Econometrica Supplementary Material

SUPPLEMENT TO "USING THE SEQUENCE-SPACE JACOBIAN TO SOLVE AND ESTIMATE HETEROGENEOUS-AGENT MODELS"

(*Econometrica*, Vol. 89, No. 5, September 2021, 2375–2408)

ADRIEN AUCLERT
Department of Economics, Stanford University, CEPR, and NBER

BENCE BARDÓCZY
Federal Reserve Board of Governors

MATTHEW ROGNLIE
Department of Economics, Northwestern University and NBER

LUDWIG STRAUB
Department of Economics, Harvard University and NBER

APPENDIX A: GENERALIZING THE FAKE NEWS ALGORITHM

A.1. *Direct Applications of the Existing Framework*

FIRST, WE IDENTIFY several ways in which the existing framework can be adapted to include model elements differing from our examples, with either no change or limited changes to the algorithm.

Non-Grid Representations of the Value Function. Framework (10)–(12) assumes that the distribution is discretized as a finite grid, and that y and Λ give the value of the output Y at each point and the transition probabilities between points. None of this places any restriction, however, on how the value function is discretized as $\mathbf{v}$. Our algorithm therefore accommodates a variety of discrete representations of $\mathbf{v}$ (splines, Chebyshev polynomials, parametric, etc.) without any modification.

Higher Moments. At first glance, (12) seems to require that we are taking the mean $\mathbf{y}_t'\mathbf{D}_t$ of some individual outcome $\mathbf{y}_t$. But if we redefine the individual outcome as $(\mathbf{y}_t)^k$, then we can calculate the kth (non-centered) power moment $((\mathbf{y}_t)^k)'\mathbf{D}_t$ as well. Applying this strategy as necessary for different k and combining the results using a simple block, we can obtain the Jacobian for any transformation of these moments, such as the variance, the coefficient of variation, or a CES price index.

This allows us to calculate many moments of interest, though not all; for instance, for some distributional moments like the Gini coefficient, we need the general framework of the next section.[1]

Adrien Auclert: aauclert@stanford.edu
Bence Bardóczy: bardoczy@u.northwestern.edu
Matthew Rognlie: matthew.rognlie@northwestern.edu
Ludwig Straub: ludwigstraub@fas.harvard.edu

[1]Note, however, that this is only necessary if we need Jacobians for these moments. If, instead, we only need impulse responses for these moments (and the moments themselves are not needed to solve for general equilibrium), we can apply the linearized (10)–(12) to the equilibrium impulse responses for $\mathbf{X}_t$ and recover impulse responses $\mathbf{y}_t$ and $\mathbf{D}_t$, then directly compute any desired moments from these.

Leads and Lags. The equations (10)–(12) include only contemporaneous $\mathbf{X}_t$, without any leads or lags. What if, instead, a lagged or future variable appears, such as $\mathbf{X}_{t-1}$ or $\mathbf{X}_{t+1}$? In the case of leads like $\mathbf{X}_{t+1}$, the algorithm works without any change: Lemma 1 goes through without modification, so that iterating backward from a shock at $T-1$ still gives the $d\mathbf{y}_0^s$ and $d\Lambda_0^s$ needed in Proposition 1. Intuitively, this is because our backward iteration already incorporates the effects of a future shock working through the value function, and nothing more is needed to handle the case where future X also appears directly in (10)–(12).

If, on the other hand, a lag like $\mathbf{X}_{t-1}$ appears in (10)–(12), then it is no longer true that $\mathbf{y}_t^s = \mathbf{y}_{ss}$ and $\Lambda_t^s = \Lambda_{ss}$ for $t = s+1$ in (16), because both are affected by the lagged shock. Lemma 1 fails, and our method—which does not account for the possibility that "past" shocks affect current individual outcomes at a particular point in the state space—no longer works.

The simplest solution is to transform variables outside the heterogeneous-agent block, for example, define a new variable $\tilde{\mathbf{X}}_t \equiv \mathbf{X}_{t-1}$ (which can be the output of a simple block taking in $\mathbf{X}$), so that within the algorithm, only a contemporaneous variable $\tilde{\mathbf{X}}_t$ appears, matching the exact form of (10)–(12).[2]

Discrete Choice With Taste Shocks. The models we simulate in this paper all have the feature that policy functions are continuous in the underlying idiosyncratic state variables. This is no longer generally the case for models that feature discrete choices, such as lumpy adjustment of durables, price setting with menu costs, or a discrete labor-leisure choice (see, e.g., Bardóczy (2020)). For such models, if the problem is discretized using a grid, linearization can give extremely misleading results: if none of the grid points at a point where the discrete choice changes, then the first-order response of the discrete choice to any shock is zero.

This problem is common to all perturbation methods. One standard solution is to assume continuously-distributed i.i.d. taste shocks affecting the value of each discrete choice. The probability of each discrete choice then varies continuously with the (pre-taste shock) state.[3] To write the model in the form (10)–(12), $\mathbf{D}_t$ should then be the discretized pre-taste shock distribution, and $\mathbf{v}_t$, $\mathbf{y}_t$, Λ_t should be the expected values at each state in this distribution.

An alternative to taste shocks, which we discuss in the next section, is to use a continuous representation of the distribution rather than a discrete grid.

Endogenous Distribution. The distribution $\mathbf{D}_t$ in equation (11) is assumed to be unaffected by the current shock $\mathbf{X}_t$ and the value function $\mathbf{v}_{t+1}$. In short, it is predetermined at date t. What if we want events at date t to affect the distribution—for instance, if shocks at date t can affect capital gains on wealth at date t, or can affect the probability of unemployment at date t?

Within the framework (10)–(12), the solution is to keep $\mathbf{D}_t$ predetermined at date t, and incorporate these shocks into the functions v, Λ, y instead. For instance, in our two-asset

[2]To implement the fake news algorithm directly with lags, we would need to calculate $\mathbf{y}_0^s$ and Λ_0^s for all s from $-u$ to $T-1$, where u is the maximum lag length, use these to build a fake news matrix $\mathcal{F}$ with columns $s = -u, \dots, T-1$, then apply the recursion $\mathcal{J}_{t,s} = \mathcal{J}_{t-1,s-1} + \mathcal{F}_{t,s}$ in step 4 starting from this new leftmost column $-u$. In our experience, this is more difficult and error-prone than the $\tilde{\mathbf{X}}_t$ solution above.

[3]One particularly convenient approach is to use extreme value taste shocks as in Iskhakov, Jørgensen, Rust, and Schjerning (2017), which are smooth and lead to logit choice probabilities. Bardóczy (2020) implemented the fake news algorithm using this approach.

HANK example, the date-0 return on the illiquid asset includes an endogenous capital gain. The distribution $\mathbf{D}_0$ gives the state prior to this capital gain, and then the ex post return on illiquid assets, r_0^a, is included as part of $\mathbf{X}_0$ as an input to v, Λ, y.

Similarly, if the probability of unemployment is endogenous at date t, $\mathbf{D}_t$ should still be the state *prior* to the realization of the idiosyncratic unemployment shock, and then v, Λ, y should take *expectations* over the realizations of this shock.

Although this procedure can virtually always be used to put a model into the framework of (10)–(12), it becomes unwieldy in complex cases. In Appendix A.3, we describe how to apply the fake news algorithm to a model where the distribution evolves over multiple subperiods within each period. This provides a more formal, structured approach.

A.2. *Nonlinear Y or D*

We now generalize our algorithm to the case of nonlinear functions for $\mathbf{D}_{t+1}$ and $\mathbf{Y}_t$ in (11)–(12). The key is the following generalization of Proposition 1.

PROPOSITION 1: *Assume that equations* (11) *and* (12) *are replaced, respectively, by*

$$\mathbf{D}_{t+1} = D(\mathbf{v}_{t+1}, \mathbf{X}_t, \mathbf{D}_t), \tag{39}$$

$$\mathbf{Y}_t = Y(\mathbf{v}_{t+1}, \mathbf{X}_t, \mathbf{D}_t), \tag{40}$$

for some functions $D(\mathbf{v}, \mathbf{X}, \mathbf{D})$ and $Y(\mathbf{v}, \mathbf{X}, \mathbf{D})$. Then Proposition 1 *still holds, provided that Definition* 1 *is changed to*

$$\mathcal{E}_t \equiv \left(D'_{\mathbf{D}}\right)^t Y'_{\mathbf{D}}, \tag{41}$$

where $D_{\mathbf{D}} \equiv \frac{\partial D}{\partial \mathbf{D}}(\mathbf{v}_{ss}, \mathbf{X}_{ss}, \mathbf{D}_{ss})$ and $Y_{\mathbf{D}} \equiv \frac{\partial Y}{\partial D}(\mathbf{v}_{ss}, \mathbf{X}_{ss}, \mathbf{D}_{ss})$ are the $n_D \times n_D$ Jacobian and $1 \times n_D$ gradient of D and Y with respect to $\mathbf{D}$, respectively.

PROOF: In the proof of Lemma 2, we replace (19) by $dY_t^s = Y_{\mathbf{D}}\, d\mathbf{D}_t^s + Y_{\mathbf{v}} d\mathbf{v}_{t+1}^s + Y_{\mathbf{X}}\, d\mathbf{X}_t^s$. Subtracting dY_t^s and dY_{t-1}^{s-1} and using $d\mathbf{v}_{t+1}^s = d\mathbf{v}_t^{s-1}$ from (16) and $d\mathbf{X}_t^s = d\mathbf{X}_{t-1}^{s-1}$ by construction, we get $\mathcal{F}_{t,s} \cdot dx = Y_{\mathbf{D}}(d\mathbf{D}_t^s - d\mathbf{D}_{t-1}^{s-1})$, which is identical to (20) except with $\mathbf{y}'_{ss}$ replaced by $Y_{\mathbf{D}}$.

Similarly, replacing (21) with $d\mathbf{D}_t^s = D_{\mathbf{D}} \cdot d\mathbf{D}_{t-1}^s + D'_{\mathbf{v}}\, d\mathbf{v}_t^s + D'_{\mathbf{X}}\, d\mathbf{X}_{t-1}^s$, we follow the same steps to show that $d\mathbf{D}_t^s - d\mathbf{D}_{t-1}^{s-1} = (D_{\mathbf{D}})^{t-1} d\mathbf{D}_1^s$, which is identical to (22) except with Λ'_{ss} replaced by $D_{\mathbf{D}}$. The modified Lemma 2 follows, with $\mathbf{y}'_{ss}$, Λ'_{ss} replaced by $Y_{\mathbf{D}}$, $D_{\mathbf{D}}$. Replacing these in the definition of $\mathcal{E}_t$, the proof of Proposition 1 goes through. *Q.E.D.*

Remarkably, the only change needed in Proposition 1, relative to Proposition 1, is to redefine $\mathcal{E}_t$ as $(D'_{\mathbf{D}})^t Y'_{\mathbf{D}}$ rather than $(\Lambda_{ss})^t \mathbf{y}_{ss}$. This redefinition is natural: the Jacobian $D_{\mathbf{D}}$, which gives the first-order effect of yesterday's distribution on today's, is the generalized counterpart of the forward iteration matrix Λ'_{ss}, and the gradient $Y_{\mathbf{D}}$, which gives the first-order effect of today's distribution on the aggregate output, is the generalized counterpart of $\mathbf{y}'_{ss}$.

Given this redefined $\mathcal{E}_t$, which can be calculated recursively via $\mathcal{E}_t = (D'_{\mathbf{D}})\mathcal{E}_{t-1}$ and $\mathcal{E}_0 = Y'_{\mathbf{D}}$, the fake news algorithm is otherwise unchanged. We now discuss some applications.

Entry and Exit. In general, if we modify our original framework to allow for entry and exit, we have an equation (39) of the more specific form

$$\mathbf{D}_{t+1} = \Lambda(\mathbf{v}_{t+1}, \mathbf{X}_t)\mathbf{D}_t + D^{\text{entry}}(\mathbf{v}_{t+1}, \mathbf{X}_t), \tag{42}$$

where Λ is a Markov matrix with rows that may sum to less than 1 (because of exit, which may be endogenous) and D^{entry} accounts for the possibly-endogenous entry of agents. If, additionally, new entrants show up in the aggregate output, then we also have an equation (40) of the form

$$\mathbf{Y}_t = y(\mathbf{v}_{t+1}, \mathbf{X}_t)'\mathbf{D}_t + Y^{\text{entry}}(\mathbf{v}_{t+1}, \mathbf{X}_t), \tag{43}$$

where Y^{entry} accounts for the effect of the new entrants.

Note that from (42) and (43), we have $D_{\mathbf{D}} = \Lambda'_{ss}$ and $Y_{\mathbf{D}} = \mathbf{y}'_{ss}$. Hence the expectation vector (41) is the same as our original definition from Section 3, and Proposition 1 and the fake news algorithm apply in their original form.

Alternative Representations of the Distribution. In our original equations (11)–(12), we assumed that the distribution vector $\mathbf{D}_t$ consisted of probability masses at discrete grid points. Now, in (39)–(40), $\mathbf{D}_t$ can be an arbitrary vector describing the distribution. For instance, suppose that the state is one-dimensional and continuous. Then, if $\mathbf{D}_t$ is a vector of parameters[4] encoding a density $f(\theta; \mathbf{D}_t)$ for $\theta \in (-\infty, \infty)$, we can write a function $D(\mathbf{v}_{t+1}, \mathbf{X}_t, \mathbf{D}_t)$ that specifies how these parameters evolve over time in our problem. We can also define the aggregate output Y as the average of some idiosyncratic outcome $y(\theta; \mathbf{v}_{t+1}, \mathbf{X}_t)$ of interest:

$$Y(\mathbf{v}_{t+1}, \mathbf{X}_t, \mathbf{D}_t) \equiv \int_{-\infty}^{\infty} y(\theta; \mathbf{v}_{t+1}, \mathbf{X}_t) \cdot f(\theta; \mathbf{D}_t)\, d\theta. \tag{44}$$

Assuming that we already have a way to calculate D and Y, all we need to implement the fake news algorithm is $D_{\mathbf{D}}$ and $Y_{\mathbf{D}}$. If $\mathbf{D}$ is not too high-dimensional, then numerical differentiation is usually a simple strategy to calculate these, although automatic differentiation or (in special cases) analytical differentiation may also be useful.

Moments of the Distribution. Suppose that we want the Jacobian for some moment that can not be represented as a transformation of power moments as in the previous section. For instance—to take a simple example—suppose that $\mathbf{D}$ is a vector of parameters describing the distribution of assets, and we want the uth quantile of this asset distribution. This is a nonlinear function $Y(\mathbf{D}_t)$, and to apply the fake news algorithm we only need to calculate the gradient $Y_{\mathbf{D}}$, which (as above) can be done using either numerical or automatic differentiation.

If $\mathbf{D}$ is instead a simple discretized distribution, then the uth quantile function is discontinuous, consisting of many steps, and its Jacobian is therefore essentially meaningless (wherever it can be calculated, it is identically zero). We could obtain a more interesting object, however, by converting this function to be piecewise linear, interpolating between the discrete mass points. With many grid points, numerical differentiation might be impractical in this case, but thanks to the simplicity of the linearly interpolated quantile function, one can write the gradient $Y_{\mathbf{D}}$ analytically instead.

[4]For an example of a parametric family of distributions often used with heterogeneous-agent models, see Algan, Allais, Den Haan, and Rendahl (2014). In some cases, another possibility is to represent the distribution with a more flexible set of basis functions, such as Chebyshev polynomials.

Discrete Choice Without Taste Shocks. As discussed in Appendix A.1, first-order methods are misleading for endogenous discrete choices on a state space that has been discretized to a grid, since locally these choices will not respond to shocks unless the grid points happen to be at the discontinuities (in which case there is instead a singularity). The suggestion of Appendix A.1 was to assume i.i.d. taste shocks, so that the probabilities of each discrete choice vary continuously.

If $\mathbf{D}$ is a vector of parameters parameterizing a smooth density, however, then the integral (44) aggregating a discrete choice y will generally vary smoothly in $\mathbf{X}_t$ even if y itself is discontinuous. Similarly, the law of motion (39) should also vary smoothly. At this point, there is no particular computational problem posed by discrete choice, and we can apply Proposition 1 as long as we can calculate $D_{\mathbf{D}}$ and $Y_{\mathbf{D}}$, just like above.[5]

A.3. *Multi-Stage Problems*

In Appendix A.1, we observe that in cases where the "distribution" at time t seems endogenous to events at time t (e.g., unemployment risk), our basic framework (10)–(12) can be applied if we interpret $\mathbf{D}_t$ as being the distribution prior to any time-t events, and v, Λ, y as taking expectations over these events. But as models become more complex, with more within-period structure, implementing this approach manually can become difficult.

We therefore further generalize our framework to account for multiple "stages" $j \in \{0, \dots, J-1\}$ within a given period t. We now assume that the three equations (10)–(12) are replaced by

$$\mathbf{v}_{t,j} = v_j(\mathbf{v}_{t,j+1}, \mathbf{X}_{t,j}), \tag{45}$$

$$\mathbf{D}_{t,j+1} = D_j(\mathbf{v}_{t,j+1}, \mathbf{D}_{t,j}, \mathbf{X}_{t,j}), \tag{46}$$

$$\mathbf{Y}_{t,j} = Y_j(\mathbf{v}_{t,j+1}, \mathbf{D}_{t,j}, \mathbf{X}_{t,j}), \tag{47}$$

where we adopt the convention that $\cdot_{t,J} = \cdot_{t+1,0}$, and assume that the initial distribution $\mathbf{D}_{0,0}$ is given exogenously.

At each stage j, we allow for stage-specific inputs $\mathbf{X}_{t,j}$ and outputs $\mathbf{Y}_{t,j}$.[6] Given a path for $\{\mathbf{X}_{t,j}\}$, one solves (45)–(47) to obtain $\{\mathbf{Y}_{t,j}\}$ in the standard way, except that all iterations go through both t and, within each t, through each stage j. For instance, iterating backward over (45), starting from some steady-state $\mathbf{v}_{T,0} = \mathbf{v}_{ss,0}$ would involve iterating through

$$\mathbf{v}_{T-1,J-1}, \mathbf{v}_{T-1,J-2}, \dots, \mathbf{v}_{T-1,0}, \mathbf{v}_{T-2,J-1}, \mathbf{v}_{T-2,J-2}, \dots,$$

and so on.

[5]One caveat, however, is that a smooth density is not always realistic in models with discrete choice. For instance, if agents always reset to the same ideal state, then there will be a mass point at that state; further, if uncertainty in the model is discrete, then there will be mass points corresponding to each finite sequence of shocks that might be realized after that state. To avoid having the distribution consist entirely of mass points in this way, it is useful to introduce some stochastic variables that are drawn from a continuous distribution (e.g., in household models, i.i.d. lognormal income risk in each period, in addition to whatever other income risk is present). This is also true if we want to avoid mass points in a model with occasionally binding constraints (e.g., in household models, a borrowing constraint).

[6]If some stages have either no outputs or no inputs, we can simply disregard the relevant terms. If multiple stages include the same input, then we can use the algorithm to calculate Jacobians with respect to the input at each stage individually, and then sum the Jacobians.

Write $D^j_{\mathbf{D}} \equiv \frac{\partial D_j}{\partial \mathbf{D}}$ and $Y^k_{\mathbf{D}} \equiv \frac{\partial Y_k}{\partial D}$, and use these to define $\mathcal{E}^k_{t,j}$ recursively, iterating backward over (t, j) starting with the initial condition $\mathcal{E}^k_{0,k} = (Y^k_{\mathbf{D}})'$ and then writing

$$\mathcal{E}^k_{t,j} \equiv \left(D^j_{\mathbf{D}}\right)' \mathcal{E}^k_{t,j+1} \tag{48}$$

for all $t > 0$ or $t = 0$ and $j < k$. For any s, the vector $\mathcal{E}^k_{t,j}$ gives the first-order impact of the distribution $\mathbf{D}_{s,j}$ at time s, stage j on the output $\mathbf{Y}_{s+t,k}$ at the time $s + t$, stage k.

Next, assuming a shock dx to $\mathbf{X}^k_s$, we define

$$\mathcal{F}^{j,k}_{t,s} \cdot dx \equiv \begin{cases} dY^{s,k}_{0,j}, & t = 0, \\ \mathcal{E}^j_{t-1,0}\, d\mathbf{D}^{s,k}_{1,0}, & t \geq 1, \end{cases} \tag{49}$$

and have the following further refinement of Proposition 1.

PROPOSITION 2: *Assume $\mathbf{D}_{0,0} = \mathbf{D}_{ss,0}$. The Jacobian $\mathcal{J}$ of h satisfies the recursion $\mathcal{J}^{j,k}_{t,s} = \mathcal{J}^{j,k}_{t-1,s-1} + \mathcal{F}^{j,k}_{t,s}$ for $t, s \geq 1$, with $\mathcal{J}^{j,k}_{t,s} = \mathcal{F}^{j,k}_{t,s}$ for $t = 0$ or $s = 0$, and is therefore given by*

$$\mathcal{J}^{j,k}_{t,s} = \sum_{u=0}^{\min\{s,t\}} \mathcal{F}^{j,k}_{t-u,s-u}, \tag{50}$$

where $\mathcal{F}^{j,k}_{t,s}$ is defined in (49).

Obtaining entries of the Jacobian is therefore no more complicated than in our original case, conditional on being able to evaluate (49) to obtain $\mathcal{F}^{j,k}_{t,s}$.

For each k, we can still obtain $dY^{s,k}_{0,j}$ and $d\mathbf{D}^{s,k}_{1,0}$ for all $s = 0, \dots, T-1$ and j by iterating backward from a shock at date $T - 1$. This is slightly more involved than before, however. One must first obtain $d\mathbf{v}^{s,k}_{0,j}$ for each s and j through backward iteration, then for each s, combine this with (46) and (47), iterating forward through the j's, to obtain $dY^{s,k}_{0,j}$ for all j's and finally $d\mathbf{D}^{s,k}_{1,0}$. (This is in contrast to the original algorithm, where obtaining the dY^s_0 involved no forward iteration at all.)

APPENDIX B: MODEL DESCRIPTIONS AND CALIBRATION

For notational simplicity, we use subscript i to denote household-level outcomes instead of writing them explicitly as functions of state variables as in the main text. Tables B.I–B.III show our calibration of the three models.

B.1. *Krusell–Smith*

We describe the model in Section 2. We assume that $P(e, e')$ discretizes a log AR(1) process

$$\log e_{it} = \rho \log e_{it-1} + \sigma \epsilon_{it}$$

with normal innovations $\epsilon_{it} \sim \mathcal{N}(0, 1)$ and use the Rouwenhorst method for discretization. In the high-dimensional ("HD") version, we set $n_e = 50$ and $n_k = 5000$.

TABLE B.I

CALIBRATION OF OUR KRUSELL–SMITH ECONOMY

Parameter		Value
r	Real interest rate	0.01
σ	Risk aversion	1
α	Capital share	0.11
δ	Depreciation rate	0.025
ρ	Skill mean reversion	0.966
$\sigma/\sqrt{1-\rho^2}$	Cross-sectional std of log earnings	0.5
n_e	Points in Markov chain for e	7
n_k	Points on asset grid	500

B.2. *One-Asset HANK*

Textbook NK model with HA household sector similar to McKay, Nakamura, and Steinsson (2016).

Households. Relative to the KS model, households also choose their hours worked n_{it}. They pay taxes and receive dividends from the ownership of firms according to incidence rules $\overline{\tau}(e)$ and $\overline{d}(e)$. The Bellman equation is

$$V_t(e_{it}, a_{it-1}) = \max_{c_{it}, n_{it}, a_{it}} \left\{ \frac{c_{it}^{1-\sigma}}{1-\sigma} - \varphi \frac{n_{it}^{1+\nu}}{1+\nu} + \beta \mathbb{E}_t V_{t+1}(e_{it+1}, a_{it}) \right\},$$

$$c_{it} + a_{it} = (1+r_t)a_{it-1} + w_t e_{it} n_{it} - \tau_t \overline{\tau}(e_{it}) + d_t \bar{d}(e_{it}),$$

$$a_{it} \geq \underline{a}.$$

Firms. A competitive final goods firm aggregates a continuum of intermediate goods, indexed by j, with a constant elasticity of substitution $\mu/(\mu-1) > 1$. Intermediate goods are produced by monopolistically competitive firms with production function $y_{jt} = F(n_{jt}) \equiv Z_t n_{jt}$. To maintain symmetry, we assume that every firm employs a representative workforce. Each firm sets the price of its product p_{jt} subject to quadratic adjustment costs $\psi_t(p_{jt}, p_{jt-1}) = \frac{\mu}{\mu-1}\frac{1}{2\kappa}[\log(p_{jt}/p_{jt-1})]^2 Y_t$. In the symmetric equilibrium, aggregate inflation $1+\pi_t \equiv P_t/P_{t-1}$ evolves according to the Phillips curve

$$\log(1+\pi_t) = \kappa\left(\frac{w_t}{F'(N_t)} - \frac{1}{\mu}\right) + \frac{1}{1+r_{t+1}}\frac{Y_{t+1}}{Y_t}\log(1+\pi_{t+1}) \tag{51}$$

and dividends equal output net of labor and price adjustment costs $d_t = Y_t - w_t N_t - \psi_t$.

Policy. The fiscal authority spends G_t, issues one-period nominal bonds B, and adjusts the level of taxes τ_t to balance its budget period by period $\tau_t = r_t B + G_t$. Monetary policy sets the nominal rate on bonds according to a standard Taylor rule $i_t = r_t^* + \phi\pi_t + \phi_y(Y_t - Y_{ss})$. The Fisher equation is $r_t = (1+i_{t-1})/(1+\pi_t)$.

Market Clearing. The final good is used for private consumption, public consumption, and price adjustment costs $Y_t = \int c_{it}\,di + G_t + \psi_t$. Aggregate household savings equals government bonds $B = \int a_{it}\,di$. Labor demand equals supply in efficiency units $N_t = \int e_{it} n_{it}\,di$.

TABLE B.II

CALIBRATION OF OUR ONE-ASSET HANK ECONOMY

Parameter		Value	Target
Households			
β	Discount factor	0.982	$r = 0.005$
φ	Disutility of labor	0.786	$N = 1$
σ	Inverse IES	2	
ν	Inverse Frisch	2	
$\underline{b}$	Borrowing constraint	0	
ρ_e	Autocorrelation of earnings	0.966	
σ_e	Cross-sectional std of log earnings	0.5	
Firms			
μ	Steady-state markup	1.2	
κ	Slope of Phillips curve	0.1	
Policy			
B	Bond supply	5.6	
G	Government spending	0	
ϕ	Taylor rule coefficient on inflation	1.5	
ϕ_y	Taylor rule coefficient on output	0	
Discretization			
n_e	Points in Markov chain for e	7	
n_a	Points on asset grid	500	

B.3. *Two-Asset HANK*

Richer NK model with wage as well as price stickiness, and capital with adjustment costs. Households have access to a liquid and an illiquid account as in Kaplan, Moll, and Violante (2018).

Households. Relative to the one-asset model, households allocate their savings between liquid assets b_{it} and illiquid assets a_{it} subject to a convex portfolio adjustment cost $\Phi_t(a_{it}, a_{it-1})$. Hours worked N_t are the same for all households and are pinned down by labor demand from firms as explained below. The Bellman equation is

$$\begin{aligned} V_t(e_{it}, b_{it-1}, a_{it-1}) = \max_{c_{it}, b_{it}, a_{it}} & \left\{ \frac{c_{it}^{1-\sigma}}{1-\sigma} - \varphi \frac{N_t^{1+\nu}}{1+\nu} + \beta \mathbb{E}_t V_{t+1}(e_{it+1}, b_{it}, a_{it}) \right\} \\ \text{s.t.} \quad & c_{it} + a_{it} + b_{it} = (1-\tau_t) w_t N_t e_{it} + \left(1 + r_t^a\right) a_{it-1} \\ & \quad + \left(1 + r_t^b\right) b_{it-1} - \Phi_t(a_{it}, a_{it-1}), \\ & a_{it} \geq 0, \qquad b_{it} \geq \underline{b}. \end{aligned}$$

We specify the adjustment cost function, with $\chi_0, \chi_1 > 0$ and $\chi_2 > 1$, as

$$\Phi_t(a_{it}, a_{it-1}) = \frac{\chi_1}{\chi_2} \left| \frac{a_{it} - \left(1 + r_t^a\right) a_{it-1}}{\left(1 + r_t^a\right) a_{it-1} + \chi_0} \right|^{\chi_2} \left[\left(1 + r_t^a\right) a_{it-1} + \chi_0\right].$$

Financial Intermediary. A representative financial intermediary takes liquid and illiquid deposits from households and invests them in government bonds B_t^g and firm equity

p_t. It performs liquidity transformation at proportional cost $\omega \int b_{it}\,di$. No arbitrage requires that the economy-wide ex ante return $\mathbb{E}_t[1+r_{t+1}]$ equals the expected returns on nominal government bonds and on equity. The competitive financial intermediary passes these returns on to households, subject to intermediation costs:

$$\mathbb{E}_t[1+r_{t+1}] = \frac{1+i_t}{\mathbb{E}_t[1+\pi_{t+1}]} = \frac{\mathbb{E}_t[d_{t+1}+p_{t+1}]}{p_t} = \mathbb{E}_t\big[1+r^a_{t+1}\big] = \mathbb{E}_t\big[1+r^b_{t+1}\big]+\omega.$$

The ex post returns r_t, r^a_t, r^b_t, however, are subject to surprise inflation and capital gains. Assuming that capital gains accrue to the illiquid account, we have $1+r_t = (1+i_{t-1})/(1+\pi_t) = 1+r^b_t+\omega$ and

$$1+r^a_t = \Theta_p\left(\frac{d_t+p_t}{p_{t-1}}\right) + (1-\Theta_p)(1+r_t),$$

where Θ_p denotes the share of equity in the illiquid portfolio.

Firms. Relative to the one-asset HANK model, intermediate goods firms have a Cobb–Douglas production function $y_{jt} = F(k_{jt-1}, n_{jt}) \equiv k^{\alpha}_{jt-1} n^{1-\alpha}_{jt}$. Firms choose their own capital stock subject to quadratic adjustment costs $\zeta(\frac{k_{jt}}{k_{jt-1}})k_{jt-1}$ with $\zeta(x) \equiv x-(1-\delta)+\frac{1}{2\delta\epsilon_I}(x-1)^2$, where $\delta > 0$ is depreciation and $\epsilon_I > 0$. The Phillips curve is analogous to (51), with marginal cost $mc_t = w_t/F_N(K_{t-1}, N_t)$. Let $I_t = K_t-(1-\delta)K_{t-1}+\zeta(\frac{K_t}{K_{t-1}})K_{t-1}$ denote aggregate investment. Dividends equal output net of investment, labor costs, and price adjustment costs $d_t = Y_t - w_tN_t - I_t - \psi_t$. Finally, Tobin's Q and capital evolve according to

$$Q_t = 1+\frac{1}{\delta\epsilon_I}\frac{K_t-K_{t-1}}{K_{t-1}}, \tag{52}$$

$$\begin{aligned}(1+r_{t+1})Q_t = {} & \alpha\frac{Y_{t+1}}{K_t}mc_{t+1} - \left[\frac{K_{t+1}}{K_t}-(1-\delta)+\frac{1}{2\delta\epsilon_I}\left(\frac{K_{t+1}-K_t}{K_t}\right)^2\right] \\ & + \frac{K_{t+1}}{K_t}Q_{t+1}.\end{aligned} \tag{53}$$

Unions. A competitive labor packer aggregates a continuum of labor services, indexed by k, with a constant elasticity of substitution $\mu_w/(\mu_w-1) > 1$. The wage for each labor type is set by a different labor union. To ensure symmetry, we assume that every household supplies every labor type, thus all unions represent all households. Unions set wages to maximize the average utility of households, taking as given their consumption-savings decisions. Setting a nominal wage W_{kt} incurs quadratic adjustment cost $\psi^w_t(W_{kt}, W_{kt-1}) = \frac{\mu_w}{\mu_w-1}\frac{1}{2\kappa_w}[\log(W_{kt}/W_{kt-1})]^2$ in utils. In the symmetric equilibrium, aggregate wage inflation $1+\pi^w_t = (1+\pi_t)w_t/w_{t-1}$ evolves according to the Phillips curve

$$\log\big(1+\pi^w_t\big) = \kappa_w\left(\varphi N^{1+\nu}_t - \frac{(1-\tau_t)w_tN_t}{\mu_w}\int e_{it}c^{-\sigma}_{it}\,di\right) + \beta\log\big(1+\pi^w_{t+1}\big).$$

TABLE B.III

CALIBRATION OF OUR TWO-ASSET HANK ECONOMY

Parameter		Value	Target
Households			
β	Discount factor	0.976	$r = 0.0125$
σ	Inverse IES	2	
χ_0	Portfolio adj. cost pivot	0.25	
χ_1	Portfolio adj. cost scale	6.416	$B = 1.04Y$
χ_2	Portfolio adj. cost curvature	2	
$\underline{b}$	Borrowing constraint	0	
ρ_e	Autocorrelation of earnings	0.966	
σ_e	Cross-sectional std of log earnings	0.92	
Labor unions			
φ	Disutility of labor	2.073	$N = 1$
ν	Inverse Frisch elasticity	1	
μ_w	Steady state wage markup	1.1	
κ_w	Slope of wage Phillips curve	0.1	
Firms			
Z	TFP	0.468	$Y = 1$
α	Capital share	0.33	$K = 10Y$
μ_p	Steady-state markup	1.015	$p + B^g = 14Y$
δ	Depreciation	0.02	
κ_p	Slope of price Phillips curve	0.1	
Financial intermediary			
ω	Liquidity premium	0.005	
Policy			
τ	Labor tax	0.356	budget balance
G	Government spending	0.2	
B^g	Bond supply	2.8	
ϕ	Taylor rule coefficient	1.5	
ϕ_y	Taylor rule coefficient on output	0	
Discretization			
n_e	Points in Markov chain for e	3	
n_b	Points on liquid asset grid	50	
n_a	Points on illiquid asset grid	70	

Policy. Monetary and fiscal policies are the same as in the one-asset HANK model, with the slight modification that τ_t denotes a proportional tax on labor, and thus budget balance requires that $\tau_t w_t N_t = r_t B^g + G_t$.

Market Clearing. The final good is used for private consumption, public consumption, investment, price adjustment costs, liquidity transformation, and portfolio adjustment cost $Y_t = \int c_{it}\,di + G_t + I_t + \psi_t + \omega \int b_{it-1}\,di + \int \Phi_t(a_{it}, a_{it-1})\,di$. Total saving by households equals the value of firm equity and government bonds $p_t + B^g = \int a_{it} + b_{it}\,di$.

APPENDIX C: COMPUTATIONAL DETAILS

C.1. *Numerical and Automatic Differentiation Details*

A key implementation question is how to obtain the two objects dY_0^s and $d\mathbf{D}_1^s$. As discussed in the main text, given some dx, one starts a backward iteration from $T-1$, obtaining $\mathbf{y}_0^s = \mathbf{y}_{T-1-s}^{T-1}$ and $\Lambda_0^s = \Lambda_{T-1-s}^{T-1}$ for all $s = 0, \dots, T-1$, and then $dY_0^s = (d\mathbf{y}_0^s)'\mathbf{D}_{ss}$ and $d\mathbf{D}_1^s = (d\Lambda_0^s)'\mathbf{D}_{ss}$. There are two important practical complications:

1. We only get the correct derivative when dx is infinitesimal.
2. In typical applications, we have not solved for the steady state solving (10) exactly (i.e., such that $\mathbf{v}_{ss} = v(\mathbf{v}_{ss}, X_{ss})$ exactly), but instead for a steady state such that (10) holds up to some numerical tolerance (i.e., such that $\|\mathbf{v}_{ss} - v(\mathbf{v}_{ss}, X_{ss})\| < 10^{-9}$). As a result, iterating backward will generally give $d\mathbf{y}_0^s \neq 0$ and $d\Lambda_0^s \neq 0$, even if $dx = 0$.

The first issue is about how to do differentiation, and is common to all perturbation methods. The second issue is more specific to our approach. We now describe three ways to perform differentiation—addressing the first issue—and, within each, discuss how to deal with the second issue.

One-Sided Numerical Differentiation. Here, we simply choose some small but non-zero dx and then iterate backward as described above. As is standard, dx should be chosen to trade off error from second-order effects (which grow with dx) and from numerical issues like rounding error (which shrink with dx). In our application, one potential source of the latter error is, as discussed above, that the steady state is not exact. This will often be worse than the typical rounding error from floating-point numbers, since we usually pick a numerical tolerance for value function convergence (e.g., 10^{-9}) that is larger than machine precision ($\sim 10^{-16}$).

There are two ways to address this issue:

(a) Do an additional full backward iteration from $T-1$ to 0 (a "ghost run") starting from $dx = 0$, and denote the results as $\tilde{\mathbf{y}}_0^s$ and $\tilde{\Lambda}_0^s$. Set $d\mathbf{y}_0^s = \mathbf{y}_0^s - \tilde{\mathbf{y}}_0^s$ and $d\Lambda_0^s = \Lambda_0^s - \tilde{\Lambda}_0^s$.

(b) At each step, subtract off $v(\mathbf{v}_{ss}, X_{ss})$ and recenter around the steady state. Starting with the shock dx, calculate $d\mathbf{v}_0^0 = v(\mathbf{v}_{ss}, X_{ss} + dx) - v(\mathbf{v}_{ss}, X_{ss})$. Then, for each s, calculate

$$d\mathbf{v}_0^s = v\big(\mathbf{v}_{ss} + d\mathbf{v}_0^{s-1}, X_{ss}\big) - v(\mathbf{v}_{ss}, X_{ss}). \tag{54}$$

Do the same for the functions Λ and y as well (e.g., at each step $s \geq 1$, calculate $d\mathbf{y}_0^s = y(\mathbf{v}_{ss} + d\mathbf{v}_0^{s-1}, X_{ss}) - y(\mathbf{v}_{ss}, X_{ss})$).

We can think of approach (a) as follows: if $\mathbf{y}_0^s$ and Λ_0^s are functions of the shock dx, we are using one-sided numerical differentiation around $dx = 0$ to calculate $d\mathbf{y}_0^s/dx$ and $d\Lambda_0^s/dx$ for each s. Approach (b) is related, but instead effectively uses one-sided numerical differentiation at *each* step of the backward iteration.[7]

Approach (b) is usually more efficient than (a), since it does not require a full backward iteration from $T-1$ to 0 with $dx = 0$, and instead only requires $v(\mathbf{v}_{ss}, X_{ss})$, $y(\mathbf{v}_{ss}, X_{ss})$, and $\Lambda(\mathbf{v}_{ss}, X_{ss})$, which take a single step to compute. Approach (b) is also more accurate, since it corrects for error in the steady state at each step and does not allow these errors to compound. We therefore use (b) as our default for one-sided calculations in this paper (with $dx = 10^{-4}$).

One advantage of (a) is that it may be easier to implement with minimal changes to existing code, since it involves two complete backward iterations from $T-1$ to 0, and does not require changing the steps themselves as in (54).

[7]To make this interpretation clearer, we could divide the right-hand side of (54) by dx to get $d\mathbf{v}_0^s/dx$, and then use $dx \cdot (d\mathbf{v}_0^{s-1}/dx)$ as an input. Practically, however, this involves unnecessary offsetting divisions and multiplications by dx.

Two-Sided Numerical Differentiation. Here, we have the following two analogues of approaches (a) and (b) above.

(a) Iterate backward from $T-1$ to 0 for shocks at $T-1$ of dx and $-dx$, denoting the results by $(\mathbf{y}_0^{s+}, \Lambda_0^{s+})$ and $(\mathbf{y}_0^{s-}, \Lambda_0^{s-})$, respectively, and set $d\mathbf{y}_0^s = (\mathbf{y}_0^{s+} - \mathbf{y}_0^{s-})/2$ and $\Lambda_0^s = (\Lambda_0^{s+} - \Lambda_0^{s-})/2$.

(b) Iterate backward from $T-1$ to 0, recentering around the steady state in each step. Specifically, starting with the shock dx, calculate $d\mathbf{v}_0^0 = (v(\mathbf{v}_{ss}, X_{ss} + dx) - v(\mathbf{v}_{ss}, X_{ss} - dx))/2$. Then, for s, calculate

$$d\mathbf{v}_0^s = \frac{v(\mathbf{v}_{ss} + d\mathbf{v}_0^{s-1}, X_{ss}) - v(\mathbf{v}_{ss} - d\mathbf{v}_0^{s-1}, X_{ss})}{2}. \tag{55}$$

Do the same for the functions Λ and y as well (e.g., at each step $s \geq 1$, calculate $d\mathbf{y}_0^s = (y(\mathbf{v}_{ss} + d\mathbf{v}_0^{s-1}, X_{ss}) - y(\mathbf{v}_{ss} - d\mathbf{v}_0^{s-1}, X_{ss}))/2$).

Analogously to above, approach (a) effectively does two-sided numerical differentiation on the entire backward iteration process, while approach (b) does two-sided numerical differentiation at each step of the process. Note that it is no longer necessary to calculate any responses with $dx = 0$.

Both approaches (a) and (b) have similar efficiency and accuracy. Approach (b) is in principle more accurate for the same reason as above, since it immediately recenters rather than allowing errors in the steady state to build up, but in practice this accuracy advantage seems minor. For consistency with the above, we use approach (b) as our default for two-sided calculations (also with $dx = 10^{-4}$).

Automatic Differentiation. Automatic differentiation allows us to calculate the two objects in (26), dY_0^s and $d\mathbf{D}_1^s$, for infinitesimal dx, getting exact derivatives dY_0^s/dx and $d\mathbf{D}_1^s/dx$. One option is to take whatever code iterates backward from $T-1$ to 0, starting with some shock dx, and simply feed it into an automatic differentiation package, telling it to differentiate with respect to dx.

Though this approach works, it also suffers from error in the steady state: since $\mathbf{v}_{ss}$ is not exactly the same as $v(\mathbf{v}_{ss}, X_{ss})$, the package will be differentiating around a slightly different "steady state" at each step, which may be inefficient. It is therefore beneficial to apply automatic differentiation to a backward iteration routine that recenters around the steady state at each step, as in (54), so that differentiation will be done around the same steady state at each step.

In our implementation, we go slightly further, pre-calculating all derivatives $\partial v/\partial \mathbf{v}$, $\partial v/\partial X$, $\partial y/\partial \mathbf{v}$, and so on around the steady state $(\mathbf{v}_{ss}, X_{ss})$, and then using these derivatives to iterate backward starting with infinitesimal dx. Specifically, we first use the Python automatic differentiation package "jax" to calculate all derivatives.[8] Then, we do backward iterations, starting with $d\mathbf{v}_0^0/dx = \partial v/\partial X$, and then iterating backward $d\mathbf{v}_0^s/dx = (\partial v/\partial \mathbf{v}) \cdot (d\mathbf{v}_0^0/dx)$. We similarly calculate each $d\mathbf{y}_0^s/dx$ and $d\Lambda_0^s/dx$.

[8]This required extensive modifications to our code to make it compatible with jax. For instance, jax requires a more functional style—it does not allow operations that overwrite existing arrays—and it cannot immediately differentiate our routines written to be compiled by Numba (a Python just-in-time compiler). Further, one major source of inefficiency is that Jacobians like $\partial v/\partial \mathbf{v}$ tend to be highly sparse, but jax (like most automatic differentiation packages) cannot internally use sparse array operations. Although we convert the derivatives provided by jax into SciPy's sparse matrix representation before doing additional computations with them, jax's internal computations are still slow in high-dimensional cases because of this limitation. Implementation difficulties of this kind are why we chose numerical differentiation to be our primary approach. (See Ahn, Kaplan, Moll, Winberry, and Wolf (2018) for an example of a paper employing a custom-built automatic differentiation toolkit that makes use of sparsity internally.)

Applying to Direct Method. In Appendix D.1, we apply one-sided, two-sided, and automatic differentiation to the direct method of computing columns s of the Jacobian. For one-sided differentiation, we apply the direct method exactly as described at the beginning of Section 3.2, calculating the impulse response to a small shock $dx = 10^{-4}$ at date s. However, in the spirit of (a) above, we subtract off the results from a "ghost run" with a shock $dx = 0$ to eliminate inaccuracy from an imperfect steady state. For two-sided differentiation, we calculate the impulse responses to small shocks dx and $-dx$ at date s and then take half the difference between the two. Finally, for automatic differentiation, we use automatic differentiation to pre-calculate all derivatives as above, and use them to evaluate the linearized equations (10)–(12) and obtain a linear impulse response to a shock at each date s.

C.2. *Equilibrium Computation as a Directed Acyclic Graph*

In this section, we introduce a formal way of representing the variable substitutions that underlie the **H** and **M** functions. The idea is to organize the model as a set of blocks arranged along a directed acyclic graph, or DAG. While the technical definition requires some formalism, constructing models this way is highly intuitive and facilitates efficient computation.

The general type of model whose Jacobians we can compute consists of any combination of heterogeneous-agent problems (or *heterogeneous-agent blocks*), characterized by the mapping (13), and *simple blocks*, which capture typical aggregate relationships in dynamic macro models. Formally, we define simple blocks as mappings between inputs $\mathbf{X}$ and outputs $\mathbf{Y}$ for which there exist $k, l \in \mathbb{N}$ and a time-invariant function h such that $\mathbf{Y}_t$ is only a function of neighboring $\mathbf{X}_t$'s, that is,

$$\mathbf{Y}_t = h(\mathbf{X}_{t-k}, \dots, \mathbf{X}_{t+l}).$$

For instance, a neoclassical firm sector can be represented as a simple block mapping $\mathbf{X}_t = (K_t, Z_t)$ to $\mathbf{Y}_t = (Y_t, r_t, w_t)$. Combining such a sector with a heterogeneous-agent block mapping $\mathbf{X}_t = (r_t, w_t)$ to $\mathbf{Y}_t = \mathcal{K}_t(\{r_s, w_s\})$, as well as a simple block mapping $\mathbf{X}_t = (\mathcal{K}_t, K_t)$ to market clearing $\mathbf{Y}_t = \mathcal{K}_t - K_t$, we obtain the Krusell–Smith model of Section 2. Jacobians of simple blocks are straightforward to compute explicitly.

We call "sequence-space model" any combination of these blocks that maps *shocks* (like Z_t) and *unknowns* (like K_t) to *targets* (like asset market clearing) along a directed acyclic graph.

DEFINITION 1: A *sequence-space model* is defined by:

1. A set of sequence indices $\mathcal{N} = \mathcal{Z} \cup \mathcal{U} \cup \mathcal{O}$, where $\mathcal{Z}$ are *exogenous shocks*, $\mathcal{U}$ are *unknowns*, $\mathcal{O}$ are *outputs*, and $\mathcal{H} \subset \mathcal{O}$ are *targets*,
2. A set of *blocks*, each either simple or heterogeneous-agent blocks, indexed by $\mathcal{B}$, where each block $b \in \mathcal{B}$ has *inputs* $\mathcal{I}_b \subset \mathcal{N}$ and *outputs* $\mathcal{O}_b \subset \mathcal{O}$, such that each output $o \in \mathcal{O}$ belongs to exactly one block, and for each output $o \in \mathcal{O}_b$, block b provides a function $h^o(\{\mathbf{X}^i\}_{i \in \mathcal{I}_b})$ mapping the block's input sequences to this output sequence,

such that (a) the number of unknowns and targets are equal, that is, $n_u = n_h$, and (b) the *directed graph* of blocks, formed by drawing an edge from b to b' whenever some output $o \in \mathcal{O}_b$ is used as an input $o \in \mathcal{I}_{b'}$, is *acyclic*.

DEFINITION 2: An *equilibrium* of a sequence-space model, given sequences $\{\mathbf{X}^i\}_{i \in \mathcal{Z}}$ for the exogenous shocks, is a set of sequences $\{\mathbf{X}^i\}_{i \in \mathcal{U} \cup \mathcal{O}}$ such that: (a) $\mathbf{X}^o = h^o(\{\mathbf{X}^i\}_{i \in \mathcal{I}_b})$ for any output $o \in \mathcal{O}$, and (b) $\mathbf{X}^o = 0$ for any target $o \in \mathcal{H}$.

DEFINITION 3: A *steady-state equilibrium* is an equilibrium in which all sequences are constant over time, $\mathbf{X}^i_t = \mathbf{X}^i_{ss}$ for all $i \in \mathcal{N}$.

A sequence-space model thus consists of a combination of blocks that are linked along a directed graph. Each individual block covers a different aspect of the economy and computes either equilibrium conditions themselves (i.e., outputs that are also "targets"), or variables that are useful for other blocks (i.e., outputs that are also inputs). Any such variable can be viewed as having been "substituted out" and need not be carried around as an unknown.

An important property we require is that the directed graph that connects blocks be *acyclic*, that is, it does not feature circular dependencies across blocks. Instead, there is always an ordering $b^1, \dots, b^{n_b}$ of the blocks (formally, a "topological sort") such that all input variables used in later blocks, for example, b^3 or b^4, are either output variables from earlier blocks, for example, b^1 or b^2 (in which case those variables were substituted out) or they are shocks or unknowns. Recursively, this implies that, starting with shocks and unknowns $\{\mathbf{X}^i, i \in \mathcal{Z} \cup \mathcal{U}\}$, we can follow along this ordering, computing each block's output, one-by-one, starting with block b^1 (which only uses inputs that are either shocks or unknowns), then moving to block b^2 (whose inputs can also be outputs of b^1), and so on. When we are done, we will have calculated all outputs $\{\mathbf{X}^o\}_{o\in\mathcal{O}}$.

Thus, the recursive mapping from shocks and unknowns $\{\mathbf{X}^i, i \in \mathcal{Z} \cup \mathcal{U}\}$ to targets $\{\mathbf{X}^o\}_{o\in\mathcal{H}}$ represents $\mathbf{H}(\mathbf{U}, \mathbf{Z}) = 0$, and the mapping to other outputs $\{\mathbf{X}^o\}_{o\in\mathcal{O}\setminus\mathcal{H}}$ represents $\mathbf{M}(\mathbf{U}, \mathbf{Z}) = \mathbf{X}$.

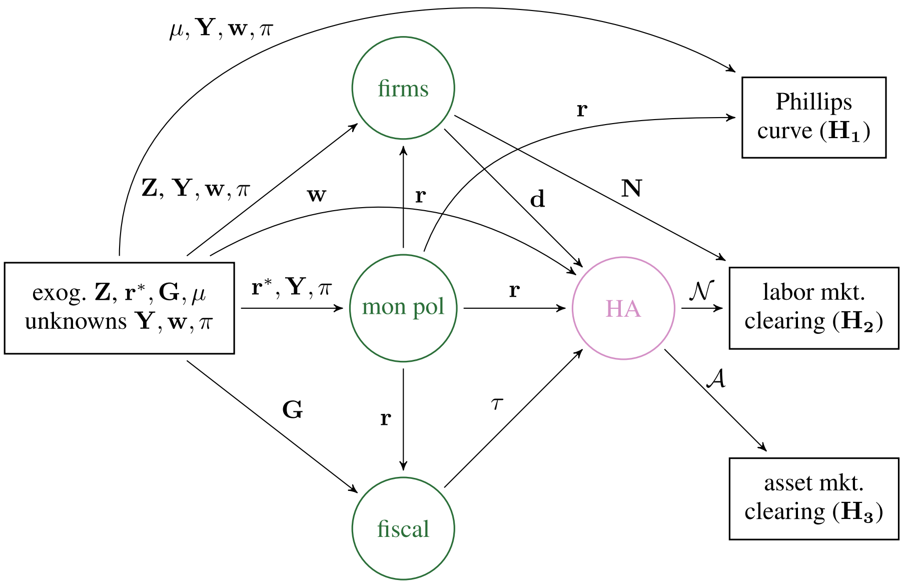
Note that acyclicality does not place any restrictions on the economic model itself, only on its representation as a directed graph. If we start with a cyclic graph, we can always break the cycle by adding additional unknowns and targets; for instance, if block b^1 is required for b^2, b^2 for b^3, and b^3 for b^1, we can add the required outputs of b^3 as unknowns that are direct inputs to b^1, and then add a target enforcing consistency between these unknowns and the actual outputs of b^3.[9]

Example: Krusell–Smith Model. Figure 3 visualizes the DAG for the Krusell–Smith model that corresponds to the variable substitutions we made in Section 2. It has three blocks (neoclassical firms, heterogeneous households "HA," and a block to compute the asset market-clearing condition H), one exogenous shock (productivity $\mathcal{Z} = \{Z\}$), one unknown (capital $\mathcal{U} = \{K\}$), four outputs (capital return, wage, household savings, asset market clearing, so $\mathcal{O} = \{r, w, \mathcal{K}, H\}$), and one target (asset market clearing $\mathcal{H} = \{H\}$).[10] The DAG is best read from left to right. The "firms" block maps the unknown sequence of capital stocks K and the exogenous shocks Z into the interest rate and wage sequences r, w. Those are then used to substitute out r, w in the "HA" block, before asset market clearing H is computed.

One and Two-Asset HANK Models. The Krusell–Smith model allows for a relatively straightforward DAG that reduces the number of unknowns to $n_u = 1$. Figure C.1 gives a DAG for a more complex case: a one-asset HANK model from Appendix B.2. This model combines standard NK elements—sticky prices, flexible wages, and a Taylor rule

[9]An alternative approach to resolve cycles is to use a "solved block," discussed in Appendix C.5.

[10]Not visualized are firm production Y or household consumption $\mathcal{C}$, which could be additional outputs of the firm and HA blocks, respectively, but are not strictly necessary since we are using asset rather than goods market clearing to define equilibrium.



$\mu, \mathbf{Y}, \mathbf{w}, \pi$
firms
$\mathbf{r}$
Phillips curve ($\mathbf{H_1}$)
$\mathbf{Z}, \mathbf{Y}, \mathbf{w}, \pi$
$\mathbf{w}$
$\mathbf{r}$
$\mathbf{d}$
$\mathbf{N}$
exog. $\mathbf{Z}, \mathbf{r}^*, \mathbf{G}, \mu$ unknowns $\mathbf{Y}, \mathbf{w}, \pi$
$\mathbf{r}^*, \mathbf{Y}, \pi$
mon pol
$\mathbf{r}$
HA
$\mathcal{N}$
labor mkt. clearing ($\mathbf{H_2}$)
$\mathcal{A}$
$\mathbf{G}$
$\mathbf{r}$
τ
asset mkt. clearing ($\mathbf{H_3}$)
fiscal

FIGURE C.1.—DAG representation of one-asset HANK economy.

for monetary policy, but no capital—with a one-asset incomplete market HA household sector where labor supply is endogenous.

As Figure C.1 shows, the DAG for this model features three unknowns (wages w, output Y, and inflation π), which are used to compute six intermediate outputs, ultimately yielding three targets (a Phillips curve condition H_1, labor market clearing H_2, and asset market clearing H_3). In other words, the DAG substitutes out six variables that would otherwise have to be included as unknowns. We introduce four exogenous shocks (productivity Z, Taylor rule intercept r^*, government spending G, and markups μ).

The DAG makes it easy to visualize the dependencies between macroeconomic aggregates that are embedded in the model: for instance, the dividends from firms are distributed to households (according to a certain rule), so the output d of the firm block is an input to the HA block. Similarly, the real interest rate r affects the taxes required for the government to achieve its balanced-budget target, so r is an input to the fiscal block, which has an output τ that is an input to the HA block.

Even as models grow in complexity beyond this one, they often still admit DAGs with small numbers of unknowns and targets. Figure C.2 shows our preferred DAG for the two-asset HANK model from Appendix B.3. This model has $n_x = 21$ endogenous variables, but 18 can be substituted out along the DAG.

C.3. *Jacobians and Impulse Responses*

We now show how to use the DAG representation of the model to automatically evaluate the Jacobians of **H** and **M**. To do so, we systematically apply the chain rule along the model's DAG, implementing a technique known as forward accumulation in the auto-

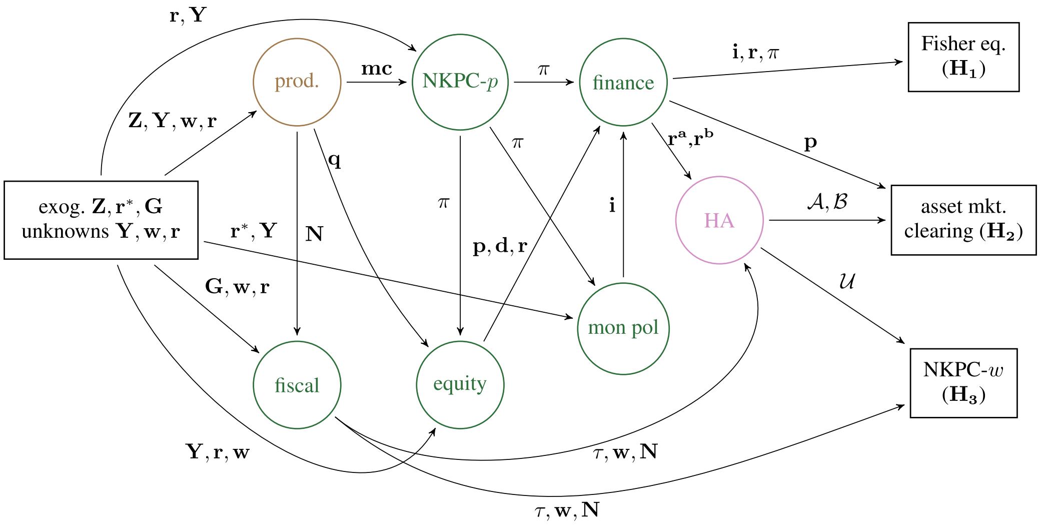


FIGURE C.2.—DAG representation of two-asset HANK economy.

matic differentiation literature (Griewank and Walther (2008)). This technique combines the Jacobians of individual blocks to build up $\mathbf{H_U}$, $\mathbf{H_Z}$ and $\mathbf{M_U}$, $\mathbf{M_Z}$.[11]

Total Jacobians J. To start, we need a new concept. For any exogenous shock or unknown $i \in \mathcal{Z} \cup \mathcal{U}$ and any output o, let the $\mathbf{J}^{o,i}$ denote the *total Jacobian* of o with respect to i when o is evaluated along the DAG. For instance, in the one-asset HANK model in Figure C.1, the total Jacobian $\mathbf{J}^{\mathcal{N},w}$ of household labor supply with respect to wages combines two forces: the direct effect of w on household decisions, and the indirect effect working through the influence of w on firm profits and therefore the dividends d received by households. This is in contrast to $\mathcal{J}^{\mathcal{N},w}$, which is a partial Jacobian that captures only the direct effect.

To obtain $\mathbf{J}^{o,i}$ through forward accumulation, we first initialize $\mathbf{J}^{i,i}$ to the identity for each $i \in \mathcal{Z} \cup \mathcal{U}$. We then go through blocks following the ordering (topological sort) $b^1, \dots, b^{n_b}$ one-by-one beginning with b^1. For each block b, we evaluate the total Jacobian of all its outputs $o \in \mathcal{O}_b$ with respect to shocks and unknowns $i \in \mathcal{Z} \cup \mathcal{U}$:

$$\mathbf{J}^{o,i} = \sum_{m \in \mathcal{I}_b} \mathcal{J}^{o,m} \mathbf{J}^{m,i}. \tag{56}$$

This systematically applies the chain rule: for each input m, (56) takes the product of the partial Jacobian $\mathcal{J}^{o,m}$ with the already-calculated total derivative $\mathbf{J}^{m,i}$ of m with respect to i. (When $m = i$, then the latter is the identity and the term is just the partial Jacobian $\mathcal{J}^{o,i}$.) The benefit of building up the $\mathbf{J}^{o,i}$ progressively via forward accumulation is that the chain rule is applied in an efficient way, without redundant computations.

[11]In actual computations, the methods in this section will be applied on Jacobians that are truncated to some horizon $T \times T$. For simple blocks, we use a simple sparse representation of the Jacobian, described in Appendix E.2, and do not need to truncate.

General Equilibrium Jacobians G. Using the $\mathbf{J}$ matrices, we can compute the total Jacobians of all targets with respect to all unknowns and shocks, $\mathbf{H_U} = \mathbf{J}^{\mathcal{H},\mathcal{U}}$ and $\mathbf{H_Z} = \mathbf{J}^{\mathcal{H},\mathcal{Z}}$. By equation (30), these are the objects needed to solve the equilibrium response of unknowns, $d\mathbf{U} = -\mathbf{H}_\mathbf{U}^{-1}\mathbf{H_Z}d\mathbf{Z}$. We can compute this response for any arbitrary shock vector $d\mathbf{Z}$ by simple multiplication with the matrix $\mathbf{G}^{\mathcal{U},\mathcal{Z}} = -\mathbf{H}_\mathbf{U}^{-1}\mathbf{H_Z}$. We refer to this matrix as *general equilibrium Jacobian* of unknowns to shocks.

To compute $\mathbf{G}^{o,\mathcal{Z}}$ for the remaining outputs $o \in \mathcal{O} \setminus \mathcal{H}$, we trace the same forward accumulation steps as before, and build $\mathbf{G}^{o,\mathcal{Z}}$ recursively, using[12]

$$\mathbf{G}^{o,\mathcal{Z}} = \sum_{m\in\mathcal{I}_b} \mathcal{J}^{o,m}\mathbf{G}^{m,\mathcal{Z}}. \tag{57}$$

Each $\mathbf{G}^{o,\mathcal{Z}}$ has n_zT columns, each of which can be interpreted as the impulse response of o to some news shock. One way to think about this approach, therefore, is that we are simultaneously calculating n_zT general equilibrium impulse responses. For our Krusell–Smith, one-asset HANK, and two-asset HANK models, n_zT is $1 \times 300 = 300$, $3 \times 300 = 900$, and $3 \times 300 = 900$, respectively.[13]

Comparing Table C.I to Table II, we see that computing $\mathbf{G}$'s is, in each of our cases, significantly cheaper than applying the fake news algorithm to obtain $\mathcal{J}$'s for the heterogeneous-agent block; for instance, it takes about 50 milliseconds for the one-asset HANK model, while the fake news algorithm took 320 milliseconds. This shows the power of $\mathcal{J}$'s as sufficient statistics: once we have them, it is just a matter of linear algebra to obtain a full characterization of equilibrium.

How important is the pattern of variable substitution along the DAG to efficiently solving heterogeneous-agent models? Table C.II compares the times needed to compute the $\mathbf{G}$ matrices using our preferred DAG ("efficient DAG") and using no substitution of endogenous variables ("flat DAG"). The latter approach results in a greater number of unknowns and is substantially slower, by a factor that ranges from 2 to 10. The reason is that the dimensionality of the linear system becomes so high that solving it is quite costly.

TABLE C.I
COMPUTING TIMES FOR G

	Krusell–Smith	One-Asset HANK	Two-Asset HANK
Total	3.3 ms	50.6 ms	173.5 ms
step 1 (forward accumulate $\mathbf{H_U}$ and $\mathbf{H_Z}$)	0.6 ms	7.5 ms	27.0 ms
step 2 (compute $\mathbf{G}^{\mathcal{U},\mathcal{Z}} = -\mathbf{H}_\mathbf{U}^{-1}\mathbf{H_Z}$)	1.2 ms	25.9 ms	51.6 ms
step 3 (forward accumulate for all $\mathbf{G}^{o,\mathcal{Z}}$)	1.5 ms	17.2 ms	95.0 ms
No. of unknowns	1	3	3
No. of exogenous shocks	1	3	7

[12]An alternative is to re-use total Jacobians and write $\mathbf{G}^{o,\mathcal{Z}} = \mathbf{J}^{o,\mathcal{U}}\mathbf{G}^{\mathcal{U},\mathcal{Z}} + \mathbf{J}^{o,\mathcal{Z}}$. In our experience, the recursive approach tended to be more efficient.

[13]If one only needs to compute one impulse response, it is possible to obtain this impulse response faster using an alternative method described in Appendix C.4. Interestingly, it is not too much more expensive to calculate the full set of impulse responses in $\mathbf{G}$: in our one-asset HANK example, obtaining 900 rather than one impulse response only takes about 5 times as long. This is possible because we only need to calculate $\mathbf{H_U}$ once, independent of shocks.

TABLE C.II

COMPUTING TIMES FOR **G**, EFFICIENT VERSUS FLAT DAG

	Krusell–Smith	One-Asset HANK	Two-Asset HANK
Total with efficient DAG	4.7 ms	57.8 ms	198.0 ms
Total with flat DAG	33.0 ms	167.5 ms	1452.7 ms
No. of unknowns (efficient DAG)	1	3	3
No. of unknowns (flat DAG)	3	7	18
No. of exogenous shocks	1	3	7

C.4. *Fast Solution for Individual Impulse Responses*

In the case where we are only interested in a single impulse response, we only need to do full forward accumulation (56) for $i \in \mathcal{U}$ to obtain $o \in \mathcal{H}$, which gives $\mathbf{H_U} = \mathbf{J}^{\mathcal{H},\mathcal{U}}$. Then, to deal with shocks, we do forward accumulation on *vectors* rather than matrices, writing

$$\mathbf{J}^{o,\mathcal{Z}}\, d\mathbf{Z} = \sum_{m \in \mathcal{I}_b} \mathcal{J}^{o,m} \mathbf{J}^{m,\mathcal{Z}}\, d\mathbf{Z}. \tag{58}$$

This gives $\mathbf{H_Z}\, d\mathbf{Z} = \mathbf{J}^{\mathcal{H},\mathcal{Z}}\, d\mathbf{Z}$. We then solve the linear system $\mathbf{H_U}\, d\mathbf{U} = -\mathbf{H_Z} d\mathbf{Z}$ to obtain $d\mathbf{U}$. Finally, to obtain equilibrium impulse responses $d\mathbf{X}^o$ for $o \notin \mathcal{Z} \cup \mathcal{U}$, we need to calculate

$$d\mathbf{X}^o = \mathbf{J}^{o,\mathcal{Z}}\, d\mathbf{Z} + \mathbf{J}^{o,\mathcal{U}}\, d\mathbf{U}. \tag{59}$$

The first term, $\mathbf{J}^{o,\mathcal{Z}}\, d\mathbf{Z}$, has already been calculated in (58). For the second term, we do forward accumulation on vectors as in (58), just solving for $\mathbf{J}^{o,\mathcal{U}}\, d\mathbf{U}$ rather than $\mathbf{J}^{o,\mathcal{Z}}\, d\mathbf{Z}$.[14]

Table C.III shows the time each step of this process takes for our three models, starting from the Jacobians $\mathcal{J}$ for each model block. In general, this process is very cheap, with the only costly parts being the steps that involve matrices rather than vectors: forward accumulation in step 1 to get $\mathbf{H_U} = \mathbf{J}^{\mathcal{H},\mathcal{U}}$, and second, solving the linear system $\mathbf{H_U}\, d\mathbf{U} = -\mathbf{H_Z} d\mathbf{Z}$ for $d\mathbf{U}$ in step 3.

TABLE C.III

COMPUTING TIMES FOR IMPULSE RESPONSES

	Krusell–Smith	One-Asset HANK	Two-Asset HANK
Total	0.9 ms	15.5 ms	30.1 ms
step 1 (forward accumulate $\mathbf{H_U}$)	0.4 ms	6.2 ms	19.7 ms
step 2 (forward accumulate $\mathbf{J}^{o,\mathcal{Z}}\, d\mathbf{Z}$)	0.1 ms	0.1 ms	0.4 ms
step 3 (solve linear system for $d\mathbf{U}$)	0.4 ms	9.0 ms	8.9 ms
step 4 (forward accumulate $\mathbf{J}^{o,\mathcal{U}}\, d\mathbf{U}$, get $d\mathbf{X}^o$)	0.1 ms	0.3 ms	1.2 ms
No. of unknowns	1	3	3
No. of exogenous shocks	1	3	7

[14]Another approach is to use the $\mathbf{J}^{o,\mathcal{U}}$ that we already calculated as part of the initial forward accumulation to obtain $\mathbf{H_U} = \mathbf{J}^{\mathcal{H},\mathcal{U}}$, and directly apply these to $d\mathbf{U}$. This approach has similar (and low) cost, but is less useful in general because it does give o that were not necessary in calculating $\mathbf{H_U}$.

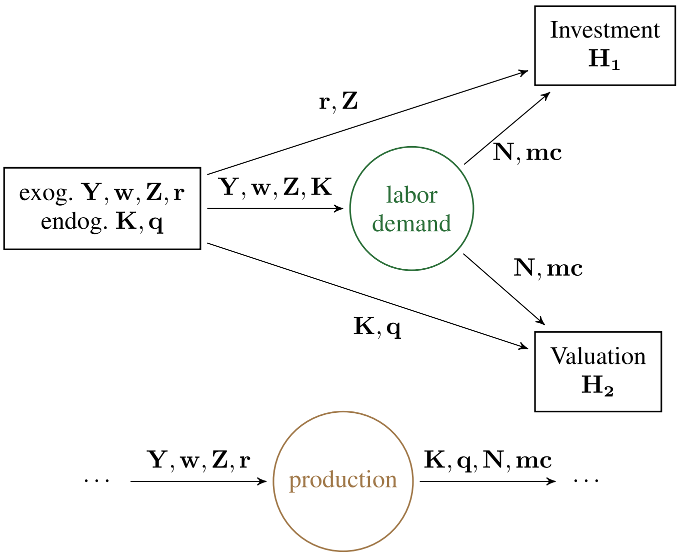

Investment
$\mathbf{H_1}$
$\mathbf{r}, \mathbf{Z}$
$\mathbf{N}, \mathbf{mc}$
exog. $\mathbf{Y}, \mathbf{w}, \mathbf{Z}, \mathbf{r}$
endog. $\mathbf{K}, \mathbf{q}$
$\mathbf{Y}, \mathbf{w}, \mathbf{Z}, \mathbf{K}$
labor demand
$\mathbf{N}, \mathbf{mc}$
$\mathbf{K}, \mathbf{q}$
Valuation
$\mathbf{H_2}$
$\mathbf{Y}, \mathbf{w}, \mathbf{Z}, \mathbf{r}$
production
$\mathbf{K}, \mathbf{q}, \mathbf{N}, \mathbf{mc}$

FIGURE C.3.—The concept of a solved block, applied to the production block of our two-asset HANK model.

Since these steps are costly because they involve the shock-independent matrix $\mathbf{H_U}$, there are clear economies of scale from computing the impulse response to multiple shocks. We can calculate $\mathbf{H_U}$ a single time, and then also calculate $\mathbf{H_U^{-1}}$ (or, better, an LU factorization of $\mathbf{H_U}$) a single time, at which point the marginal cost of computing additional impulse responses is very low. This is the approach we use in Section 5.3 to evaluate the likelihood when redrawing model parameters, since this involves finding impulse responses to each shock simultaneously. Taking this idea to its fullest extent, we can calculate the impulse responses to all shocks simultaneously, which is the "**G** matrix" approach in Section C.3.

C.5. *Solved Blocks*

The DAG of the two-asset model in Figure C.2 includes a brown "production" block. Production with adjustment costs is well known to involve the joint determination of investment and q, and it is natural to solve for these two jointly inside a block. This leads us to introduce a "solved block" concept, as follows.

DEFINITION 4: A *solved block b* has an underlying sequence-space model with shocks $\tilde{\mathcal{Z}}$, unknowns $\tilde{\mathcal{U}}$, outputs $\tilde{\mathcal{O}}$, and targets $\tilde{\mathcal{T}}$, and an equilibrium that is locally unique around the steady state, where we define:

1. The inputs of the solved block to be the shocks of the underlying sequence-space model: $\mathcal{I}_b \equiv \tilde{\mathcal{Z}}$.
2. The outputs of the solved block to be the unknowns and outputs, minus targets, of the underlying sequence-space model: $\mathcal{O}_b \equiv \tilde{\mathcal{U}} \cup (\tilde{\mathcal{O}} \setminus \tilde{\mathcal{T}})$.
3. For each output $o \in \mathcal{O}_b$, the function $h^o(\{\mathbf{x}^i\}_{i \in \mathcal{I}_b})$ is the locally unique equilibrium path of o in the underlying sequence-space model given sequences $\{\mathbf{x}^i\}_{i \in \tilde{\mathcal{Z}}}$ for the exogenous shocks in that model (recalling that $\mathcal{I}_b = \tilde{\mathcal{Z}}$).

Informally, a solved block is a sequence-space model, turned into a block. Figure C.3 illustrates how this concept works in the case of the production block of the two-asset HANK model. Given the exogenous inputs Y, w, Z, r, the solved block solves for the endogenous paths for K and Q that jointly satisfy the q theory equations, so that its outputs are K, Q as well as labor demand N and marginal costs mc.

C.6. *Reiter Method Implementation*

We now briefly describe our implementation of the "Reiter method." The idea is to arrange the equations governing equilibrium into a system of nonlinear equations with at most a single lead and a single lag, at which point we can use standard linear rational expectations methods to obtain the first-order solution.

Krusell–Smith Model. Here, we construct a stacked vector $\mathbf{X}_t$ of length $2n_g + 1$, where n_g is the number of points in our grid. This includes:

- the entire vector $\mathbf{v}_t$ representing the value function at time t (in our implementation, this is the length n_g vector giving the derivative of the value function at each point)
- the distribution $\mathbf{D}_{t+1}$ excluding the last entry, which equals 1 minus the other elements and is therefore redundant (length $n_g - 1$)
- capital K_t (scalar)
- productivity Z_t (scalar)

Note that since, in our model, the distribution $\mathbf{D}_{t+1}$ is determined by information available at time t, we include it in $\mathbf{X}_t$ in line with the usual timing convention for these models.

We now build a function $\mathbf{F}(\mathbf{X}_{t-1}, \mathbf{X}_t, \mathbf{X}_{t+1}, \epsilon_t)$ with a $2n_g + 1$-dimensional output, which includes all equilibrium conditions:

- n_g entries for the equation (10) that determines $\mathbf{v}_t$ given $\mathbf{v}_{t+1}$ and the inputs K_{t-1} and Z_t (which together determine r_t and w_t, entering into the household's problem)
- $n_g - 1$ entries for the equation (11) that determines $\mathbf{D}_{t+1}$ given $\mathbf{v}_{t+1}$, $\mathbf{D}_t$, and the inputs K_{t-1} and Z_t (note that again, we drop the last entry, which is redundant since the distribution sums to 1)
- one entry for the equation (12) that expresses aggregate K_t as the total of individual holdings given by $\mathbf{D}_{t+1}$
- one entry for the assumed AR(1) law of motion $\log(Z_t/Z_{ss}) = \rho \log(Z_{t-1}/Z_{ss}) + \epsilon_t$ for productivity.

Recursive stochastic equilibrium corresponds to the condition $\mathbb{E}_t F(\mathbf{X}_{t-1}, \mathbf{X}_t, \mathbf{X}_{t+1}, \epsilon_t) = 0$. We use the automatic differentiation package jax to linearize this as

$$A \mathbb{E}_t \, d\mathbf{X}_{t+1} + B \, d\mathbf{X}_t + C \, d\mathbf{X}_{t-1} + E \epsilon_t = 0, \tag{60}$$

where A, B, and C are $(2n_g + 1) \times (2n_g + 1)$ matrices and E is a $(2n_g + 1) \times 1$ vector. Equation (60) is a standard form for a linear rational expectations model, and can be solved using a variety of standard techniques. We use Alisdair McKay's Python toolkit, which implements a version of Sims's gensys algorithm to solve (60).[15] This gives us a solution, expressed as the recursive law of motion

$$d\mathbf{X}_t = P \, d\mathbf{X}_{t-1} + Q \epsilon_t, \tag{61}$$

[15]See https://alisdairmckay.com/Notes/HetAgents/index.html.

where P is a $(2n_g + 1) \times (2n_g + 1)$ matrix and Q is a $(2n_g + 1) \times 1$ vector. Note that the first n_g columns of P are all zeros, since $\mathbf{v}_{t-1}$ is not a state and has no direct impact on $\mathbf{X}_t$. We can then unstack (61) to obtain the linear law of motion relating the individual components of $\mathbf{X}_t$ ($\mathbf{v}_t$, $\mathbf{D}_{t+1}$, K_t, and Z_t) to $\mathbf{D}_t$, K_{t-1}, Z_{t-1}, and ϵ_t.

To obtain the impulse response to a unit shock to ϵ_0 (i.e., a unit productivity shock) for comparison to our sequence-space solution, we plug $d\mathbf{X}_{t-1} = 0$ and $\epsilon_0 = 1$ into (61) and iterate forward to get $d\mathbf{X}_0, d\mathbf{X}_1, \dots$.

One-Asset HANK Model. Since our implementation here is mostly the same as above, we will only describe the differences.

The stacked vector $\mathbf{X}_t$ now has length $2n_g + 4$, including n_g entries $\mathbf{v}_t$ giving the derivative of the value function at each grid point, the $n_g - 1$ first entries of $\mathbf{D}_{t+1}$, and then w_t, Y_t, π_t, r_t, and Z_t.

The function $F(\mathbf{X}_{t-1}, \mathbf{X}_t, \mathbf{X}_{t+1}, \epsilon_t)$ now also has output of length $2n_g + 4$, including n_g entries for (10), $n_g - 1$ entries for (11), two entries for (12) corresponding to the aggregation of assets and labor (and clearing in the respective markets), one entry for the Phillips curve, one entry for the AR(1) law of motion $\log(Z_t/Z_{ss}) = \rho \log(Z_{t-1}/Z_{ss}) + \epsilon_t$ for productivity, and one entry for the combined Taylor rule and Fisher equation giving the ex post real rate, $r_t = (1 + r^*_{t-1} + \phi\pi_{t-1})/(1 + \pi_t) - 1$.

Note that in addition to the equations for $\mathbf{v}_t$, $\mathbf{D}_{t+1}$, and Z_t, we have one equation in F for each of the targets in Figure 3 (asset market clearing, labor market clearing, and the Phillips curve), but also an additional equation for r_t. Similarly, we have one entry in $\mathbf{X}_t$ for each of the unknowns in Figure 3 (Y, w, and π), but also r_t as an unknown. We calculate each target in F by starting with the unknowns and progressively calculating the date-t output of each block in Figure 3. Since we need r_{t+1} as an input to the calculation of the Phillips curve,[16] however, we include r as part of $\mathbf{X}$ and add the equation from the "monetary" block explicitly to F.

C.7. *Recovering the State-Space Law of Motion*

From the linearized solution in the sequence space, given a particular shock process expressed in state-space form, it is possible to recover the equivalent state-space law of motion. The general idea is to determine the effect that a perturbation to any state, and any innovation to the shock process, has on all states in the following period. This can be done in three steps: one first finds (a) the effect of the perturbation on the targets $d\mathbf{H}$, then (b) the response of unknowns to targets $d\mathbf{U}$, and finally (c) the response of next-period states to unknowns. Since the distribution of agents is a state, this process requires two pieces of information about the distribution, which are both computed by the fake news algorithm in Section 3.2: the expectation vectors $\mathcal{E}^o_t$ for step (a), and the distribution perturbation vectors $\mathcal{D}^i_1$ for step (c). One can then use the state-space law of motion for any standard application, such as simulation or estimation using state-space methods.

As an example, here we explain how to recover the state-space law of motion in the Krusell–Smith model with AR(1) TFP shocks of persistence ρ. The state then consists of the n_g points of the distribution and the two aggregate states K_{t-1} and Z_{t-1}, so the law of

[16]This appears in the nonlinear F but actually falls out to first order, so is irrelevant to the calculation we will perform.

motion reads

$$\begin{pmatrix} \mathbf{D}_{t+1} \\ K_t \\ Z_t \end{pmatrix} = \mathbf{A} \begin{pmatrix} \mathbf{D}_t \\ K_{t-1} \\ Z_{t-1} \end{pmatrix} + \mathbf{B}\epsilon_t, \tag{62}$$

with $\mathbf{A}$ an $(n_g + 2) \times (n_g + 2)$ matrix indicating the dependence of states on past states, and $\mathbf{B}$ an $(n_g + 2) \times 1$ vector indicating how states respond to innovations ϵ_t to the TFP process. To elicit $\mathbf{A}$ and $\mathbf{B}$ from a sequence-space model, we treat the initial states (ϵ_0, $\mathbf{D}_0$, K_{-1}, and Z_{-1}) as exogenous "shocks" and then look for their effects on the state the following period, ($\mathbf{D}_1$, K_0, Z_0).

Using the sequence-space model, we can compute the effect of shocks to (ϵ_0, $\mathbf{D}_0$, K_{-1}, and Z_{-1}) on the time paths of targets $d\mathbf{H}$. This is straightforward to do for aggregates, while for the distribution $\mathbf{D}_0$, these perturbations are directly given by the expectation vectors $\mathcal{E}_t^{\mathcal{K}}$ from Definition 1. Next, we solve for the equilibrium path of capital $d\mathbf{K} = -\mathbf{H}_{\mathbf{K}}^{-1}\, d\mathbf{H}$ that results from the perturbation. The date-0 element of this vector delivers the rows of $\mathbf{A}$ and $\mathbf{B}$ corresponding to K. Finally, using the general equilibrium Jacobian matrices $J^{w,K}$ and $J^{r,K}$, we can recover the effect on the time paths of wages and interest rates $d\mathbf{w}$ and $d\mathbf{r}$ for all shocks (ϵ_0, $\mathbf{D}_0$, K_{-1}, and Z_{-1}). Then, using the equation $d\mathbf{D}_1 = \mathcal{D}_1^r\, d\mathbf{r} + \mathcal{D}_1^w d\mathbf{w}$, where the $\mathcal{D}_t$ vectors are discussed in Section 3.2, we obtain the effect of all shocks (ϵ_0, $\mathbf{D}_0$, K_{-1}, and Z_{-1}) on the distribution at date 1. This gives us the first n_g rows of $\mathbf{A}$ and $\mathbf{B}$. Finally, the last row is just the exogenous law of motion of the shock process. Altogether, this procedure allows us to construct a state-space law of motion for the Krusell–Smith model. The procedure can easily be generalized to any alternative sequence-space model with a known state space.

APPENDIX D: EVALUATION OF ACCURACY

D.1. *Accuracy of Alternative Methods to Compute $\mathcal{J}$*

Figures D.1 and D.2 verify the relative accuracy of various methods for computing the Jacobian of aggregate assets with respect to the interest rate ($\mathcal{J}^{\mathcal{K},r}$ or $\mathcal{J}^{\mathcal{A},r}$) in our Krusell–Smith and one-asset HANK models. These are the two models for which it is feasible for us to compute the model with automatic differentiation. Our benchmark is the model computed using the direct method under automatic differentiation, which we will

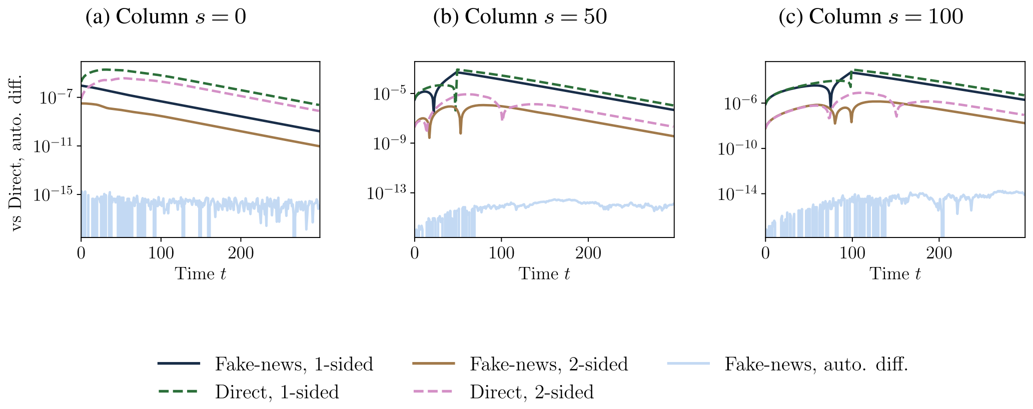


FIGURE D.1.—Accuracy of methods for computing $\mathcal{J}$ for Krusell–Smith model.

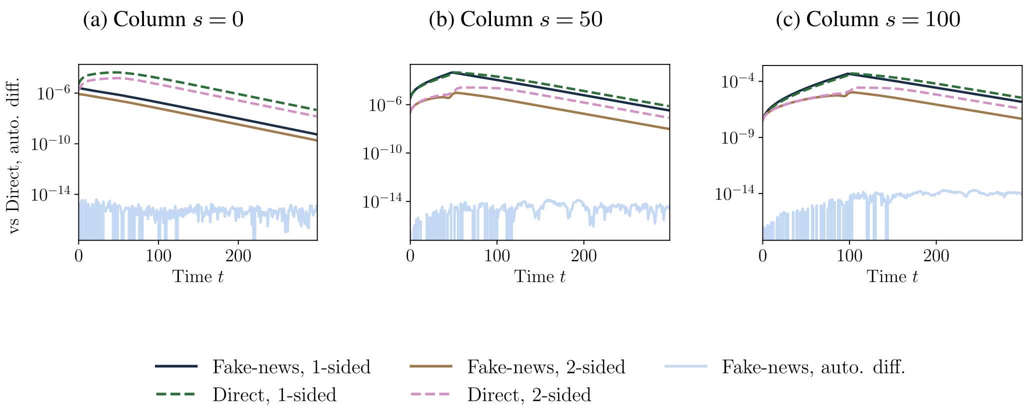


FIGURE D.2.—Accuracy of methods for computing $\mathcal{J}$ for one-asset HANK model.

refer to as the "true" impulse response. The impulse response in levels for the Krusell–Smith model are then those displayed in Figure 2, panel (a). For the one-asset HANK model, the levels are very similar. Here, we focus on the differences between various methods, for columns $s = 0, 50, 100$.[17]

We first compare impulse response obtained using our fake news algorithm under automatic differentiation to the "truth." The pale blue line on all graphs shows errors of the order of 10^{-14}. In other words, the direct and the fake news method yield the same answer to machine precision: this verifies Proposition 1.

Next, we compare the impulse response obtained with one-sided numerical differentiation (dark blue and dark green lines). There, the errors can get as large as 10^{-3}, or 0.01% of the peak of the level response (which is around 10). Two-sided numerical differentiation (brown and pink lines) mitigate this error by one to two digits of accuracy. In practice, two-sided numerical differentiation is just as simple to implement as one-sided numerical differentiation and only twice as costly in terms of computation time, so this may provide a useful alternative when very good accuracy is required.

Finally, we note that, unless automatic differentiation is used, the fake news method actually generally has better performance than the direct method. This is because it imposes some of the linearity implications of the true first-order derivative. Hence, the fake news impulse response is not only around T times faster to compute than the direct impulse response, it also tends to be more accurate.

APPENDIX E: ADDITIONAL COMPUTATIONAL DETAILS AND ACCURACY

E.1. *Two-Asset Household Model Algorithm*

In this section, we describe an efficient algorithm, based on the endogenous grid points approach of Carroll (2006), to solve the two-asset household model with convex adjustment costs.

Generic Setup. Households' individual state variables are (exogenous) income $z_{it} \in \{z_1, \dots, z_m\}$, liquid assets $b_{it-1} \in [\underline{b}, \infty)$, and illiquid assets $a_{it-1} \in [0, \infty)$. The Bellman

[17]Using the sup norm over the entire Jacobian, as well as other Jacobians obtained via this method, yields the same findings.

equation is

$$V_t(z_{it}, b_{it-1}, a_{it-1}) = \max_{c_{it}, b_{it}, a_{it}} u(c_{it}) + \beta \mathbb{E}_t V_{t+1}(z_{it+1}, b_{it}, a_{it})$$

$$\text{s.t.} \quad c_{it} + a_{it} + b_{it} = z_{it} + \left(1 + r_t^a\right) a_{it-1} + \left(1 + r_t^b\right) b_{it-1} - \Phi(a_{it}, a_{it-1}),$$

$$a_{it} \geq 0, \qquad b_{it} \geq \underline{b}.$$

The adjustment cost function is

$$\Phi(a_{it}, a_{it-1}) = \frac{\chi_1}{\chi_2} \left| \frac{a_{it} - \left(1 + r_t^a\right) a_{it-1}}{\left(1 + r_t^a\right) a_{it-1} + \chi_0} \right|^{\chi_2} \left[\left(1 + r_t^a\right) a_{it-1} + \chi_0\right], \tag{63}$$

with $\chi_0, \chi_1 > 0$ and $\chi_2 > 1$. Note that $\Phi(a_{it}, a_{it-1})$ is bounded, differentiable, and convex in a_{it}.

First-Order and Envelope Conditions. The Bellman equation can be rewritten more compactly as

$$V_t(z_{it}, b_{it-1}, a_{it-1}) = \max_{b_{it}, a_{it}} u\left(z_{it} + \left(1 + r_t^a\right) a_{it-1} + \left(1 + r_t^b\right) b_{it-1} - \Phi(a_{it}, a_{it-1}) - a_{it} - b_{it}\right)$$
$$+ \lambda_{it}(b_{it} - \underline{b}) + \mu_{it} a_{it} + \beta \mathbb{E} V_{t+1}(z_{it+1}, b_{it}, a_{it}).$$

The first-order conditions with respect to b_{it} and a_{it} are

$$u'(c_{it}) = \lambda_{it} + \beta \mathbb{E} \partial_b V_{t+1}(z_{it+1}, b_{it}, a_{it}), \tag{64}$$

$$u'(c_{it})\left[1 + \Phi_1(a_{it}, a_{it-1})\right] = \mu_{it} + \beta \mathbb{E} \partial_a V_{t+1}(z_{it+1}, b_{it}, a_{it}), \tag{65}$$

and the envelope conditions are

$$\partial_b V_t(z_{it}, b_{it-1}, a_{it-1}) = \left(1 + r_t^b\right) u'(c_{it}), \tag{66}$$

$$\partial_a V_t(z_{it}, b_{it-1}, a_{it-1}) = \left[1 + r_t^a - \Phi_2(a_{it}, a_{it-1})\right] u'(c_{it}). \tag{67}$$

It is convenient to define the *post-decision value function* $W_t(z_{it}, b_{it}, a_{it}) \equiv \beta \mathbb{E}_t V_{t+1}(z_{it}, b_{it}, a_{it})$.

Algorithm. We start from a guess for the (discretized) partials of the value function and iterate backward until convergence. We use (z', b', a') to refer to tomorrow's grid and (z, b, a) to today's grid. Let Π denote the transition matrix of the exogenous state z. The key trick is to include Lagrange multipliers in the backward iteration whenever the household is partially constrained. We also exploit the fact that the constraint on the illiquid asset will never be binding unless the constraint on the liquid asset is also binding (otherwise, a simple variation will improve utility).

1. *Initial guess.* Guess $V_a(z', b', a')$ and $V_b(z', b', a')$.
2. *Common $z' \to z$.* By definition

$$W_b\left(z, b', a'\right) = \beta \Pi V_b\left(z', b', a'\right), \tag{68}$$

$$W_a\left(z, b', a'\right) = \beta \Pi V_a\left(z', b', a'\right). \tag{69}$$

3. *Unconstrained* $\underline{a' \to a}$. Assuming that no constraints bind, $\lambda_{it} = \mu_{it} = 0$, and (64) and (65) become

$$u'(c) = W_b(z, b', a'), \tag{70}$$

$$u'(c)\left[1 + \Phi_1(a', a)\right] = W_a(z, b', a'). \tag{71}$$

Combine these to get

$$0 = F(z, b', a, a') \equiv \frac{W_a(z, b', a')}{W_b(z, b', a')} - 1 - \Phi_1(a', a), \tag{72}$$

which characterizes $a'(z, b', a)$. Use this to map $W_b(z, b', a')$ into $W_b(z, b', a)$ by interpolation, then compute consumption as

$$c(z, b', a) = W_b(z, b', a)^{-\frac{1}{\sigma}}. \tag{73}$$

4. *Unconstrained* $\underline{b' \to b}$. Now using $a'(z, b', a)$ and $c(z, b', a)$ from the previous step, use the budget constraint to obtain

$$b(z, b', a) = \frac{c(z, b', a) + a'(z, b', a) + b' - (1 + r^a)a + \Phi(a'(z, b', a), a) - z}{1 + r^b}.$$

We invert this function via interpolation to get $b'(z, b, a)$. The same interpolation weights can be used to do $a'(z, b', a) \to a'(z, b, a)$.

5. *Liquidity constrained* $\underline{a' \to a}$. This branch is analogous to the unconstrained case. Assuming that the liquidity constraint is binding, $\lambda_{it} > 0$, and (64) and (65) become

$$u'(c) = \lambda + W_b(z, 0, a'),$$

$$u'(c)\left[1 + \Phi_1(a', a)\right] = W_a(z, 0, a').$$

To help with scaling, let us define $\kappa \equiv \lambda / W_b(z, 0, a')$ and rewrite the first equation as

$$u'(c) = (1 + \kappa)W_b(z, 0, a').$$

Divide and rearrange to get

$$0 = F(z, \kappa, a, a') \equiv \frac{1}{1 + \kappa}\frac{W_a(z, 0, a')}{W_b(z, 0, a')} - 1 - \Phi_1(a', a). \tag{74}$$

We solve this for $a'(z, \kappa, a)$, and compute consumption as

$$c(z, \kappa, a) = \left[(1 + \kappa)W_b(z, \kappa, a)\right]^{-\frac{1}{\sigma}}. \tag{75}$$

6. *Liquidity constrained* $\underline{\kappa \to b}$. Now using $a'(z, \kappa, a)$ and $c(z, \kappa, a)$ from the previous step, use the budget constraint to obtain

$$b(z, \kappa, a) = \frac{c(z, \kappa, a) + a'(z, \kappa, a) + \underline{b} - (1 + r^a)a + \Phi(a'(z, \kappa, a), a) - z}{1 + r^b}.$$

We invert this function via interpolation to get $\kappa(z,b,a)$. The same interpolation weights can be used to map $a'(z,\kappa,a)$ into $a'(z,b,a)$. We already know that $b'(z,b,a)=\underline{b}$.

7. *Update guesses.* The final $b'(z,b,a)$ is the element-wise maximum of its unconstrained and liquidity-constrained counterparts. Replace the unconstrained $a'(z,b,a)$ with constrained one at the exact same points. Compute consumption from the budget constraint as

$$\begin{aligned} c(z,b,a) = z + \left(1+r^a\right)a + \left(1+r^b\right)b - \Phi\left(a'(z,b,a),a\right) \\ - a'(z,b,a) - b'(z,b,a). \end{aligned} \tag{76}$$

Finally, use the envelope conditions (66) and (67) to update the guesses

$$V_b(z,b,a) = \left(1+r^b\right)c(z,b,a)^{-\sigma}, \tag{77}$$

$$V_a(z,b,a) = \left[1+r^a-\Phi_2\left(a'(z,b,a),a\right)\right]c(z,b,a)^{-\sigma}. \tag{78}$$

Go back to step 2, repeat until convergence.

E.2. *Efficient Multiplication of Simple Jacobians*

One important detail underlying the speeds in Table C.I is a set of special routines that efficiently handle the Jacobians of simple blocks. These simple blocks comprise the majority of our DAGs. Their Jacobians are easy to obtain to high accuracy (for instance, with symmetric numerical differentiation), and have a special sparse structure: they can be expressed as linear combinations of a few *shift* operators S_i on sequences.

For positive i, S_i maps $(x_0,x_1,\dots)\to(0,\dots,0,x_0,x_1,\dots)$, with i zeros inserted at the beginning, and for negative $-i$, S_{-i} maps $(x_0,x_1,\dots)\to(x_i,x_{i+1},\dots)$. The former takes an i-period *lag* in sequence space, while the latter takes an i-period *lead* in sequence space.[18] For instance, in the one-asset HANK economy depicted in Figure C.1, the Jacobian $\mathcal{J}^{H_1,\pi}$ of the Phillips curve condition with respect to price inflation π is $S_0-\frac{1}{1+r}S_{-1}$.[19]

For the most part, these operators obey simple rules: if i and j are both positive, $S_iS_j=S_{i+j}$, and so on. However, as is well known from an older literature that works with the lag algebra (e.g., Whiteman (1983)), the S are not quite closed under multiplication. To take the simplest example, S_1S_{-1}, a one-period lag of a one-period lead, maps $(x_0,x_1,x_2,\dots)\to(0,x_1,x_2,\dots)$, zeroing out the first entry of a sequence and leaving everything else unchanged. Fortunately, we have found a more general set of operators that includes the S and is closed under multiplication following an easy-to-compute rule, as we derive in the following proposition.

PROPOSITION 3: *Let S_i be the shift operator on sequences, and Z_m be the "zero" operator that replaces the first m entries of a sequence with zeros. If we define*

$$Q_{i,m} \equiv \begin{cases} S_iZ_m, & i>0, \\ Z_mS_i, & i<0, \end{cases} \tag{79}$$

[18]In matrix form, S_i has zeros everywhere, except for 1's on the ith diagonal below the main diagonal.

[19]This corresponds to a linearized curve of the form $\pi_t=\cdots+\frac{1}{1+r}\mathbb{E}_t\pi_{t+1}$.

then $Q_{i,m}Q_{j,n} = Q_{k,l}$, *where*

$$k = i + j, \tag{80}$$

and

$$l = \begin{cases} \max(m-j,n), & i,j \geq 0, \\ \max(m,n)+\min(i,-j), & i \geq 0, j \leq 0, \\ \max(m-i-j,n), & i \leq 0, j \geq 0, i+j \geq 0, \\ \max(n+i+j,m), & i \leq 0, j \geq 0, i+j \leq 0, \\ \max(m,n+i), & i,j \leq 0. \end{cases} \tag{81}$$

This proposition nests the shift operators S_i in a more general class of operators $Q_{i,m}$.[20] This has two advantages. First, it makes multiplying the Jacobians of simple blocks vastly more efficient: rather than doing matrix multiplication with large $T \times T$ matrices, we just need to apply rules (80) and (81) a few times. Second, it is computationally easy to multiply $Q_{i,m}$ and an ordinary matrix Jacobian (or vector), since this is a combination of shifting and zeroing elements. Together, these features make forward accumulation on the DAG, which consists mostly of simple blocks, vastly more efficient.

In our online code, we implement this by simply overriding the matrix multiplication operator, so that sparse linear combinations of $Q_{i,m}$ and ordinary matrices can be used interchangeably. With this in place, the methods of Section C.3 can be applied without any outwardly visible modification.

Exploiting sparsity has played a prominent role in both the heterogeneous-agent literature (e.g., Achdou, Han, Lasry, Lions, and Moll (2020)) and the literature on solving for perfect-foresight paths using Newton's method (e.g., Juillard (1996)). Our approach builds on the latter, but our much more compact representation of Jacobians offers additional efficiencies. For instance, to store $0.5 \cdot Q_{1,1}$, we only need a few numbers, while a conventional $T \times T$ sparse matrix representation not taking advantage of this structure would need $T-2$ separate entries, and still create some truncation error.

PROOF: Here, we derive the rules for multiplication of the operator (79), where S_i is the shift operator on sequences by i and Z_m zeros out the first m elements of sequences, by doing case-by-case analysis on the product $Q_{i,m}Q_{j,n}$. In our derivation, we will exploit the following fact about multiplication of S_i:

$$S_iS_j = \begin{cases} S_{i+j}Z_{-j}, & i>0, j<0, i+j>0, \\ Z_iS_{i+j}, & i>0, j<0, i+j<0, \\ S_{i+j}, & \text{otherwise}, \end{cases}$$

and the rules $S_{-i}Z_j = Z_{\max(j-i,0)}S_{-i}$ and $Z_jS_i = S_iZ_{\max(j-i,0)}$ for multiplication of S and Z.

Case 1: positive i, positive j. Here we have

$$\begin{aligned} Q_{i,m}Q_{j,n} &= S_iZ_mS_jZ_n \\ &= S_iS_jZ_{\max(m-j,0)}Z_n \end{aligned}$$

[20]The matrix representation of $Q_{i,m}$ is the same as that of S_i, except that the first m entries on the diagonal are zeros.

$$= S_{i+j} Z_{\max(m-j,n)}$$
$$= Q_{i+j,\max(m-j,n)}. \tag{82}$$

Case 2: positive i, negative j. Here we have

$$\begin{aligned} Q_{i,m} Q_{j,n} &= S_i Z_m Z_n S_j \\ &= S_i Z_{\max(m,n)} S_j. \end{aligned}$$

If $i + j > 0$, then we write

$$Z_{\max(m,n)} S_j = S_j Z_{\max(m,n)-j}$$

and then

$$\begin{aligned} S_i Z_{\max(m,n)} S_j &= S_i S_j Z_{\max(m,n)-j} \\ &= S_{i+j} Z_{-j} Z_{\max(m,n)-j} \\ &= S_{i+j} Z_{\max(m,n)-j} \\ &= Q_{i+j,\max(m,n)-j}. \end{aligned}$$

If $i + j < 0$, then we write

$$S_i Z_{\max(m,n)} = Z_{\max(m,n)+i} S_i$$

and then

$$\begin{aligned} S_i Z_{\max(m,n)} S_j &= Z_{\max(m,n)+i} S_i S_j \\ &= Z_{\max(m,n)+i} Z_i S_{i+j} \\ &= Z_{\max(m,n)+i} S_{i+j} \\ &= Q_{i+j,\max(m,n)+i}. \end{aligned}$$

Both these cases boil down to the simpler form

$$Q_{i,m} Q_{j,n} = Q_{i+j,\max(m,n)+\min(i,-j)}. \tag{83}$$

Case 3: negative i, positive j. Then we have

$$\begin{aligned} Q_{i,m} Q_{j,n} &= Z_m S_i S_j Z_n \\ &= Z_m S_{i+j} Z_n. \end{aligned}$$

If $i + j > 0$, then we write $Z_m S_{i+j} = S_{i+j} Z_{\max(m-i-j,0)}$ and get

$$Q_{i,m} Q_{j,n} = Q_{i+j,\max(m-i-j,n)}. \tag{84}$$

If $i + j < 0$, then we write $S_{i+j} Z_n = Z_{\max(n+i+j,0)} S_{i+j}$ and get

$$Q_{i,m} Q_{j,n} = Q_{i+j,\max(n+i+j,m)}. \tag{85}$$

Case 4: negative i, negative j. Then we have

$$\begin{aligned} Q_{i,m}Q_{j,n} &= Z_m S_i Z_n S_j \\ &= Z_m Z_{\max(n+i,0)} S_i S_j \\ &= Z_{\max(m,n+i)} S_{i+j} \\ &= Q_{i+j,\max(m,n+i)}. \end{aligned} \tag{86}$$

Combined, equations (82)–(86) give (80) and (81) in Proposition 3. *Q.E.D.*

E.3. *Simulating Panels of Individuals*

Here, we briefly describe how to simulate a panel of individuals. The first option is to recover the state-space law of motion, as described at the end of Section C.3, augmented with policies. Then, one can simulate using the state space.

The second option is to recover the MA for policies. For example, in the Krusell–Smith model, the MA for the capital policy, truncated to T, in response to innovations $\boldsymbol{\epsilon}_t$ to TFP, takes the form

$$d\mathbf{k}_t = \sum_{s=0}^{T} \mathbf{K}_s \boldsymbol{\epsilon}_{t-s}, \tag{87}$$

where the ith entry in the vector $d\mathbf{k}_t$ corresponds to the change in the capital policy of households in state i. Hence, in order to simulate a panel of individuals, we need to recover the $N \times 1$ vectors $\mathbf{K}_s$. This can be done as follows. Using the policy function symmetry property from Lemma 1, we know that

$$\begin{aligned} \begin{pmatrix} d\mathbf{k}_0 \\ d\mathbf{k}_1 \\ d\mathbf{k}_2 \\ \vdots \end{pmatrix} &= \begin{pmatrix} \frac{\partial \mathbf{k}_0}{\partial w_0} & \frac{\partial \mathbf{k}_0}{\partial w_1} & \frac{\partial \mathbf{k}_0}{\partial w_2} \\ 0 & \frac{\partial \mathbf{k}_0}{\partial w_0} & \frac{\partial \mathbf{k}_0}{\partial w_1} \\ 0 & 0 & \frac{\partial \mathbf{k}_0}{\partial w_0} \end{pmatrix} \begin{pmatrix} dw_0 \\ dw_1 \\ dw_2 \\ \vdots \end{pmatrix} + \begin{pmatrix} \frac{\partial \mathbf{k}_0}{\partial r_0} & \frac{\partial \mathbf{k}_0}{\partial r_1} & \frac{\partial \mathbf{k}_0}{\partial r_2} \\ 0 & \frac{\partial \mathbf{k}_0}{\partial r_0} & \frac{\partial \mathbf{k}_0}{\partial r_1} \\ 0 & 0 & \frac{\partial \mathbf{k}_0}{\partial r_0} \end{pmatrix} \begin{pmatrix} dr_0 \\ dr_1 \\ dr_2 \\ \vdots \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial \mathbf{k}_0}{\partial w_0} & \frac{\partial \mathbf{k}_0}{\partial w_1} & \frac{\partial \mathbf{k}_0}{\partial w_2} \cdots \end{pmatrix} \begin{pmatrix} dw_0 & 0 & 0 \\ dw_1 & dw_0 & \ddots \\ dw_2 & dw_1 & \ddots \\ dw_3 & dw_2 & \ddots \\ \vdots & \vdots & \ddots \end{pmatrix} \\ &\quad + \begin{pmatrix} \frac{\partial \mathbf{k}_0}{\partial r_0} & \frac{\partial \mathbf{k}_0}{\partial r_1} & \frac{\partial \mathbf{k}_0}{\partial r_2} \cdots \end{pmatrix} \begin{pmatrix} dr_0 & 0 & 0 \\ dr_1 & dr_0 & \ddots \\ dr_2 & dr_1 & \ddots \\ dr_3 & dr_2 & \ddots \\ \vdots & \vdots & \ddots \end{pmatrix}. \end{aligned} \tag{88}$$

The derivatives of the first-period policy $\mathbf{k}_0$ with respect to w_t and r_t are byproducts of step 1 of the fake news algorithm. Further, from the procedure to recover the MA representation described in Section 5.1, we obtain the matrices $M^{w,\epsilon}$ and $M^{r,\epsilon}$ satisfying

$$\begin{pmatrix} dw_0 \\ dw_1 \\ dw_2 \\ \vdots \end{pmatrix} = M^{w,\epsilon} \begin{pmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \end{pmatrix}, \qquad \begin{pmatrix} dr_0 \\ dr_1 \\ dr_2 \\ \vdots \end{pmatrix} = M^{r,\epsilon} \begin{pmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \end{pmatrix}. \tag{89}$$

Combining (88) and (89) delivers the coefficients $\mathbf{K}_t$ in (87). Note that equation (88) is fast to implement, since it involves a multiplication of $N \times T$ matrices by $T \times T$ matrices that can be formed efficiently from (89).

Finally, to simulate a panel of individuals from the MA for policies, one can then take these perturbed policy functions and apply them to simulated individuals.

E.4. *Fast Fourier Transform to Compute Analytical Second Moments*

Consider any sequences $a_0, \dots, a_{T-1}$ and $b_0, \dots, b_{T-1}$ of real scalars. If we define the sequences

$$(\hat{a}_0, \dots, \hat{a}_{2T-2}) = (a_0, \dots, a_{T-1}, 0, \dots, 0),$$
$$(\hat{b}_0, \dots, \hat{b}_{2T-2}) = (b_0, \dots, b_{T-1}, 0, \dots, 0)$$

to be a and b each padded by $T-1$ zeros, then

$$a_0 b_u + a_1 b_{u+1} + \dots + a_{T-1-u} b_{T-1} = \sum_{\ell=0}^{2T-2} \hat{a}_\ell \hat{b}_{u+\ell}, \tag{90}$$

where $\hat{b}_{u+\ell} \equiv \hat{b}_{u+\ell-(2T-2)}$ when $u + \ell \geq 2T-2$. It then follows from the standard properties of the discrete Fourier transform $\mathcal{F}$ that, for any $u \in 0, \dots, T-1$,

$$\sum_{\ell=0}^{2T-2} \hat{a}_\ell \hat{b}_{u+\ell} = \big(\mathcal{F}^{-1}\big(\mathcal{F}(\hat{a})^* \cdot \mathcal{F}(\hat{b})\big)\big)_s, \tag{91}$$

where $*$ denotes complex conjugation.[21]

Since the discrete Fourier transform is a linear operator, we can extend this method to apply to the matrices $d\mathbf{X}_0, \dots, d\mathbf{X}_{T-1}$ in (34), where we interpret $\mathcal{F}$ as applying element-by-element to a sequence of matrices. Letting $d\hat{\mathbf{X}}_0, \dots, d\hat{\mathbf{X}}_{2T-2}$ denote the sequence padded with zeros like above, we have from (90) and (91), substituting $u = t' - t$, that

$$\begin{aligned} \sum_{s=0}^{T-1-(t'-t)} (d\mathbf{X}_s)(d\mathbf{X}_{s+t'-t})' &= [d\mathbf{X}_0][d\mathbf{X}_{t'-t}]' + \dots + [d\mathbf{X}_{T-1-(t'-t)}][d\mathbf{X}_{T-1}]' \\ &= \big(\mathcal{F}^{-1}\big(\mathcal{F}(d\hat{\mathbf{X}})^* \mathcal{F}(d\hat{\mathbf{X}}')\big)\big)_{t'-t}, \end{aligned} \tag{92}$$

[21]The padding with zeros to create $\hat{a}$ and $\hat{b}$ is necessary so that the wraparound $\hat{b}_{s+\ell}$ terms for large $s + \ell$ do not affect the sum.

where $d\hat{\mathbf{X}}$ is the stacked sequence $d\hat{\mathbf{X}}_0, \dots, d\hat{\mathbf{X}}_{2T-2}$, the transpose $d\hat{\mathbf{X}}'$ is applied individually to each matrix in the sequence, and $\mathcal{F}(d\hat{\mathbf{X}})^*\mathcal{F}(d\hat{\mathbf{X}}')$ is the product of each pair of matrices in the frequency-by-frequency sequence.[22]

We simply apply (92), using the fast Fourier transform for $\mathcal{F}$, to calculate the covariances in (34) for each $t' - t$. Since the two key operations—the FFT and matrix multiplication—have extremely efficient implementations widely available, this can be done very quickly, taking only a few milliseconds in Table III for the examples in this paper. It is far faster than a naive calculation of the sum in (34).

This procedure is closely related to the standard FFT approach to calculating the empirical autocovariance function (although many implementations only apply to one-dimensional series, missing the efficiencies from exploiting linearity in equation (92)).[23] It is also similar to the standard formulas for the spectral density of an MA, and for inverting this spectrum (see, e.g., Hansen, Peter, and Sargent (1981)).

E.5. *Equivalence Between SSJ and Dynare for Representative-Agent Models*

In order to illustrate the accuracy of our routines to calculate impulse responses for representative-agent models, here we perform two tests of the accuracy of impulse responses on two classic representative-agent models. Specifically, we simulate the Smets and Wouters (2007) model and the benchmark model described in Herbst and Schorfheide (2015), with the parameters at the estimated mode presented in these sources, using both our method and Dynare. Figure F.1 compares the impulse responses of output to all shocks for the Smets–Wouters exercise: the left panel shows the level of

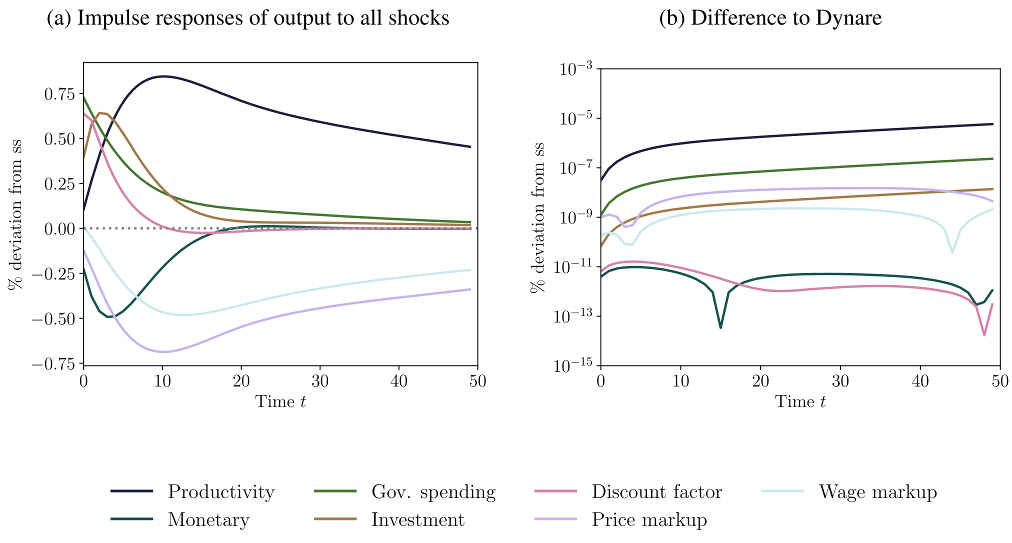


FIGURE F.1.—Equivalence between SSJ and Dynare for the Smets and Wouters (2007) model.

[22]Since the inputs are real, the full transform is redundant and we can deal only with the first T entries; the final $T - 2$ are complex conjugates of entries 1 through $T - 1$. This economizes on the time for $\mathcal{F}$ and also for matrix multiplication.

[23]For instance, in the Python "statsmodels" package, the now-default "fft = True" option uses the FFT to calculate the autocovariances of a one-dimensional time series.

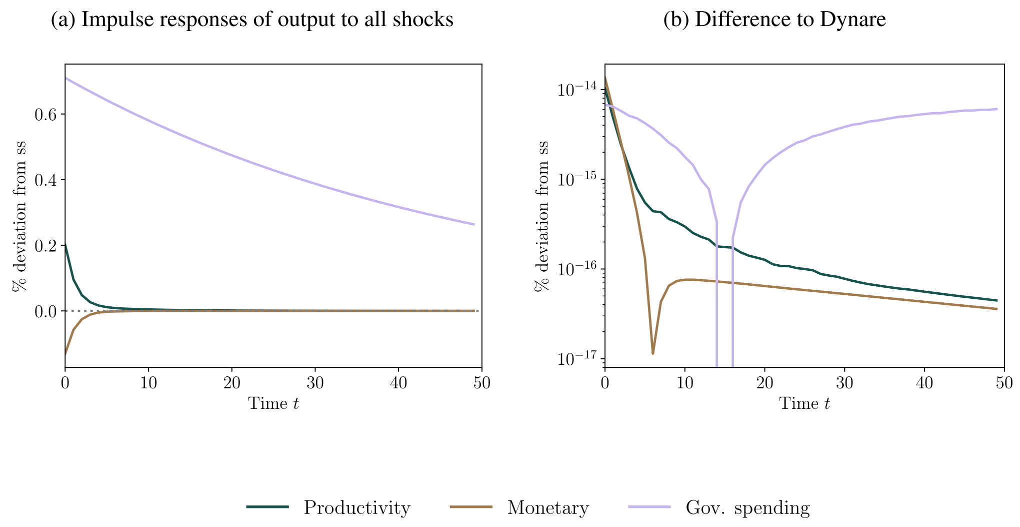


FIGURE F.2.—Equivalence between SSJ and Dynare for the Herbst and Schorfheide (2015) model.

the impulse responses, and the right panel displays the difference to Dynare. Figure F.2 repeats this exercise with the Herbst–Schorfheide model. As can be seen, our method delivers the same impulse responses for these benchmark representative-agent models with very high accuracy.

APPENDIX F: BAYESIAN ESTIMATION RESULTS

F.1. *Accuracy of Likelihood Computation for Representative-Agent Models*

Here, we continue the exercise of Section E.5 by showing that, for both the Smets and Wouters (2007) model and the Herbst and Schorfheide (2015) model, estimating the posterior mode on the original data set yields the same answer with our method as it does with the routines offered in Dynare (which uses the Kalman filter on a state-space representation of the model). Table F.I shows that the posterior mode is identical to three-digit accuracy. As discussed in the main text, this verifies that our method to compute the likelihood for these benchmark models is accurate.

F.2. *Estimating Heterogeneous-Agent Models*

Tables F.II–F.IV summarize the prior and posterior distributions of the estimated parameters. Figures F.3–F.12 show recursive means and posterior modes for all our estimated models.

For simplicity, the DAG of the two-asset model in Figure C.2 is drawn with only three shocks: to TFP Z_t, monetary policy r_t^*, and government spending G_t. In our estimation, we add to these four additional shocks: we let the price markup μ_p vary over time (a price markup shock), the wage markup μ_w vary over time (a wage markup shock), the discount rate of households β vary over time (a preference shock), and we add a spread r_{It} to the interest rate r_t that enters the firm valuation equation (53), and let that spread vary over time (an investment shock).

TABLE F.I

ESTIMATED PARAMETERS FOR THE SMETS–WOUTERS AND HERBST–SCHORFHEIDE ECONOMIES

Model	Method	Parameters							
Smets–Wouters		σ_a	ρ_a	σ_b	ρ_b	σ_g	ρ_g	σ_I	ρ_I
	SSJ	0.446	0.978	0.246	0.250	0.589	0.971	0.461	0.662
	Dynare	0.446	0.978	0.245	0.252	0.589	0.970	0.460	0.663
		σ_i	ρ_i	σ_p	ρ_p	ρ_p^{ma}	σ_w	ρ_w	ρ_w^{ma}
	SSJ	0.229	0.086	0.135	0.975	0.740	0.256	0.976	0.925
	Dynare	0.229	0.086	0.133	0.975	0.735	0.256	0.975	0.924
Herbst–Schorfheide		τ	κ	ψ_1	ψ_2	$\bar{r}$	$\bar{\pi}$	$\bar{y}$	
	SSJ	2.3162	1.0000	1.9684	0.4754	0.3043	3.4468	0.6214	
	Dynare	2.3164	1.0000	1.9684	0.4753	0.3051	3.4472	0.6213	
		σ_r	ρ_r	σ_g	ρ_g	σ_z	ρ_z		
	SSJ	0.1905	0.7978	0.6531	0.9908	0.1855	0.9252		
	Dynare	0.1905	0.7978	0.6530	0.9903	0.1855	0.9252		

TABLE F.II

ESTIMATED PARAMETERS FOR OUR KRUSELL–SMITH ECONOMY

			Posterior		
Shock		Prior Distribution	Mode	Mean	[0.05, 0.95] CI
TFP shock	s.d.	Invgamma(0.4, 4)	0.179	0.182	[0.165, 0.200]
	AR-1	Beta(0.5, 0.2)	0.908	0.908	[0.862, 0.950]
	MA-1	Beta(0.5, 0.2)	0.032	0.047	[0.015, 0.095]

TABLE F.III

ESTIMATED PARAMETERS FOR OUR ONE-ASSET HANK ECONOMY

			Posterior (Shocks)			Posterior (Shocks + Model)		
Parameter / shock		Prior Distribution	Mode	Mean	[0.05, 0.95] CI	Mode	Mean	[0.05, 0.95] CI
MP shock	s.d.	Invgamma(0.4, 4)	0.429	0.434	[0.393, 0.478]	0.419	0.430	[0.385, 0.481]
	AR-1	Beta(0.5, 0.2)	0.529	0.525	[0.478, 0.569]	0.463	0.460	[0.393, 0.524]
G shock	s.d.	Invgamma(0.4, 4)	0.580	0.584	[0.514, 0.662]	0.569	0.581	[0.508, 0.662]
	AR-1	Beta(0.5, 0.2)	0.872	0.870	[0.838, 0.900]	0.833	0.821	[0.771, 0.868]
P markup shock	s.d.	Invgamma(0.4, 4)	0.099	0.101	[0.091, 0.112]	0.092	0.096	[0.067, 0.128]
	AR-1	Beta(0.5, 0.2)	0.881	0.878	[0.849, 0.905]	0.913	0.909	[0.875, 0.942]
ϕ		Gamma(1.5, 0.25)				1.320	1.352	[1.231, 1.495]
ϕ_y		Gamma(0.5, 0.25)				0.126	0.143	[0.061, 0.250]
κ		Gamma(0.1, 0.1)				0.140	0.144	[0.105, 0.186]

TABLE F.IV

ESTIMATED PARAMETERS FOR OUR TWO-ASSET HANK ECONOMY

Parameter / Shock		Prior Distribution	Posterior (Shocks) Mode	Posterior (Shocks) Mean	Posterior (Shocks) [0.05, 0.95] CI	Posterior (Shocks + Model) Mode	Posterior (Shocks + Model) Mean	Posterior (Shocks + Model) [0.05, 0.95] CI
TFP shock	s.d.	Invgamma(0.4, 4)	0.072	0.073	[0.066, 0.080]	0.071	0.072	[0.066, 0.080]
	AR-1	Beta(0.5, 0.2)	0.994	0.941	[0.911, 0.968]	0.970	0.951	[0.921, 0.979]
G shock	s.d.	Invgamma(0.4, 4)	0.437	0.441	[0.400, 0.487]	0.467	0.643	[0.543, 0.783]
	AR-1	Beta(0.5, 0.2)	0.503	0.499	[0.448, 0.548]	0.292	0.952	[0.914, 0.986]
β shock	s.d.	Invgamma(0.4, 4)	0.093	0.093	[0.085, 0.103]	0.093	0.096	[0.087, 1.104]
	AR-1	Beta(0.5, 0.2)	0.941	0.938	[0.906, 0.967]	0.971	0.943	[0.913, 0.969]
r_I (investment) shock	s.d.	Invgamma(0.4, 4)	0.174	0.179	[0.147, 0.214]	0.089	0.413	[0.360, 0.473]
	AR-1	Beta(0.5, 0.2)	0.779	0.775	[0.731, 0.816]	0.867	0.656	[0.594, 0.714]
Monetary policy shock	s.d.	Invgamma(0.4, 4)	0.1442	0.146	[0.123, 0.172]	0.655	0.663	[0.487, 0.844]
	AR-1	Beta(0.5, 0.2)	0.830	0.827	[0.798, 0.856]	0.844	0.737	[0.671, 0.874]
P markup shock	s.d.	Invgamma(0.4, 4)	0.091	0.092	[0.083, 0.101]	0.059	0.049	[0.032, 0.070]
	AR-1	Beta(0.5, 0.2)	0.904	0.903	[0.881, 0.923]	0.888	0.891	[0.851, 0.923]
W markup shock	s.d.	Invgamma(0.4, 4)	0.373	0.377	[0.343, 0.414]	0.142	0.155	[0.116, 0.202]
	AR-1	Beta(0.5, 0.2)	0.875	0.872	[0.844, 0.899]	0.648	0.586	[0.432, 0.734]
ϕ		Gamma(1.5, 0.25)				1.203	1.297	[1.021, 1.764]
ϕ_y		Gamma(0.5, 0.25)				0.086	2.932	[2.564, 3.519]
κ^p		Gamma(0.1, 0.1)				0.035	0.030	[0.010, 0.064]
κ^w		Gamma(0.1, 0.1)				0.009	0.011	[0.007, 0.017]
ϵ_I		Gamma(4, 2)				0.267	0.502	[0.349, 0.670]

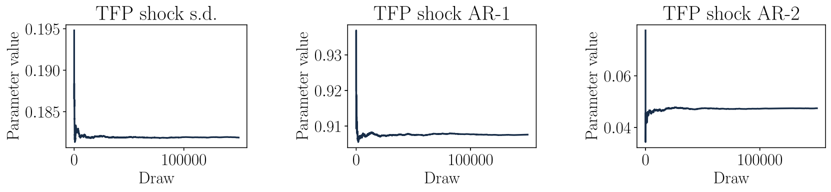


FIGURE F.3.—Recursive means for the RWMH estimation of the Krusell–Smith model.

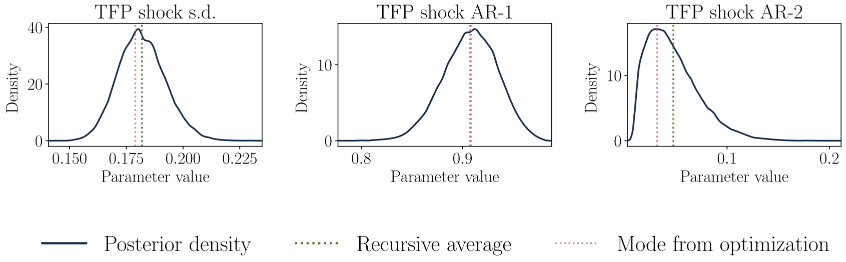


FIGURE F.4.—Posterior distributions for the RWMH estimation of the Krusell–Smith model.

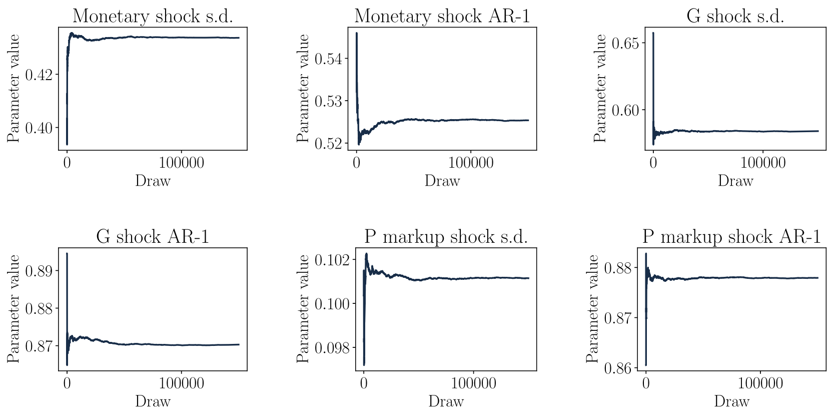


FIGURE F.5.—Recursive means for the RWMH estimation of the one-asset HANK model with shocks.

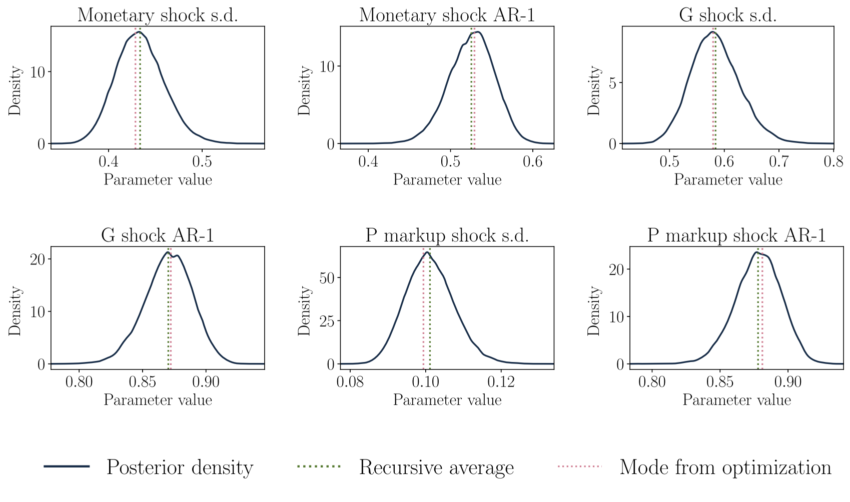


FIGURE F.6.—Posterior distributions for the RWMH estimation of the one-asset HANK model with shocks.

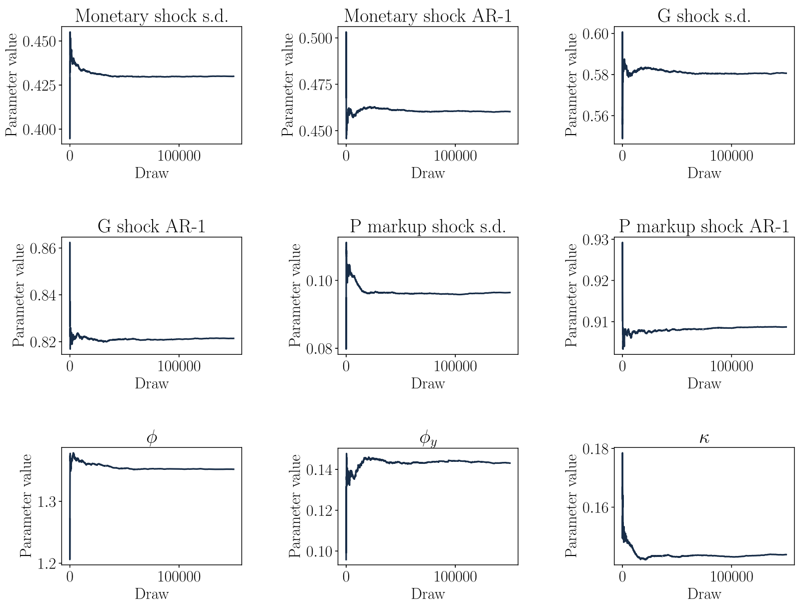


FIGURE F.7.—Recursive means for the RWMH estimation of the one-asset HANK model with shocks and parameters.

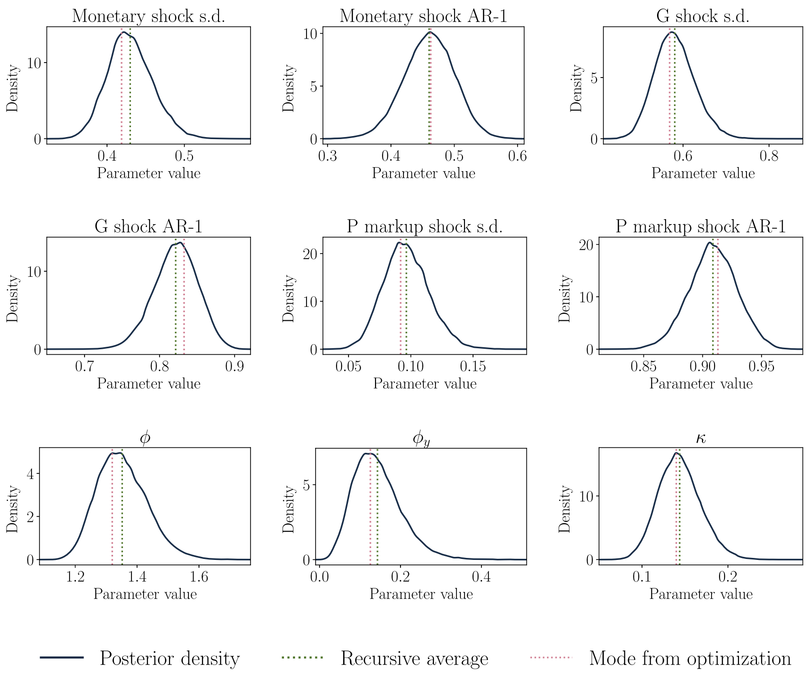


FIGURE F.8.—Posterior distributions for the RWMH estimation of the one-asset HANK model with shocks and parameters.

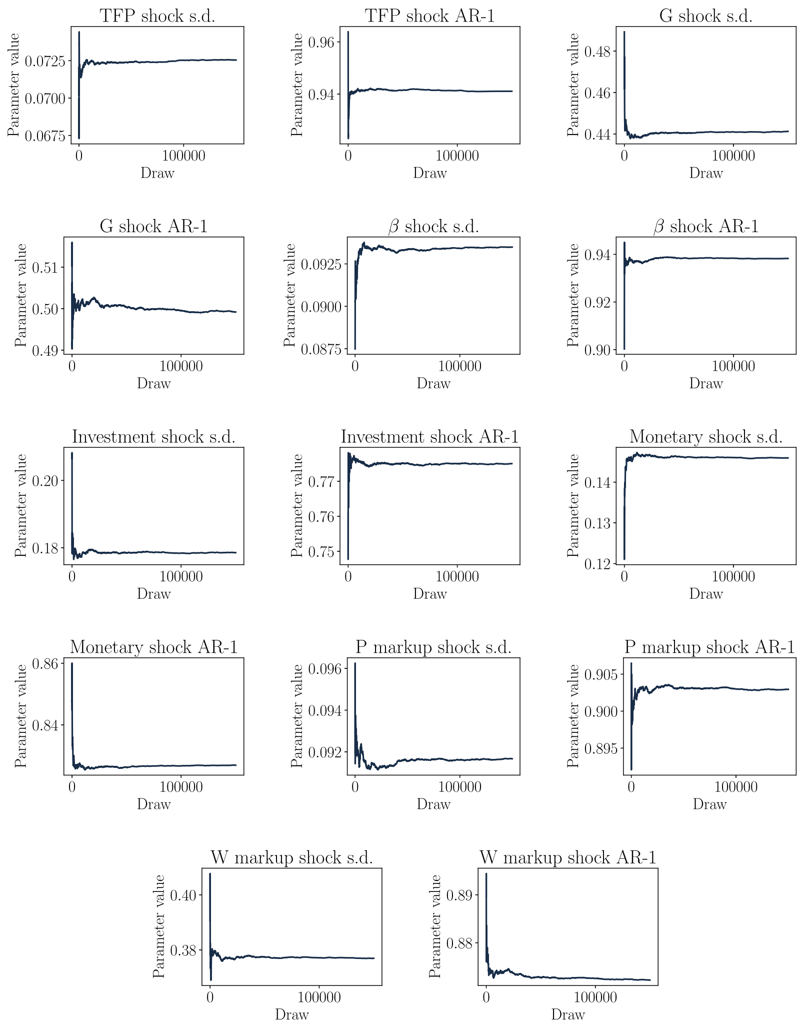


FIGURE F.9.—Recursive means for the RWMH estimation of the two-asset HANK model with shocks.

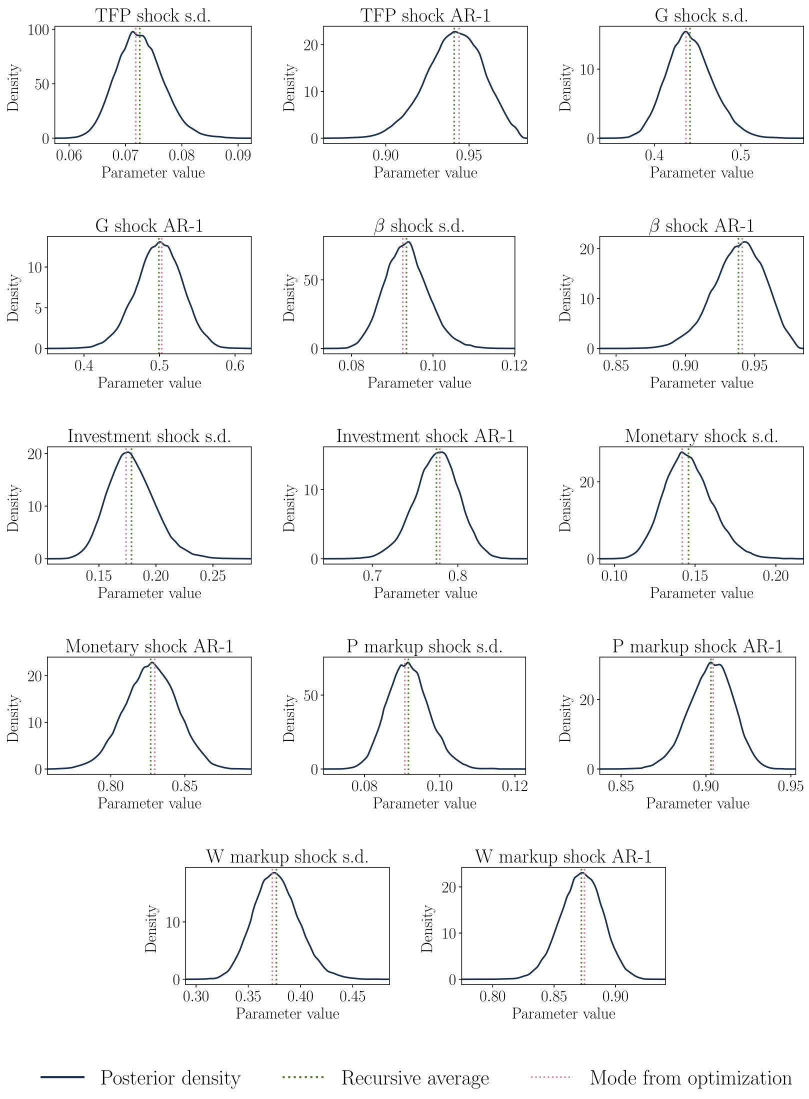


FIGURE F.10.—Posterior distributions for the RWMH estimation of the two-asset HANK model with shocks.

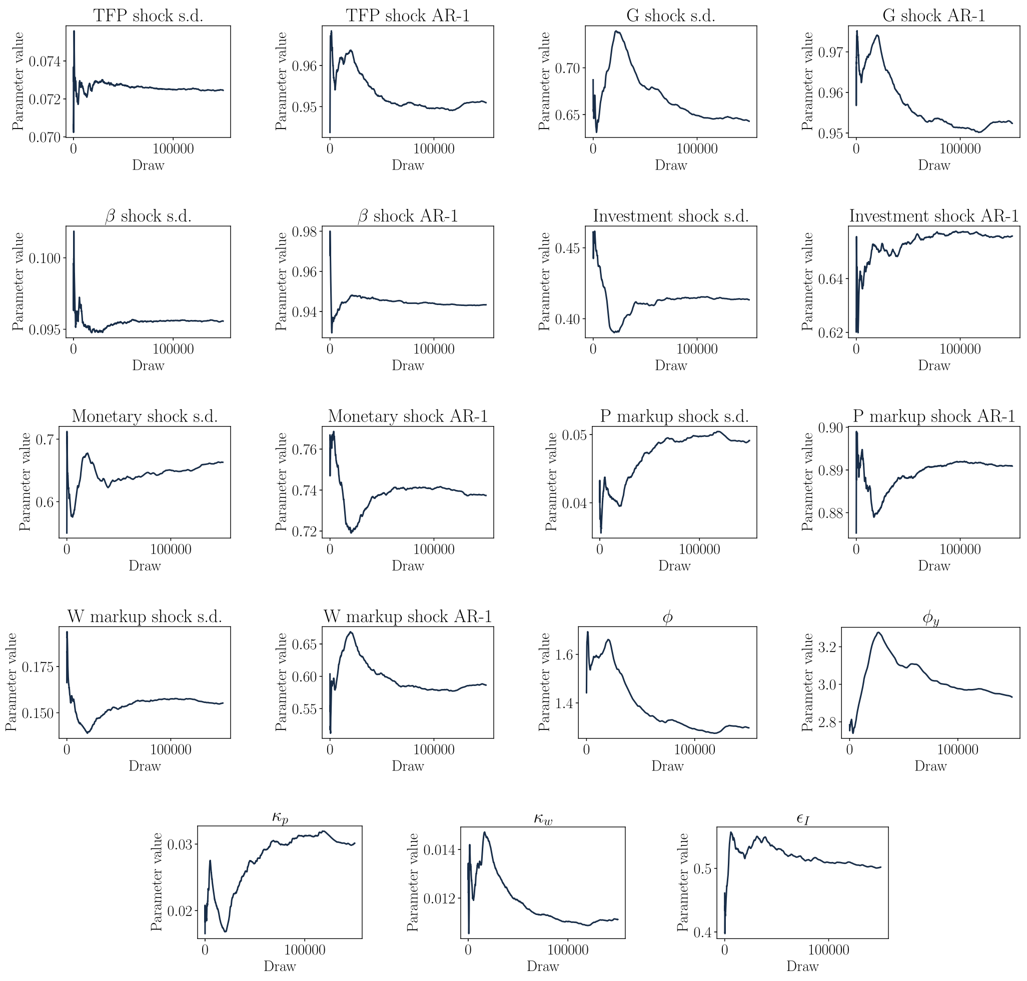


FIGURE F.11.—Recursive means for the RWMH estimation of the two-asset HANK model with shocks and parameters.

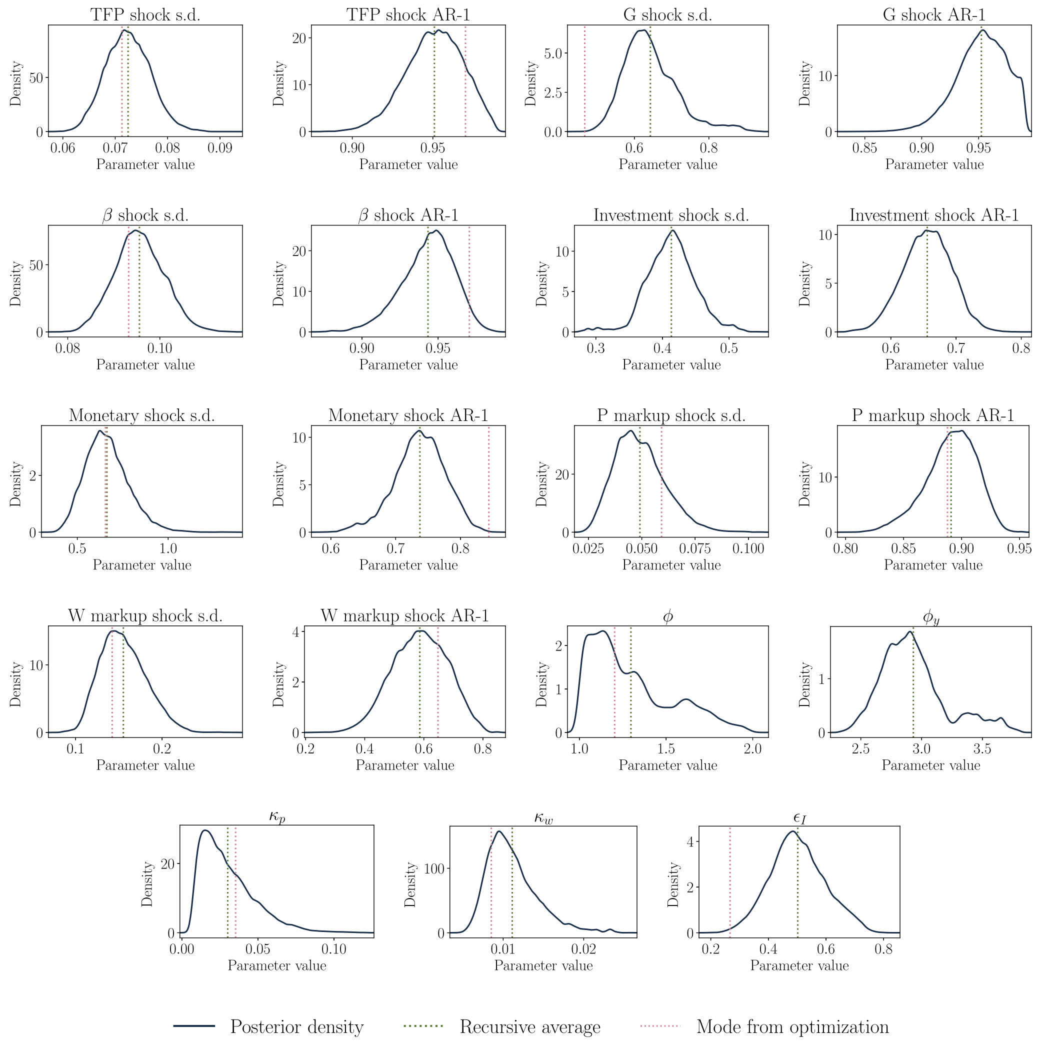


FIGURE F.12.—Posterior distributions for the RWMH estimation of the two-asset HANK model with shocks and parameters.

REFERENCES

ACHDOU, Y., J. HAN, J.-M. LASRY, P.-L. LIONS, AND B. MOLL (2020): "Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach," *Review of Economic Studies* (Forthcoming). https://doi.org/10.1093/restud/rdab002. [27]

AHN, S., G. KAPLAN, B. MOLL, T. WINBERRY, AND C. WOLF (2018): "When Inequality Matters for Macro and Macro Matters for Inequality," *NBER Macroeconomics Annual*, 32 (1), 1–75. [12]

ALGAN, Y., O. ALLAIS, W. J. DEN HAAN, AND P. RENDAHL (2014): "Chapter 6—Solving and Simulating Models With Heterogeneous Agents and Aggregate Uncertainty," in *Handbook of Computational Economics*, Vol. 3, ed. by K. Schmedders and K. L. Judd. Elsevier, 277–324. [4]

BARDÓCZY, B. (2020): "Spousal Insurance and the Amplification of Business Cycles," Job Market Paper, Northwestern University. [2]

CARROLL, C. D. (2006): "The Method of Endogenous Gridpoints for Solving Dynamic Stochastic Optimization Problems," *Economics Letters*, 91 (3), 312–320. [23]

GRIEWANK, A., AND A. WALTHER (2008): *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation* (Second Ed.) SIAM. [16]

HANSEN, L. PETER, AND T. J. SARGENT (1981): "Exact Linear Rational Expectations Models: Specification and Estimation," Federal Reserve Bank of Minneapolis Staff Report. [31]

HERBST, E. P., AND F. SCHORFHEIDE (2015): *Bayesian Estimation of DSGE Models*. Princeton University Press. [31,32]

ISKHAKOV, F., T. H. JØRGENSEN, J. RUST, AND B. SCHJERNING (2017): "The Endogenous Grid Method for Discrete-Continuous Dynamic Choice Models With (or Without) Taste Shocks," *Quantitative Economics*, 8 (2), 317–365. [2]

JUILLARD, M. (1996): "Dynare: A Program for the Resolution and Simulation of Dynamic Models With Forward Variables Through the Use of a Relaxation Algorithm," CEPREMAP Working Paper no 9602. [27]

KAPLAN, G., B. MOLL, AND G. L. VIOLANTE (2018): "Monetary Policy According to HANK," *American Economic Review*, 108 (3), 697–743. [8]

MCKAY, A., E. NAKAMURA, AND J. STEINSSON (2016): "The Power of Forward Guidance Revisited," *American Economic Review*, 106 (10), 3133–3158. [7]

SMETS, F., AND R. WOUTERS (2007): "Shocks and Frictions in US Business Cycles: A Bayesian DSGE Approach," *American Economic Review*, 97 (3), 586–606. [31,32]

WHITEMAN, C. (1983): *Linear Rational Expectations Models: A User's Guide*. Univ. of Minnesota Press. [26]

Co-editor Giovanni L. Violante handled this manuscript.

Manuscript received 28 June, 2019; final version accepted 27 March, 2021; available online 19 April, 2021.