Lumpy Investment, Business Cycles, and Stimulus Policy,* 1, January. https://doi.org/10.1257/aer.20161723.