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. 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 each key aspect of the code in a verbal Pseudocode format. 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, which is implemented in `compute_MC_Residual_Histogram.py`. This code 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)

Next we run `compute_MC_Residual_poly.py` to compute the estimated parametric form of the SS distribution and a new guess of K_{SS}. It 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.

Finally, we graph our extract policy functions for savings and consumption for employed $\epsilon = 1$ and unemployed $\epsilon = 0$ individuals and 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 near-exactly match the original output from Winberry.

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 `*_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.