Replication Package Readme for "The 2000s Housing Boom With 2020 Hindsight: A Neo-Kindlebergerian View"

Gabriel Chodorow-Reich, Adam Guren, and Timothy McQuade

October 2022

This preliminary readme document explains the code used in our paper (it will be expanded and updated soon). Section 1 explains the code for the results in sections 2 and 4 and Appendix B. Section 2 explains how we solve the model computationally and explains the files used to calculate the global model solution and impulse responses. Section 3 explains how we conduct our simulated method of moments estimation and calculate our final model results.

1 Files for Sections 2 and 4 and Appendix B

The subfolder "data/code" contains eponymous Stata .do files to create each table and figure in Sections 2 and 4 and Appendix B. For example, running fig1.do creates figure 1 of the paper. The file masterfile.do creates all of the tables and figures in these sections. The file masterfile.do also calls two files, smm-data.do and ltv-scf.do, that produce data sets used in the model estimation. For completeness, the folder also contains the files cbsa-merged-data.do and prep-cbsa-data.do which were used to create our CBSA-level data set.

2 Model Computation

In this appendix, we detail the computational approach we employ for solving the model. We first discuss approximating the Fokker-Plank partial differential equation via polynomial projection methods. We then discuss how we use a sparse grids approach for our global solution method. Finally, we discuss how Monte-Carlo simulation methods are employed in our iterative global solution method to compute equilbirum mortgage pricing.

2.1 Approximating the Fokker-Plank Equation

The infinite-dimensional distribution of mortgage balances is a key state variable in our model. The measure density $g(M,t)$ of this distribution follows the Fokker-Plank equation:

$$\frac{\partial}{\partial t}g(M,t) = (I_t + \iota) H_t \phi(M/P_t)/P_t - \iota g(M_t) \tag{1}$$

where ι is the arrival rate of liquidity shocks and $\phi(\cdot)$ is the LTV origination distribution. We approximate the loan balance distribution with a Chebyshev series:

$$g(M,t) = \sum_{i=0}^{N} \alpha_i(t) T_i(M),$$

$$\text{with } T_i(M) = \cos\left(i * \arccos\left(\frac{M - \left(M^u + M^l\right)/2}{\left(M^u - M^l\right)/2}\right)\right)$$

the (scaled) i^{th} Chebyshev polynomial of the 1st kind and M^u, M^l the upper and lower bounds of the mortgage balance distribution. The coefficients $\alpha_i(t)$ are set so that the polynomial series interpolates the true measure density at $(M_1, ..., M_N)$ collocation points. We set the collocation nodes to be the (scaled) Chebyshev-Gauss-Lobato (CGL) points:

$$M_i = \frac{1}{2}\left(M^u + M^l\right) - \frac{1}{2}\left(M^u - M^l\right)\cos\left(\frac{i\pi}{N}\right),$$

equal to the $(N-1)$ extrema of the N^{th} Chebyshev polynomial plus the endpoints. We thus require:

$$\sum_{i=1}^{N} \alpha_i(t) T_i(M_j) = g(M_j, t) \text{ for } j = 1, ..., N.$$

Using (1), we find the system of differential equations governing the coefficients:

$$\sum_{i=0}^{N} \alpha_i'(t) T_i(M_j) = (I_t + \iota) H_t \phi(M_j/P_t)/P_t - \sum_{i=0}^{N} \iota \alpha_i(t) T_i(M_j).$$

Letting $\mathcal{A}^* T_i(M) = \iota \alpha_i(t) T_i(M)$ we have:

$$\boldsymbol{\alpha}'(t) = T(\boldsymbol{M})^{-1}\left[\mathcal{A}^* T(\boldsymbol{M})\boldsymbol{\alpha}(t) + (I_t + \iota) H_t \phi(\boldsymbol{M}/P_t)/P_t\right],$$

$$\text{where: } \boldsymbol{\alpha}(t) = \begin{bmatrix} \alpha_0(t) \\ \vdots \\ \alpha_N(t) \end{bmatrix}, \; T(\boldsymbol{M}) = \begin{bmatrix} T_0(M_0) & \cdots & T_N(M_0) \\ \vdots & \ddots & \vdots \\ T_0(M_N) & \cdots & T_N(M_N) \end{bmatrix},$$

which provides a finite-dimensional system of differential equations governing the evolution of the coefficients of the Chebyshev expansion.

2.2 Sparse Grids

Using a Chebyshev series to approximate the loan balance distribution, the state variables in the model are the current dividend D_t, current beliefs about the dividend growth rate m_t^θ, the current housing stock H_t, the moving average of past investment rates $\bar{I}_t$, and the vector of Chebyshev coefficients $\boldsymbol{\alpha}(t)$. Full grid methods for the global solution quickly run into the curse of dimensionality. We thus employ sparse grid techniques to get the global solution of the model. Our approach follows Judd et al. (2014).

We use the Smolyak construction for the sparse grids, once again utilizing Chebyshev-Gauss-Lobatto (CGL) points, that is extrema of Chebyshev polynomials of the 1st kind. In particlar, let d denote the number of state variables. The Smolyak construction proceeds as follows. We first extract a subsequence of unidimensional grid points $S_1, S_2, ...$ from the extrema of the Chebyshev polynomials satisfying $|S_1| = 1$ $|S_i| = 2^{i-1} + 1$ for $i > 1$ and $S_i \subset S_{i+1}$. The first such four nested sets are:

$$
\begin{aligned}
S_1 &= \{0\} \\
S_2 &= \{0, -1, 1\} \\
S_3 &= \left\{0, -1, 1, \frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right\} \\
S_4 &= \left\{0, -1, 1, \frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{-\sqrt{2+\sqrt{2}}}{2}, \frac{-\sqrt{2-\sqrt{2}}}{2}, \frac{\sqrt{2-\sqrt{2}}}{2}, \frac{\sqrt{2+\sqrt{2}}}{2}\right\},
\end{aligned}
$$

equal to the extrema of the 1st, 3rd, 5th, and 7th Chebyshev polynomials of the 1st kind.

To form multidimensional grid points, we can take d-fold products of the unidimensional sets above. In particular, let:

$$\mathcal{K}^{\boldsymbol{i}} = \prod_{j=1}^{d} S_{i_j}$$

for $\boldsymbol{i} = (i_1, ..., i_d)$. Finally, let $\mu \geq 1$ be the order of the approximation. Then, the Smolyak sparse grid is formed as:

$$\mathcal{H}^{d,\mu} = \bigcup_{d \leq |\boldsymbol{i}| \leq d+\mu} \mathcal{K}^{\boldsymbol{i}}$$

where $|\boldsymbol{i}| = i_1 + \cdots + i_d$.

We then construct an approximation to the true global solution $f(x) : \mathbb{R}^d \to \mathbb{R}$ as:

$$\hat{f}(x) = \sum_{n=1}^{M} c_n \Upsilon_n(x)$$

where $\Upsilon_n : \mathbb{R}^d \to \mathbb{R}$ for $n = 1, ..., M$ are a set of d-dimensional basis functions. We then set the coefficients $(c_n)_{n=1}^M$ by minimizing the L^2-norm:

$$\boldsymbol{c} = \arg\min_{\boldsymbol{c}} \left\| f\left(\mathcal{H}^{d,\mu}\right) - \sum_{n=1}^{M} c_n \Upsilon_n\left(\mathcal{H}^{d,\mu}\right) \right\|_2,$$

where $f\left(\mathcal{H}^{d,\mu}\right), \Upsilon_n\left(\mathcal{H}^{d,\mu}\right) \in \mathbb{R}^{\left|\mathcal{H}^{d,\mu}\right|}$ are vectors which evaluate the respective functions at each element of the sparse grid $\mathcal{H}^{d,\mu}$.

2.3 Monte-Carlo Simulation and Global Solution Algorithm

The key challenge for the global solution is determining equilibrium mortgage pricing, which takes the form of mortgage points:

$$W_t = \mathbb{E}_t\left[e^{-r(\tau - t)}\left(M_t - \psi R P_\tau\right)\right].$$

The difficulty is that the mortgage points depend (nonlinearly) on the equilibrium house price function, but of course equilibrium house prices depend on equilibrium mortgage points.

We therefore follow an iterative procedure, in conjunction with Monte-Carlo simulation, to solve for the global solution. Let $W^0\left(\mathcal{H}^{d,\mu}\right)$ be an initial guess for equilibrium mortgage points on the sparse grid $\mathcal{H}^{d,\mu}$. Then at iteration j:

1. Given the current solution $W^j\left(\mathcal{H}^{d,\mu}\right)$, solve for the equilibrium price function $P^j\left(\mathcal{H}^{d,\mu}\right)$ and equilibrium investment function $I^j\left(\mathcal{H}^{d,\mu}\right)$ on the sparse grid.

2. Construct appproximants to the house price and investment functions $\hat{P}^j$ and $\hat{I}^j$ with coefficients $\boldsymbol{c}_P^j, \boldsymbol{c}_I^j$ using the procedure described in the previous subsection.

3. At each point of the sparse grid $\mathcal{H}^{d,\mu}$, use Monte-Carlo simulation with N trials to simulate house prices forward.

   (a) At each point of the sparse grid $\mathcal{H}^{d,\mu}$, simulate dividends and beliefs forward using Euler-Maruyama method for the SDE system.

   (b) Use the investment function approximant $\hat{I}^j$ to simulate forward the housing stock and the house price approximant $\hat{P}^j$ to construct house prices at each step of the simulation.

4. At each point of the sparse grid $\boldsymbol{x} \in \mathcal{H}^{d,\mu}$, compute:

$$W(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_P\left[e^{-r\tau}\left(M(\boldsymbol{x}) - \psi R P_\tau\right) 1\left[\tau_L < \tau_C\right]\right],$$

   where the expectation $\mathbb{E}_P[\cdot]$ is conditional on the simulated future house prices for that Monte-Carlo trial. This gives $W^{j+1}\left(\mathcal{H}^{d,\mu}\right)$.

If $\left\|W^{j+1}\left(\mathcal{H}^{d,\mu}\right) - W^{j}\left(\mathcal{H}^{d,\mu}\right)\right\| < \varepsilon$ for some specified tolerance level $\varepsilon > 0$, then terminate and construct the approximant $\hat{W}^{*}$ with coefficient $\boldsymbol{c}_{W}^{*}$. If not, move to iteration $j+1$ and return to Step 1. This procedure has the advantage of being highly parallelizable.

3 Simulated Method of Moments Estimation Procedure

We run our simulated method of moments on the Boston University Shared Computing Cluster (SCC) using Matlab in five steps. First, we grid over parameters to find a coarse optimum for quartile four. We use the cluster for this step to break a large grid with millions of parameter combinations into thousands of small chunks that can be run independently and in parallel on separate nodes (we use over 1,000 at one time) and then stitched together. Second, we refine to a finer grid and find a fine optimum for quartile four; this grid is the grid pre-programmed into our replication code. Third, we run fine grids for the other quartiles holding fixed some of the parameters estimated in quartile four; these grids are pre-programmed into our replication code as well. To economize on computing resources, we run the model without priced mortgages for the grids, so before moving on, in a fourth step we check that the model with priced mortgages is very close to the model without priced mortgages. This is the case for all but quartile one, so we run another grid for quartile one (also pre-programmed) with priced mortgage. Fifth, we calculate rents (which is quite computationally intensive) for the optimal parameters for each of the four quartiles and make our final tables and figures.

We include in the replication package not only the Matlab code but also the files used to run the Matlab files. In particular the BU SCC uses a batch scheduler, and we include the .sh "shell" files used to submit jobs to the SCC cluster.

3.1 Step-By-Step Guide

The step-by-step guide to run this on the SCC (or a similar server is) is:

1. Create a folder called "SMM" on your server. Upload the "Template,", "data_moments", and "shared_libraries" folders as well as the .sh files in the "shell_files" folder to the SMM folder.

    - The "Template" folder is the folder that is copied for each chunk and contains all the code to run the simulations for each parameter combination.
    - The "shared_libraries" folder contains shared Matlab computational libraries used in the numerics.
    - The "data_moments" folder contains the data moments we load in to set some parameters exogenously.
    - You will need to update the directory paths in the .sh files.

- It is useful to run build_mex.m in matlab now in the "Template" directory, which will set up the mex files for parallelization on your server (you should also do this in "model_main" on your local computer.

2. Run "smm_nopoints.sh"'. This submits 3,000 jobs to the server, each of which analyzes a chunk of parameter combinations. It also runs the collector when everything is done running. The collector's output is "smm_combined.mat." We have included the parameter grid we used for our fine final run; if you change something you may need to adjust the parameter grid which you can do by finding it in "run_smm_nopoints.m."

3. When the collector is done, download "smm_combined.mat" to the "model_main" folder on your local machine and run "smm_moments.m". (Note that "model_main" includes "smm_combined.mat" after we ran things on the server for our final grid.) This loads the moments from the "data_moments" folder, finds the coarse optimum, and saves it in "optimal_params.mat." At this point, if you change parameters, you may need to adjust the grids in "run_smm_nopoints.m" and re-run until the coarse optimum is in the interior of the grid. To clean things up, delete all the "smm_[number]" folders that were created by "run_smm_nopoints.m." Also rename "smm_combined.mat" on the server.

4. Run "smm_nopoints_xc.sh." This submits 3,000 jobs to the server, each of which analyzes a chunk of parameter combinations for the cross-sectional version (quartiles 1-4), holding ϕ, κ, and ρ fixed at the values estimated for quartile four. It also runs the collector when everything is done running. We have included the parameter grid we used; if you change something you may need to adjust the parameter grid which you can do by finding it in "run_smm_nopoints_xc.m." If you have a different optimum from the one we found in step two, you will need to update ϕ, κ, and ρ in "run_smm_nopoints_xc.m" as these are hard coded.

5. When the collector is done, rename the output "smm_combined.mat" to "smm_combined_xc.mat." Download it to the "model_main" folder on your local machine and run "smm_moments_xc.m." (Note that "model_main" includes "smm_combined_xc.mat" after we ran things on the server for our final grid.) This loads the moments from the "data_moments" folder, finds the coarse optimum, and saves it in "optimal_params_xc.mat." At this point, if you change parameters, you may need to adjust the grids in "run_smm_nopoints.m" and re-run until the coarse optimum is in the interior of the grid.

6. Run "figures_pointscompare.m" in the "model_main" folder. This creates a plot that compares the model wihtout priced mortgages with the model with priced mortgages to see if we need to run a grid with priced mortgages. We find that the unpriced mortgage model deviates very little from the priced mortgage model except in quartile 1.

7. Run "smm_points_xc.sh." This submits 3,000 jobs to the server, each of which analyzes a chunk of parameter combinations for quartile 1 only with priced mortgages. We do

not parallelize the calculation of points, so this runs very slowly.

8. When the collector is done, rename the output "smm_combined.mat" to "smm_combined_xc_q1.mat." Download it to the "model_main" folder on your local machine and run "smm_moments_xc_q1.m." (Note that "model_main" includes "smm_combined_xc_q1.mat" after we ran things on the server for our final grid.) This loads the moments from the "data_moments" folder, finds the coarse optimum, and overwrites "optimal_params_xc.mat" with the priced mortgage optimum for q1. At this point, if you change parameters, you may need to adjust the grids in "run_smm_points.m" and re-run until the coarse optimum is in the interior of the grid.

9. Copy the "Template" folder on the server and name it "postgrid_rents." Upload "optimal_params_xc.mat" to the folder and then Run "postgrid_rents.sh" on the server. This calculates the rents, which is computationally intensive, on the server and creates a file called "postgrid_rents.mat" that includes the calculated rents. When it is finished, download this to the "model_main" folder on your local machine.

10. Create the tables and figures by running: "figure_main.m" which creates the tables and figures in the text and "figures_pf.m" which creates the lender perfect foresight figure in the appendix, all of which are saved in "model_main/output." In particular:

    - moments.text, which is the bottom half of Table 2. The top half is entered manually based on "optimal_params_xc.mat"
    - paper_crossicites.pdf, which is Figure 7.
    - paper_rents.pdf, which is Figure 8.
    - paper_structuralcombined.pdf, which is Figure 9.
    - slides_lenderforesight.pdf, chich is Figure C.1.

3.2 Ancillary Calibration Targets

Most of the calibration targets are loaded from csv files created in Stata as described in section 1 (these can be found in the "data_moments" folder or hard coded.

Two sets of calibration targets are worth further discussion. First, the repeat sales residual standard deviation from DataQuick is used as a moment in our calibration procedure. As described in Section 5.3 of the paper, we load all of the DataQuick deeds data from 1988-2013 for non-distressed ales of single family homes and condominiums and run a repeat sales regression of log price on house fixed effects and census tract-by-quarter fixed effects and use the residuals to create this moment. The do file to run this analysis is "dq_residuals.do" in the "data_moments" folder. We unfortunately cannot share the underlying data, which was obtained from DataQuick by Harvard University. Second, we load the initial loan balance distribution from the 1995 Survey of Consumer Finances. This file "ltv-scf-1995.csv" in teh "data_moments" folder is created by the do files described in Section 1. The "import_LTV.m" file in "model_main" loads this and saves it in "LTV_DATA", which is used by the rest of the model files.

4 File By File Summary

Coming Soon