Computational Appendix for Inequality and Aggregate Demand

Adrien Auclert Matthew Rognlie

January 2018

1 Household algorithm

We use the method of endogenous gridpoints, described in Carroll (2006), to solve the household's decision problem. There are two steps: first, we do backward iteration to obtain policy functions; and second, we iterate forward on the distribution of states using these policy functions to obtain the distribution in each period.

We represent asset positions by discrete points on a dense grid $\mathcal{A} \subset [0, \overline{a}]$, where $\overline{a}$ is chosen high enough such that the steady-state policy functions always draw down assets starting at $\overline{a}$ (which we then verify ex post).

Part 1: backward iteration. Suppose we start with a policy function $c_{t+1}(s_{t+1}, a_t)$ for date $t+1$ consumption as a function of date t assets $a_t \in \mathcal{A}$ and the date $t+1$ exogenous state s_{t+1}, and that we have the paths $\{r_t\}$ and $\{y_t(s)\}$ of real interest rates and the after-tax income process (constant if we are solving for the steady state)

Given this, for each (s_t, a_t) pair, whenever the household is not constrained, the following Euler equation can be solved for the consumption level c_t that is consistent with the choice of a_t and the next period's policy:

$$c_t^{-\nu^{-1}} = \beta(1 + r_t) \cdot \mathbb{E}[c_{t+1}(s_{t+1}, a_t)^{-\nu^{-1}} | s_t] \tag{1}$$

The household budget constraint can then be used to solve for the corresponding a_{t-1}:

$$c_t + a_t = (1 + r_{t-1})a_{t-1} + y_t(s_t) \tag{2}$$

Together, (1) and (2) give a mapping from $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, which are the state at time t and the choice of a_t at time t, to the income asset position a_{t-1} at time t that is consistent with this choice. These a_{t-1} are not on the grid $\mathcal{A}$, but we can use linear interpolation to invert this mapping and obtain the mapping from $(s_t, a_{t-1}) \in \mathcal{S} \times \mathcal{A}$ to a_t for all a_{t-1} on the grid $\mathcal{A}$. This mapping, adjusted so that all choices $a_t < 0$ are replaced with the minimum permitted asset $a_t = 0$, is our asset policy function $a_t(s_t, a_{t-1})$. We then write:

$$c_t(s_t, a_{t-1}) = (1 + r_{t-1})a_{t-1} + y_t(s_t) - a_t(s_t, a_{t-1}) \tag{3}$$

to obtain the consumption policy function at date t as a function of date $t-1$ assets $a_{t-1} \in \mathcal{A}$ and the date t exogenous state s_t.

This completes the description of an iteration that starts with the policy function $c_{t+1}(s_{t+1}, a_t)$ for date $t+1$ consumption and gives the policy functions $a_t(s_t, a_{t-1})$ and $c_t(s_t, a_{t-1})$ for date t assets and consumption. This iteration is either repeated until we reach the initial period $t = 0$ (when computing transitional dynamics) or repeated until some criterion for convergence is met (when computing the steady state). As our criterion for convergence, we use the sup norm for the function $c_t(s_t, a_{t-1})$ and a threshold of $\|c^{(n)} - c^{(n-1)}\| < 10^{-8}$.

To begin the backward iteration, when doing transitional dynamics we choose the terminal policy function $c_T(s_T, a_{T-1})$ to be the policy function in the terminal steady state. When solving for the steady state, we can seed the backward iteration with an arbitrary initialization of c; we either use $c(s, a) = y(s) + \max(0.04, r) \cdot a$ as an approximation to the true consumption policy, or the $c(s, a)$ obtained in a previous steady state computation.

Part 2: forward iteration on the distribution. Given a distribution $\Psi_t(s_t, a_{t-1})$ of agents at time t, we obtain the next period's distribution $\Psi_{t+1}(s_{t+1}, a_t)$ as follows.

First, we use the asset policy function to construct the distribution $\Phi_t(s_t, a_t)$. We start by initializing $\Phi_t(s_t, a_t)$ to zero. Next, for each (s_t, a_{t-1}), we interpolate to find the λ such that $a_t(s_t, a_{t-1}) = \lambda a^1 + (1 - \lambda)a^2$, where a^1 and a^2 are the gridpoints closest on either side to $a_t(s_t, a_{t-1})$, and then add $\lambda \cdot \Psi_{t-1}(s_t, a_{t-1})$ to $\Phi_t(s_t, a^1)$ and $(1 - \lambda) \cdot \Psi_{t-1}(s_t, a_{t-1})$ to $\Phi(s_t, a^2)$.[1] Finally, we go from the distribution $\Phi_t(s_t, a_t)$ to $\Psi_{t+1}(s_{t+1}, a_t)$ by using the Markov transition matrix Π for s_t.[2]

When doing transitional dynamics, we start with a distribution Ψ_0 obtained from the ergodic distribution of the initial steady state, with positions a_{-1} changed due to revaluation effects following the unanticipated shock, as described in appendix A.2. We implement the revaluation effects by treating the mapping from old a_{-1} to modified a_{-1} as a "policy function", and then using the procedure described in the previous paragraph to update assets.

[1]Since the asset policy function generically does not put us at a point on the asset grid, this approximates the policy function by sending the right portion of mass to each of the neighboring two gridpoints. The λ and $1 - \lambda$ turn out to be the same as the interpolation weights from interpolation in the backward iteration; since interpolating to find λ is not a bottleneck with efficient interpolation code, however, we have not found significant benefits from making use of this fact.

[2]We have found that the most efficient way to perform this sequence of steps is to use efficient, compiled code (in our case, a Matlab mex file written in C) to go from Ψ_t to Φ_t, and then use simple multiplication by the transition matrix to go from Φ_t to Ψ_{t+1}.

When solving for the steady-state distribution, we can seed the forward iteration with an arbitrary initialization of Ψ, which we choose to be either a uniform distribution over the grid $\mathcal{S} \times \mathcal{A}$ or the steady-state distribution obtained in a previous steady state computation.

Given the distribution of agents and the policy function at each t, we can easily compute any aggregate (such as consumption or assets) needed for the aggregate general equilibrium computations.

2 Calibrating the steady state

Using the calibration targets in section 2.3, we solve for the remaining parameters of the model and the resulting steady state.

First, since the calibration targets steady-state $L = 0.97 < 1$, it follows from complementary slackness (62) that the rate of wage inflation is 0, and therefore in steady state that the rate of price inflation is $\pi = 0$ as well.

Steady state requires that $q = 1$ from (37′). Then, evaluating (39) at steady state, we get $F_K(K, L) = r + \delta$. Since we assume F is Cobb-Douglas, $r + \delta = F_K(K, L) = (1 - \alpha)Y/K$, which is consistent with our calibration targets in section 2.3. For simplicity, we normalize $Y = 1$, which then pins down the TFP parameter A of the Cobb-Douglas production function.

From this we can compute steady-state real wages $\frac{W}{P} = F_L(K, L)$ using (40), and we can solve for steady-state τ using the government budget constraint (57) together with the calibrated values of G_{ss} and B_{ss}.

The only remaining parameter is the household discount rate β, and the only remaining constraint is steady-state version of the consolidated asset market clearing condition (66), which in this case is

$$\int a \cdot d\Psi(s, a) = K + B_{ss} \tag{4}$$

where $\Psi(s, a)$ is the steady-state stationary distribution. The right side of this equation is fixed and the left side is upward-sloping in β, and we use Brent's method to find the unique solution β, obtaining $\Psi(s, a)$ on each iteration using the methods described in appendix 1. Convergence is quick, even given a convergence threshold for (4) of 10^{-15}.

3 Solving for other steady states

Starting from the baseline calibration from appendix 2, we consider the effects of various permanent shocks to the household income process—specifically, to the map $e_t(\cdot)$ from states to endowments. What are the new steady-state equilibria implied by these shocks?

We specify steady state equilibrium as a system of two equations in two unknowns: L and r. We set up these equations by composing several steps:

- Given r and L, we can invert $r + \delta = F_K(K, L)$ to solve for K.
- Given K and L, we can obtain real wages $\frac{W}{P} = F_L(K, L)$.
- Given L, from (14) and (13) we can find steady-state $\bar{B}$ and $\bar{G}$ relative to their initial values:

$$\begin{aligned}\bar{B} - B_{ss} &= -\frac{\epsilon_{DL}}{\epsilon_{DB}}(L - L_{ss}) \\ \bar{G} - G_{ss} &= -\epsilon_{GL}(L - L_{ss})\end{aligned}$$

- Given $\frac{W}{P}$, r, L, $\bar{B}$, and $\bar{G}$, from (57) we can obtain the steady-state labor tax τ.
- Given the after-tax aggregate labor income level $\frac{W}{P}(1-\tau)L$, the rationing given by $\gamma(s, L)$, and the interest rate r, we can solve the household problem to obtain the distribution $\Psi(s, a)$.

From the output of these two steps, we finally set up the two equations:

- **Asset market clearing** $\int a \cdot d\Psi(s, a) = K + \bar{B}$.
- **Monetary policy condition**, which is either $i = 0$ for the ZLB case, $r = r^*$ for the constant-r case, or $L = 1$ for the neoclassical case. (The assumption for the ZLB case, which can then be verified in each experiment, is that if we start from the calibrated depressed steady state with a binding ZLB, the new steady state will remain in it, and therefore $\pi_w = \pi = -\kappa$ and $i = 0$ in steady state.)

The outer loop to solve the steady state iterates over L and r to find some (L, r) that satisfies these two equations.

The approach is to use a modified version of Newton's method: differentiate and use the chain rule on the steps above to find the Jacobian J of the mapping from (L, r) to the errors $(\varepsilon^A, \varepsilon^M)$ in the asset market clearing and monetary policy conditions; and then given these errors, update (L, r) using

$$\begin{pmatrix} L^{updated} \\ r^{updated} \end{pmatrix} = \begin{pmatrix} L \\ r \end{pmatrix} - J^{-1} \begin{pmatrix} \varepsilon^A \\ \varepsilon^M \end{pmatrix} \tag{5}$$

and continue updating using (5) until the errors are very small ($|\varepsilon^A| < 10^{-12}$, $|\varepsilon^M| < 10^{-12}$). This usually takes 4–10 iterations.

For the most part, finding the Jacobian J is simple, since each part of the model above is analytical and has a simple closed-form derivative. The one exception is the household-side mapping from after-tax aggregate labor income $\frac{W}{P}(1-\tau)L$, employment L (which determines rationing), and the interest rate r to total steady-state asset demand $\int a \cdot d\Psi(s, a)$. Numerical derivatives must be used instead (except for $\frac{W}{P}(1-\tau)L$, with respect to which asset demand has a unit elasticity). To avoid repeating the computation for a numerical derivative, however, we compute and store these derivatives when the steady state is first calibrated, and use them for the initial Newton update (5). We then use secant-based updating for the derivative of asset demand with respect to r on the first few iterations, replacing the derivative with whatever derivative exactly rationalizes the change in ε^A on each iteration.

This procedure is somewhat more elaborate than necessary for our simple monetary rules, for each of which we could solve the asset market clearing condition using a simple bisection or Brent's method, but it is easily extensible to more complex monetary and fiscal rules, and it allows us to have a single function that solves for steady-state equilibrium given all such rules.

4 Solving for transition paths for capital

Given the paths $\{r_t\}$ and $\{L_t\}$, and the initial level K_{-1} of capital, the paths $\{q_t\}$ and $\{K_t\}$ solve the equations

$$\frac{1}{\delta\epsilon_I}\left(\frac{K_t - K_{t-1}}{K_{t-1}}\right) = q_t - 1 \tag{37}$$

and

$$(1+r_{t-1})q_{t-1} = F_K(K_{t-1}, L_t) - \left(\frac{K_t}{K_{t-1}} - (1-\delta) + \frac{1}{2\delta\epsilon_I}\left(\frac{K_t - K_{t-1}}{K_{t-1}}\right)^2\right) + \frac{K_t}{K_{t-1}}q_t \tag{39'}$$

for each $t \geq 0$. We truncate at some sufficiently far-out time T, imposing (37) for all $t = 0, \dots, T-1$ and (39′) for all $t = 0, \dots, T$, and using them to solve for $(K_0, \dots, K_{T-1})$ and $(q_{-1}, \dots, q_{T-1})$. For (39′) when $t = T$, we set the forward-looking term q_T to its steady-state level 1.[3]

We treat these $2T+1$ equations as a system of nonlinear equations in $2T+1$ unknowns, to be solved using Newton's method. More concretely, define $\mathbf{q} = (q_{-1}, \dots, q_{T-1})$ and $\mathbf{K} = (K_0, \dots, K_{T-1})$ and the functions $\mathbf{g}(\mathbf{q}, \mathbf{K})$ and $\mathbf{h}(\mathbf{q}, \mathbf{K})$ as the errors in (39′) and (37) respectively, writing

$$\begin{aligned} g_t(\mathbf{q}, \mathbf{K}) &= (1+r_{t-1})q_{t-1} - F_K(K_{t-1}, L_t) - \left(\frac{K_t}{K_{t-1}} - (1-\delta) + \frac{1}{2\delta\epsilon_I}\left(\frac{K_t - K_{t-1}}{K_{t-1}}\right)^2\right) + \frac{K_t}{K_{t-1}}q_t \\ h_t(\mathbf{q}, \mathbf{K}) &= \frac{1}{\delta\epsilon_I}\left(\frac{K_t - K_{t-1}}{K_{t-1}}\right) - (q_t - 1) \end{aligned}$$

where $\mathbf{g} = (g_0, \dots, g_T)$ and $\mathbf{h} = (h_0, \dots, h_{T-1})$.

Then, we stack into a single system $\mathbf{f}(\mathbf{X})$, where $\mathbf{f} = \begin{bmatrix}\mathbf{g} \\ \mathbf{h}\end{bmatrix}$ and $\mathbf{X} = \begin{bmatrix}\mathbf{q} \\ \mathbf{K}\end{bmatrix}$, and solve $\mathbf{f}(\mathbf{X}) = 0$

[3] Since this is a two-dimensional system, it is straightforward to find the linearized dynamics around the steady state, which can be used to substitute a more precisely accurate value of q_{T+1}. We experimented with this but found that it made minimal difference in our application, since for other reasons we always choose T such that q_{T+1} is extremely close to its steady-state value.

using Newton's method. To do so, we need the Jacobian J corresponding to $\mathbf{f}(\mathbf{X})$. This takes the form

$$J = \begin{bmatrix} \frac{\partial \mathbf{g}}{\partial \mathbf{q}} & \frac{\partial \mathbf{g}}{\partial \mathbf{K}} \\ \frac{\partial \mathbf{h}}{\partial \mathbf{q}} & \frac{\partial \mathbf{h}}{\partial \mathbf{K}} \end{bmatrix} \tag{6}$$

(6) is quite sparse, with a limited set of nonzero entries: g_t depends only on q_{t-1}, q_t, K_{t-1}, and K_t, while h_t depends only on q_t, K_{t-1}, and K_t. The corresponding partial derivatives are

$$\frac{\partial g_t}{\partial q_{t-1}} = 1 + r_{t-1}; \qquad \frac{\partial g_t}{\partial q_t} = \frac{K_t}{K_{t-1}}$$

$$\frac{\partial g_t}{\partial K_{t-1}} = -F_{KK}(K_{t-1}, L_t) + \left(q_t - 1 - \frac{1}{2\delta\epsilon_I}\left(\frac{K_t - K_{t-1}}{K_{t-1}}\right)^2 \right) \frac{K_t}{K_{t-1}^2} + \frac{1}{\delta\epsilon_I}\left(\frac{K_t - K_{t-1}}{K_{t-1}}\right)$$

$$\frac{\partial g_t}{\partial K_t} = \left(q_t - 1 - \frac{1}{2\delta\epsilon_I}\left(\frac{K_t - K_{t-1}}{K_{t-1}}\right)^2 \right) \frac{1}{K_{t-1}} + \frac{1}{\delta\epsilon_I}\left(\frac{K_t - K_{t-1}}{K_{t-1}}\right)$$

$$\frac{\partial h_t}{\partial q_t} = -1; \qquad \frac{\partial h_t}{\partial K_{t-1}} = -\frac{1}{\delta\epsilon_I}\frac{K_t}{K_{t-1}^2}; \qquad \frac{\partial h_t}{\partial K_t} = \frac{1}{\delta\epsilon_I}\frac{1}{K_{t-1}}$$

Computing these and populating J in (6), we can then apply Newton's method to solve $\mathbf{f}(\mathbf{X}) = 0$ for $\mathbf{X} = \begin{bmatrix} \mathbf{q} \\ \mathbf{K} \end{bmatrix}$. Starting with some arbitrary initial iterate $\mathbf{X}^{(0)}$, which we usually pick[4] to be a path that is already at the new steady state, iterating gives

$$\mathbf{X}^{(n)} = \mathbf{X}^{(n-1)} - J^{-1}\mathbf{f}(\mathbf{X}^{(n-1)}) \tag{7}$$

Since the J used here is exact and analytical, (7) achieves second-order convergence.[5]

Advantages over alternative methods. This method of solving for q and K has two crucial advantages over alternative methods.

First, unlike many shooting-based methods, it is inherently stable. This stability stems from the underlying stability of the economic model: values of g_t and h_t different from zero can be interpreted as wedges in the valuation and investment equations, respectively, and the solution

[4]When we are solving for $\mathbf{q}$ and $\mathbf{K}$ as part of the general equilibrium transition dynamics discussed in the next subsection, it is natural to start instead with the $\mathbf{X}^{(0)}$ equal to the paths from the last iteration, and additional speedup is possible by doing step (7) first using the J^{-1} from the previous iteration.

[5]For very large T, inverting J using standard methods might become costly enough that it would be advantageous to take advantage of the sparsity of J and use methods specific to sparse matrices with small bandwidth (reordering the nonzero entries so that they are in a small band around the diagonal). We have generally used $T = 250$ or $T = 500$, which turn out not to be large enough for this to make a nontrivial difference.

of a distorted investment problem subject to these wedges is well-behaved. Errors at t in one iteration have little impact on distant t', because these errors correspond to wedges and the economic impact of wedges on the allocation is steadily dampened in this model as we move away from the period in which the wedges occur.[6]

Second, obtaining J^{-1} to use in (7) is essential to the general equilibrium transition algorithm in subsection 5, because it gives the first-order response of $\mathbf{K}$ and $\mathbf{q}$ to shocks to the paths of r and L, as we discuss next.

First-order impact of changing inputs to the investment problem. The paths of both r and L appear in $\mathbf{f}$. Defining $\mathbf{r} = (r_0, \ldots, r_{T-1})$ and $\mathbf{L} = (L_0, \ldots, L_{T-1})$, and making the dependence of $\mathbf{f}$ on them implicit by writing the equations as $\mathbf{f}(\mathbf{X}; \mathbf{r}, \mathbf{L}) = 0$, the implicit function theorem implies that

$$\frac{d\mathbf{X}}{d\mathbf{r}} = -J^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{r}}; \qquad \frac{d\mathbf{X}}{d\mathbf{L}} = -J^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{L}} \tag{8}$$

Note that the only dependence of $\mathbf{f}$ on either $\mathbf{r}$ or $\mathbf{L}$ comes through $\mathbf{g}$, and that the only nonzero entries in $\frac{\partial \mathbf{g}}{\partial \mathbf{r}}$ and $\frac{\partial \mathbf{g}}{\partial \mathbf{L}}$ are

$$\frac{\partial g_t}{\partial L_t} = -F_{KL}(K_{t-1}, L_t); \qquad \frac{\partial g_t}{\partial r_{t-1}} = q_{t-1} \tag{9}$$

Combining (8) and (9) gives us $\frac{d\mathbf{X}}{d\mathbf{r}}$ and $\frac{d\mathbf{X}}{d\mathbf{L}}$. These first-order sensitivities of $\mathbf{X} = \begin{bmatrix} \mathbf{q} \\ \mathbf{K} \end{bmatrix}$ to $\mathbf{r}$ and $\mathbf{L}$ will be crucial in the next subsection, where they allow us to construct the Jacobian in a Newton algorithm computation of general equilibrium.

5 Solving for the general equilibrium transition paths

General approach. Our general approach to solving for GE transition dynamics is to write equilibrium as a nonlinear system of equations, and then use Newton iterations over time paths to find the solution to this system. Specifically, we:

a) Truncate the model at some date T sufficiently far after the shock, and assume that at this point the terminal steady-state equilibrium has been reached.

b) Rewrite the system of equations characterizing general equilibrium for these truncated time paths as a function f from a reduced set of unknown time paths to a reduced set of equilibrium conditions. This function f is written as the composition of many simpler functions $f_1, \ldots, f_k$. (All model equations that are not listed as part of the reduced set of equilibrium conditions are inverted and incorporated into the function.)

[6]More broadly, we think that is a highly desirable property for iterative solution algorithms, and it will generally be the case when we write equilibrium conditions such that errors correspond to economically interpretable wedges.

c) Given a guess for the unknowns, apply f to obtain the errors for each equilibrium condition.

d) Find the approximate Jacobian J of f at the point evaluated in step 2. To do so, find the analytical or numerical derivative of each $f_1, \ldots, f_k$ whenever practical, and specify an *approximate* derivative otherwise. (In our case, the latter is only necessary for the f_i corresponding to the household decision problem, since everything else is analytical.) Then, use the chain rule to combine these into an approximate Jacobian for f.

e) Use the errors in step 2 and the approximate Jacobian in step 3 to do a Newton iteration to find a new guess for the unknowns, then return to step 2. Continue iterating until the equilibrium condition errors are sufficiently small.

This approach, which iterates over time paths to obtain general equilibrium transition dynamics, is closely related to the approach used in several recent papers, including the seminal work of Guerrieri and Lorenzoni (2017). Our key innovation is to replace the usual ad-hoc iterations with approximate Newton iterations, which make use of an approximate Jacobian. This makes each iteration much more accurate, substantially cutting the number of iterations needed to find equilibrium. It is also more robust to different parameterizations and changes in the model equations—unlike ad-hoc methods, which often need to be tweaked for convergence as different variations of a model are tried, the Newton approach adjusts automatically.

The major barrier to applying this method is calculating the Jacobian of f. As discussed above, we simplify this by individually differentiating the functions $f_1, \ldots, f_k$, most of which have easily computed derivatives, and then combining the results with the chain rule.[7] When the derivative of some f_i cannot be efficiently computed, we instead use an approximation of the function's behavior, often based on a simplified economic model. These approximate derivatives mean that we cannot achieve the second-order convergence in the standard Newton method, since the approximation error in J does not vanish as we converge to equilibrium. Nonetheless, we can still achieve very rapid first-order convergence, which in practical terms is nearly as good and requires few iterations. It is worth emphasizing although there are approximations in calculating the Jacobian J for each iteration, these do not imply any approximation in the final solution, which is exact (up to the chosen tolerance).

Formal notation. Denote the unknown time paths in the truncated model by $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Choose some subset of m of them to be the unknowns over which we will iterate in the outer Newton loop; without loss of generality, suppose that these are $\mathbf{x}_1, \ldots, \mathbf{x}_m$.

[7]Splitting the calculation in this way has an added benefit: for each f_i, it is possible to calculate the extent to which the actual value calculated on one iteration deviates from a first-order approximation based on the previous iteration. This makes it easy to see where the inaccuracies in J are coming from, thereby identifying the bottlenecks to rapid convergence. Generally, unless there is a programming error or the model is extremely nonlinear, these inaccuracies should only be in the non-analytically differentiable f_i, especially after the first few iterations.

Now, choose some subset of m equations to be the equilibrium conditions that we target in the iteration. Invert in order to write all other equilibrium equations as functions $f_1, \ldots, f_k$. Each of these functions f_i takes in a set of unknowns indexed by $\mathcal{I}_i$, and outputs some mix[8] of other unknowns and equilibrium condition errors, indexed by $\mathcal{O}_i^U$ and $\mathcal{O}_i^E$ respectively:

$$(\{\mathbf{x}_j\}_{j \in \mathcal{O}_i^U}, \{\mathbf{e}_j\}_{j \in \mathcal{O}_i^E}) = f_i(\{\mathbf{x}_j\}_{j \in \mathcal{I}_i})$$

We require that

$$j \in \mathcal{I}_i \implies j \in \{1, \ldots, m\} \text{ or } j \in \mathcal{O}_{i'}^U \text{ for some } i' < i$$

i.e. that each input to some f_i is either one of the unknowns over which we perform the outer iteration, $\mathbf{x}_1, \ldots, \mathbf{x}_m$, or an unknown that is output by a previous $f_{i'}$. This allows $f_1, \ldots, f_k$ to be evaluated sequentially, taking in $\mathbf{x}_1, \ldots, \mathbf{x}_m$ as initial inputs.

We also require that the sets $\mathcal{O}_i^U$ and $\mathcal{O}_i^E$ are all pairwise disjoint, and that

$$\begin{aligned} \mathcal{O}_1^U \cup \cdots \cup \mathcal{O}_k^U &= \{m+1, \ldots, n\} \\ \mathcal{O}_1^E \cup \cdots \cup \mathcal{O}_k^E &= \{1, \ldots, m\} \end{aligned}$$

i.e. that together, $f_1, \ldots, f_k$ produce as outputs all unknowns $\mathbf{x}_{m+1}, \ldots, \mathbf{x}_n$ and also all errors for the equilibrium conditions $\mathbf{e}_1, \ldots, \mathbf{e}_m$ that we are targeting. Together, these assumptions imply that we can compose $f_1, \ldots, f_k$ to produce a function f

$$(\mathbf{e}_1, \ldots, \mathbf{e}_m) = f(\mathbf{x}_1, \ldots, \mathbf{x}_m)$$

that takes the unknowns over which we iterate to the errors in equilibrium conditions. Solving for equilibrium is then equivalent to finding a root of f.

To differentiate f, we apply the chain rule sequentially for $f_1, \ldots, f_k$. Specifically, at f_i, for each output $\mathbf{x}_o$ or $\mathbf{e}_o$, we write for each $\mathbf{x}_l \in \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$

$$\begin{aligned} \frac{d\mathbf{x}_o}{d\mathbf{x}_l} &= \sum_{j \in \mathcal{I}_i} \frac{\partial \mathbf{x}_o}{\partial \mathbf{x}_j} \cdot \frac{d\mathbf{x}_j}{d\mathbf{x}_l} \\ \frac{d\mathbf{e}_o}{d\mathbf{x}_l} &= \sum_{j \in \mathcal{I}_i} \frac{\partial \mathbf{e}_o}{\partial \mathbf{x}_j} \cdot \frac{d\mathbf{x}_j}{d\mathbf{x}_l} \end{aligned}$$

where $\frac{d\mathbf{x}_j}{d\mathbf{x}_l}$ has already been calculated in a previous iteration (note that if $\mathbf{x}_j \in \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, then it is the identity if $j = l$ and zero if $j \neq l$). Note that in some cases, it may not be computationally practical to compute $\frac{\partial \mathbf{x}_o}{\partial \mathbf{x}_j}$ or $\frac{\partial \mathbf{e}_o}{\partial \mathbf{x}_j}$ exactly,

Continuing this process for all f_i, we end up with derivatives $\frac{d\mathbf{e}_o}{d\mathbf{x}_l}$ for each $\mathbf{e}_o \in \{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$

[8]It would be simpler notationally to write each f_i as having only one output. In practice, however, often several outputs are generated together as part of a joint computation, and we therefore choose to write functions in this way to more closely hew to the code.

and $\mathbf{x}_l \in \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$. Stacking these, we get the Jacobian matrix J for f:

$$J = \begin{pmatrix} \frac{d\mathbf{e}_1}{d\mathbf{x}_1} & \cdots & \frac{d\mathbf{e}_1}{d\mathbf{x}_m} \\ \vdots & \ddots & \vdots \\ \frac{d\mathbf{e}_m}{d\mathbf{x}_1} & \cdots & \frac{d\mathbf{e}_m}{d\mathbf{x}_m} \end{pmatrix}$$

On each iteration of the Newton algorithm, we start with guesses for $(\mathbf{x}_1, \ldots, \mathbf{x}_m)$ and then apply $f_1, \ldots, f_k$ to obtain the corresponding $(\mathbf{e}_1, \ldots, \mathbf{e}_m)$. Then, calculating J as above, we update

$$\begin{pmatrix} \mathbf{x}_1^{new} \\ \vdots \\ \mathbf{x}_m^{new} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_m \end{pmatrix} - J^{-1} \begin{pmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_m \end{pmatrix}$$

and repeat, continuing until $(\mathbf{e}_1, \ldots, \mathbf{e}_m)$ are sufficiently close to zero.

Implementation for this paper. For the exercises in this paper, we set $m = 3$, choosing L, r, and $\overline{B}$ to be the unknowns over which we iterate, and having the equilibrium conditions be asset market clearing, the condition governing monetary policy, and the fiscal flow budget constraint.

We found that these choices made it easy for us to use the same code to evaluate different parameterizations of the model, particularly under different monetary and fiscal rules. Although, for instance, r is trivial under constant-r policy and L is trivial under neoclassical policy, having both L and r as unknowns in the iteration allows us to combine both cases, just substituting different conditions governing monetary policy and then following the same iterative algorithm.[9]

The list of functions is below, including discussion of derivatives when their computation is nontrivial.

- $(\mathbf{q}, \mathbf{K}) = f_1(\mathbf{L}, \mathbf{r})$ uses the algorithm from appendix 4 to solve the investment problem. It obtains their derivatives using (8) and (9).
- $(\mathbf{Y}, \mathbf{W}/\mathbf{P}) = f_2(\mathbf{L}, \mathbf{K})$ uses the production function F and its derivative F_L to get output and real wages from aggregate factor inputs.
- $\pi = f_3(\mathbf{W}/\mathbf{P})$ obtains price inflation from the path of real wages, assuming that downward nominal wage rigidity is binding (as it is in all our experiments except where the monetary rule is such that π is irrelevant).

[9]Similarly, although in principle we could derive $\overline{B}$ from L and r given some fiscal rule rather than iterating over it in the outer loop, we found that this made the algorithm less robust and stable, and it also could not accommodate active, non-Ricardian fiscal rules (though these are not in the paper).

- $(\boldsymbol{\tau}, \mathbf{G}) = f_4(\mathbf{L}, \mathbf{r}, \overline{\mathbf{B}}, \mathbf{W})$ obtains the labor tax rate and government spending implied by the fiscal rule (which can be general).

- $(\mathbf{1} - \boldsymbol{\tau})\mathbf{W} = f_5(\mathbf{W}, \boldsymbol{\tau})$ simply combines wages and taxes to obtain aggregate after-tax income, which is all that matters for the household problem.

- $\mathbf{A} = f_6(\overline{\mathbf{B}}, \mathbf{q}, \mathbf{K})$ simply combines bonds, capital, and its value to obtain the overall supply of assets $A = \overline{B} + qK$.

- $\mathbf{e}_1 = f_7(\mathbf{L}, \mathbf{r}, \boldsymbol{\pi})$ evaluates the condition governing monetary policy (e.g. $L = 1$ for neoclassical, $r = r^*$ for constant-r, somewhat more complex for Taylor rules with and without ZLB).

- $\mathbf{e}_2 = f_8(\mathbf{r}, \mathbf{L}, \overline{\mathbf{B}}, \mathbf{W}, \boldsymbol{\tau}, \mathbf{G})$ evalutes the flow government budget constraint.

- $\mathbf{e}_3 = f_9((\mathbf{1} - \boldsymbol{\tau})\mathbf{W}, \mathbf{L}, \mathbf{r}, \mathbf{A})$ evaluates the asset market clearing constraint. Asset supply is already known from $\mathbf{A}$, and to obtain asset demand this solves the household problem as discussed in appendix 1. An exact derivative is not practical to calculate, and our solution is discussed below.

Composing $f_1, \ldots, f_9$ gives a function $f : (\mathbf{L}, \mathbf{r}, \overline{\mathbf{B}}) \to (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$, and we solve for equilibrium by finding the zero of this function using the Newton method outlined earlier. Generally, this takes 5 to 15 iterations to achieve accuracy to within a tolerance of 10^{-8} for all three equilibrium conditions; each iteration takes a few seconds on an ordinary laptop.

Approximate derivatives for the household problem. There is no simple way to compute a full matrix of partial derivatives for the partial equilibrium household problem in f_9. Directly computing this numerically would require a separate run of the procedure in appendix 1 when differentiating with respect to each input at each point in time, which would be extremely expensive.

Instead, we use an *approximate* model of the household.[10] Specifically, we replace the heterogenous household sector with a hypothetical, single representative agent with a discount rate that depends on its asset position

$$\beta_t(a_t) \equiv \beta \cdot \left(\frac{a_t}{\overline{a}_t}\right)^{-\zeta}$$

with some elasticity $\zeta > 0$ (where $\overline{a}_t$ is the actual path of assets in the most recent iteration). Locally, household optimization subject to an endogenous discount rate of this form bears some

[10]In recent work on these methods joint with Ludwig Straub, we have experimented with instead nonparametrically obtaining an approximate matrix of partial derivatives by interpolating and extrapolating a few partial runs of the household problem. This appears promising and may deliver much greater accuracy, at no additional cost, when compared to the parametric approximate model discussed here, cutting the number of Newton iterations needed even further. These techniques are employed in Straub (2017), and will likely be employed in future algorithms.

similarity to aggregate household behavior in our heterogenous agent, incomplete markets model: for instance, MPCs are above the permanent-income level. But it is easy to solve in closed form for the local behavior of this model in response to shocks, thereby obtaining an approximate derivative for function f_9 above.

Of course, it is not clear ζ should be. For shocks to $(1-\tau)W$, L, and r, before starting our algorithm we choose the ζ that causes the representative agent model to best match the actual household model's impulse responses to shocks occurring at $t = 5$ and $t = 30$. We construct a full approximate derivative assuming this ζ separately in response to each shock.[11] We then reuse these ζ across many calculations.

References

Carroll, Christopher D., "The Method of Endogenous Gridpoints for Solving Dynamic Stochastic Optimization Problems," *Economics Letters*, 2006, *91* (3), 312–320.

Guerrieri, Veronica and Guido Lorenzoni, "Credit Crises, Precautionary Savings, and the Liquidity Trap," *Quarterly Journal of Economics*, August 2017, *132* (3), 1427–1467.

Straub, Ludwig, "Consumption, Savings, and the Distribution of Permanent Income," *Manuscript*, November 2017.

[11] For shocks to initial assets A_{-1}, which incorporate revaluation effects from the unanticipated shock, we sidestep this by directly calculating the impulse response of the actual around the steady state, since this only requires a single forward iteration to solve for the evolution of the distribution.