Technical Appendix of "Estimating HANK: Macro Time Series and Micro Moments"*

Reca Sarfati†

December 1, 2019

Abstract

We show how to use sequential Monte Carlo methods, which can be parallelized, to estimate a heterogeneous agent New Keynesian (HANK) model featuring both nominal and real rigidities. We demonstrate how the posterior distribution may be specified as the product of the standard macro time series likelihood and a prior enforcing several steady state distributional moments, including the average marginal propensity to consume and fraction of agents with zero liquid wealth. We ask whether there exists a tension between fitting macroeconomic time series and distributional moments, ultimately finding there is none. For instance, even after relaxing the prior, the posterior based solely on macro time series features a marginal propensity to consume well below one, broadly in line with existing microeconomic evidence.

This technical appendix is split into two sections. Appendix A discussions the estimation approach. Appendix B details the augmentation of the time series likelihood function so as to fit both micro and macro data.

Keywords: Heterogeneous agent New Keynesian models, adaptive algorithms, Bayesian inference, online estimation, sequential Monte Carlo methods

JEL Codes: C11, C32, C53, E32, E37, E52

*This is the technical appendix of the working paper, "Estimating HANK: Macro Time Series and Micro Moments," joint with Sushant Acharya, Michael Cai, Marco Del Negro, Keshav Dogra, and Ethan Matlin. While part of a joint work, I have written this appendix solely myself.

†Federal Reserve Bank of New York (email: rebecca.sarfati@ny.frb.org). The views expressed in this paper are those of the authors and do not necessarily reflect the position of the Federal Reserve Bank of New York nor the Federal Reserve System.

A Sequential Monte Carlo Algorithms

Computing moments of the posterior distribution of a dynamic stochastic general equilibrium (DSGE) model is often analytically intractable, necessitating the use of simulation techniques. To date, the prevailing tool for Bayesian estimation has been the random walk Metropolis-Hastings (RWMH) algorithm. However, the complexity of many modern DSGE models renders RWMH ineffective, for three primary reasons. First, RWMH can be impractically slow; when sequences of model parameter draws exhibit high serial correlation, a larger number of draws is demanded to ensure proper convergence to the posterior, greatly extending overall runtime. By construction, RWMH cannot be executed in parallel—as the transition process is Markov, each iteration must occur sequentially. Secondly, RWMH is susceptible to getting stuck at local modes, in turn failing to explore the full posterior distribution by the termination of the algorithm. Finally, RWMH is very inflexible in its execution: following new realizations of data or minor modifications to one's model, an estimation must be relaunched from scratch. This rigidity hampers both the model development process and practical deployment of DSGE models for regular forecasting and policy analysis.

Sequential Monte Carlo (SMC) methods are adaptable, parallelizable tools to conduct Bayesian inference of posterior distributions, solving for aforementioned limitations of the RWMH algorithm. Heavily distributing the computational load across computing cores, SMC methods can produce more accurate estimates of posterior moments in a fraction of the time. Further, SMC methods can easily be modified so as to enable "online" estimation, as discussed in another paper of ours, Cai et al. (2019). Herbst and Schorfheide (2014) and Cai et al. (2019) have also shown SMC methods to be robust to multimodality.

Appendix A begins with a formal exposition of the estimation problem (Section A.1), reviews the standard SMC algorithm (Section A.2), describes our generalized tempering approach as put forth in Cai et al. (2019) (Section A.3), and discusses the adaptive tuning of parameters (Section A.4). Finally, Section A.5 discusses the use of Chandrasekhar recursions for fast computation of the likelihood.

A.1 Problem Statement

Given data Y, prior $p(\theta)$, and the likelihood $p(Y|\theta)$, we wish to compute moments of the posterior distribution $p(\theta|Y)$ of static parameter θ, with support Θ. The posterior density is given by

$$p(\theta|Y) = \frac{p(Y|\theta)p(\theta)}{p(Y)}, \quad \text{where } p(Y) = \int p(Y|\theta)p(\theta)d\theta$$

The function $p(Y|\theta)p(\theta)$ can be evaluated analytically for linearized DSGE models with Gaussian innovations and regular prior densities. However, the normalization constant $\int p(Y|\theta)p(\theta)d\theta$ lacks a closed-form expression and can be very costly to compute. Numerical integration techniques, for instance, suffer from a "curse of dimensionality," whereby the requisite computation time to proffer a desired degree of precision increases exponentially with the dimension of the problem. We thus aim to approximate the posterior expectations of θ and functions $h(\theta)$ with Monte Carlo simulation methods, whose convergence rates are independent of the dimensionality of θ.

A.2 Overview of SMC Algorithm

This section provides an overview of the structure of the SMC algorithm. Those seeking even greater detail may refer to Chopin (2002), Del Moral et al. (2012), Creal (2012), and Herbst and Schorfheide (2014).

SMC conjoins pieces of importance sampling and modern Markov Chain Monte Carlo methods. Recalling the construction of classic importance sampling, let $\pi(\cdot)$ be an arbitrary density to estimate. We seek to approximate $\pi(\cdot)$ with another density $g(\theta)$ that is easier to sample from. Letting $\theta^i \overset{iid}{\sim} g(\theta)$ for $i = 1, ..., N$, and Z be a normalization constant, we have the identity

$$\mathbb{E}_\pi[h(\theta)] = \int h(\theta)\pi(\theta)d\theta = \frac{1}{Z}\int_\Theta h(\theta)w(\theta)g(\theta)d\theta, \quad \text{where } w(\theta) = \frac{f(\theta)}{g(\theta)},\ \pi(\theta) = \frac{f(\theta)}{Z}$$

Foundational to the motivation of SMC is the Monte Carlo average

$$\overline{h} = \sum_{i=1}^{N} \tilde{W}^i h(\theta^i) \quad \xrightarrow{a.s.} \quad \mathbb{E}_\pi[h(\theta)] \qquad \text{as } N \to \infty$$

where the $\tilde{W}^i$ are normalized importance weights of parameter draws θ^i

$$\tilde{W}^i = \frac{w(\theta^i)}{\sum_{j=1}^{N} w(\theta^j)}$$

Critically, the accuracy of the approximation is dependent on the "closeness" of the function $g(\cdot)$ to $f(\cdot)$. Yet, finding such a $g(\cdot)$ is typically very difficult. SMC methods thus aim to construct a sequence of bridge distributions $\{\pi_n(\theta)\}_{n=0}^{N_\phi}$ converging to the target posterior: $\pi_{N_\phi}(\theta) = \pi(\theta)$. In the standard SMC algorithm, the bridge posterior distributions $\pi_n(\theta)$ are given by the stage-n likelihood functions

$$\pi_n(\theta) = \frac{p_n(Y|\theta)p(\theta)}{\int p_n(Y|\theta)p(\theta)d\theta}$$

SMC methods represent each intermediate posterior density $\pi_n(\theta)$ by a swarm of particles $\{\theta_n^i, W_n^i\}_{n=0}^{N}$, such that the Monte Carlo average

$$\overline{h}_{n,N} = \frac{1}{N}\sum_{i=1}^{N} W_n^i h(\theta^i) \xrightarrow{a.s.} \mathbb{E}_{\pi_n}[h(\theta_n)] \quad \text{as } N \to \infty, \quad \text{for each } n = 0, ..., N_\phi.$$

The SMC algorithm iterates from $n = 0$ to N_ϕ. At $n = 0$, we initialize the particle cloud $\{\theta_0^i, 1\}_{i=1}^{N}$ with values θ_0^i drawn *iid* from the prior density $p(\theta)$, and weights W_0^i set to 1. There are three steps within each stage n of the SMC algorithm: (1) *correction*: reweighting stage $n-1$ particles to reflect the new density of stage n, (2) *selection*: resampling particles by weight so as to avoid degeneracy, and (3) *mutation*: propagating particles forward using a Markov transition kernel, so as to update particle values in reflection of the stage-n bridge density. When $n = N_\phi$, the likelihood will have converged to its true value,

$p_{N_\phi}(Y|\theta) = p(Y|\theta)$. The path of the likelihood sequence depends on the tempering strategy; we utilize an adaptive tempering schedule for this paper, as discussed in Section A.3. A generic implementation of the algorithm is detailed below.

Algorithm 1: Basic SMC Algorithm

---

1. *Initialization.* ($\phi_0 = 0$). **for** $i = 1, ..., N$ **do**

   Draw initial particles from the prior: $\theta_1^i \overset{iid}{\sim} p(\theta)$

   Initialize weights: $W_1^i \leftarrow 1$

   end

2. *Recursion.* **for** $n = 1, ..., N_\phi$ **do**

   (a) *Correction.* Reweight the particles from stage $n-1$ by defining the incremental weights

   $$\tilde{w}_n^i = \frac{p_n(Y|\theta_{n-1}^i)}{p_{n-1}(Y|\theta_{n-1}^i)}$$

   and normalized weights

   $$\tilde{W}_n^i = \frac{\tilde{w}_n^i W_{n-1}^i}{\frac{1}{N}\sum_{i=1}^N \tilde{w}_n^i W_{n-1}^i}, \quad i = 1, ..., N$$

   (b) *Selection (Optional).* Resample swarm of particles $\{\theta_{n-1}^i, \tilde{W}_n^i\}_{i=1}^N$ by normalized weights. Denote resampled swarm by $\{\hat{\theta}_{n-1}^i, W_n^i\}_{i=1}^N$, where $W_n^i = 1$ for all i.

   (c) *Mutation.* Propagate particles $\{\hat{\theta}_{n-1}^i, W_n^i\}_{i=1}^N$ via N_{MH} steps of a Metropolis-Hastings algorithm with transition density $\theta_n^i \sim K_n(\theta_n|\hat{\theta}_n^i; \zeta_n)$ and stationary distribution $\pi_n(\theta)$. An approximation of $\mathbb{E}_{\pi_n}[h(\theta)]$ is given by

   $$\overline{h}_{n,N} = \frac{1}{N}\sum_{i=1}^N h(\theta_n^i) W_n^i$$

   end

3. For $n = N_\phi$ ($\phi_{N_\phi} = 1$), the importance sampling approximation of $\mathbb{E}_\pi[h(\theta)]$ is given by:

   $$\overline{h}_{N_\phi,N} = \sum_{i=1}^N h(\theta_{N_\phi}^i) W_{N_\phi}^i$$

One will notice the mutation step may be performed in parallel across particles. The mutation step is also the most computationally intensive; the likelihood function must be

evaluated at each Metropolis-Hastings step. The number of ϕ-stages N_ϕ and number of Metropolis-Hastings steps N_{MH} are tuning parameters to be calibrated. The vector ζ_n comprises the tuning parameters of the Metropolis-Hastings algorithm, discussed in Section A.4.

A.2.1 Selection

The equalizing of particle weights solves for issues of degeneracy, increasing the accuracy of importance sampling approximations in ensuing stages. However, resampling particles comes at the cost of adding noise to the Monte Carlo approximation, and thus it is inadvisable to resample unnecessarily. To determine the stages at which to resample, it is common to use a threshold rule based on the variance of the particle weights. As in Herbst and Schorfheide (2014), we compute the "effective sample size"

$$\widehat{ESS}_n = N/\left(\frac{1}{N}\sum_{i=1}^{N}(\tilde{W}_n^i)^2\right)$$

and pick a threshold $\underline{N}$, such that we resample during any period where $\widehat{ESS}_n < \underline{N}$.

A.2.2 Mutation

The mutation step enables particles to adjust to the new posterior density by taking Metropolis-Hastings steps N_{MH} from their present values at the start of the period. Were particles never to mutate, they would remain fixed in value once drawn from the prior during initialization, reducing the SMC algorithm to classic importance sampling. As noted earlier, it is very difficult to choose a "close" function to the true posterior; as such, the prior is generally a poor distribution for approximating the posterior. The critical distinction in performance between SMC and vanilla importance sampling is thus the mutation of particles.

Several tuning parameters and modifications may increase the probability of accepting a proposed mutation, such as the number of steps taken, N_{MH}, as well as grouping the parameters θ into subsets and allowing subsets to mutate, blockwise. We modify the mutation

step as in Herbst and Schorfheide (2014) to make the proposal distribution adaptive, such that the mean and covariance matrix during the nth iteration are functions of the particles from the preceding iteration, $n-1$. Further, as in Kohn et al. (2010), we utilize a mixture density for the proposal distribution. The amalgamation of these adjustments is explicated in the following pseudocode.

Algorithm 2: Mutation Step *(Step 2. (c) of SMC Algorithm)*

---

1. *Parameter blocking.*
   · Draw $r_i \sim U[0,1]$ corresponding to each parameter θ^i, $n = 1, ..., N$.
   · Sort θ^i by r_i. Let the b-th block of parameters, denoted $\theta_{n,b}$, consist of the sorted parameters $(b-1) \cdot N_{blocks}, ..., b \cdot N_{blocks}$.
   · Define θ_b^* and Σ_b^* to be the partitions of θ^* and Σ^* corresponding to parameter block $\theta_{n,b}$.

2. *Mutation steps.* **for** $n = 1, ..., N_{MH}$ **do**
   for $b = 1, ..., N_b$ **do**

   (a) Denote $\theta^i_{n,b,m}$ as parameter values for $\theta^i_{n,b}$ on the m-th MH step. Let $\theta^i_{n,b,0} \leftarrow \theta^i_{n-1,b}$

   (b) Generate proposal draw ϑ_b from the mixture distribution

$$\vartheta_b \mid (\theta^i_{n,b,m-1}, \theta^i_{n,-b,m}, \theta_b^*, \Sigma_b^*)$$
$$\sim \alpha N\left(\theta^i_{n,b,m-1}, c^2\Sigma_b^*\right) + \frac{1-\alpha}{2} N\left(\theta^i_{n,b,m-1}, c^2 diag(\Sigma_b^*)\right) + \frac{1-\alpha}{2} N\left(\theta_b^*, c^2\Sigma_b^*\right)$$

   denoting the density of the mixture proposal by $q(\vartheta_b | \theta^i_{n,b,m-1}, \theta^i_{n,-b,m}, \theta_b^*, \Sigma_b^*)$

   (c) Define the acceptance probability

$$\alpha = \min\left\{1, \frac{p^{\phi_n}(Y \mid \vartheta_b, \theta^i_{n,-b,m}) p(\vartheta_b, \theta^i_{n,-b,m}) / q(\vartheta_b \mid \theta^i_{n,b,m-1}, \theta^i_{n,-b,m}, \theta_b^*, \Sigma_b^*)}{p^{\phi_n}(Y \mid \theta^i_{n,b,m-1}, \theta^i_{n,-b,m}) p(\theta^i_{n,b,m-1}, \theta^i_{n,-b,m}) / q(\theta^i_{n,b,m-1} \mid \vartheta_b, \theta^i_{n,-b,m}, \theta_b^*, \Sigma_b^*)}\right\}$$

   (d) Draw $r \sim U[0,1]$.

   if $r < \alpha$ **then**
   Accept proposed mutation of block: $\theta^i_{n,b,m} \leftarrow \vartheta_b$
   else
   Reject proposed mutation of block: $\theta^i_{n,b,m} \leftarrow \theta^i_{n,b,m-1}$
   end
   end
   end

3. Set values to those of last MH step: $\theta^i_{n,b} \leftarrow \theta_{n,b,M}$, for $b = 1, ..., N_{blocks}$.

A.3 Tempering Schedules: Likelihood, Data, and Generalized

We have so far remained silent on how to construct the sequence of bridge posterior distributions $\{\pi_n\}_{n=0}^{N_\phi}$. This section discusses two methods of tempering—likelihood and data—before describing their unification in the "generalized" tempering approach, as put forward in another paper of ours, Cai et al. (2019).

A.3.1 Likelihood Tempering

In likelihood tempering, the parameter ϕ_n determines the weight of the likelihood function in the evaluation of the bridge posterior

$$\pi_n(\theta) \propto p(Y|\theta)^{\phi_n} p(\theta)$$

where ϕ_n follows a monotonically increasing path from 0 to 1 (i.e. $\phi_0 = 0$ and $\phi_{N_\phi} = 1$). The question of tempering is thus how to tune the path of ϕ_n. An advantage of likelihood tempering is that one can design bridge distributions to be arbitrarily close to one another, which may be useful so as to avoid rapid decay of the effective sample size. For instance, it is advisable to introduce the likelihood function slowly during initial stages, so as to avoid many particles being assigned near-zero weight during the correction step, since the likelihood may have a very different shape from the prior. The trade-off of having too many stages, N_ϕ, is that each added stage requires additional (costly) evaluations of the likelihood function.

The ϕ-schedule may be chosen and fixed *a priori*, or determined adaptively over the course of the SMC algorithm. Herbst and Schorfheide (2014) propose the following tempering schedule

$$\phi_n = \left(\frac{n-1}{N_\phi - 1}\right)^\lambda$$

where λ is a parameter to be calibrated (the tuning of which is discussed in their paper). Larger values of λ will keep successive bridge distributions closer when n is small and farther apart when n is close to N_ϕ. The value $\lambda = 2$ is recommended for DSGE model applications.

A.3.2 Data Tempering

Data tempering, by contrast, gradually adds sets of observations to the likelihood function. That is,

$$p_n(Y|\theta) = p(y_{1:\lfloor \phi_n T \rfloor} \mid \theta)$$

where the function $\lfloor \phi_n T \rfloor$ returns the largest time step less than or equal to $\phi_n T$, and ϕ_n again follows a path from 0 to 1. Data tempering has intuitive applications to time series contexts, but the proximity of successive bridge distributions is limited by the coarseness of the data, restricting the extent to which one can temper to avoid particle degeneracy.

A.3.3 Generalized Tempering

"Generalized" tempering encompasses both data and likelihood tempering. Consider abstractly drawing from the posterior

$$\tilde{\pi}(\theta) \propto \tilde{p}(\tilde{Y} \mid \theta) p(\theta)$$

where $\tilde{\pi}(\theta)$ is permitted to vary from the true posterior π either by a different sample of *data* ($\tilde{Y}$ versus Y), a distinct *model* ($\tilde{p}(Y|\theta)$ versus $p(Y|\theta)$), or both. The likelihood function at stage n may be expressed as

$$p_n(Y|\theta) = [p(Y|\theta)]^{\phi_n} [\tilde{p}(\tilde{Y}|\theta)]^{1-\phi_n}$$

To temper the likelihood alone, one may set $\tilde{p}(\cdot) = 1$. To temper the data alone, one may set $\tilde{p}(\cdot) = p(\cdot)$, $Y = y_{1:\lfloor \phi_m T \rfloor}$, and $\tilde{Y} = y_{1:\lfloor \phi_{m-1} T \rfloor}$, where the schedule $\{\phi_m\}_{m=1}^{N_{\phi_T}}$ may be specified as desired. The generalized tempering approach allows for the additional flexibility of "smoothing" between data realizations. This is particularly useful during periods for which the inclusion of the data sample $y_{1:\lfloor \phi_{m-1} T \rfloor} : y_{1:\lfloor \phi_m T \rfloor}$ has a significant effect on the likelihood, such as during the onset of a recession.

A.4 Adaptive Choice of Tuning Parameters

The transition kernel in Algorithm 1 consists of the following parameters ζ_n to be calibrated

$$\zeta_n = \{\hat{c}_n, \theta_n^*, \Sigma_n^*\}$$

As in Herbst and Schorfheide (2014), we choose the tuning parameters $\hat{c}_n, \theta_n^*, \Sigma_n^*$ adaptively. The parameter $\hat{c}_n$ determines the scaling factor of the covariance matrix, and is selected so as to yield an acceptance rate of approximately 25%, as discussed in Geweke and Durham (2019). The selection of these parameters is detailed below.

Algorithm 3: Adaptive Particle Mutation *(Before Step 2. (c) of SMC Algorithm)*

---

1. Compute importance sampling approximations of $\mathbb{E}_{\pi_n}[\theta]$ and $\mathbb{V}_{\pi_n}[\theta]$ to specify θ_n^* and Σ_n^*:

$$\theta_n^* = \sum_{n=1}^{N} \tilde{W}_n^i \cdot \theta_{n-1}^i, \qquad \Sigma_n^* = \sum_{n=1}^{N} \tilde{W}_n^i \cdot (\theta_{n-1}^i - \theta_n^*) \cdot (\theta_{n-1}^i - \theta_n^*)^T$$

2. Compute the average empirical rejection rates $\hat{R}_{n-1}(\hat{\zeta}_{n-1})$ of the mutation step in iteration $n-1$.

3. Compute the scaling factor $\hat{c}_n$:

$$\hat{c}_n = \hat{c}_{n-1} f(\hat{R}_{n-1}(\hat{\zeta}_{n-1}))$$

   where, as in Herbst and Schorfheide (2014), Algorithm 10, we define $f(\cdot)$ as

$$f(x) = 0.95 + 0.10 \frac{e^{16(x-0.25)}}{1 + e^{16(x-0.25)}}$$

4. **return:** $\hat{\zeta}_n = \{\hat{c}_n, \theta_n^*, \Sigma_n^*\}$

The final parameter to be chosen is ϕ_n. As in Cai et al. (2019), we may choose ϕ_n adaptively, so as to target a particular rate of degeneration of the effective sample size.

Pursuant to this, we define:

$$w^i(\phi) = [p(Y|\theta^i_{n-1})]^{\phi-\phi_{n-1}}, \quad W^i(\phi) = \frac{w^i(\phi)W^i_{n-1}}{\frac{1}{N}\sum_{i=1}^N w^i(\phi)W^i_{n-1}}, \quad \widehat{ESS}(\phi) = N/\left(\frac{1}{N}\sum_{i=1}^N(\tilde{W}^i_n(\phi))^2\right)$$

We choose the parameter ϕ as follows:

$$f(\phi) = \widehat{ESS}(\phi) - \rho\widehat{ESS}_{n-1} = 0$$

and choose ρ to capture the permissible rate of deterioration in effective sample size. An optimized algorithm for efficient root-finding is given below.

Algorithm 4: Adaptive Tempering Schedule *(Before Step 2.(a) of SMC Algorithm)*

---

1. *Initialization.* Initialize variables: $j = 2, n = 2, \phi_1 = 0, \tilde{\phi} = 0$
2. Construct vector of incremental upper bounds. $\vec{\hat{\phi}} = \{\hat{\phi}_1, ..., \hat{\phi}_{N-1}, \hat{\phi}_N\}$
3. *Root finding.* **while** $\phi_n < 1$ **do**

   $f(\phi) = ESS(\phi) - \rho ESS_{n-1}$

   while $f(\tilde{\phi}) \geq 0$ *and* $j \leq N$ **do**

   $\tilde{\phi} = \hat{\phi}_j$

   $j = j + 1$

   end

   if $f(\tilde{\phi}) < 0$ **then**

   $\phi_n = root(f, [\phi_{n-1}, \tilde{\phi}])$

   else

   $\phi_n = 1$

   end

   $n = n + 1$

   end
4. **return:** ϕ_n

A.5 Chandrasekhar Recursions for Fast Likelihood Computation

When estimating linearized DSGE models with Gaussian innovations, the evaluation of the Kalman filter likelihood typically dominates other parts of the algorithm in contribution to total runtime. Likelihood evaluations of HANK models are especially costly; the standard Kalman filter involves many operations with $k \times k$ matrices, where k is the number of states. Consequently, as the number of states grows, so too does the runtime of the likelihood. This is concerning for our purposes; in the state-space representation of our HANK model, the state vector includes the full discretized wealth distribution and individual decision variables.

The Chandrasekhar recursions, outlined in Morf (1974), reconfigure the algorithm to instead operate on $n \times n$ matrices, where n is the number of observables. Abstracting away the cost of the fixed set of operations C shared by both algorithms, the original Kalman filter has a runtime on the order of $O(k^4 + k^3 n + C)$, while the Chandrasekhar recursions run in $O(n^3 + kn^2 + C)$. DSGE models, and HANK models in particular, are especially suited for the use of these recursions, as the number of states tends to far exceed the number of observables ($k >> n$).

This section will walk through details of implementation. For a more thorough treatment and proofs of validity, consult Morf (1974). For empirical performance, Herbst (2015) compares the runtime of both algorithms using four DSGE models of varying scale.

We take as input the state-space representation of our model,

$$s_t = T s_{t-1} + R\epsilon_t, \quad \epsilon_t \sim N(0, Q)$$
$$y_t = Z s_t + D + \eta_t, \quad \eta_t \sim N(0, H)$$

where s_t is the k-length vector of states, T is the $k \times k$ transition matrix, ϵ_t is the vector of structural shocks, and R is the shock transmission matrix. In the measurement equation, y_t is the length-n vector of observables, with measurement error vector η_t such that $\mathbb{E}[\epsilon_t \eta_t'] = 0$. To use the Chandrasekhar recursions, we further assume (1) the system matrices

(T, R, Q, Z, D, H) are time-invariant, and (2) the process $\{s_t\}_{t=1}^T$ is stationary. This means the algorithm does not allow missing observations, as the system matrices are not permitted to change in size over the time sample. Our goal is to compute the log-likelihood $\mathcal{L}(y_{1:T}|\theta)$.

Algorithm 5: Chandrasekhar Recursions

Inputs: Data: $y_{1:T}$, Transition: T, R, Q, Measurement: Z, D, H

1. *Initialization.*
   (a) Solve the discrete Lyapunov equation $\bar{P} = T\bar{P}T' + RQR'$ for the unconditional variance, $\bar{P}$.
   (b) $\hat{s}_{1|0} \leftarrow \mathbf{0}$
   (c) $K_1 \leftarrow T\bar{P}Z', \quad F_1 \leftarrow Z\bar{P}Z' + H$
   (d) $W_1 \leftarrow K_1, \qquad M_1 \leftarrow -F_1^{-1}$
2. *Iteration.* **for** $t = 1, ..., T$ **do**
   (a) Compute forecast error ν_t:
   $$\nu_t = y_t - D - Z\hat{s}_{t|t-1}$$
   (b) Compute intermediate likelihood calculation of ν_t with respect to $N(\mathbf{0}, F_t)$:
   $$L_t = n\ln(2\pi) + \ln|F_t| + \nu_t' F_t^{-1}\nu_t$$
   (c) Compute $\hat{s}_{t+1|t}$:
   $$\hat{s}_{t+1|t} = T\hat{s}_{t|t-1} + K_t F_t^{-1}\nu_t$$
   (d) Update forecast error variance:
   $$F_{t+1} = F_t + ZW_t M_t W_t' Z'$$
   (e) Update K_{t+1} (note: Kalman gain given by $K_{t+1}F_{t+1}^{-1}$):
   $$K_{t+1} = K_t + TW_t M_t W_t' Z'$$
   (f) Compute W_{t+1} and M_{t+1}:
   $$W_{t+1} = (T - K_t F_t^{-1} Z)W_t$$
   $$M_{t+1} = M_t + M_t W_t' Z' F_t^{-1} Z W_t M_t$$

   end
3. **return:** $\mathcal{L}(y_{1:T}|\theta) = -\frac{1}{2}\sum_{t=1}^T L_t$

B Fitting Micro and Macro Data

B.1 Augmenting Time Series Likelihood

We have two datasets against which we hope to fit our HANK model: macro time series data Y^m and micro distributional data Y^d. The macro time series data ($Y^m = y^m_{1:T}$) consists of the set of observables used in standard DSGE model estimation. We include the same observables as Smets and Wouters (2007) for clarity of message. The likelihood $p(y^m_{1:T}|\theta)$ is similarly the standard macro time series likelihood, which may be computed using the Kalman filter or Chandrasekhar recursions (provided there are no missing observations).

We also want the model to fit micro distributional data, Y^d. We target a set of moments $\bar{m}(Y^d)$ and affiliated standard deviations σ_d from the steady state cross-sectional income and liquid wealth distributions. The moments $\bar{m}(Y^d)$ are chosen from Kaplan et al. (2018), and include the fraction of zero-liquid wealth agents, average marginal propensity to consume, and cross-sectional variance in income.

To date, there have been several approaches to combining micro and macro data in the estimation of HANK models (Chang et al. (2018); Liu and Plagborg-Møller (2019), etc.). Our approach differs in that we are not seeking to fit repeated cross-sections of micro data, but rather only a set of moments from the steady state distribution. Taking the approach of Del Negro and Schorfheide (2008), we construe the micro moments as providing a form of *a priori* knowledge about a subset of the parameters, and define the posterior distribution of θ to be

$$p(\theta|Y^m, Y^d) \propto \underbrace{p(y^m_{1:T}|\theta)}_{\text{likelihood}} \cdot \underbrace{p(Y^d|\theta)p(\theta)}_{\text{prior}}$$

The prior $p(\theta)$ is broadly specified to be the same as Smets and Wouters (2007) for parameters which do not affect the steady state. The "penalty function" $p(Y^d|\theta)$ is constructed as follows

$$\log p(Y^d|\theta) = -\frac{1}{2}(\log \bar{m}(\theta) - \log \bar{m}(Y^d))' \Sigma_d^{-1} (\log \bar{m}(\theta) - \log \bar{m}(Y^d))$$

where $\bar{m}(\theta)$ are the moments implied by the parameters of our model, and Σ_d is a sparse (positive definite) matrix with the vector of empirical standard deviations σ_d^2 along the diagonal.

B.2 Assessing Trade-offs in Model Fit

We wish to construct a measure of the "trade-off" of fitting macro and micro data. Using our earlier formulation, we may introduce scaling parameters Υ^m and Υ^d, which determine the relative weights of the standard macro time series likelihood $p(y_{1:T}^m|\theta)$ and penalty term $p(Y^d|\theta)$ in the construction of the posterior, respectively:

$$p(\theta|Y^m, Y^d) \propto \underbrace{p(y_{1:T}^m|\theta)^{\Upsilon^m}}_{\text{likelihood}} \cdot \underbrace{p(Y^d|\theta)^{\Upsilon^d} p(\theta)}_{\text{prior}}$$

When $\Upsilon^d = 0$, we return to the standard statement of the posterior distribution, considering only macro data. By contrast, as $\Upsilon^d \to \infty$, the model is forced to meet micro targets $\bar{m}(Y^d)$. Intuitively, this is equivalent to specifying that $\Sigma_d \to 0$.

Alternatively interpreted, $p(Y^d|\theta)$ serves an expression for the "likelihood" as invoked in micro studies, yielding the additional interpretation of our expression as being one of a "composite likelihood," so discussed by Canova and Matthes (2018) with regards to the estimation of DSGE models

$$p(\theta|Y^m, Y^d) \propto \underbrace{p(y_{1:T}^m|\theta)^{\Upsilon^m} \cdot p(Y^d|\theta)^{\Upsilon^d}}_{\text{composite likelihood}} \cdot \underbrace{p(\theta)}_{\text{prior}}$$

With this formulation, we have the flexibility of asking whether, for lower specifications of Υ^d, our estimated model still returns reasonable values for micro moments of interest. We estimate our model using SMC, just as discussed in Appendix A, with the minor modification that we utilize the "composite likelihood" in the place of the standard macro time series likelihood.

References

Cai, Michael, Marco Del Negro, Edward Herbst, Ethan Matlin, Reca Sarfati, and Frank Schorfheide, "Online Estimation of DSGE Models," Staff Reports 893, Federal Reserve Bank of New York 2019.

Canova, Fabio and Christian Matthes, "A Composite Likelihood Approach for Dynamical Structural Models," Staff Reports 18-12, Federal Reserve Bank of Richmond 2018.

Chang, Minsu, Xiaohong Chen, and Frank Schorfheide, "Heterogeneity and Aggregate Fluctuations," 2018.

Chopin, Nicolas, "A Sequential Particle Filter for Static Models," *Biometrika*, 2002, *89* (3), 539–551.

Creal, Drew, "A Survey of Sequential Monte Carlo Methods for Economics and Finance," *Econometric Reviews*, 2012, *31* (3), 245–296.

Del Negro, Marco and Frank Schorfheide, "Forming Priors for DSGE Models (and How it Affects the Assessment of Nominal Rigidities)," *Journal of Monetary Economics*, 2008, *55* (7), 1191–1208.

Geweke, John and Garland Durham, "Sequentially adaptive Bayesian learning algorithms for inference and optimization," *Journal of Econometrics*, 2019, *210* (1), 4–25.

Herbst, Edward, "Using the "Chandrasekhar Recursions" for Likelihood Evaluation of DSGE Models," *Computational Economics*, 2015, *45* (4), 693–705.

_ **and Frank Schorfheide**, "Sequential Monte Carlo Sampling for DSGE Models," *Journal of Applied Econometrics*, 2014, *29* (7), 1073–1098.

Kaplan, Greg, Benjamin Moll, and Giovanni L Violante, "Monetary policy according to HANK," *American Economic Review*, 2018, *108* (3), 697–743.

Kohn, Robert, Paolo Giordani, and Ingvar Strid, "Adaptive Hybrid Metropolis-Hastings Samplers for DSGE Models," *Riksbank Manuscript*, 2010.

Liu, Laura and Mikkel Plagborg-Møller, "Full-Information Estimation of Heterogeneous Agent Models Using Macro and Micro Data," 2019.

Moral, Pierre Del, Arnaud Doucet, and Ajay Jasra, "An Adaptive Sequential Monte Carlo Method for Approximate Bayesian Computation," *Statistical Computing*, 2012, *22*, 1009–1020.

Morf, Martin, "Fast Algorithms for Multivariable Systems." PhD dissertation, Stanford 1974.

Smets, Frank and Raf Wouters, "Shocks and Frictions in US Business Cycles: A Bayesian DSGE Approach," *American Economic Review*, 2007, *97* (3), 586 – 606.