Bayesian Inference for ARFIMA Models

Garland Durham[*], John Geweke[†], Susan Porter-Hudak[‡], and Fallaw Sowell[§]

August 23, 2018

Abstract

This paper develops practical methods for Bayesian inference in the ARFIMA model using the exact likelihood function, any proper prior distribution, and time series that may have thousands of observations. These methods utilize sequentially adaptive Bayesian learning, a sequential Monte Carlo algorithm that can exploit massively parallel desktop computing with graphics processing units (GPUs). The paper identifies and solves several problems in the computation of the likelihood function that apparently have not been addressed in the literature. Four applications illustrate the utility of the approach. The most ambitious is an ARFIMA(2,d,2) model for the Campito tree ring time series (length 5,405), for which the methods developed in the paper provide an essentially uncorrelated sample of size 16,384 from the exact posterior distribution in under four hours. Less ambitious applications take as little as 4 minutes without exploiting GPUs.

[*]California State Polytechnic University, Orfalea College of Business, gbdurham@calpoly.edu

[†]University of Washington, Department of Economics, jgeweke@uw.edu; University of Technology Sydney, School of Business, John.Geweke@uts.edu.au

[‡]Northern Illinois University, sporterhudak@niu.edu

[§]Carnegie Mellon University, Tepper School of Business, fs0v@andrew.cmu.edu

1 Introduction

Approaches to inference in long memory models have proliferated since Geweke and Porter-Hudak (1983) proposed the first simple estimation scheme. These approaches differ in several dimensions, including their joint specification of long and short memory time series components. Augmenting the standard autoregressive moving average (ARMA) model with a long memory component leads to the autoregressive fractionally integrated moving average (ARFIMA) model. Using the exact likelihood function is an important consideration for all likelihood-based methods because it obviates questions about the impact of using likelihood function approximations. Thus the development of practical methods for exact likelihood function evaluation with ARFIMA models in Sowell (1992) was an important advance, and the model has subsequently proved to be one of the most enduring and widely used specifications. The new methods reported in this paper build on that work.

Bayesian approaches to inference in time series seamlessly integrate data and other information to produce predictive distributions. These advantages are particularly attractive in long memory models where the persistence of shocks impacts both inference for parameters and the joint predictive distribution in forecasts with long horizons. Pai and Ravishanker (1996) and Koop et al. (1997) used Markov chain Monte Carlo methods to create samples from the posterior distribution of parameters based on the exact likelihood function. While these approaches extend the advantages of Bayesian inference to ARFIMA models, they require extensive application-specific tuning. Their computational demands preclude application with thousands of observations, including time series in finance and geophysics in which long memory properties are important.

This paper uses sequentially adaptive Bayesian learning (SABL), a generic sequential Monte Carlo method developed recently in Durham and Geweke (2015) and Geweke and Durham (2018). In most cases, including ARFIMA models, it entails no application-specific tuning, making it available to applied statisticians who are not specialists in Bayesian posterior sampling methods. Because the algorithm is pleasingly parallel it is well-suited to the massively parallel desktop computing environments now available using graphics processing units (GPUs). GPU implementation reduces computing time by a factor of roughly 100, making possible application to time series with thousands of observations. Section 2 provides an overview of SABL.

In our implementation of the ARFIMA model in SABL we discovered that consistently reliable and efficient evaluation of the likelihood function requires close attention to some numerical issues that arise in the approach of Sowell (1992). To the best of our knowledge these have not been discussed in the literature. Section 3 outlines and addresses the problems in the context of a self-contained development of the ARFIMA likelihood function. It is the core analytical contribution of the paper. Section 3 also discusses and illustrates the factors that drive required computation time and provides guidance on whether to use a CPU or GPU platform.

Section 4 illustrates the approach with four classic applications of long memory models: consumer price inflation (70 observations), growth in per capita GDP (70 observations), annual Nile River minima (663 observations), and the Campito tree ring data (5,405 observations). For each of these applications the implementation of the ARFIMA model in SABL produces an essentially uncorrelated sample drawn from the exact posterior distribution.

To the best of our knowledge, this paper is the first report of exact posterior simulation in ARFIMA models using time series as long as those in our third and fourth applications. The examples use ARFIMA(p, d, q) models with orders $p = 0, 1, 2$ and $q = 0, 1, 2$. CPU implementation suffices for the first two applications; computation time for a posterior sample of 65,536 particles ranges from 4 to 43 minutes depending on the size of the model. The last two applications require GPU implementation; computation time for a posterior sample of 16,384 particles ranges from 1 to 8 minutes for the Nile data and from 40 to 210 minutes for the tree ring data.

2 The SABL algorithm for Bayesian inference

The sequentially adaptive Bayesian learning algorithm is an instance and extension of sequential Monte Carlo methods that can be used to simulate from posterior distributions and solve optimization problems. Relative to other approaches it is particularly advantageous when the number of parameters (Bayesian inference) or objective function arguments (optimization) is relatively small but the posterior density or objective function is analytically intractable or irregular. Here we are concerned only with Bayesian inference. This section provides a brief summary of the method and discusses features that are most pertinent in its application to ARFIMA posterior distributions in this paper. For full details see Durham and Geweke (2015), Geweke (2016) and especially Geweke and Durham (2018).[1]

To establish notation let A denote a model with the parameter vector $\theta_A \in \Theta_A \subseteq \mathbb{R}^{k_A}$; let $p(\theta_A \mid A)$ denote the proper prior density function of θ_A; let x_t $(t = 1, \ldots, T)$ denote the time series of outcomes observable in T successive time periods; denote $x_{1:t} = (x_1, \ldots, x_t)'$; and let

$$p(x_t \mid x_{1:t-1}, \theta_A, A) \qquad (t = 1, ..., T)$$

be the specified conditional probability density functions for observable outcomes.

Denote the observed outcomes x_t^o $(t = 1, \ldots, T)$. Then

$$L(\theta_A) = p(x_{1:T}^o \mid \theta_A, A) = \prod_{t=1}^{T} p(x_t^o \mid x_{1:t-1}^o, \theta_A, A)$$

is the likelihood function. The marginal likelihood (also known as the marginal data density) is

$$k(x_{1:T}^o \mid A) = \int_{\Theta_A} p(\theta_A \mid A) p(x_{1:T}^o \mid \theta_A, A)\, d\theta_A. \tag{1}$$

The posterior density of the parameter vector θ_A is

$$p(\theta_A \mid x_{1:T}^o, A) = \frac{p(\theta_A \mid A) p(x_{1:T}^o \mid \theta_A, A)}{k(x_{1:T}^o \mid A)}.$$

The notation can be extended in straightforward fashion to J models for $x_{1:T}$ by substituting A_j for A $(j = 1, \ldots, J)$. Given prior model probabilities $p(A_j)$ $(j = 1, \ldots, J)$

[1]SABL software and an accompanying handbook for users are available at `http://depts.washington.edu/savtech/help/software`.

posterior model probabilities are

$$p(A_j \mid x^o_{1:T}) = \frac{p(A_j)k(x^o_{1:T} \mid A_j)}{\sum_{i=1}^{J} p(A_i)k(x^o_{1:T} \mid A_i)} \qquad (j = 1, ..., J).$$

The SABL algorithm begins with a random sample $\theta^{(0)}_{A,n}$ $(n = 1, \ldots, N)$ from the prior distribution. For historical reasons the N members of the sample are known as *particles*. For short, write $\theta^{(0)}_{A,n} \sim p(\theta_A \mid A)$ $(n = 1, \ldots, N)$. The algorithm then constructs a sequence of distributions for θ_A with probability densities

$$p^{(\ell)}(\theta_A \mid x^o_{1:T}, A) \propto p(\theta_A \mid A)p(x^o_{1:T} \mid \theta_A, A)^{r_\ell} \tag{2}$$

in a sequence of cycles $\ell = 1, \ldots, L$. In this sequence $0 < r_1 < \ldots < r_L = 1$, so the sequence of distributions with densities (2) becomes increasingly concentrated in successive cycles in a process known as *power tempering.*[2] At the end of each cycle ℓ the particles have been updated to $\theta^{(\ell)}_{A,n}$ $(n = 1, \ldots, N)$, identically distributed with density (2).

Three phases comprise each cycle ℓ.

- Correction phase. Introduce new information by using the increment $r_\ell - r_{\ell-1}$ that achieves a pre-set relative effective sample size (Liu and Chen, 1998). Construct weights $w^{(\ell)}_n = p(x^o_{1:T} \mid \theta^{(\ell-1)}_{A,n}, A)^{r_\ell - r_{\ell-1}}$ $(n = 1, \ldots, N)$ such that the set of weighted particles $(\theta^{(\ell-1)}_{A,n}, w^{(\ell)}_n)$ $(n = 1, \ldots, N)$ represents the updated density (2).

- Selection phase. Resample with replacement the collection of particles $\theta^{(\ell-1)}_{A,n}$ $(n = 1, \ldots, N)$ with probabilities proportional to their weights $w^{(\ell)}_n$, creating a set of identically distributed particles $\theta^{(\ell,0)}_{A,n}$ $(n = 1, \ldots, , N)$ that represents (2).

- Mutation phase. Execute a series of Gaussian random walk Metropolis steps for the particles, using the sample variance of the particles $\theta^{(\ell,0)}_{A,n}$ $(n = 1, \ldots, , N)$ to tune the variance matrix, yielding the sequence $\theta^{(\ell,\kappa)}_{A,n}$ $(\kappa = 1, 2, \ldots)$ for each particle $n = 1, \ldots, N$. The sequence in κ terminates at the first Metropolis step $\kappa = K_\ell$ at which the particles $\theta^{(\ell,\kappa)}_{A,n}$ satisfy a mixing criterion. The particles at the end of cycle ℓ are then $\theta^{(\ell)}_{A,n} = \theta^{(\ell,K_\ell)}_{A,n}$ $(n = 1, \ldots, N)$. This is the most computationally intensive of the three phases in each cycle and the execution time for SABL tends to be driven by the total number of Metropolis steps $K = \sum_{\ell=1}^{L} K_\ell$.

The SABL algorithm has quite a few attractive properties. Several are especially important for Bayesian inference in the ARFIMA model.

1. The computations are mainly in the mutation phase and only a trivial portion are in the selection phase. The correction and mutation phases of the SABL algorithm are embarrassingly parallel. Consequently the algorithm efficiently exploits the capabilities

[2] An alternative is *data tempering*, in which the sequence of kernels is $p(\theta_A)p(x_{1:t_\ell} \mid x_{1:t_{\ell-1}}, \theta_A, A)$ with $0 = t_0 < \ldots < t_L = T$. SABL facilitates data tempering as well. We have not investigated the efficiency of data tempering for posterior inference in the ARFIMA model.

of massively parallel desktop computing with GPUs. This is important for Bayesian inference in ARFIMA models for very long time series and essential when there are hundreds (Section 4.3) or thousands (Section 4.4) of observations.

2. Like a smartphone or digital camera the SABL algorithm can be adjusted manually in many dimensions, for example the value of the relative effective sample size criterion used for terminating the correction phase and the criteria for sufficient mixing in the mutation phase. Like those devices, the SABL algorithm also has default settings that produce good results in a wide range of situations. Of the 27 settings that can be adjusted in SABL the examples in Section 4 used default settings for all except for the choice of number of particles.

3. Using SABL for inference in a particular model requires code for creating an i.i.d. sample from the prior distribution, for evaluating the prior distribution at specified parameter values (the particles), and for evaluating the log likelihood function at specified parameter values. There are no ancillary computations analogous to the evaluation of derivatives in maximum likelihood (Sowell 1992), creating and sampling from conditional posterior distribution in Gibbs sampling for Bayesian inference (Chib and Greenberg 1994), or design of proposal distributions for importance sampling (Koop et al. 1997) or Metropolis chains (Pai and Ravishanker 1996). Consequently there is often a great reduction in time devoted to tuning, tinkering, trial and error compared with alternative approaches.

4. SABL approximates the marginal likelihood (1) as a byproduct, together with a numerical standard error of approximation just as it does for posterior moments. Approximating the marginal likelihood has been a notoriously difficult computational problem for Bayesian inference. It is especially useful here in comparing models with and without long (or short) memory components and in comparing non-nested variants of the ARFIMA model.

3 The ARFIMA model

Following the notation in Sowell (1992) the autoregressive fractional integrated moving average model for a univariate time series $\{x_t\}$ is

$$\Phi(L)(1-L)^d(x_t-\mu)=\Theta(L)\varepsilon_t \tag{3}$$

where L is the conventional lag operator,

$$(1-L)^d = \sum_{j=0}^{\infty}\begin{pmatrix} d \\ j \end{pmatrix}(-1)^j L^j=\sum_{j=0}^{\infty}\frac{\Gamma(j-d)}{\Gamma(-d)\Gamma(j+1)}L^j, \tag{4}$$

$$\Phi(L)=1-\sum_{j=1}^{p}\phi_j L^j, \quad \Theta(L)=1+\sum_{j=1}^{q}\theta_j L^j, \tag{5}$$

and ε_t is white noise with variance σ^2. Our likelihood function and applications specify Gaussian white noise. We refer to (4) as the long memory component of the model and to (5) as the short memory component. The model has $3 + p + q$ parameters μ, σ^2, d, $\phi = (\phi_1, \ldots, \phi_p)'$ and $\theta = (\theta_1, \ldots, \theta_q)'$. The parameters are restricted to a subset of $\mathbb{R}^{3+p+q}$ such that the following conditions are satisfied.

1. The roots of $\Phi(L)$ and $\Theta(L)$ are strictly outside the unit circle.
2. $|d| < 1/2$.
3. $\Phi(L)$ has no repeated roots and $\phi_p \neq 0$.
4. $\sigma^2 > 0$.

The first condition is standard. Given the first condition, the time series $\{x_t\}$ is stationary if $d < 1/2$ and the lag operator $(1 - L)^d$ is invertible if $d > -1/2$. The third condition is a technical one (Section 3.1); it eliminates a subset of Lebesgue measure zero of the parameter space. These restrictions are imposed by truncating the support of the prior distribution as detailed in Section 3.3.

Bayesian inference in the ARFIMA model requires specification of the likelihood function and a prior distribution for the parameters. Using SABL to access the posterior distribution requires code that reliably and efficiently evaluates the likelihood function, simulates from the prior distribution and evaluates the prior density. Evaluation of the likelihood function proceeds in two steps: compute the autocovariance function given the parameters (Section 3.1) and then evaluate the log likelihood function (Section 3.2). Section 3.3 discusses prior distributions.

3.1 The autocovariance function

Since $\{x_t\}$ is a covariance stationary process, it has an autocovariance function

$$\operatorname{cov}(x_t, x_{t-s}) = \gamma(s; \sigma^2, d, \phi, \theta).$$

Sowell (1992) shows that the autocovariance function (ACF) can be expressed

$$\gamma(s) = \sigma^2 \sum_{j=1}^{p} \sum_{\ell=-q}^{q} \psi(\ell) \zeta_j C(d, p + \ell - s, \rho_j)$$

where

$$\psi(\ell) = \sum_{s=\max(0,\ell)}^{\min(q,q-\ell)} \theta_s \theta_{s-\ell}. \tag{6}$$

Condition 3 implies that $\Phi(L)$ has p distinct roots. We denote their inverses ρ_j $(j = 1, \ldots, p)$. Then define

$$\zeta_j = \left[\rho_j \prod_{i=1}^{p} (1 - \rho_i \rho_j) \prod_{m \neq j} (\rho_j - \rho_m) \right]^{-1} \quad (j = 1, \ldots, p), \tag{7}$$

and

$$\begin{aligned} C(d, h, \rho) &= \frac{\Gamma(1 - 2d)\Gamma(d + h)}{\Gamma(1 - d + h)\Gamma(1 - d)\Gamma(d)} \\ &\quad \times \left[\rho^{2p} F(d + h, 1; 1 - d + h, \rho) + F(d - h, 1; 1 - d - h, \rho) - 1 \right] \end{aligned} \tag{8}$$

where F is the hypergeometric function.

Several computational challenges become evident when implementing these calculations, some of which pertain specifically to the GPU implementation.

1. Evaluating (6) is completely straightforward, including GPU implementation.

2. Solving for the roots of $\Phi(L)$ is also straightforward using standard math libraries (e.g., the Matlab `roots()` function). Floating point arithmetic issues can arise in rare cases, for example, if $p = 2$ and $|\phi_2| < 2^{-m}$ where m is the number of mantissa bits in floating point representation. In this case, ϕ_2 is for all practical purposes zero. Our experience has been that floating point arithmetic issues here can be safely ignored.

3. If $\Phi(L)$ has repeated roots, (7) is infinite, hence Condition 3. But it is possible for (7) to evaluate to infinity in floating point arithmetic even if all the Conditions 1 - 4 are satisfied. We remove such points from the support of the prior distribution. Occurrence of this is unlikely in practice and eliminating these points from the prior has negligible effect. However, our experience has been that SABL is able to explore the parameter space very intensively, and it is important to make allowance for such edge cases.

4. There are several considerations specific to (8).

   (a) When $d = 0$ (8) reduces to

   $$C(0, h, \rho) = \begin{cases} \rho^h, & h \geq 0 \\ \rho^{2p-h}, & h < 0 \end{cases}. \tag{9}$$

   (b) Problems can emerge when d is very close to zero since $\lim_{x \to i} \Gamma(x)$ is undefined for any negative integer i. We avoid this problem by reverting to (9) if $|d| < 1000 \cdot 2^{-m} \approx 2.2 \times 10^{-13}$, where m is the number of mantissa bits in floating point representation.

   (c) Efficient and reliable computation of the hypergeometric function is essential to the efficiency and reliability of the entire algorithm. The standard infinite series representation of F is

   $$F(a, 1; c, \rho) = \sum_{i=0}^{\infty} g_i(a, c, \rho) = \sum_{i=0}^{\infty} \left(\prod_{j=0}^{i-1} \left(\frac{a+j}{c+j} \right) \right) \rho^i. \tag{10}$$

   As $|\rho|$ approaches one, convergence of this series can be very slow. Care is required in terminating the summation and all of the off-the-shelf code we were able to locate failed in doing so, producing inaccurate results for values of ρ very close to 1.

      i. Our algorithm terminates the summation in (10) at the first s such that $g_{s+1}(a, c, \rho)$ has no significant digits relative to $\sum_{i=0}^{s} g_i(a, c, \rho)$ at machine precision. We restrict $|\rho| < 0.9999$ to ensure that this termination occurs in reasonable time. For values close to this boundary, accurate evaluation of

(10) may still require many millions of terms. There are round-off issues that need to be addressed when summing such long sequences. This is especially a problem when the terms are decreasing in size, as is the case here. There are well-known ways to handle such issues and we have taken pains to ensure that these kinds of round-off errors do not occur.

ii. The GPU evaluates threads in "warps" of 32 threads running in same instruction multiple data (SIMD) fashion. Recall that each thread calculates the likelihood for a single particle. If (10) converges more quickly for some threads in a warp than others, computational power is wasted doing no work in the threads for which the series has already converged. To address this issue, we have found it to be useful to sort the particles based on $|\rho|$ before evaluating (10).

(d) As pointed out by Sowell (1992), the computation (10) can be streamlined by exploiting the recursion

$$F(a,1;c,\rho) = \frac{c-1}{\rho(a-1)}\left[F(a-1,1;c-1,\rho)-1\right]. \tag{11}$$

However, we have found that rounding error accumulates significantly with repeated application of this recursion and every so often one must refresh the sequence with a new evaluation of (10). Fortunately, it turns out that these errors emerge more slowly as $|\rho|$ approaches 1, so the values of ρ for which evaluating (10) is most costly are also the ones where it needs to be evaluated least often. We have found by trial and error that sufficient accuracy is obtained by stopping the recursion and executing a new evaluation of (10) every M iterations, where $M = \text{int}(4/(1.0005-|\rho|))$. In GPU implementation all threads in a warp use the value of M corresponding to the smallest value of $|\rho|$ among the threads in the warp.

Evaluation of the autocovariance function becomes simpler in some relevant special cases. If $p=0$ (i.e. the short memory component (5) is MA(q)) then

$$\gamma(s) = \sigma^2 \sum_{\ell=-q}^{q} \psi(\ell)\frac{\Gamma(1-2d)\Gamma(d+s-\ell)}{\Gamma(d)\Gamma(1-d)\Gamma(1-d-s+\ell)}$$

if $d \neq 0$ and

$$\gamma(s) = \begin{cases} \sigma^2\psi(s), & |s| \leq q \\ 0, & |s| > q \end{cases}$$

if $d = 0$.

3.2 Evaluating the log likelihood function

Let $V(\gamma, T)$ denote the $T \times T$ Toeplitz matrix with entries $v_{ij}(\gamma, T) = \gamma(|i-j|)$ and let $x = (x_1, \ldots, x_T)'$. Since the distribution of x is Gaussian the log likelihood function is

$$\ell(\gamma,\mu) = -(T/2)\log(2\pi) - (1/2)\log|V(\gamma,T)| - (1/2)(x-\mu)'V(\gamma,T)^{-1}(x-\mu).$$

Evaluation of this log likelihood function using a Cholseky decomposition and Gauss-Jordan elimination entails $O(T^3)$ floating point operations. The Durbin-Levinson algorithm (Levinson 1947; Durbin 1960; Golub and van Loan 1996, Section 4.7.2) takes advantage of the Toeplitz structure of the variance matrix to evaluate the log likelihood function with $O(T^2)$ floating point operations, but can be numerically unstable. We use the adjusted version of the algorithm provided in Doornik and Ooms (2003, Appendix A) that amounts to simultaneous computation and application of the inverted Cholseky decomposition of $V(\gamma, T)$. The algorithm is embarrassingly parallel across particles, resulting in extremely fast computations in the GPU environment described in Section 2. For the special case of short memory processes ($d = 0$) it is competitive with conventional state space approaches to the evaluation of the log likelihood function of an ARMA model (Durbin and Koopman 2001).

GPU implementation is relatively straightforward with each thread evaluating the likelihood corresponding to a single particle, as usual. The only serious issue involves memory requirements. With N particles and T observations, Durbin-Levinson requires memory proportional to NT for a fully parallel implementation. However, we were able to run the application in Section 4.4 ($N = 2^{14}$, $T = 5405$) with no particular difficulty.

3.3 Prior distributions

The SABL algorithm supports very flexible prior distributions relative to some competing approaches like Gibbs sampling, which places a premium on conditionally conjugate prior distributions, and Metropolis algorithms which may need to be tuned to specific prior distributions (Geweke, 2005, Chapter 4).

Some of the work here uses the following *benchmark prior distribution*, which has four independent components.

- $\mu \sim N(m_\mu, v_\mu)$, where the hyperparameters m_μ and v_μ are application dependent.
- $\log \sigma^2 \sim N(m_\sigma, v_\sigma)$, where the hyperparameters m_σ and v_σ are application dependent.
- $d \sim \text{Uniform}\left(-\frac{1}{2}, \frac{1}{2}\right)$.
- $\phi \sim N(m_\phi, V_\phi)$ subject to all p roots of the lag polynomial $\Phi(L)$ outside the unit circle (Condition 1) and distinct (Condition 3). The prior probability density function is

  $$p(\phi \mid m_\phi, V_\phi) = \frac{p_N(\phi; m_\phi, V_\phi)}{s(m_\phi, V_\phi)}$$

  where $p_N(x; m, V)$ is the normal probability density function for $\mathrm{N}(m, V)$ and $s(m, V) = \int_{\Re} p_N(x; m, V)dx$ where $\Re$ is the subset of $\mathbb{R}^p$ satisfying the constraints. The value of $s(m, V)$ can be approximated by Monte Carlo, and this is practical unless $s(m_\phi, V_\phi)$ is very close to 0. Given values of p, m_ϕ and V_ϕ that are likely to be entertained in practice, prospects for problems on this account are nil. The approximation of s needs to be computed only once and can be done at essentially no cost when simulating from the prior at the start of the SABL algorithm. In our implementation we use straightforward acceptance sampling from the source distribution $N(m_\phi, V_\phi)$ and the approximation of $s(m, V)$ is the ratio of the number of particles N to the number of draws from the source distribution required.

Table 1: Computing time in seconds, one ARFIMA(1,d,1) likelihood function evaluation

	CPU (Matlab)							
	Autocovariance function				Durbin-Levinson algorithm			
$N =$	2^{10}	2^{12}	2^{14}	2^{16}	2^{10}	2^{12}	2^{14}	2^{16}
$T = 100$	1.6	6.3	24.8	94.7	0.1	0.4	3.3	11.0
$T = 500$	5.3	21.3	84.3	324.4	2.4	22.5	68.0	260.0
$T = 2500$	20.0	91.1	332.5	1236.5	105.6	422.2	1603.7	8147.6
	GPU (CUDA)							
	Autocovariance function				Durbin-Levinson algorithm			
$N =$	2^{10}	2^{12}	2^{14}	2^{16}	2^{10}	2^{12}	2^{14}	2^{16}
$T = 100$	0.15	0.18	0.28	0.46	0.02	0.03	0.05	0.16
$T = 500$	0.57	0.40	0.32	0.74	0.24	0.32	0.92	3.689
$T = 2500$	1.33	0.92	1.08	2.14	5.91	7.67	22.81	173.49

- $\theta \sim \mathrm{N}(m_\theta, V_\theta)$ subject to all q roots of the lag polynomial $\Theta(L)$ outside the unit circle (Condition 1). Similar considerations as for ϕ apply regarding the prior probability density function and Monte Carlo approximation of the normalizing constant.

For our *standard benchmark prior distributions* for ϕ and θ we set

$$m_\phi = 0_p,\ V_\phi = I_p,\ m_\theta = 0_q,\ V_\theta = I_q.$$

All of the applications in Section 4 utilize this prior. But depending on the application, components of the model may have substantive interpretations for which specific and informative prior distributions are available. Section 4.2 develops and applies such a prior distribution.

3.4 GPU versus CPU computation

GPU computing using the CUDA computing platform and application programming interface can yield substantial performance improvements, especially with large data sets and many particles. For small sample sizes T, overall computational cost is dominated by the cost of evaluating the hypergeometric function. The recursive formula (11) reduces the cost significantly, but evaluating the series (10) can require summing millions of terms and the number of function evaluations is proportional to NT. On the other hand, Durbin-Levinson is not costly to evaluate for small T, but the cost increases in proportion to NT^2.

Care is required in implementing both the ACF and Durbin-Levinson algorithms on the GPU, as noted in Sections 3.1 and 3.2, but the payoff can be large. Table 1 shows likelihood function wall clock computation time for several combinations of N and T, using artificial data $x_t \overset{iid}{\sim} N(0,1)$, and particles $\mu = 0$, $\sigma^2 = 1$, $\phi_1 \overset{iid}{\sim} \mathrm{Uniform}(-0.9999, 0.9999)$, $\theta_1 \overset{iid}{\sim} \mathrm{Uniform}(-0.9999, 0.9999)$, $d \overset{iid}{\sim} \mathrm{Uniform}(-0.4999, 0.4999)$. The CPU (Matlab) implementation uses no parallelization beyond the Matlab default. GPU (CUDA) implements only the computationally intensive portions of the algorithm and is a direct plug-in replacement for the corresponding Matlab code in the SABL algorithm, the core code for which is essentially unchanged whether run on the GPU or CPU. This exercise was performed on a workstation with two NVIDIA K40 GPUs (2880 processor cores running at 745 MHz and 12 GB onboard memory each), of which only one was used, and dual Intel E5-2620 CPUs (six cores each, 2.1 GHz).

The cost of evaluating the ACF dominates for small sample sizes, but Durbin-Levinson dominates for large sample sizes, with the break-even point at about $T = 500$, depending on the number of particles N and computing environment.

For code running on the CPU, computing time scales roughly as expected with respect to N and T. It should be noted that the Matlab code that evaluates the ACF requires looping over both $t = 1, \ldots, T$ and $n = 1, \ldots, N$ because the hypergeometric function is not amenable to Matlab vector operations. Writing C code and mex files to evaluate the likelihood would improve performance here. On the other hand, the Durbin-Levinson algorithm uses highly efficient Matlab vector operations across particles.

On the GPU, with small N and T overhead costs related to calling CUDA from Matlab dominate; the benefits of GPU computing become striking only with larger T and (especially) N. Recall that the number of threads run in parallel is equal to the number of particles. Because of the GPU architecture, having more threads available reduces the time the GPU cores spend idle while waiting on memory transfers, and this is apparent in the way computing time scales with respect to N. The exception is for $N = 2^{16}$ and $T = 2500$, and the longer than expected execution time here is most likely due to issues related to Matlab's memory management.

We have found that for well-suited problems, GPU computing is often around 100 times faster than on the CPU, and this is borne out here. The fact that the SABL algorithm runs well in a massively parallel environment is one of the features that makes the algorithm so appealing, and SABL takes full advantage of this with little effort by the user.

For applications where the number of observations is around a hundred or less, as is typical for annual macroeconomic data like that examined in Sections 4.1 and 4.2, CPU execution is practical. However, for samples with several hundreds, or thousands, of observations—such as those examined in Sections 4.3 and 4.4—CPU execution quickly becomes impractical and GPU implementation is highly desirable.

4 Applications

Four applications of Bayesian inference in the ARFIMA model using SABL in this section illustrate the capabilities of the approach, provide insight into the likelihood function and posterior distributions, demonstrate the flexibility of prior distributions accommodated in SABL, and illustrate the advantages of the algorithm in combination with massively parallel desktop computing using GPUs. All of the specific applications have precedent in the literature[3].

The first two applications are to economic time series: consumer price inflation (Section 4.1) and US per capita real gross domestic product (Section 4.2). Because each time series has fewer than 100 observations Bayesian inference using SABL is practical using conven-

[3]Software for the four applications in this section is available at `www.quantosanalytics.org/calpoly/arfima.zip`.

tional quadcore CPUs. These applications investigate the joint posterior distributions of parameters, illustrate Bayesian model averaging over different specifications of the short memory component, and demonstrate the utility of an informative prior distribution for the short memory component.

The third and fourth applications examine data sets commonly used in previous studies of long memory. The Nile River data is comprised of annual water level measurements over the 663 year period 622–1284 AD (Section 4.3). The Campito tree ring data is comprised of annual observations of tree ring widths over the 5,405 year period 3435 BC–1969 AD (Section 4.4). For both applications, the posterior probabilities strongly favor long memory; in both cases, the preferred models have no short-memory component. These data sets contain relatively large numbers of observations and demonstrate the utility of GPU computing with SABL. Posterior simulation using CPU alone would be tedious for the Nile River data and essentially intractable for the tree ring data. In Sections 4.3 and 4.4 we provide additional details on computational aspects of the algorithm, including the number of cycles L and total Metropolis steps K executed.

4.1 Consumer price index

In the published economics literature long memory models were first applied to time series of prices (Granger and Joyeux 1980; Geweke and Porter-Hudak 1983). There is an accompanying literature on arbitrage opportunities when d is fractional (Rogers 1997). Baillie et al. (1996), Hwang (2001), Doornik and Ooms (2004), Ling (2003) and Sela and Hurvich (2009) used maximum likelihood methods of inference in ARFIMA models applied to inflation rates. The only instance of Bayesian inference in ARFIMA-like models applied to inflation of which we are aware is Nonejad (2015), who used particle Gibbs with ancestor sampling and a stochastic volatility model for the disturbances $\{\varepsilon_t\}$ in (3).

The data in this example are the inflation rate $(P_t - P_{t-1})/P_{t-1}$, where P_t denotes the U.S. consumer price index (CPI) and t indicates the years $1948, \ldots, 2017$. The source for annual CPI is the FRED data base maintained by the Federal Reserve Bank of St. Louis. The SABL algorithm (Section 2) was executed with default settings except that the number of particles was increased from 2^{14} to $2^{16} = 65,536$. Because $T = 70$ execution using conventional quadcore CPUs is practical.

The objectives in this application are to compare alternative ARFIMA model specifications and learn about the posterior distributions of model parameters in some of these specifications. The standard benchmark prior distribution applies in all cases; the independent distributions $\mu \sim N(0, 0.04^2)$ and $\log \sigma^2 \sim N(-8, 1)$ complete the prior distribution.

There are 18 alternative ARFIMA (p, d, q) models (3), the Cartesian product of $p = 0, 1, 2$, $q = 0, 1, 2$, each with and without the restriction $d = 0$. Table 2 indicates the log marginal likelihood value of each model (top panel) and the posterior probability of each model using a prior probability of 1/18 (bottom panel). The numerical standard error of the SABL approximation of the log marginal likelihood is typically about 0.01 and never greater than 0.02.

The posterior odds ratio in favor of long memory models is 4.2:1. The ARFIMA$(0, d, 1)$ and ARFIMA$(1, d, 1)$ models have the highest posterior probabilities, 0.318 and 0.130 respectively.

Table 2: Model specifications for CPI inflation using benchmark prior distributions

	Log marginal likelihood							
	Short memory ($d = 0$)				Long memory			
	$q = 0$	$q = 1$	$q = 2$		$q = 0$	$q = 1$	$q = 2$	
$p = 0$	145.21	167.75	168.46		165.34	172.35	171.38	
$p = 1$	167.25	170.18	171.08		167.56	171.46	171.41	
$p = 2$	166.19	169.52	170.15		168.13	170.68	170.50	
	Implied posterior probabilities (equal prior probabilities)							
	Short memory ($d = 0$)				Long memory			
	$q = 0$	$q = 1$	$q = 2$	Marginal	$q = 0$	$q = 1$	$q = 2$	Marginal
$p = 0$	0.000	0.003	0.006	0.010	0.001	0.318	0.120	0.438
$p = 1$	0.002	0.036	0.089	0.128	0.003	0.130	0.123	0.256
$p = 2$	0.001	0.019	0.035	0.054	0.005	0.060	0.050	0.114
Marginal	0.003	0.058	0.131	0.192	0.008	0.507	0.293	0.808

Table 3 provides more detail for these two models, including the first four posterior moments of each parameter (first four columns for each model). It also provides two statistics related to the accuracy of the SABL approximation of the posterior mean: the numerical standard error (NSE) of the first moment approximation; and the relative numerical efficiency (RNE) of this approximation, which is the ratio of the numerical standard error in a hypothetical random sample from the posterior distribution to the one achieved by SABL. Geweke and Durham (2018) provide more detail on the construction and interpretation of NSE and RNE. For our purposes it is sufficient to note that posterior mean approximations are good to about 3 significant figures.

The last two columns of Table 3 provide maximum likelihood estimates and standard errors of the parameters, computed using SABL.[4] Due to variations in sample period and frequency and modeling of the term ε_t in (3) even rough comparisons of the results in Table 3 with other studies, e.g. Baillie et al. (1996), are not possible.

The results in Table 3 are inconsistent with a Gaussian posterior distribution. Somewhat surprisingly, among the individual parameters this is strongly characteristic of the posterior distribution of μ. The posterior moments of μ and σ^2 are similar in the two models, those of θ_1 are somewhat different, and those of d are strikingly different. In particular conclusions about d are sensitive to the specification of the short memory component of the model. Posterior standard deviations are generally higher than maximum likelihood standard errors, by as much as 58%, and they are never much lower. To understand these outcomes it helps to move closer to the posterior distribution itself.

Figure 1 plots the marginal posterior distribution of each parameter (panels on the diagonal) and the posterior distribution of every pair of parameters (panels off the diagonal) for the ARFIMA$(0, d, 1)$ model. The panels above the diagonal display the indicated components of all $2^{16} = 65,536$ particle vectors at the conclusion of the SABL algorithm, whereas the panels below the diagonal display a 1 in 64 subsample (components of $2^{10} = 1,024$ particle vectors). Thus the panels above the diagonal provide more detail on the tails of

[4]For details on optimization with SABL see Geweke and Durham (2018).

Table 3: Posterior moments and estimates of ARFIMA model parameters, CPI inflation

ARFIMA$(0,d,1)$								
	Posterior distribution						Maximum likelihood	
	Mean	St dev	Skewness	Kurtosis	NSE	RNE	Estimate	St err
μ	0.0271	0.0197	-0.8374	5.7510	0.0001	0.8580	0.0344	0.0124
d	0.3185	0.0895	-0.3053	2.6718	0.0004	0.7089	0.2606	0.0980
$\log\sigma^2$	-7.9311	0.1712	0.1570	3.0720	0.0007	0.8714	-7.9833	0.1714
θ_1	0.6943	0.1086	-0.8681	4.4595	0.0004	1.4441	0.7517	0.0922

ARFIMA$(1,d,1)$								
	Posterior distribution						Maximum likelihood	
	Mean	St dev	Skewness	Kurtosis	NSE	RNE	Estimate	St err
μ	0.0271	0.0202	-0.9001	5.6450	0.0001	1.0122	0.0339	0.0148
d	0.2661	0.2092	-1.8618	6.3571	0.0012	0.4403	0.3179	0.1321
$\log\sigma^2$	-7.9253	0.1727	0.1820	3.0532	0.0006	1.3158	-7.9911	0.1692
θ_1	0.7055	0.1462	0.9713	3.7939	0.0005	1.3226	0.7955	0.1139
ϕ_1	0.0173	0.3244	-1.3325	7.1705	0.0016	0.6609	-0.1409	0.2508

the distributions, the panels below the diagonal more detail on the portions of the bivariate distributions with the greatest mass concentration.

The joint distribution of μ and d is distinctly not elliptical. It reflects the fact that as the long memory parameter d increases, and especially as it approaches the upper limit of stationarity $d = 1/2$, there is less information about the mean, which fails to exist at $d = 1/2$. This phenomenon is driven by the likelihood function and has been noted previously, e.g. Pai and Ravishanker (1996). The posterior correlation between d and μ is -0.29: as d moves from 0.15 to 0.46, the conditional posterior mean of μ decreases substantially, from 0.033 to 0.012. This phenomenon is driven by the $N(0, 0.04^2)$ prior distribution of μ: as d increases the data provide less information about μ, the prior becomes relatively more influential, and so the conditional posterior mean of μ moves closer to the prior mean of μ. Because the joint distribution of μ and d is not remotely elliptical, the classic rationale for maximum likelihood estimates and standard errors does not apply for these parameters. This is a general feature of ARFIMA models, not specific to this particular application: given a posterior distribution for d with positive mean and a large standard deviation (say, exceeding 0.15) conclusions about d and μ will be sensitive to prior information and the classic theory of maximum likelihood estimation for d and μ will be a poor approximation.

All of the other bivariate posterior distributions are non-Gaussian, due mostly to the relevance of the stationarity condition $d < 1/2$ and invertibility condition $|\theta_1| < 1$ in this application. This can be made explicit by overplotting the bivariate maximum likelihood confidence regions, as in Geweke and Durham (2018). That has not been done here in order to maintain the legibility of Figure 1.

In the ARFIMA$(1, d, 1)$ model, which has substantial posterior probability (Table 2), the posterior distribution is highly irregular. The third and fourth posterior moments in Table 3 suggest this possibility. It is obvious in Figure 2. In the posterior distribution there is strong negative correlation between the short memory AR coefficient ϕ_1 and the long memory parameter d, due to the fact that ϕ_1 accounts for persistence that decays

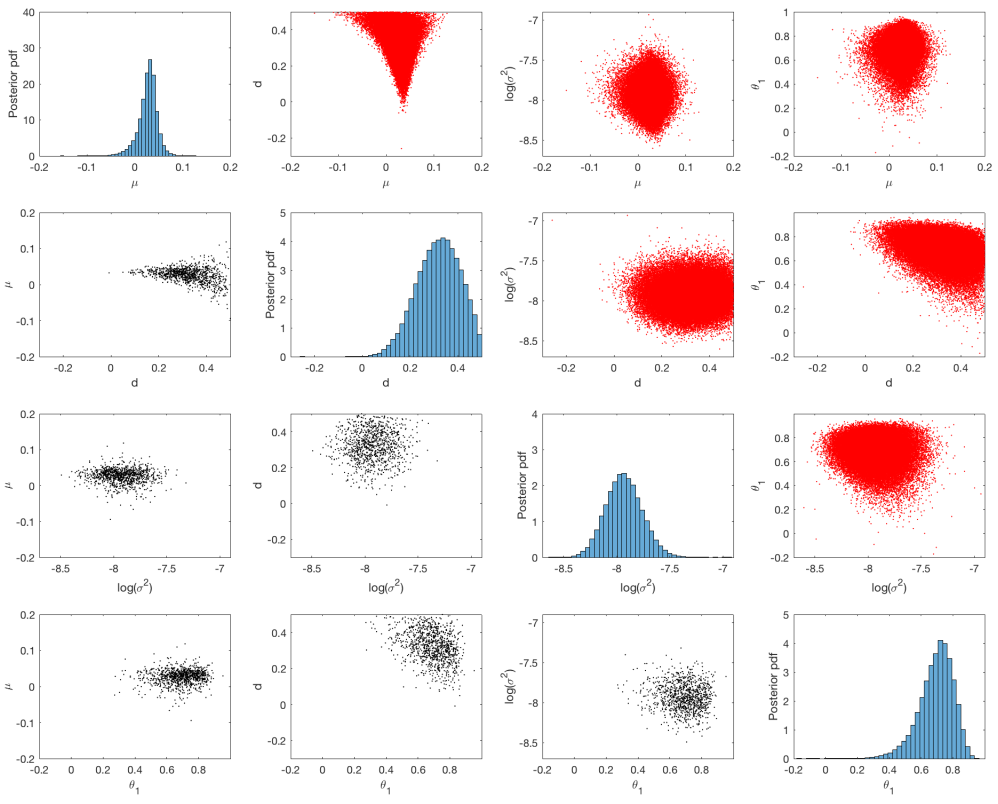


Figure 1: Aspects of the parameter posterior distribution, CPI ARFIMA$(0, d, 1)$ model.

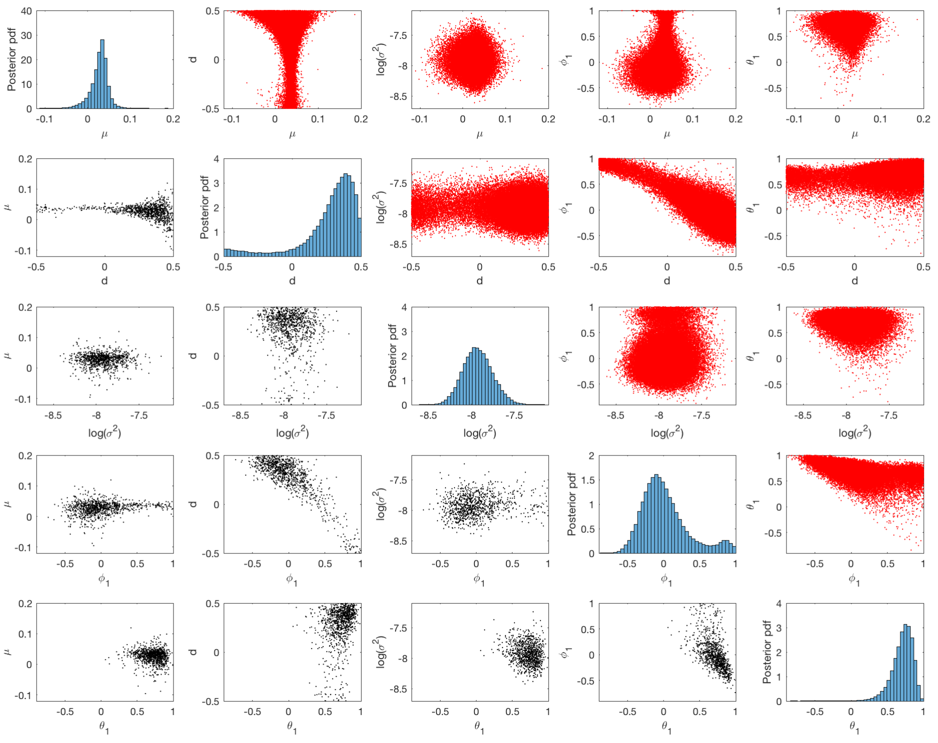


Figure 2: Aspects of the parameter posterior distribution, CPI ARFIMA$(1, d, 1)$ model.

slowly but geometrically, d accounts for persistence that decays slowly but algebraically, and the greater the degree of persistence the larger the sample required to sort them out. As a consequence d has non-negligible posterior mass on all of its support as does ϕ_1 on most of its support. Given the relation between d and the other parameters previously noted, the lack of information about (d, ϕ_1) complicates the posterior distributions of all of the parameters indirectly. This leads to the "tornado" joint posterior densities of (d, μ) and (ϕ_1, μ) in Figure 2, bimodal posterior densities of $(\phi_1, \log \sigma^2)$ and $(d, \log \sigma^2)$, and the distribution of (ϕ_1, θ_1) that reflects the constraints of stationarity and invertibility in the short memory component.

4.2 Per capita real GDP

The data in this example are the first difference of the logarithm of annual real per capita GDP for the United States, 1947 - 2017. The data were constructed from the real GDP and US total population time series in the FRED data base maintained by the Federal Reserve Bank of St. Louis. The SABL algorithm (Section 2) was executed with default settings except that the number of particles was increased from its default value of 2^{14} to $2^{16} = 65,536$. Because $T = 70$ execution using conventional quadcore CPUs is practical.

Table 4: Model specifications for GDP using benchmark prior distributions

	Log marginal likelihood							
	Short memory ($d = 0$)				Long memory			
	$q = 0$	$q = 1$	$q = 2$		$q = 0$	$q = 1$	$q = 2$	
$p = 0$	162.31	161.08	159.52		161.28	160.27	159.17	
$p = 1$	161.07	160.73	159.28		160.56	160.13	158.92	
$p = 2$	159.54	159.25	158.98		159.06	158.88	158.50	
	Implied posterior probabilities (equal prior probabilities)							
	Short memory ($d = 0$)				Long memory			
	$q = 0$	$q = 1$	$q = 2$	Marginal	$q = 0$	$q = 1$	$q = 2$	Marginal
$p = 0$	0.334	0.098	0.021	0.453	0.120	0.044	0.015	0.178
$p = 1$	0.097	0.069	0.016	0.182	0.058	0.038	0.011	0.107
$p = 2$	0.021	0.016	0.012	0.049	0.032	0.028	0.018	0.049
Marginal	0.453	0.182	0.049	0.684	0.191	0.092	0.033	0.316

The objectives in this example are to assess the evidence for long and short memory models and to compare models with different prior distributions for the short memory component.

The first prior distribution is the standard benchmark prior distribution with $\mu \sim \mathrm{N}(0, 0.04^2)$ and $\log \sigma^2 \sim \mathrm{N}(-8, 1)$. There are 18 specifications of the model (3): the Cartesian product of $p = 0, 1, 2$ with $q = 0, 1, 2$ with short memory ($d = 0$) and long memory ($d \in (-1/2, 1/2)$). Quadcore CPU execution time ranges from about 1 minute for $d = p = q = 0$ to about 15 minutes for $p = q = 2$, $d \in (-1/2, 1/2)$. The upper panel of Table 4 presents the SABL approximations of the log marginal likelihood values for the different benchmark models; the numerical standard error of approximation is about 0.01 for all values.

A prior distribution that places equal probabilities on each of the 18 models defines a super-model composed of these 18 models. The posterior distribution of the super-model includes posterior probabilities for each of the 18 specifications, shown in the lower panel of Table 4.

From the sample of parameter values provided by SABL it is straightforward to evaluate the corresponding spectral density at any specified set of ordinates. This calculation exploits the spectral density function $S(\omega; d) = [2(1 - cos(\omega))]^{-d}$ of a long memory process and the Fourier transforms of the lag operators in (5); this is much faster than the autocovariance function computation. The top panel of Figure 3 shows posterior quantiles of the log spectral density function at each ordinate, given the benchmark prior distribution with equal prior probabilities for the 18 alternative specifications. The mean width of the interquartile range is 0.32, corresponding to a ratio 1.38 of spectral density values ($1.38 = \exp(0.32)$).

For the standard benchmark prior the posterior odds ratio in favor of short memory over long memory is 2.16:1. This is consistent with the posterior probability density function of d under the benchmark prior distribution, shown in the upper left panel of Figure 4. The value of the p.d.f at $d = 0$ is 2.16, which coincides (as it must, because the support of d has measure 1) with the posterior odds ratio in favor of short memory. These findings are consistent with those of Koop at al. (1997) and Pai and Ravishanker (1996), which studied

Table 5: Comparison of models with benchmark and cyclical prior distributions

	Log marginal likelihood		Odds ratio
Prior $\rightarrow$	Benchmark	Cyclical	
Short memory	160.83	162.28	4.29:1
Long memory	160.06	161.18	2.72:1
All	160.51	161.87	3.08:1

similar data.

Alternative prior distributions may resolve some of this uncertainty. Promising priors would have two properties. First, they would be informative relative to the standard benchmark prior distribution, exhibited in posterior distributions that are concentrated relative to those shown in the upper panel of Figure 3 and the upper left panel of Figure 4. Second, they would be reasonable, exhibited in marginal likelihoods at least as great as those arising with the standard benchmark prior. Since the posterior distribution implicitly disentangles the short and long memory components from the data, more informative prior distributions for either component are candidates.

There is a ready interpretation of the short memory ARMA component of the model arising from the multiplier-accelerator model (Samuelson 1939) that has been used for Bayesian inference in short memory models of GDP growth (Geweke 1988, 2016). In the multiplier-accelerator model, economic activity is AR(2): the short memory component (5) is $\Phi(L) = 1 - \phi_1 L - \phi_2 L^2$ and $\Theta(L) = 1$. The roots (r_1, r_2) of $\Phi(L)$ are complex conjugates as well as being outside the unit circle ($|r_1| > 1$, $|r_2| > 1$). Let r_1 indicate the root with positive imaginary component. The second order polynomial $\Phi(L)$ is the same as that in the damped oscillator model with periodicity and amplitude

$$P = 2\pi / \arg(r_1) \quad \text{and} \quad \alpha = |r_1|^{-1},$$

respectively, implying $P > 2$. The half-life h corresponding to the amplitude $\alpha \in (0, 1)$ is defined by

$$\int_0^h \alpha^u \, du \Big/ \int_0^1 \alpha^u \, du,$$

implying h $= \log(1/2)/\log(\alpha)$. It is the time required for half the power of the shock ε_t to be dissipated through the short memory component of the model.

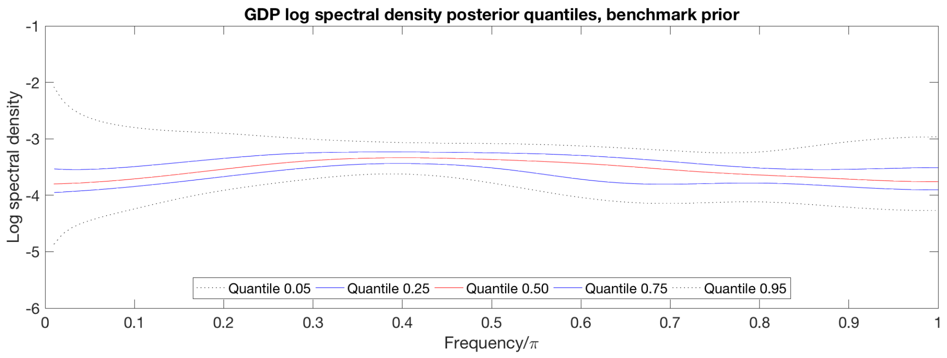
The inverse mapping from (P, α) to (ϕ_1, ϕ_2) is

$$r = \exp(2\pi i/P)/\alpha, \quad \phi_1 = 2 \cdot \text{Re}(1/r), \quad \phi_2 = -1/|r|^2.$$

Hence it is practical to replace an explicit prior distribution for ϕ with one for (P, h). Motivated by characteristics of the business cycle, P and h are independent in this *cyclical prior distribution* with

$$\log(P) \sim N\left(\log(5),\ \log(1.2)^2\right), \quad (P > 2), \quad \log(h) \sim N\left(0,\ 0.5^2\right).$$

In the SABL algorithm $\log(P)$ and $\log(h)$ replace the elements ϕ_1 and ϕ_2 of the SABL parameter vector θ. The prior distributions for μ, σ^2 and d remain the same as under



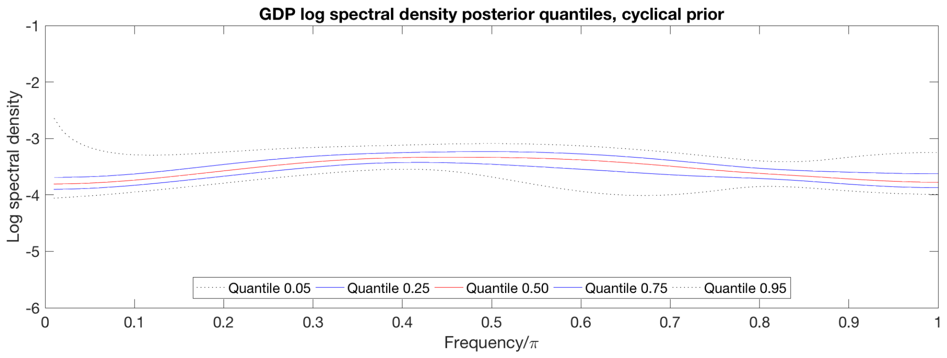


Figure 3: Posterior distributions of log spectral densities.

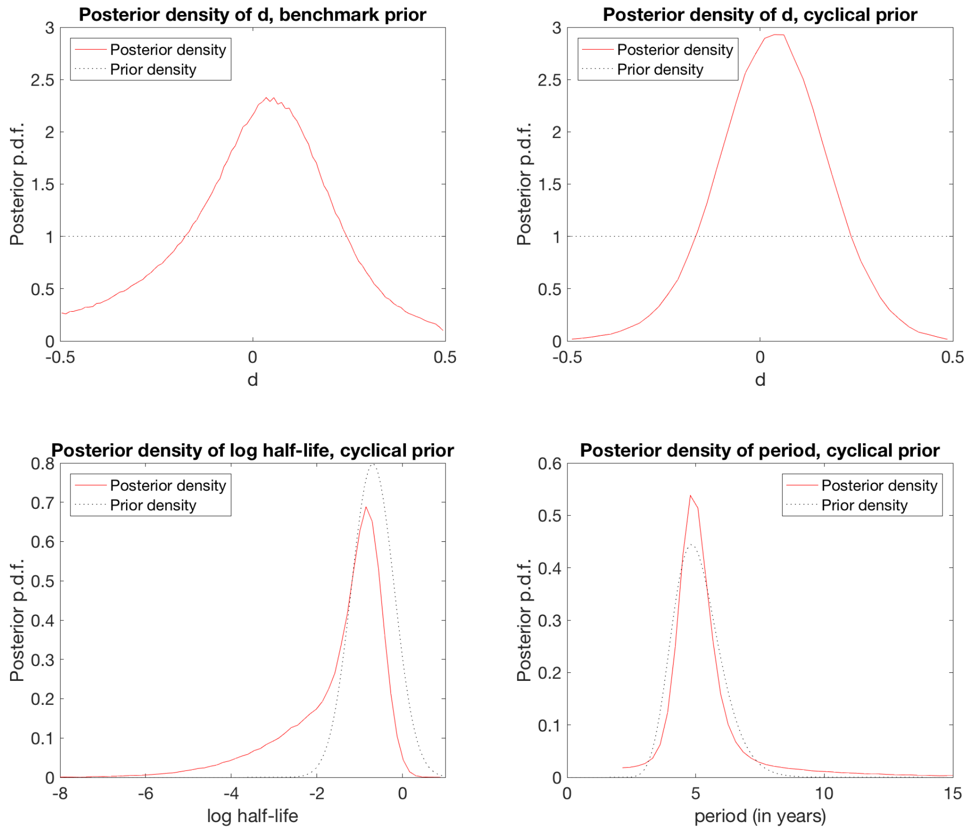


Figure 4: Some posterior distributions of parameters, GDP example.

the benchmark prior distribution. Thus the contrast between the two models is between a relatively uninformative prior distribution for the short memory component (benchmark) and a relatively informative one (cyclical).

There are just two variants of the GDP model with the cyclical prior: with and without the long memory component. Table 5 provides the log marginal likelihoods under the cyclical prior distribution (column 2) and compares them with those under the benchmark prior distribution (column 1). The first two rows, for the benchmark prior, reflect equal prior probabilities on all 9 short memory and long memory models, respectively. The last row reflects equal prior probability on the short memory and long memory model(s) under the benchmark and cyclical prior distributions. The last column provides the posterior odds ratio in favor of the cyclical prior model over the benchmark prior model. The results support the plausibility of the more informative prior distribution.

In Figure 3 the increase in information is manifest in more concentrated quantiles with the cyclical prior (bottom panel) compared to the benchmark prior (top panel). The mean interquartile range with the cyclical prior is 0.215, whereas it was 0.320 with the benchmark prior. With the cyclical prior the spectral density is higher at very low frequencies than it

Table 6: Prior and posterior moments with cyclical prior

	Prior		Posterior	
	Mean	St dev	Mean	St dev
Half life h	1.133	0.604	0.806	0.414
Period P	5.084	0.935	5.608	1.602

is with the benchmark prior. This is consistent with the greater concentration and modest upward location shift in the posterior distribution of the long memory parameter d under the cyclical prior (mean -0.013, standard deviation 0.221) relative to the benchmark prior (mean 0.032, standard deviation 0.144), as shown in the upper right and left panels respectively of Figure 4.

The bottom panels of Figure 4 show the prior and posterior densities of h and P. The posterior densities are more concentrated than the prior densities, but not by much. Table 6 compares the prior and posterior means and standard deviations. The information content of the prior is strong, relative to the data. But that information is consistent with the data, resulting in odds that favor the strong prior and an increase in the precision of the model.

Two earlier studies undertook Bayesian inference for the logarithm of US real GDP. These applications differ from the one here in (1) the use of aggregate rather than per capita GDP, (2) the use of quarterly seasonally adjusted data rather than annual data,[5] (3) the specification (or not) of a proper prior distribution and (4) data ending about 1990 rather than 2017.

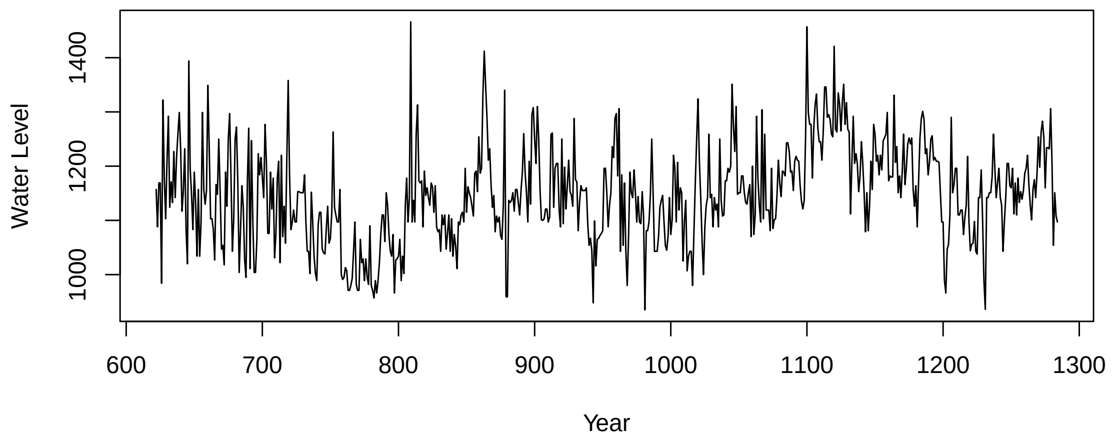
Pai and Ravishanker (1996) report posterior moments of d in five models with alternative specifications of the short memory component, but since the priors are improper there are no marginal likelihoods and hence no posterior distributions of parameters that account for posterior model uncertainty. Assuming equal marginal likelihoods the posterior mean of d is 0.074 and the posterior standard deviation is 0.240 (Pai and Ravishanker 1996, Table I).

Koop et al. (1997) adopt a proper prior distribution and report posterior moments and model probabilities (Koop et al. 1997, Table 1) for 16 alternative short memory component specifications, the Cartesian product of AR and MA components of orders 0, 1, 2, 3. The posterior probability of the ARMA(1,0) specification is 0.616 and the posterior probabilities of all other ARMA specifications are less than 0.07. The posterior mean of the long-memory d parameter is -0.243 and the standard deviation is 0.259. Since their data are quarterly the results for the short memory component would not be comparable even if the data pertained to the same years.

4.3 Nile River data

The Nile River data is comprised of annual minimum river levels in the years 622–1284 based on measurements at the Roda gauge near Cairo. This is a commonly used data set for studying long-memory, going back to seminal work by Hurst (1951), Mandelbrot and Wallis (1968) and Beran (1994). The data are available as part of the R package 'longmemo' (Beran et al. 2011).

[5]To the best of our knowledge the effects of temporal aggregation on ARFIMA models have not been studied in the published literature.



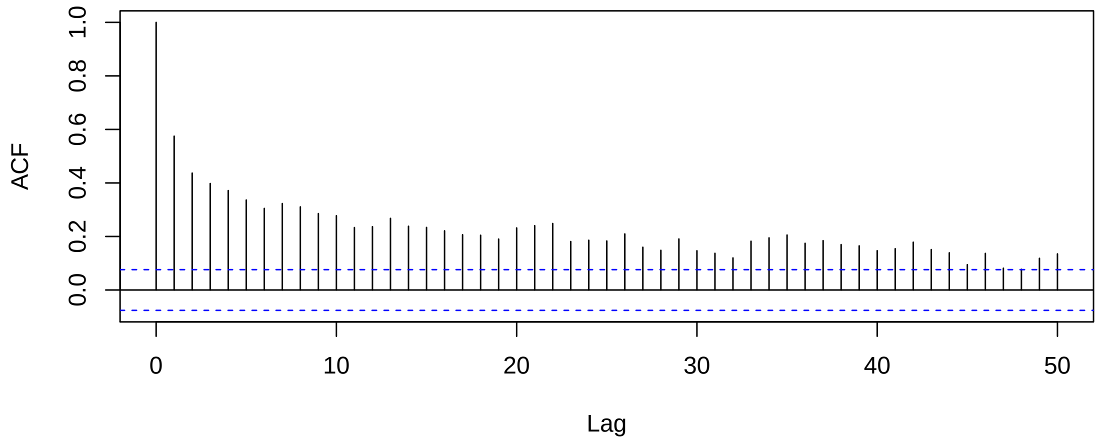


Figure 5: Nile River minimum level, 622–1284 AD. Dashed lines in the autocorrelation plot indicate 95% confidence bands about 0.

Table 7: SABL results for Nile River data ($T = 663$, $N = 2^{14}$).

Short memory

p	q	Computing Time (sec)	Cycles	K	Log ML	NSE
0	0	43.0	6	18	-3,921.40	0.032
1	0	62.1	8	24	-3,791.37	0.045
2	0	117.6	8	28	-3,785.91	0.050
0	1	62.3	8	24	-3,835.07	0.041
1	1	95.0	9	36	-3,777.62	0.073
2	1	250.6	10	69	-3,774.00	0.073
0	2	108.1	9	28	-3,814.70	0.053
1	2	236.0	10	66	-3,773.88	0.105
2	2	499.4	10	107	-3,775.30	0.049

Long memory

p	q	Computing Time (sec)	Cycles	K	Log ML	NSE
0	0	49.2	6	18	-3,765.12	0.030
1	0	69.7	8	25	-3,767.48	0.038
2	0	194.3	9	39	-3,769.90	0.044
0	1	56.5	8	24	-3,767.46	0.053
1	1	85.6	9	36	-3,768.17	0.045
2	1	295.2	9	66	-3,770.55	0.045
0	2	107.2	9	30	-3,769.87	0.044
1	2	219.5	10	64	-3,770.61	0.038
2	2	488.8	10	84	-3,771.62	0.063

Figure 5 shows a time-series plot of the data and the sample autocorrelation function. Autocorrelation decays slowly, characteristic of long-memory. We use the standard benchmark prior discussed in Section 3.3, with $\mu \sim N(1100, 100^2)$ and $\log \sigma^2 \sim N(9, 2^2)$, all default settings for SABL, and consider the same 18 models as in the preceding examples ($p = 0, 1, 2$, $q = 0, 1, 2$, with and without long memory).

Marginal likelihoods, numerical standard error, computing time, and the number of SABL cycles and total Metropolis steps K required for each model are shown in Table 7. The preferred model is $p = q = 0$ with long memory. The best short memory model is ARMA(1,2). The posterior odds ratio in favor of the preferred long-memory model over the best short-memory model is 637:1.

The posterior mean for d in the long memory model with $p = q = 0$ is 0.40 with posterior standard deviation 0.029, which is consistent with other work that has looked at these data (e.g., Beran 1994). Figure 6 shows the model residuals calculated at the posterior mean and their ACF. The residuals are suggestive of a possible structural break around year 722 (the early data seem to be more independent and more variable, as pointed out by, e.g., Beran and Terrin 1996) and perhaps some outliers, notably at years 646, 809 and perhaps

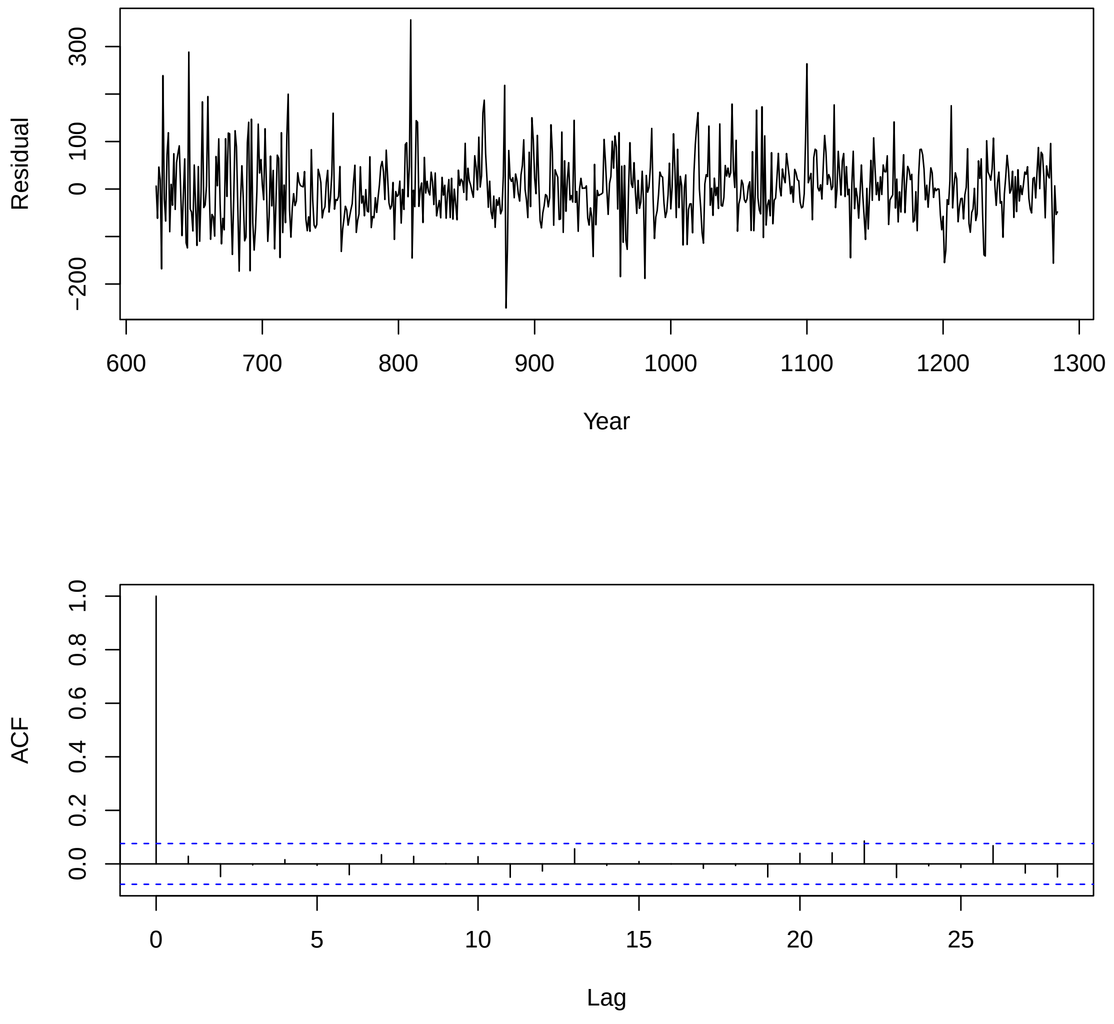


Figure 6: Nile River data, residuals of ARFIMA(0,0.40,0). Dashed lines in the autocorrelation plot indicate 95% confidence bands about 0.

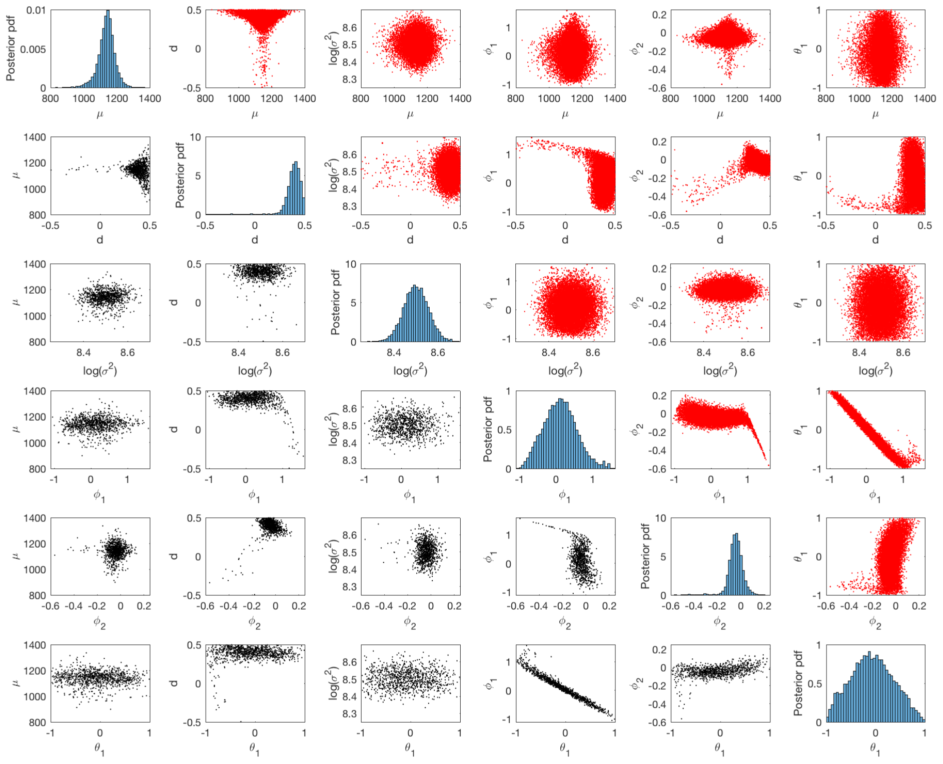


Figure 7: Nile River data, pairwise scatterplots of posterior sample for ARFIMA(2,d,1).

878 (Lévy-Leduc et al. 2011).

Especially for some of the more heavily over-parameterized long-memory models, posterior densities are highly nonspherical. For example, Figure 7 shows pairwise scatterplots of the sample of particles from the posterior for the ARFIMA(2,d,1) model. As in the preceding applications, for values of d near one-half the posterior density for μ becomes less precise, resulting in a triangular shaped joint posterior. Note also the high correlation between ϕ_1 and θ_1, reflecting over-parameterization. The right tail of ϕ_2 appears to be constrained by the stationarity constraint (Condition 1). The larger models require more cycles and many more Metropolis steps to process. Computing time ranges from 49 seconds for the long memory model with $p = q = 0$ to 499 seconds for the model with $p = q = 2$.

4.4 Tree ring data

The Campito tree ring data is another commonly-used data set for studying long memory. These data comprise 5,405 annual observations of tree ring widths of bristlecone pines on Campito Mountain, California from 3435 BC to 1969 AD. Empirical studies include Baillie and Chung (2002), Rea et al. (2011) and Kendal (2017). The data are available as part of the R package 'tseries' (Trapletti and Hornik, 2017).

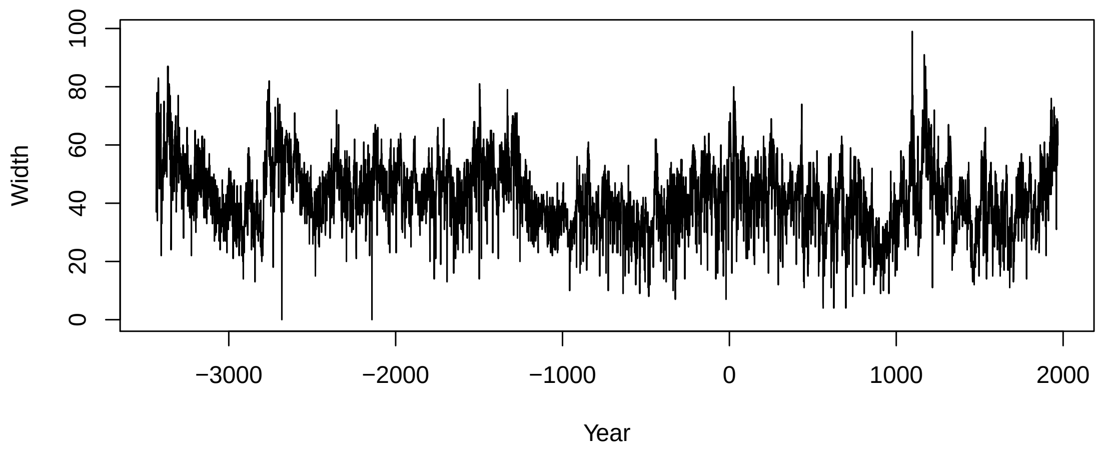
The data are shown in Figure 8 along with the autocorrelation function. As with the Nile River data examined in Section 4.3, autocorrelation decays slowly, suggestive of long memory. We use the standard benchmark prior with $\mu \sim N(42, 5^2)$ and $\log \sigma^2 \sim N(5, 2^2)$. SABL is used with the same settings and the same 18 models are used as before ($p = 0, 1, 2$, $q = 0, 1, 2$, with and without long memory).

Marginal likelihoods, numerical standard error, computing time and total Metropolis steps K required for each model are shown in Table 8. The preferred model is $p = q = 0$ with long memory, while the best of the short memory models is ARMA(2,1). The posterior odds ratio in favor of the preferred long-memory model over the best short-memory model is $(2.25 \times 10^8) : 1$. Total computing time for all 18 models was about 32 hours. Using CPU alone would have required several months (although the workload could be easily distributed across multiple CPU cores, reducing the required computing time substantially).

The posterior mean for d in the preferred long-memory model is 0.45 with a posterior standard deviation 0.01, consistent with other work that has looked at these data (e.g., Ballie and Chung, 2002). The model residuals calculated at the posterior mode and their ACF are shown in Figure 9. Similar comments regarding the nonsphericity of posterior densities for some of the more heavily over-parameterized long-memory models apply here as for the Nile River data, although to a somewhat lesser extent, reflecting the larger sample size. As with the Nile River data, the larger models require more cycles and many more Metroplis steps to process. Computing time ranges from about an hour for the long memory model with $p = q = 0$ model to just under four hours for the model with $p = q = 2$.

5 Context and conclusion

This work began when the editors invited the authors of Geweke and Porter-Hudak (1983) to contribute to this issue of *Journal of Time Series Analysis* marking the 35th anniversary of the publication of that paper. The ARFIMA model of Sowell (1992) has a handful of



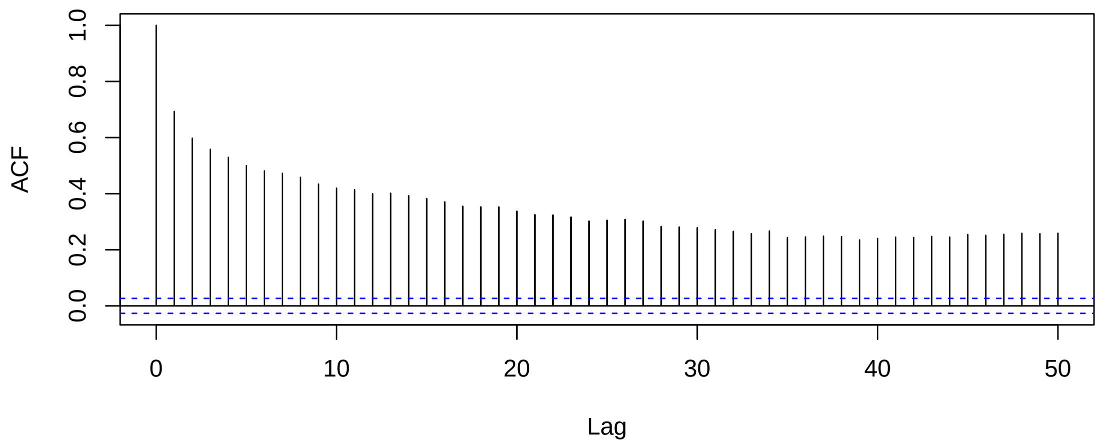


Figure 8: Campito Mountain tree ring widths measured in 0.01 mm, 3435 BC–1969 AD. Dashed lines in the autocorrelation plot indicate 95% confidence bands about 0.

Table 8: SABL results for Campito Mountain tree ring data ($T = 5405$, $N = 2^{14}$).

Short memory

p	q	Computing Time (sec)	Cycles	K	Log ML	NSE
0	0	2,361.3	7	21	-20,998.55	0.034
1	0	3,662.5	9	27	-19,229.72	0.037
2	0	4,625.7	11	34	-19,090.86	0.055
0	1	3,728.0	10	30	-19,966.03	0.066
1	1	4,971.8	12	36	-18,983.27	0.061
2	1	9,436.1	13	73	-18,928.03	0.072
0	2	4,449.9	12	36	-19,615.63	0.044
1	2	7,786.7	13	60	-18,936.33	0.071
2	2	12,594.9	14	104	-18,929.70	0.113

Long memory

p	q	Computing Time (sec)	Cycles	K	Log ML	NSE
0	0	3,576.9	8	24	-18,908.80	0.049
1	0	4,881.2	10	34	-18,912.51	0.050
2	0	6,375.0	12	49	-18,914.38	0.045
0	1	4,463.9	10	31	-18,912.50	0.045
1	1	4,841.9	10	40	-18,912.88	0.048
2	1	8,969.2	12	78	-18,915.42	0.047
0	2	5,498.7	12	40	-18,914.44	0.066
1	2	8,503.1	12	74	-18,915.37	0.114
2	2	14,257.4	12	136	-18,916.40	0.058

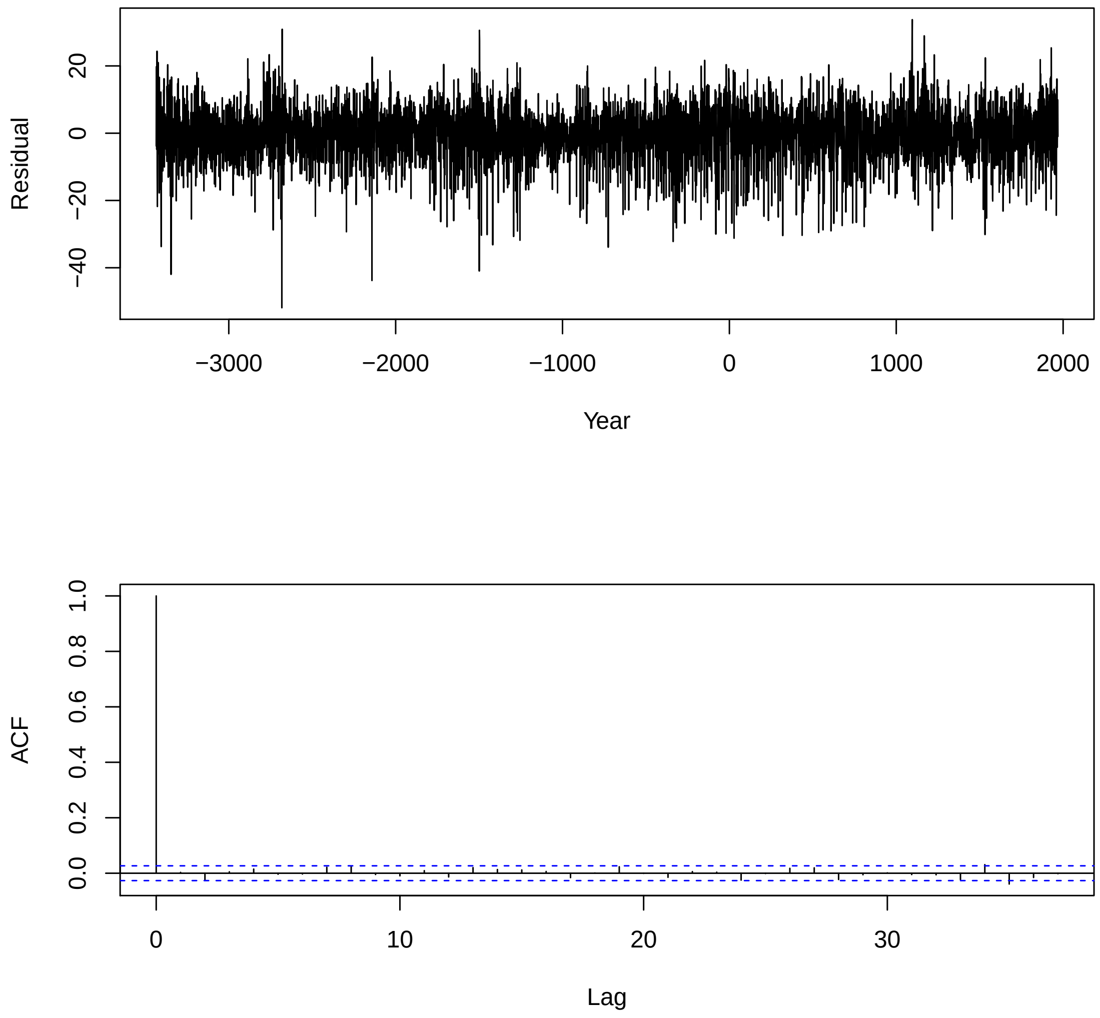


Figure 9: Campito Mountain tree ring data, residuals of ARFIMA(0,0.45,0).

parameters but an analytically and computationally complex likelihood function, features that speak to the comparative advantages of the SABL algorithm for Bayesian inference developed in Durham and Geweke (2015) and Geweke and Durham (2018). SABL is reliable when the likelihood function can be evaluated accurately, and the successful solution of this problem became the core analytical contribution of the paper.

Since 1983 the demands for understanding long time series of data related to climate change, including meteorological, biological and hydrological series, have increased by orders of magnitude. Recall that the simple approach in Geweke and Porter-Hudak (1983) is practical for annual time series of this kind regardless of length. The methods developed in this paper, implemented using GPUs, make ARFIMA models practical for studying these data. Given the tools available, current approaches are forced to use approximations to the likelihood function in these applications (Hsu and Breidt 2003; Graves et al. 2015). As illustrated in Geweke and Porter-Hudak (1983) the approximations typically used in these studies, including truncation of lag operators and finite Fourier transforms, can be treacherous precisely because of the long memory inherent in the model. Thus the methods developed in this paper are timely for these critical applications related to climate change.

References

[1] Baillie RT, Chung CF, Tieslau MA. 1996. Analysing inflation by the fractionally integrated ARFIMA-GARCH model. *Journal of Applied Econometrics* 11: 23 - 40.

[2] Baillie RT, Chung SK. 2002. Modeling and forecasting from trend-stationary long memory models with applications to climatology. *International Journal of Forecasting* 18: 215 - 226.

[3] Beran J. 1994. *Statistics for Long Memory Processes.* New York: Chapman & Hall.

[4] Beran J, Terrin N. 1996. Testing for a change of the long-memory parameter. *Biometrika* 83: 627 - 638.

[5] Beran J, Whitcher B, Maechler M. 2011. longmemo: Statistics for long-memory processes—data and functions (version 1.0-0). `https://CRAN.R-project.org/package=longmemo`.

[6] Chib S, Greenberg E. 1994. Bayes inference in regression models with ARMA(p,q) errors. *Journal of Econometrics* 64: 183 - 203.

[7] Doornik JA, Ooms M. 2003. Computational aspects of maximum likelihood estimation of autoregressive fractionally integrated moving average models. *Computational Statistics and Data Analysis* 42: 333 - 348.

[8] Doornik JA, Ooms M. 2004. Inference and forecasting for ARFIMA models with an application to US and UK inflation. *Studies in Nonlinear Dyanmics and Econometrics* 8, No. 2, Article 14.

[9] Durbin J. 1960. The fitting of time series models. *International Statistical Review* 28: 233 - 243.

[10] Durbin J, Koopman SJ. 2001. *Time Series Analysis by State Space Methods*. Oxford: Oxford University Press.

[11] Durham G, Geweke J. 2015. Adaptive sequential posterior simulators for massively parallel computing environments, in: I. Jeliazkov , D. J. Poirier (eds.) *Bayesian Model Comparison* (Advances in Econometrics, Volume 34) Emerald Group Publishing Limited, Chapter 1, 1-44.

[12] Geweke, J. 1988. The secular and cyclical behavior of real GDP in 19 OECD countries, 1957–1983. *Journal of Business and Economic Statistics* 6: 479 – 486.

[13] Geweke J. 2005. *Contemporary Bayesian Econometrics and Statistics*. New York: Wiley.

[14] Geweke J. 2016. Sequentially adaptive Bayesian learning for a nonlinear model of the secular and cyclical behavior of US real GDP. *Econometrics*, 4, 10; doi: 10.3390/econometrics4010010

[15] Geweke J, Durham G. 2018. Sequentially adaptive Bayesian learning for inference and optimization. *Journal of Econometrics*, forthcoming.

[16] Geweke J, Porter-Hudak S. 1983. The estimation and application of long memory time series models. *Journal of Time Series Analysis* 4: 221- 238.

[17] Granger CWJ, Joyeux R. 1980. An introduction to long memory time series models and fractional differencing. *Journal of Time Series Analysis* 1: 15-29.

[18] Graves T, Gramacy RB, Franzke CLE, Watkins NW. 2015. Efficient Bayesian inference for natural time series using ARFIMA processes. *Nonlinear Proccesses in Geophysics* 22: 679 - 700.

[19] Golub GH, van Loan CF. 1996. *Matrix Computations*. Baltimore: Johns Hopkins University Press.

[20] Hsu NJ, Breidt FJ. 2003. Bayesian analysis of fractionally integrated ARIMA with additive noise. *Journal of Forecasting* 22: 491 - 514.

[21] Hurst HE. 1951. Long-term storage capacity of reservoirs. *Transactions of the American Society of Civil Engineers* 116: 770 - 779.

[22] Hwang Y. 2001. Relationship between inflation rate and inflation uncertainty. *Economics Letters* 73: 179 - 186.

[23] Kendal WS. 2017. 1/f Noise and multifractality from bristlecone pine growth explained by the statistical convergence of random data. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 473. DOI: 10.1098/rspa.2016.0586.

[24] Koop G, Ley E, Osiewalski J, Steel MFJ. 1997. Bayesian analysis of long memory and persistence using ARFIMA models. *Journal of Econometrics* 76: 149 - 169.

[25] Levinson N. 1947. The Wiener RMS error criterion in filter design and prediction. *Journal of Mathematical Physics* 25:261 - 278.

[26] Lévy-Leduc C, Boistard H, Moulines E, Taqqu M, Reisen V. 2011. Robust estimation of the scale and of the autocovariance function of Gaussian short and long-range dependent processes. *Journal of Time Series Analysis* 32:132 - 156.

[27] Ling SG. 2003. Adaptive estimators and tests of stationary and nonstationary short- and long-memory ARFIMA-GARCH models. *Journal of the American Statistical Association* 98: 955 - 967.

[28] Liu JS, Chen R. 1998. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association* 93: 1032 - 1044.

[29] Mandelbrot BB, Wallis JR. 1968. Noah, Joseph, and operational hydrology. *Water Resources Research* 4: 909 - 918.

[30] Nonejad N. 2015. Particle Gibbs with ancestor sampling for stochastic volatility models with heavy tails, in mean effects, leverage, serial dependence and structural breaks. *Studies in Nonlinear Dyunamics and Econometrics* 19: 561 - 584.

[31] Pai JS, Ravishanker N. 1996. Bayesian modelling of ARFIMA processes by Markov Chain Monte Carlo methods. *Journal of Forecasting* 15: 63 - 82.

[32] Rea W, Reale M, Brown J, Oxley L. 2011. Long memory or shifting means in geophysical time series? Mathematics and computers in simulation 81:1441 - 1453.

[33] Rogers LCG. 1997. Arbitrage with fractional Brownian motion. *Mathematical Finance* 7: 95 - 105.

[34] Samuelson, P.A. 1939. Interactions between the multiplier analysis and the principle of acceleration. *Review of Economics and Statistics* 21: 75–78.

[35] Sela RJ, Hurvich CM. 2009. Computationally efficient methods for two multivariate fractionally integrated models. *Journal of Time Series Analysis* 30: 631 - 651.

[36] Sowell F. 1992. Maximum likelihood estimation of stationary univariate fractionally integrated time series models. *Journal of Econometrics* 53: 165 - 188.

[37] Trapletti A, Hornik K. 2017. tseries: Time series analysis and computational finance (version 0.10-44). `https://CRAN.R-project.org/package=tseries`.