Computationally Efficient Gaussian Maximum Likelihood Methods for Vector ARFIMA Models

Rebecca J. Sela and Clifford M. Hurvich*
New York University

July 31, 2008

Abstract: In this paper, we discuss two distinct multivariate time series models that extend the univariate ARFIMA model. We describe algorithms for computing the covariances of each model, for computing the quadratic form and approximating the determinant for maximum likelihood estimation, and for simulating from each model. We compare the speed and accuracy of each algorithm to existing methods and measure the performance of the maximum likelihood estimator compared to existing methods. We also fit models to data on unemployment and inflation in the United States, to data on goods and services inflation in the United States, and to data about precipitation in the Great Lakes.

Contents

*Thanks to Alexei Karlovich, Torsten Ehrhardt, and Lei Li for helpful advice regarding determinant computation. Thanks to Rohit Deo, Amy Finkbiner and Karen Paul for many helpful comments. All mistakes are my own.

1 Introduction

While time series often come in groups that could be analyzed together, much time series work focuses on the analysis of univariate time series. This has led to the creation of a wide variety of models that can handle many types of correlation structure, including long memory processes which have slowly-decaying autocorrelations (see Granger and Joyeux (1980) and Hosking (1981) for some of the earliest work in this area). In the case of multiple stationary time series, the most widely-used model is a vector autoregressive-moving average (ARMA) model, in which the autocorrelations of each component series and therefore the cross-correlations between pairs of series decay exponentially fast. Such a restriction on the autocorrelations has been found to be too strong in a variety of univariate cases. Instead, many authors suggest applying a long memory model such as an fractionally integrated ARMA (ARFIMA) model, to such time series (see Baillie (1996) for a discussion of applications to geophysical sciences, macroeconomics, prices, and more). In this paper, we discuss two vector versions of the ARFIMA model, both of which are multivariate generalizations of the traditional univariate ARFIMA model.

To make a time series model suitable for practical use, it is desirable to be able to determine its covariance structure, estimate its parameters through maximum likelihood, and simulate from it. Ideally, all of these tasks must be done both quickly and precisely. In the case of univariate and multivariate ARMA models, the conditional likelihood function, in which some initial values

of the time series are assumed to be fixed, provides a simple approximation to the full likelihood function. The application of the EM algorithm of Dempster et al. (1977) to the state-space representation of a multivariate ARMA process provides an alternative estimation method. (See Hamilton (1994, chapter 11 and section 13.4) for more information.) However, neither of these methods is applicable to long memory models, because one cannot condition on a finite number of observations and because long memory models do not have state space representations (Baillie (1996) and others). For univariate ARFIMA models, more recent work (Bertelli and Caporin, 2002; Deo et al., 2006; Davies and Harte, 1987) has found efficient methods for computing the autocovariances of an ARFIMA process, computing the likelihood function of an ARFIMA process, and simulating from an ARFIMA process. Previously, Sowell (1989a,b) described exact methods for computing the covariances from one particular type of vector ARFIMA model and for computing the exact likelihood and simulating from general multivariate processes. However, his calculation methods are often slow, with the likelihood and simulation calculations taking $O(T^2)$ time, where T is the number of observations in the dataset; reliance on these algorithms makes the use of vector ARFIMA models prohibitively expensive for large datasets. In this paper, we present methods which will accomplish the tasks of computation and simulation fast enough to make the use of vector ARFIMA models more practical.

Beyond the application of the newly proposed methods to estimating vector ARFIMA models, our algorithms for computing the quadratic form and for simulation are applicable to any multivariate time series for which the covariance structure is known. This provides additional value to people who wish to compute the quadratic form of or to simulate from a multivariate time series that does not have state space representations or other methods for exact computation.

To make our notation precise, suppose that we observe $k = 1, ..., K$ time series over $t = 1, ..., T$ periods, with X_{kt} denoting the t^{th} period of the k^{th} time series and $X_t = (X_{1t}, ..., X_{Kt})'$. In this paper, unless stated otherwise, we will assume that all time series are stationary with zero mean. We will consider these observations grouped either by series or by time. In the former case, we will write $X = (X_{1\cdot}, ..., X_{K\cdot})'$, where $X_{k\cdot} = (X_{k1}, ..., X_{kT})'$. In the latter case, we will write $\tilde{X} = (X_1', ..., X_T')'$. Notice that $X = P\tilde{X}$, where P is a permutation matrix. Suppose we have a model for X described by a vector of parameters, θ. Define $\Omega(\theta)$ as the $KT \times KT$ matrix, $\mathrm{Cov}(X)$. Note that $\Omega(\theta)$ consists of K^2 blocks, with the (i, j) block equal to the $T \times T$ matrix containing $E(X_{i\cdot}X_{j\cdot}')$. Since the multivariate process is stationary, each block is Toeplitz, with the same number along each diagonal. Alternatively, we may consider $\tilde{\Omega}(\theta) = \mathrm{Cov}(\tilde{X})$. Then, $\tilde{\Omega}(\theta)$ consists of T^2 blocks containing $\mathrm{Cov}(X_t, X_{t-r})$, arranged in a Toeplitz fashion, so that the blocks along each diagonal are identical. We may then write the Gaussian log likelihood as:

$$l(\theta|X) = -\frac{1}{2}\log|\Omega(\theta)| - \frac{1}{2}X'\Omega(\theta)^{-1}X \quad (1)$$

$$= -\frac{1}{2}\log|\tilde{\Omega}(\theta)| - \frac{1}{2}\tilde{X}'\tilde{\Omega}(\theta)^{-1}\tilde{X} \quad (2)$$

We will discuss how to compute the autocovariances which could be used to create $\Omega(\theta)$ and

$\tilde{\Omega}(\theta)$ in section 5, how to compute the term containing the quadratic form in section 6, and how to approximate the determinant term in section 7.

In section 2 we provide some background on long memory processes and a discussion of two distinct models that appear as we move from the univariate case to the multivariate case. Section 3 describes block circulant and block Toeplitz matrices, which are the basis of many of the methods we will use. In section 4, we discuss existing computational methods that have been applied to maximum likelihood estimation in multivariate ARFIMA models. In sections 5, 6, and 7, we present computationally efficient methods for the distinct tasks in estimating vector ARFIMA models with maximum likelihood: computing covariances, computing the quadratic form in the likelihood function, and computing the determinant in the likelihood function. The methods for computing the covariances and computing the quadratic form are extensions of univariate algorithms which have been discussed in the time series literature, and we review those methods in the corresponding sections. In section 8, we discuss simulating from a vector ARFIMA process. This section also includes a description of the existing method for univariate ARFIMA processes that our algorithm extends. After presenting these methods, we discuss the performance of the maximum likelihood estimator in section 9. We apply our estimator to econometric and meteorological data in section 10. Section 11 concludes.

2 Long Memory Processes

2.1 Univariate ARFIMA Processes

A univariate long memory process with differencing parameter, d, is one in which the autocovariances, $\omega(r)$, decay at a hyperbolic rate; that is, $\lim_{|r|\to\infty} \frac{\omega(r)}{|r|^{2d-1}}$ is constant. Equivalently, a univariate long memory process is a process in which the spectral density, defined as $f(\lambda) = \frac{1}{2\pi}\sum_{r=-\infty}^{\infty} \omega(r)\exp(-ir\lambda)$, obeys $f(\lambda) \sim C|1-e^{-i\lambda}|^{-2d}$ when λ is near 0. We must have $0 \leq |d| < \frac{1}{2}$, for this spectrum to be integrable and for the process to be stationary; the process is said to have short memory when $d = 0$ and long memory for any $0 < |d| < \frac{1}{2}$. Long memory processes have long been studied in the literature. (See Granger and Joyeux (1980) and Hosking (1981) for early work on long memory and Brockwell and Davis (1993, section 13.2) or Baillie (1996) for more background.)

The simplest case of long memory is fractionally integrated white noise, $\{y_t\}$. Fractionally integrated white noise is defined by $(1-L)^d y_t = \epsilon_t$, where ϵ_t is white noise with variance σ^2, and L is the lag operator, $Lx_t = x_{t-1}$. Even though d is not an integer, we can define $(1-L)^d$

by the binomial expansion:

$$\begin{aligned}(1-L)^d &= \sum_{j=0}^{\infty}(-1)^j\binom{d}{j}L^j \\ \binom{d}{j} &= \frac{d(d-1)\cdots(d-j+1)}{j!}\end{aligned}$$

The spectral density of $\{y_t\}$ is given by $f_y(\lambda) = \frac{\sigma^2}{2\pi}|1-e^{-i\lambda}|^{-2d}$. The coefficients of the infinite order autoregressive representation, the infinite order moving average representation, and the autocovariances of $\{y_t\}$ are available in closed form (Brockwell and Davis, 1993, see, for example, [Theorem 13.2.1).

ARFIMA models are a more general class of univariate long memory processes. A time series, $\{x_t\}$, follows an $ARFIMA(p,d,q)$ process if it can be written as $a(L)(1-L)^d x_t = b(L)\epsilon_t$, where $a(L)$ and $b(L)$ are lag polynomials of degree p and q respectively. We generally assume that $a(L)$ and $b(L)$ have no common roots and that all of their roots are outside the unit circle. Together with the assumption that $|d| < \frac{1}{2}$, these conditions ensure that $\{x_t\}$ is a stationary and invertible process. Notice that we may think of $\{x_t\}$ in two different ways that are equivalent in the univariate case but will not be equivalent for multivariate models. First, $\{x_t\}$ is an $ARMA(p,q)$ process driven by fractionally integrated white noise, which can be written as:

$$a(L)x_t = b(L)[(1-L)^{-d}\epsilon_t]$$

Second, we may describe $\{x_t\}$ as an ordinary $ARMA(p,q)$ process which has been fractionally integrated:

$$x_t = (1-L)^{-d}\left(\frac{b(L)}{a(L)}\epsilon_t\right)$$

Since the composition of linear filters is commutative in the univariate case, the two descriptions are identical.

2.2 Vector ARFIMA Processes

The composition of linear filters does not commute in the multivariate case, so there are multiple possible extensions of a univariate ARFIMA process to a vector ARFIMA process. In this paper, we will focus primarily on models with autoregressive but not moving average components, because of the additional complications associated with moving average components, particularly in a multivariate setting (see Dunsmuir and Hannan (1976, page340) for a description of the structure needed to identify the parameters in vector ARMA models). Because a vector ARMA model can be written as a vector AR model (Hamilton, 1994, page 259), many of our results generalize to models with MA components; we will identify cases in which that occurs.

Let $A(L) = A_0 + A_1L + ... + A_pL^p$, where A_0 is the $K \times K$ identity matrix, I_K, and $A_1, ..., A_p$ are any matrices such that $|A(L)|$ has all of its roots outside the unit circle. If $p = 1$, this

condition is equivalent to the requirement that A_1 has all of its singular values less than 1, or equivalently that all of the eigenvalues of $A^T A$ are less than one. Let $D(L)$ be the diagonal matrix with diagonal entries $(1-L)^{d_1}, ..., (1-L)^{d_K}$, where $d_1, ..., d_K \in (-\frac{1}{2}, \frac{1}{2})$, to ensure stationarity and invertibility. Let $\{\epsilon_t\}$ be a sequence of K-variate white noise, with $E(\epsilon_t \epsilon_s') = 0$ when $t \neq s$ and $E(\epsilon_t \epsilon_t') = \Sigma$, with Σ positive definite. Given the parameters $D(L)$, $A(L)$ and Σ, we may define two distinct vector ARFIMA models; versions of the models including moving average components were presented by Lobato (1997).

In the first model, called Model A by Lobato, we have:

$$A(L)D(L)X_t = \epsilon_t$$

We may understand the properties of the process, X_t, by defining it in two steps. First, define $X_t = D(L)^{-1} Z_t$, so that $X_{k\cdot} = (1-L)^{-d_k} Z_{k\cdot}$. Then, assume that $\{Z_t\}$ follows a vector autoregressive (VAR) model, $A(L)Z_t = \epsilon_t$. Combining these two parts, we see that X_t is a fractionally integrated vector autoregression, which we will called a FIVAR model in this paper. When we wish to specify p and $\vec{d} = (d_1, ..., d_K)$, we will call this a $FIVAR(p, \vec{d})$ model.

Permuting the matrices $A(L)$ and $D(L)$ gives us what Lobato calls Model B:

$$D(L)A(L)X_t = \epsilon_t$$

This model is a vector autoregressive model, $A(L)X_t = Y_t$, driven by fractionally integrated white noise, $D(L)^{-1}\epsilon_t$. In this paper, we will refer to this model as a $VARFI(p, \vec{d})$ model, where p is the order of the lag polynomials in $A(L)$ and $\vec{d}$ is the vector of differencing parameters, as before.

In the univariate case, these two models are identical. Since the composition of $A(L)$ and $D(L)$ is not necessarily commutative, however, these models differ in most cases when $K > 1$. The distinction between the models is also apparent when we write down the the spectral densities of the models:

$$\begin{aligned} f_{FIVAR}(\nu) &= \frac{1}{2\pi} D(e^{-i\nu})^{-1} A(e^{-i\nu})^{-1} \Sigma (A(e^{-i\nu})^{-1})^* (D(e^{-i\nu})^{-1})^* \\ f_{VARFI}(\nu) &= \frac{1}{2\pi} A(e^{-i\nu})^{-1} D(e^{-i\nu})^{-1} \Sigma (D(e^{-i\nu})^{-1})^* (A(e^{-i\nu})^{-1})^* \end{aligned}$$

These spectral densities are identical when the matrices describing the linear filters, $D(\cdot)$ and $A(\cdot)$, commute. In particular, they are identical when $D(L)$ is a scalar multiple of the identity matrix; this occurs when all of the series have equal differencing parameters. Also, they are identical when $A(L)$ and Σ are both diagonal; in that case, the individual series, $X_{k\cdot}$, are uncorrelated univariate ARFIMA series.

In Figures 1, 2, and 3, we plot the autocovariance sequences and cross-covariance sequences of $FIVAR(1, \vec{d})$ and $VARFI(1, \vec{d})$ processes with identical $A(L)$, Σ, and d. The covariance

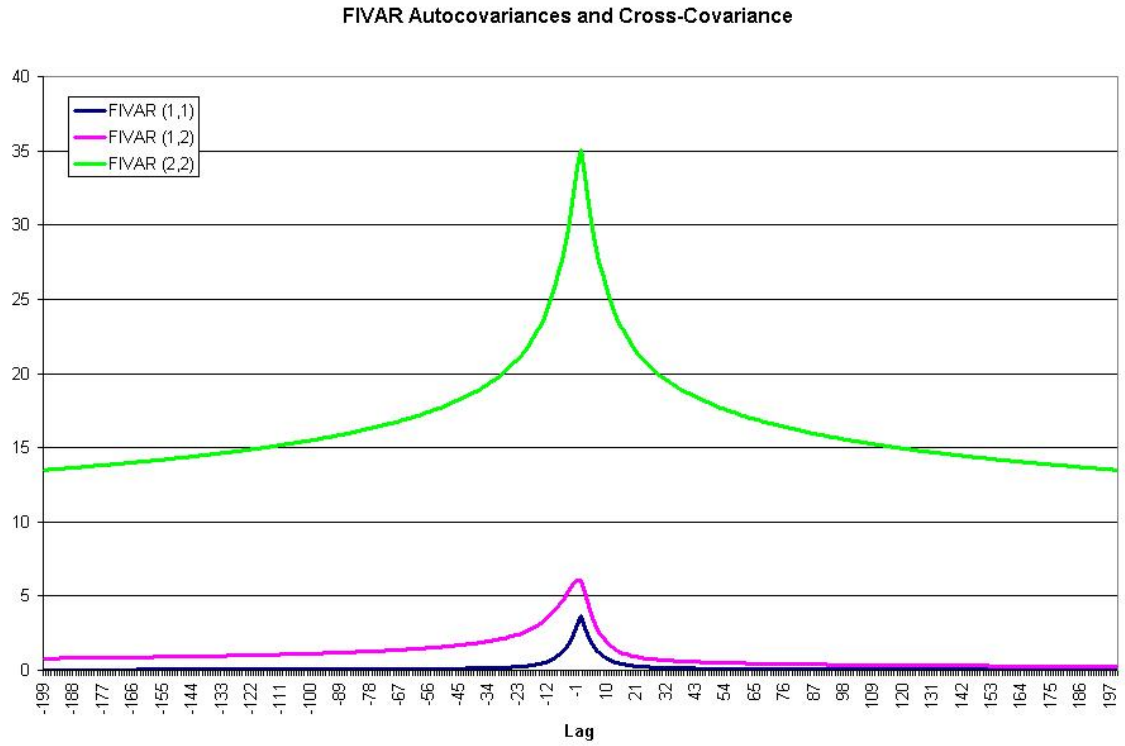

Figure 1: The theoretical autocovariance sequences and cross-covariance sequence of a $FIVAR(1, \vec{d})$ process with parameters $d = (0.1, 0.4)$, $A_1 = (0.7, 0.1, 0.2, 0.6)$, and $\Sigma = (1, .5, .5, 2)$ for lags -199 to 199.

sequences differ dramatically. The autocovariance sequences of the two variables decay more rapidly in the VARFI process than in the FIVAR process. The cross-covariance sequences show an even larger difference; the FIVAR process shows much more asymmetry in the cross-covariances.

Besides producing different autocovariance sequences, the two models differ in their implications; a FIVAR model cannot produce anything like fractional cointegration because the stationary VAR series are integrated separately. However, in most cases, a VARFI model will have linear combinations of X_t and up to p lags which are integrated of a lower order. Extending the analysis of Lobato (1997, page 141) from the bivariate case to a general multivariate case, we give a simple formula that describes the cointegrating relationships. Let $A_{\cdot,k}(L)$ be the k^{th} row of $A(L)$. Then, $A_{\cdot,k}(L)X_t = (1-L)^{d_k}\epsilon_{kt}$. Suppose that $d_k < \max(\vec{d})$ and at least two elements of $A_{\cdot,k}(L)$ are non-zero, and that the corresponding X_{kt} are integrated of order $\max(\vec{d})$. Then $A_{\cdot,k}(L)X_t$ is a linear combination of present and past variables which is fractionally integrated

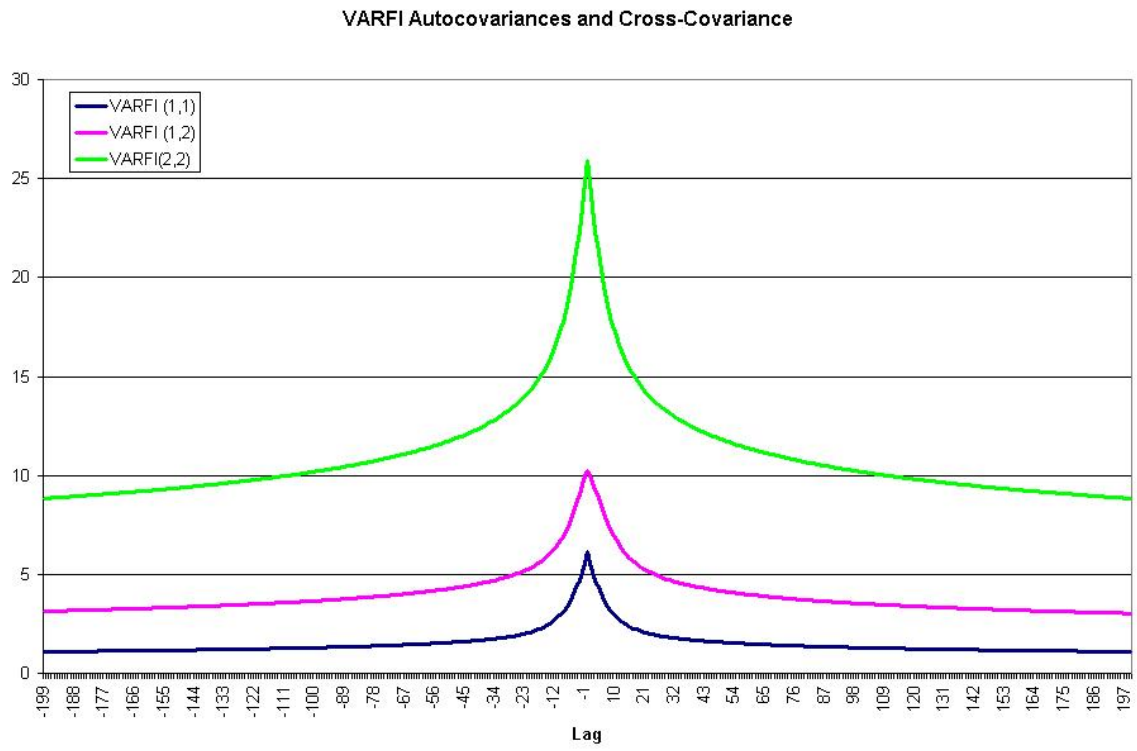

Figure 2: The theoretical autocovariance sequences and cross-covariance sequence of a $VARFI(1, \vec{d})$ process with parameters $d = (0.1, 0.4)$, $A_1 = (0.7, 0.1, 0.2, 0.6)$, and $\Sigma = (1, .5, .5, 2)$ for lags -199 to 199.

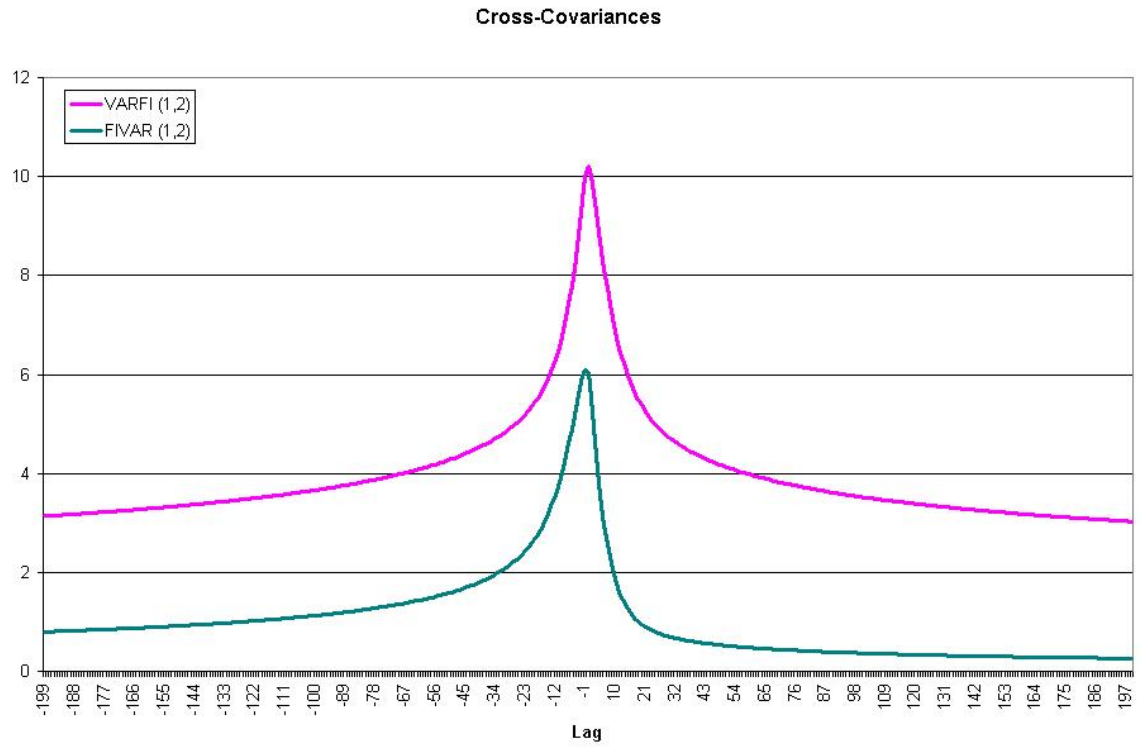

Figure 3: The theoretical cross-covariance sequences of a $FIVAR(1, \vec{d})$ process and a $VARFI(1, \vec{d})$ process, both with parameters $d = (0.1, 0.4)$, $A_1 = (0.7, 0.1, 0.2, 0.6)$, and $\Sigma = (1, .5, .5, 2)$ for lags -199 to 199.

of a lower order than the individual variables are. This relationship will include both present and past values of the variables; since A_0 is the identity matrix, exactly one variable will enter with its present value.

True fractional cointegration occurs when there is some vector, a, such that $a'X_t$ is fractionally integrated of a lower order than any of the elements of X_t. Unlike the relationship we found for VARFI models, this relationship depends only on contemporaneous values of X_t. To produce true cointegration in a FIVAR model, we must include an additional linear filter in our description of the series (Sowell (1989b) uses this method as well). We motivate this addition through the simple bivariate fractional cointegration model of Robinson and Hualde (2003); Hualde and Robinson (2007). Their model can be written as:

$$D(L)VX_t = \epsilon_t \tag{3}$$

where $V = \begin{pmatrix} 1 & -\nu \\ 0 & 1 \end{pmatrix}$; unlike them, we do not assume that $\epsilon_{kt} = 0$ when $t < 0$, because we consider only stationary cases. We may generalize the formulation in (3) by applying the V matrix to a FIVAR model, yielding a cointegrated FIVAR model:

$$A(L)D(L)VX_t = \epsilon_t$$

where V is a matrix with ones along the diagonal and which is block diagonal according to which sets of series are cointegrated; see Sowell (1989b, section 5) for more details. In the case of a bivariate model, Sowell defines $V = \begin{pmatrix} 1 & 0 \\ \nu & 1 \end{pmatrix}$. In our analysis of cointegration in sections 5.4 and 7.4, any parameterization of the cointegrating matrix could be used. In our parameter estimation, we will use the parameterization of Sowell for identification. Then, the spectral density of multivariate cointegrated time series is given by:

$$f_{coint}(\nu) = \frac{1}{2\pi} V^{-1} D(e^{-i\nu})^{-1} A(e^{-i\nu})^{-1} \Sigma (A(e^{-i\nu})^{-1})^* (D(e^{-i\nu})^{-1})^* (V^{-1})^*$$

One could also introduce the V matrix into a VARFI model, though it would not generally lead to cointegration, because the series $A(L)^{-1}D(L)^{-1}\epsilon_t$ may all have the same order of integration even before the addition of V.

Thus far, we have assumed that all series have mean zero. In practice, it is likely that each time series will have an unknown mean. Consider the series $Y_t = X_t + \vec{\mu}$, where X_t follows one of the models with mean zero discussed above. A variety of possibilities exist for the estimation of the parameters of X_t and the estimation of μ. A common approach in the literature (for example Brockwell and Davis, 1993, page 238) is to subtract the sample mean from each time series $Y_{k.}$, and to proceed with estimation based on the demeaned observations. However, the variance of the sample mean of a long memory process is $O(\frac{1}{n^{1-2d}})$, where d is the differencing parameter; thus, when $d > 0$, the variance declines more slowly than the traditional short memory variance

of the mean, $O(\frac{1}{n})$. Despite these problems, demeaning is straightforward, and we will use this method in our data analysis. An alternative method is to use restricted maximum likelihood (REML) as described by Harville (1977). In REML, the data is transformed to remove nuisance parameters, and then maximum likelihood is applied to the transformed data. In the case where the means of the individual time series are the only nuisance parameters, it is enough to take the first difference of each time series individually. Because differencing decreases each d_k by one, this method should be applied when we may assume that the original data has each $d_k \in (0.5, 1.5)$. One could also include the mean directly as part of maximum likelihood estimation. We will not pursue REML or inclusion of the mean further in this paper.

It is more common in the literature (for example Sowell, 1989b; Hosoya, 1996; Martin and Wilkins, 1999; **?**; Ravishanker and Ray, 1997, 2002) to analyze FIVAR models. Tsay (2007) is a notable exception. We will present algorithms for computing the covariances of FIVAR and VARFI models in sections 5.2 and 5.3, respectively. In section 7.1, we present an algorithm for approximating $|\Omega|$ which can be used with either FIVAR or VARFI models; section 7.3 contains a second algorithm which can be used only for VARFI models. The algorithms which we will present for computing the quadratic form, $X'\Omega^{-1}X$, and simulating from a multivariate time series apply to either of the models, because they depend only on knowing the covariance structure.

3 Block Circulant and Toeplitz Matrices

We begin by discussing some properties of circulant and Toeplitz matrices. These properties will be integral to many of the computational methods we will present. First, we recall the definitions of these two types of matrices. A Toeplitz matrix is one in which all of the elements along each diagonal are constant. That is, the value of element A_{ij} depends only on $i - j$. The covariance matrix of a sequence of observations of a univariate time series, $x = (x_1, ..., x_T)'$, is a symmetric Toeplitz matrix. In general, Toeplitz matrices need not be symmetric. A circulant matrix is a matrix in which each row shifts the elements of the previous row one space to the right and moves the right-most element to the beginning of the row; a circulant matrix is a special case of a Toeplitz matrix.

Once we begin to consider multiple time series, we must use block matrices, that is, a matrices that can be partitioned into square blocks, each of which has a certain property. A block circulant matrix is a matrix which can be partitioned into blocks, each of which is a circulant; a block Toeplitz matrix is defined analogously. In this paper, we will generally consider $KT \times KT$ matrices which can be partitioned into K^2 Toeplitz or circulant blocks, each of which is of dimension $T \times T$.

In this section, we will present suggestions for the storage of block circulant and block Toeplitz matrices and algorithms for computing powers of block circulant matrices and for multiplying

by block circulant and block Toeplitz matrices. Most of these algorithms are well-known; the algorithm for computing powers of circulant matrices is a new generalization of an algorithm presented by Chan and Olkin (1994) for computing inverses of block circulant matrices.

3.1 Efficient Storage

In this and the following sections, we discuss how we can use the properties of Toeplitz and circulant matrices to make the operations of the algorithms more efficient. In this section, as an introduction to the structure of these matrices, we discuss the simplest way: the repeated elements in each kind of matrix mean that there are more efficient ways to store them than just writing down all the elements.

The most obvious way to store a block circulant matrix would be to store all K^2T^2 elements. However, because a circulant is completely defined by its first row, it is sufficient to store the first row of each block in a $K \times K \times T$ array, which is a dramatic reduction in the required storage space when T is large. In fact, one can store any T elements which uniquely define the first row of a circulant; we actually store the Fourier transform of the first row, as we will discuss in section 3.3.

In addition, it is not efficient to store the entire block Toeplitz matrix, Ω, since it would require the same large amount of space. Any Toeplitz matrix can be completely described by the first row and first column, and we store those elements instead of storing the entire matrix. In particular, we specify the Toeplitz matrix by the vector of elements:

$$[a(T-1,0),...,a(1,0),a(0,0),a(0,1),...,a(0,T-1)],$$

where we number the rows and columns starting at 0. When this Toeplitz matrix is the (i,j) block of Ω, we may describe the elements in relation to the covariances of $\{X_{it}\}$ and $\{X_{jt}\}$.. Note that, in block $A_{i,j}$, the $(0,r)$ element is $\omega_{ij}(-r) = \mathrm{Cov}(X_{i,t}, X_{j,r+t})$, and the $(r,0)$ element is $\omega_{ij}(r) = \mathrm{Cov}(X_{i,t}, X_{j,t-r})$. Thus, the elements of the first row and column as ordered above are simply

$$[a(T-1,0),...,a(1,0),a(0,0),a(0,1),...,a(0,T-1)] = [\omega_{ij}(T-1),...,\omega_{ij}(-(T-1))]$$

To describe a block Toeplitz matrix, we combine all of these vectors of length $2T-1$ into a three-dimensional array of size $K \times K \times (2T-1)$, in which each $K \times K$ layer is $\omega(r) = \mathrm{Cov}(X_t, X_{t-r})$ for $r = -(T-1),...,(T-1)$.

3.2 Computing a power of a block circulant matrix

As we will see, the methods for computing the quadratic form and for simulation both depend on computing a power of a block circulant matrix; the quadratic form requires computing an

inverse, while simulation requires computing a square root. In this section, we describe a fast way to compute an arbitrary power of a matrix, assuming that it is well-defined. The algorithm given in this section is a generalization of the one given by Chan and Olkin (1994), which describes only how to compute the inverse.

We first describe how the α^{th} power of a block circulant matrix, C, could be computed in theory. Let C_{ij} be the (i,j) block of C. The eigenvalue decomposition of that block is $C_{ij} = F^* \Lambda_{ij} F$, where F is the Fourier matrix with entries $F_{jk} = \frac{1}{\sqrt{T}} \exp\left(\frac{2\pi jk\sqrt{-1}}{T}\right)$ and Λ_{ij} is the diagonal matrix with diagonal equal to the Fourier transform of the first row of C_{ij} (see, for example, Brockwell and Davis, 1993, section 4.5). Throughout this section, when we refer to the eigenvalues of a circulant, we order them as in the Fourier transform of the first row of C_j.

We now consider the matrix, C, as a whole. Let L be the $KT \times KT$ matrix consisting of the diagonal blocks, Λ_{ij}. We may write $C = (I \otimes F^*) L (I \otimes F)$, where $\otimes$ is the Kronecker product. Since $(I \otimes F)^* = (I \otimes F^*)^{-1}$, we may write $C^\alpha = (I \otimes F^*) L^\alpha (I \otimes F)$. Thus, it remains only find an expression for L^α.

Notice that L consists of K^2 blocks of size $T \times T$, each of which is zero except on the diagonal. Therefore, we may find a permutation matrix, P, such that $L = PBP'$, where B is a matrix with T blocks of size $K \times K$ along the diagonal and zeroes everywhere else. In particular, we choose P such that the t^{th} block along the diagonal of B consists of the t^{th} elements along the diagonal of each block in L; this moves all K^2T non-zero elements of L to the blocks along the diagonal of B. This is the same permutation matrix described in the introduction. The resulting blocks are not necessarily diagonal or Toeplitz. (See Chan and Olkin (1994, section three) for more details, particularly the graphic on page 94.)

Consider the spectral decomposition, $V_B \Lambda_B V_B^{-1}$, of B. Since B is block diagonal, we may choose V_B to be block diagonal as well. Combining this decomposition with $(I \otimes F)$ and P yields the eigenvector decomposition of C:

$$C = (I \otimes F^*) P V_B \Lambda_B V_B^{-1} P^{-1} (I \otimes F^*)^{-1}$$

The spectral decomposition allows us to compute powers of C in a simple form. To do this, we first find B^α using the spectral decomposition for each block separately. (Though there no structure on the individual blocks in B, finding the eigenvalues is not computationally intensive if K is small.) We then find that $L^\alpha = PB^\alpha P'$. Since B^α is a block diagonal matrix and P is the same permutation matrix, L^α has the same diagonal block structure as L. Multiplying by $(I \otimes F^*)$ and $(I \otimes F)$, we find the formula for C^α:

$$C^\alpha = (I \otimes F^*) P B^\alpha P' (I \otimes F)$$

Not only does this give a method for computing C^α in theory, but it also shows that C^α is block circulant.

In a small number of cases, a block, B_r, of the matrix B might be defective, so that it has

no spectral decomposition. While this means that general powers of B_r cannot be computed, algorithms exist for computing B_r^α for certain α. The inverse, $\alpha = -1$, can be computed using Gaussian elimination, as long as B_r is invertible. When $\alpha = \frac{1}{2}$, the algorithm of Denman and Beavers (1976) can be used to compute a square root. These are the two cases which will be required in this paper. When all of the B_r have spectral decompositions, the algorithm we have presented can be used for any α.

Though the formula above gives a straightforward method for describing C^α, it is not efficient to write down all K^2T^2 elements of C^α nor to multiply by permutation matrices. Instead, we create a $K \times K \times T$ array, Γ, to completely describe C in a way that makes computation simpler. First, we consider what is in each block, B_{rr}, of the block diagonal matrix, B. For a permutation matrix which moves the r^{th} diagonal element of L_{ij} to the (i,j) location in the r^{th} block, the (i,j) element of B_{rr} is the r^{th} eigenvalue of C_{ij}. Thus, as we compute the eigenvalues for each block, C_{ij}, we may store them as $\Gamma(i,j,\cdot)$, so that each column of Γ corresponds to the eigenvalues of one block of C. Once Γ has been stored in this way, B_{rr} is simply $\Gamma(\cdot,\cdot,r)$. Define $\tilde{\Gamma}$ as the array that stores the elements of C^α in the same fashion. Then, since B is block diagonal, $\tilde{\Gamma}$ is obtained from Γ by computing the power of each layer, $\Gamma(\cdot,\cdot,r)$. This yields the following algorithm for obtaining the eigenvalues of the blocks of C^α:

Algorithm 1 Computing a Representation of a Power of a Block Circulant Matrix

- *Create two $K \times K \times T$ arrays, Γ and $\tilde{\Gamma}$, for storage.*
- *Loop over all pairs, (i,j), with $i = 1,...,K$ and $j = 1,...,K$:*
  - *Set $\Gamma(i,j,\cdot)$ to the Fast Fourier Transform of the first row of C_{ij}.*
- *For $r = 1,..,T$, set $\tilde{\Gamma}(\cdot,\cdot,r) = [\Gamma(\cdot,\cdot,r)]^\alpha$.*

The resulting array holds the eigenvalues of the individual blocks of the power of the circulant preconditioner, which can be used for multiplication by C^α as shown in the next section.

3.3 Efficient Multiplication Methods

Multiplying a $T \times T$ matrix by a $T \times 1$ vector, v, requires $O(T^2)$ steps in general. If, however, the matrix, G, is a circulant, we can speed up this multiplication to $O(T \log T)$ steps, again using the fact that $G = F^* \Lambda F$. The following algorithm an be used for efficient multiplication by a circulant:

Algorithm 2 Multiplication by a Circulant, $G = F^* \Lambda F$.

- *Compute Fv as the Fourier transform of v.*
- *Compute ΛFv by multiplication by a diagonal matrix.*
- *Compute $F^*\Lambda Fv$ as the inverse Fourier transform of the previous result.*

Computing the Fourier transforms in the first and third steps takes $O(T \log T)$ operations, while the second step takes only $O(T)$ operations. In total, this multiplication takes $O(T \log T)$ time.

Algorithm 2 can also be used to compute Av, where A is a Toeplitz matrix and v in any vector. The extension to Toeplitz matrices requires circulant embedding. First, we create a $2T \times 2T$ circulant matrix, $\tilde{A}$, with diagonal blocks equal to A and off-diagonal blocks filled in with the elements of A necessary to make the matrix into a circulant. That is, if we number the row and column indices from 0 as before, the first row of $\tilde{A}$ is $[A(0,0), A(0,1), ..., A(0,T-1), A(0,0), A(T-1,0), ..., A(1,0)]$, and the circulant structure defines the remaining elements of $\tilde{A}$. Second, we extend v to a vector of length $2T$, $\tilde{v}$, by appending T zeroes to the end. We may then use Algorithm 2 to compute $\tilde{A}\tilde{v}$. Then, the first T elements of $\tilde{A}\tilde{v}$ are identical to the elements of Av.

Multiplication by circulant and Toeplitz matrices may be extended to multiplication by block circulant and block Toeplitz matrices. This takes advantage of the block-Toeplitz and block-circulant structures to reduce the number of operations required for multiplication to $O(K^2 T \log T)$ steps. Consider the general block matrix, B, with $T \times T$ blocks, B_{ij}, and vector, v, of length TK, partitioned into K subvectors, v_k, of length T. Then, we compute:

$$\begin{pmatrix} B_{11} & \cdots & B_{1K} \\ \vdots & \ddots & \vdots \\ B_{K1} & \cdots & B_{KK} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_K \end{pmatrix} = \begin{pmatrix} B_{11}v_1 + \cdots + B_{1K}v_K \\ \vdots \\ B_{K1}v_1 + \cdots + B_{KK}v_K \end{pmatrix}$$

If the blocks of B are circulants, then each of the multiplications can be computed using the method for multiplying by a circulant. If the blocks of B are Toeplitz, then each of the multiplications can be computed using circulant-embedding. Computing K^2 such multiplications and then adding them up to get the final vector will take $O(K^2 T \log T)$ steps. We will see the usefulness of these multiplication methods in section 6.

4 Previous Computational Methods for Multivariate Models

4.1 Existing approximations to the likelihood of vector ARFIMA models

The most commonly used approximation to the likelihood in the frequency domain is the Whittle approximation first given in Whittle (1963). The estimation is based on the periodogram matrix,

$$I(\lambda) = \frac{1}{2\pi T} \sum_{t=1}^{T} \sum_{s=1}^{T} X_t X_s' \exp(i\lambda(t-s))$$

According to Dunsmuir and Hannan (1976), the log likelihood is approximately a constant plus:

$$-\frac{T}{2} \log|\Sigma| - \frac{1}{2} \sum_{j=1}^{T} tr\left(f^{-1}\left(\frac{2\pi j}{T}\right) I\left(\frac{2\pi j}{T}\right)\right)$$

where $tr(\cdot)$ is the trace operator and f is the spectral density described in 2.2. Hosoya (1996) discusses this approximation in more detail. The first term uses the approximation $|\Omega| = T|\Sigma|$ of Grenander and Szego (1958), which Dunsmuir and Hannan (1976, page 344) note might not work well for small T even in the ARMA case, but which is very easy to compute. We discuss this approximation and some possible modifications in more detail in section 7.1. In the univariate case, Hannan (1970, chapter 6, section 6) notes that the second term is based on the approximation $\Omega^{-1} \approx F\Lambda F^*$, where F is the Fourier matrix in section 3.2 and Λ is a diagonal matrix with $\frac{2\pi}{\sigma^2} f\left(\frac{2\pi j}{T}\right)$ at the (j,j) location. Notice that this approximates the Toeplitz matrix Ω by a circulant matrix. (See also Brockwell and Davis, 1993, Proposition 4.5.2.) Since $f(0)$ may be infinite in long memory models, this approximation may not be as accurate for vector ARFIMA models.

In the time domain, Luceno (1996) finds an approximation to the quadratic form in the likelihood expression; he neglects the determinant because he says (page 605) that importance declines in relation to the importance of the quadratic form for large T. As we will show in section 9, inclusion of an accurate approximation to the determinant can be quite important in the sample sizes we consider. He then finds an asymptotic approximation to Ω^{-1} in terms of "inverse-transpose" autocovariances, δ_i. These inverse-transpose autocovariances are defined by $\delta_i = \delta_{-i}' = \sum_{j=0}^{\infty} \pi_{t+j}' \Sigma^{-1} \pi_j$, for $i \geq 0$, where π_j are the $AR(\infty)$ coefficients and Σ is the innovation covariance for the process X_t. Using these inverse-transpose autocovariances, an exact expression for the quadratic form is given by:

$$X'\Omega^{-1}X = tr(\delta_0 P_0) + 2\sum_{i=1}^{\infty} tr(\delta_i P_i)$$

where he defines

$$P_i = \sum_{t=-\infty}^{\infty} \hat{X}_{t+i} \hat{X}_t'$$

and $\hat{X}_t$ is the observed series for $t = 1, ..., T$ and the forecast or backcast, $E(X_t|X_1, ..., X_T)$, of the series otherwise. While this expression is exact, the forecasts and backcasts may be costly to compute, and the exact sum must be truncated for computational purposes. Therefore, Luceno recommends approximating:

$$\begin{aligned} P_i &\approx \begin{cases} \sum_{t=1}^{T-i} X_{t+i} X_t' & 0 \leq i \leq T-1 \\ 0 & T \leq i \end{cases} \\ P_i &= P_{-i}', i < 0 \end{aligned}$$

This approximation has an error which of the order $\frac{1}{T}$. Using the expression given above, the quadratic form can be computed by truncating the infinite sums. Luceno notes that the inverse-transpose autocovariances "frequently" are of the same model type as the original autocovariances (page 608); that is, the inverse-transpose autocovariances of a scalar ARFIMA model will be the autocovariances of a different ARFIMA model. However, he does not give a general method for computing the inverse-transpose autocovariance sequence, which makes them infeasible for the general case.

Martin and Wilkins (1999) avoid the likelihood functions altogether by applying indirect estimation to FIVAR models. In this approach, they estimate a $VAR(2)$ using the data and then find parameter values for a FIVAR model that lead to simulated data with identical estimates in a $VAR(2)$. We do not pursue this approach, though we note that indirect estimation would benefit from the simulation algorithm we propose in section 8.

4.2 Existing exact likelihood algorithms for vector ARFIMA models

The most comprehensive set of exact methods for maximum likelihood estimation for vector ARFIMA models can be found in two papers of Sowell (1989b,a). The second paper presents algorithms of computing the autocovariances of a vector ARFIMA process of the FIVAR type, while the first paper presents methods for computing the inverse and determinant of a block Toeplitz matrix, which could be associated with any multivariate process.

First, we discuss Sowell's (1989b) algorithm for computing the autocovariances of a FIVAR process. Consider the autocovariances of a $FIVAR(p, \vec{d})$ process, where $\frac{B(L)}{a(L)}$ is the moving average representation of the vector $ARMA(p, q)$ part of the model, with $B(L)$ a matrix of lag polynomials of order at most $(K-1)p + q$ and $a(L)$ a scalar lag polynomial of order at most $H = Kp$. Let v_{ij} be the (i, j) entry of the cointegration matrix described in section 2.2. Sowell (1989b) finds that $\omega_{ij}(s) = \text{Cov}(X_{i,t}, X_{j,t-s})$ can be written as:

$$\omega_{ij}(s) = \sum_{l=-M}^{M} \sum_{m=1}^{H} \sum_{n=1}^{K} \sum_{r=1}^{K} v_{in} v_{jr} \psi_{ij}(l) \zeta_m C(d_i, d_j, H + l - s, \rho_m)$$

with

$$C(w,v,h,\rho) = \Gamma(1-w-v)\left(\rho^{2H}\sum_{m=0}^{\infty}\frac{\rho^m(-1)^{h+m}}{\Gamma(1-w+h+m)\Gamma(1-v-h-m)} + \sum_{n=1}^{\infty}\frac{\rho^n(-1)^{h-n}}{\Gamma(1-w+h-n)\Gamma(1-v-h+n)}\right)$$

and where the ρ_n, ζ_n, and $\psi_{ij}(l)$ satisfy:

$$\begin{aligned} a(\xi) &= \prod_{j=1}^{H}(1-\rho_j\xi) \\ \zeta_j &= \frac{1}{\rho_j\prod_{i=1}^{H}(1-\rho_i\rho_j)\prod_{m=1,m\neq j}^{H}(\rho_j-\rho_m)} \\ \psi_{ij}(l) &= \sum_{h=1}^{K}\sum_{t=1}^{K}\sum_{s=\max(0,l)}^{\min(M,M-l)}\Sigma_{ht}B_{ih}(s)B_{jt}(s-l) \end{aligned}$$

These sums must be evaluated using the hypergeometric function, which has no closed form in general (Weisstein, 2008). While this gives an exact expression for the covariances, the sums are slow to evaluate, as we will show in section 4. Furthermore, Sowell's method does not apply to VARFI models.

An alternative method to compute the covariances of either a FIVAR or a VARFI model is to use the relationship between the spectral density and the autocovariances. For any multivariate time series with cross-spectral density, f, we may compute the autocovariance function as:

$$\omega(h) = \int_{-\pi}^{\pi} e^{ih\lambda} f(\lambda) d\lambda$$

(See, for example, Brockwell and Davis, 1993, section 11.6.) This gives a straightforward method for computing the autocovariance sequence for either type of model. However, as we will show in Tables 2 and 4, it is also a computationally intensive method.

Sowell (1989a) describes methods to compute the determinants, inverses, and simulated realizations of any stationary multivariate process, including a vector ARFIMA process. In this paper, Sowell uses a version of the Durbin-Levinson algorithm (see also Brockwell and Davis, 1993, Proposition 11.4.1) to decompose the autocovariance matrix $\Omega = \mathrm{Var}(\tilde{X})$, where $\omega(j) = \mathrm{Cov}(X_t, X_{t-j})$, into a series of matrices that are useful for computation.

Algorithm 3 Sowell/Durbin-Levinson Covariance Matrix Decomposition (Sowell, 1989a). *Set*

the initial values:

$$\begin{aligned} v(0) &= \bar{v}(0) = \omega(0) \\ D(1) &= \omega(1) \\ \bar{D}(1) &= \omega(-1) \end{aligned}$$

For $n = 1, ..., T$ and $k = 1, ..., n$, compute the following quantities iteratively:

$$\begin{aligned} A(n,n) &= D(n)\bar{v}(n-1)^{-1} \\ \bar{A}(n,n) &= \bar{D}(n)v(n-1)^{-1} \\ A(n,k) &= A(n-1,k) - A(n,n)\bar{A}(n-1,n-k) \\ \bar{A}(n,k) &= \bar{A}(n-1,k) - \bar{A}(n,n)A(n-1,n-k) \\ v(n) &= \omega(0) - \sum_{j=1}^{n} A(n,j)\omega(-j) \\ \bar{v}(n) &= \omega(0) - \sum_{j=1}^{n} \bar{A}(n,j)\omega(j) \\ D(n+1) &= \omega(n+1) - \sum_{j=1}^{n} A(n,n-j)\omega(j) \\ \bar{D}(n+1) &= \omega(-n-1) - \sum_{j=1}^{n} \bar{A}(n,n-j)\omega(-j) \end{aligned}$$

Because all of the $A(n,k)$ must be computed, finding this decomposition requires $O(T^2)$ operations. General algorithms for determinants, inverses, and Cholesky decompositions for general matrices are $O(T^3)$, which means that using this algorithm is an improvement. However, an algorithm which is $O(T^2)$ is still quite slow for many applications. Given this decomposition, various quantities of interest become quite straightforward to compute. The determinant of Ω is simply $\prod_{t=0}^{T-1} |v(t)|$. The inverse is given by $\Omega^{-1} = \bar{\beta}\bar{\beta}'$, where

$$\bar{\beta} = \begin{pmatrix} I_K & -\bar{A}(1,1)' & -\bar{A}(2,2)' & \cdots & -\bar{A}(T-1,T-1)' \\ 0 & I_K & -\bar{A}(2,1)' & \cdots & -\bar{A}(T-1,T-2)' \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & \cdots & & 0 & I_K \end{pmatrix} \begin{pmatrix} \bar{v}(0) & 0 & 0 & \cdots & 0 \\ 0 & \bar{v}(1) & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & & 0 & \bar{v}(T-1) \end{pmatrix}^{-1/2}$$

Notice that computation of the quadratic form, $X'\Omega^{-1}X$, using this representation would require an additional $O(T^2)$ steps, even if the decomposition were already known. Finally, Sowell points out (Result 3) that one can simulate from the distribution of X can be done by drawing a vector $U = (U_1', ..., U_T')'$ of length KT and then defining $X_1 = \bar{v}(0)^{1/2}U_1$ and $X_t = \sum_{j=1}^{t-1} \bar{A}(t-1, t-j)X_j + \bar{v}(t-1)^{1/2}U_t$. This simulation method also requires $O(T^2)$ steps, which would be

particularly problematic if many samples were drawn. All of these methods are exact. However, the computations required are daunting when T is large. In fact, Doornik and Ooms (2003) say that this method is "still rather time consuming" for a dataset in which K is 2 and T is 121. In order for exact maximum likelihood to be feasible for estimating multivariate ARFIMA models, a faster algorithm is needed.

Tsay (2007) applied Sowell's algorithms to VARFI processes and using Sowell's (1989b) expression for the autocovariances of a $VARFI(0, \vec{d})$ process. While this work avoids the slow computations of autocovariances that plagues Sowell's (1989b) algorithm for computing the covariances of a FIVAR process, it does not address the slowness of the Cholesky decomposition.

Chung (2001) presents a method for calculating the impulse response function of a FIVAR process. Also, Ravishanker and Ray (1997, 2002) discuss Bayesian methods for estimating from and forecasting FIVAR processes. We do not pursue either of these computations further.

5 Computing Autocovariances

In this section, we present algorithms for computing the autocovariances of both types of vector ARFIMA processes. First, as background, we describe the univariate splitting algorithm of Bertelli and Caporin (2002) for computing the autocovariances of a univariate ARFIMA model. In sections 5.2 and 5.3, we present fast algorithms for computing the autocovariance sequences of both FIVAR and VARFI processes. After we detail each algorithm, we will show the speed in practice and compare it to existing algorithms.

5.1 Computing the autocovariances of a univariate ARFIMA model

To compute the autocovariance sequence, $\omega(j)$, of an $ARFIMA(p, d, q)$ process with $d \in (-\frac{1}{2}, \frac{1}{2})$, Bertelli and Caporin (2002) write the covariances as the infinite convolution of the autocovariances, $\xi(j)$, of an $ARMA(p, q)$ process, and the autocovariances, $\phi(j)$, of an $ARFIMA(0, d, 0)$. Both of these autocovariance sequences have closed forms or can be computed quickly. Then, the $ARFIMA(p, d, q)$ autocovariances can be written as :

$$\omega(j) = \sum_{h=-\infty}^{\infty} \xi(h)\phi(j-h)$$

Because the autocovariances of an ARMA model decay exponentially fast, they recommend setting the $\xi(h)$ to 0 for $|h| > M$ for large M. A larger value of M may be chosen to increase the accuracy. Then, the computation of these convolutions for $j = 0, ..., T$ can be done quickly using the Fast Fourier Transform. This gives a fast and accurate method to compute the autocovariance sequence in the univariate case.

5.2 FIVAR Covariances

To generalize the univariate splitting algorithm to a FIVAR process, we use the two-step definition of a FIVAR discussed in section 2.2. In this section, for complete generality, we allow for a moving average component as well as an autoregressive component. First, we define Z_t as a vector ARMA process, so that $A(L)Z_t = B(L)\epsilon_t$, with $\text{Cov}(\epsilon_t) = \Sigma$. We assume that $A(L)$ and $B(L)$ both have all of their roots outside the unit circle. If this model were used for estimation, we would also require that $A(L)$ and $B(L)$ fit the identifiability conditions of Dunsmuir and Hannan (1976). Let $\xi(h) = E(Z_{t+h}Z_t')$ be the autocovariance sequence of Z_t. The full model, $A(L)D(L)X_t = B(L)\epsilon_t$, can be written as $D(L)X_t = Z_t$. We may write $X_t = \sum_{j=0}^{\infty} C_j Z_{t-j}$, where C_j is a diagonal matrix with (k,k) element equal to $\psi(j,d_k) = \frac{\Gamma(j+d_k)}{\Gamma(j+1)\Gamma(d_k)}$ and Γ is the gamma function. If Z_t were white noise, this would be the moving average expansion of an $ARFIMA(0,d_k,0)$ process. Using this "moving average" expansion, we find an expression for the autocovariances of $X_{\cdot t}$:

$$\begin{aligned}
\omega(h) &= \text{Cov}(X_t, X_{t-h}) && (4)\\
&= \text{Cov}\left(\sum_{i=0}^{\infty} C_i Z_{t-i}, \sum_{j=0}^{\infty} C_j Z_{t-j-h}\right) && (5)\\
&= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} C_i \text{Cov}(Z_{t-i}, Z_{t-h-j}) C_j' && (6)\\
&= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} C_i \xi(h+j-i) C_j' && (7)
\end{aligned}$$

We now focus on the (k,l) entry of $C_i\xi(h+j-i)C_j'$. Let $\xi_{kl}(h)$ be the (k,l) entry of $\xi(h)$, that is, $\xi_{kl}(h) = E(Z_{k,t+h}Z_{l,t})$. Since C_i and C_j are both diagonal matrices, the (k,l) entry of $C_i\xi(h+j-i)C_j'$ is $\psi(i,d_k)\psi(j,d_l)\xi_{kl}(h+j-i)$. Using this, we find an expression for the (k,l) entry of $\omega(h)$:

$$\begin{aligned}
\omega_{kl}(h) &= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \psi(i,d_k)\psi(j,d_l)\xi_{kl}(h+j-i) && (8)\\
&= \sum_{m=0}^{\infty}\sum_{j=m}^{\infty} \psi(j-m,d_k)\psi(j,d_l)\xi_{kl}(h+m) && (9)\\
&= \sum_{m=0}^{\infty} \xi_{kl}(h+m)\left(\sum_{j=m}^{\infty} \psi(j,d_l)\psi(j-m,d_k)\right) && (10)
\end{aligned}$$

where the second equality follows from the substitution $m = j-i$ and an interchange of the order of summation. The inner sum is the cross-covariance of an $ARFIMA(0,d_k,0)$ process and an $ARFIMA(0,d_l,0)$ process that are driven by common white noise. Writing this cross-covariance

in terms of the integral of the cross-spectrum, we find that:

$$\sum_{j=m}^{\infty} \psi(j,d_l)\psi(j-m,d_k) = \frac{1}{2\pi}\int_0^{2\pi}(1-e^{-i\lambda})^{-d_k}(1-e^{i\lambda})^{-d_l}e^{i\lambda m}d\lambda \quad (11)$$

$$= \frac{\Gamma(1-d_k-d_l)(-1)^m}{\Gamma(1-d_k-m)\Gamma(1-d_l+m)} \quad (12)$$

$$= \frac{\Gamma(1-d_k-d_l)\Gamma(d_k+m)}{\Gamma(d_k)\Gamma(1-d_k)\Gamma(1-d_l+m)} \quad (13)$$

where the last two equations follow from Sowell (1989b, Appendix II and Appendix III, equation IV.2). Notice that this agrees with the usual expression for the autocovariance of an $ARFIMA(0,d_k,0)$ process when $d_k = d_l$ (see, for example, Brockwell and Davis, 1993, Theorem 13.2.1). For notational convenience, we write $\phi_{lk}(h) = \frac{\Gamma(1-d_k-d_l)\Gamma(d_k+h)}{\Gamma(d_k)\Gamma(1-d_k)\Gamma(1-d_l+h)}$. Note that $\phi_{kl}(h) = \phi_{lk}(-h)$, as must be true for any cross-covariances.

Following Bertelli and Caporin (2002), we consider the finite approximation to the outer sum in (10), by setting $\xi_{kl}(m) = 0$ for all $|m| > M$. Because the autocovariance sequence of a vector ARMA decays exponentially fast, we may choose a relatively small M to approximate the process to a given degree of accuracy. Our choice of M depends on the parameters of the ARMA process; if $\xi(h)$ is the autocovariance sequence of an $MA(q)$ process, then we may choose $M = q$ to compute the autocovariances exactly. Otherwise, we must choose an M which accounts for how quickly the autocovariances of the vector ARMA process decay.

Because any stationary and invertible vector $ARMA(p,q)$ process can be written as a vector $AR(1)$ (see Hamilton, 1994, page 259 for details), we focus on the $AR(1)$ process,

$$Z_t = A_1 Z_{t-1} + \eta_t$$

where A_1 is a $K \times K$ matrix such that all of its eigenvalues lie inside the unit circle. Notice that rewriting an ARMA process as an $AR(1)$ may lead to an innovation variance which is positive semi-definite but not positive definite. The computations presented in this section do not depend on Σ being positive definite, so this does not pose a problem. Then, the autocovariance sequence, $\xi(h)$, satisfies (Hamilton, 1994, page 265):

$$\begin{aligned} vec(\xi(0)) &= (I_{K^2} - A_1 \otimes A_1)^{-1} vec(\Sigma) \\ \xi(h) &= A_1^h \xi(0), h > 0 \\ \xi(-h) &= \xi(h)' \end{aligned}$$

where h is a positive integer, vec is the vectorization operator, $\otimes$ is the Kronecker product, and I_{K^2} is a $K^2 \times K^2$ identity matrix. Let $G = \max_{k,l} \phi_{kl}(0)$. Let $\|\cdot\|$ be the Euclidean matrix norm, $\|Q\|_2$, where $\|Q\|_2$ is the maximum singular value of Q (see Heath, 2002, sections 3.6 and 4.7 for background). This is equal to the square root of the largest eigenvalue of $Q^T Q$, which ensures that $\|A_1\| < 1$ as long as the $VAR(1)$ process defined by A_1 is stationary. Then, we

may bound the norm of the error in truncating the infinite sum by:

$$\begin{aligned} 2G\sum_{m=M}^{\infty}\|\xi_{kl}(m)\| &= 2G\sum_{m=M}^{\infty}\|A_1^m\xi(0)\| \\ &\leq 2G\sum_{m=M}^{\infty}\|A_1\|^m\|\xi(0)\| \\ &= 2G\|\xi(0)\|\frac{\|A_1\|^M}{1-\|A_1\|} \end{aligned}$$

Thus, once we know $\xi(0)$ and A_1, we can choose M such that the norm of the error does not exceed a chosen value, δ. In particular, we must have:

$$M \geq \frac{\log(1-\|A_1\|)+\log(\delta)-\log(G)}{\log(\|A_1\|)}+1$$

Once M has been chosen, it remains to compute the sequences $\phi_{kl}(h), h=-M-T,...,M+T$ and $\xi_{kl}(h), h=-M,...,M$ and their convolutions for each (k,l). Using the naive method of summing all the products directly would require $O(M^2+MT)$ operations. Instead, for larger values of M, we recommend using the Fast Fourier Transform to speed up the process to $O((M+T)\log(M+T))$ operations for each of the K^2 convolutions. In most cases, $M>\log T$; exceptions may occur when the eigenvalues of F are far from the unit circle or T is very large; we suggest checking this condition so that the faster convolution method is used. Since there are K^2 convolutions, using an efficient method is particularly important.

Combining all of these considerations yields the splitting algorithm for a FIVAR process:

Algorithm 4 Computing $FIVAR(1,\vec{d})$ covariances to tolerance δ.

- *Set G to be the maximum singular value of $\phi_{kl}(0)$ and compute the maximum singular value of A_1, $\|A_1\|$.*
- *Set M to be the smallest power of two greater than $\frac{\log(1-\|A_1\|)+\log(\delta)-\log(G)}{\log(\|A_1\|)}+1$.*
- *Compute the covariances, ξ, for a $VAR(1)$ for lags $-M$ to M.*
- *Compute the cross-covariances, ϕ, for ARFIMA processes with differencing parameters $\vec{d}$ for lags $-(M+T)$ to $M+T$.*
- *If $M\geq\log T$, compute the convolution of ξ_{ij} with ϕ_{ij} for $i=1,...,K$ and $j=1,...,K$ using the Fast Fourier Transform:*
  - *Append enough zeroes to ξ_{ij} and ϕ_{ij} so that the total length is the smallest power of two which is greater than $length(\xi_{ij})+length(\phi_{ij})$. Call these series $\tilde{\xi}$ and $\tilde{\phi}$.*

Lag	Splitting	Sowell
0	(3.658217, 6.04877, 6.048769, 35.02676)	(3.658217, 6.04877, 6.048769, 35.02676)
1	(3.103113, 5.530935, 6.094733, 33.952608)	(3.103113, 5.530935, 6.094733, 33.952608)
10	(0.7597274, 1.855598, 3.9196162, 25.501238)	(0.7597274, 1.855598, 3.9196162, 25.501238)
100	(0.06346564, 0.3674387, 1.12644985, 15.4985175)	(0.06346564, 0.3674387, 1.12644985, 15.4985175)

Table 1: Computed values for the autocovariances of a FIVAR process with $d = (0.1, 0.4)$, $A_1 = (0.7, 0.1, 0.2, 0.6)$, and $\Sigma = (1, .5, .5, 2)$, using the Sowell (1989b) method and the splitting method.

  - *Compute the inverse Fast Fourier Transforms of $\tilde{\xi}$ and $\tilde{\phi}$ and multiply them together element-by-element.*
  - *Compute the Fast Fourier Transform of the result.*
  - *Return the first $(length(\xi_{ij}) + length(\phi_{ij}) - 1)$ elements of the result.*
  - *Extract the middle covariances from the result.*

- *If $M < \log T$, then compute the convolutions by summing all the terms directly.*

If the $VAR(1)$ process has been created from a vector $ARMA(p, q)$ process, the autocovariances of the original process are the autocovariances computed using the method above for the observed series.

Though we must truncate the sum, this method can be used to compute the autocovariances to any level of precision; more precision simply requires a larger choice of M. In Table 1, we give the computed values of some autocovariances based on the splitting method and based on Sowell's (1989b) method. The two computed results are almost identical to at least five figures.

The running time of the FIVAR splitting algorithm depends on two factors. First, as the largest singular value of A_1 moves arbitrarily close to the unit circle, M will grow infinitely large. Second, given a fixed A_1 and therefore a fixed value of M, the running time will initially grow as $O(T \log T)$ as long as $M > \log T$, and will then grow linearly with T once it is faster to use direct summation instead of the Fast Fourier Transform. Notice that both the Sowell (1989b) method and the method using integrals described in section 4.2 grow linearly with T. Furthermore, Sowell's method depends on computing an infinite sum in which the summands decay as $\rho_1, ..., \rho_K$, which turn out to be the eigenvalues of A_1 in this case. Thus, both Sowell's method and our method slow down as the roots of $I - A_1 L$ approach the unit circle. In Tables 2 and 3, we report the total elapsed processor time in seconds as reported by the R `system.time` function to compute the autocovariances in various cases. This table shows that our method

T	Splitting	Sowell	Integral-Based Method
4	0.028	0.354	4.988
8	0.029	0.672	11.025
16	0.029	1.330	*
32	0.028	2.629	*
64	0.029	4.470	*
128	0.030	8.423	*

Table 2: Processing time needed to compute the autocovariances of a FIVAR process with $d = (0.1, 0.4)$, $A_1 = (0.7, 0.1, 0.2, 0.6)$, and $\Sigma = (1, .5, .5, 2)$, using the Sowell (1989b) method, the integral-based method and the splitting method presented in section 5.2. * indicates that the integral covariances for higher lags did not converge.

Maximum Singular Value	Our Time	Sowell Time	Maximum Difference	M Required
0.8	0.028	4.552	9.808×10^{-10}	119
0.9	0.038	5.970	7.827×10^{-10}	263
0.95	0.058	8.154	7.504×10^{-10}	564
0.99	0.246	14.733	5.955×10^{-8}	3138
0.995	0.577	14.848	1.413×10^{-5}	6479
0.999	4.701	14.775	0.001285	34324

Table 3: Processing time needed to compute the autocovariances of a FIVAR process with $T = 64$, $d = (0.1, 0.4)$, $\Sigma = (1, .5, .5, 2)$, and $A_1 = \alpha(0.7, 0.1, 0.2, 0.6)$, where α is a scalar chosen to vary the maximum singular value. The fourth column shows the maximum absolute difference between Sowell's (1989b) method and our method over all 64 autocovariances. The last column shows the value of M required by the splitting algorithm. Times are the mean processing time needed for 100 repetitions of the calculation.

is much faster than either of the competing methods, for a range of T and for autoregressive matrices with singular values both near and far from the unit circle.

Now that we have seen that the splitting algorithm yields the same results as Sowell's algorithm in a fraction of the time, we will use only the splitting algorithm to compute covariances in the remainder of this paper.

5.3 VARFI Covariances

To use the splitting algorithm with a VARFI process, we first consider the spectral density of X_t. We begin with a $VARFI(1, \vec{d})$ in which $A(L) = I - A_1 L$, where $A(L)$ has all of its roots outside of the unit circle. In this case, we also assume that A_1 is not a defective matrix, so that it has K unique eigenvectors (see, for example Heath, 2002, chapter 4). Though this requirement will cause the method not to apply for certain matrices, defective matrices are quite rare and

therefore of little concern.

We first write the autocovariances of X_t in terms of the spectral density:

$$
\begin{aligned}
f_X(\lambda) &= (I - A_1 e^{-i\lambda})^{-1} D(e^{-i\lambda})^{-1} \Sigma D(e^{i\lambda})^{-1} (I - A_1^* e^{i\lambda})^{-1} \\
&= \left(\sum_{r=0}^{\infty} A_1^r e^{-i\lambda r} \right) \times \\
&\quad \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1K} \\ \Sigma_{21} & \Sigma_{22} & \ddots & \Sigma_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{K1} & \Sigma_{K2} & \cdots & \Sigma_{KK} \end{pmatrix} \bullet \\
&\quad \begin{pmatrix} (1-e^{-i\lambda})^{-d_1}(1-e^{i\lambda})^{-d_1} & (1-e^{-i\lambda})^{-d_1}(1-e^{i\lambda})^{-d_2} & \cdots & (1-e^{-i\lambda})^{-d_1}(1-e^{i\lambda})^{-d_K} \\ (1-e^{-i\lambda})^{-d_1}(1-e^{i\lambda})^{-d_2} & (1-e^{-i\lambda})^{-d_2}(1-e^{i\lambda})^{-d_2} & \ddots & (1-e^{-i\lambda})^{-d_2}(1-e^{i\lambda})^{-d_K} \\ \vdots & \vdots & \ddots & \vdots \\ (1-e^{-i\lambda})^{-d_K}(1-e^{i\lambda})^{-d_1} & (1-e^{-i\lambda})^{-d_K}(1-e^{i\lambda})^{-d_2} & \cdots & (1-e^{-i\lambda})^{-d_K}(1-e^{i\lambda})^{-d_K} \end{pmatrix} \times \\
&\quad \left(\sum_{s=0}^{\infty} (A_1^*)^s e^{i\lambda s} \right) \\
&= \sum_{r=0}^{\infty} \sum_{s=0}^{\infty} A_1^r \times \\
&\quad \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1K} \\ \Sigma_{21} & \Sigma_{22} & \ddots & \Sigma_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{K1} & \Sigma_{K2} & \cdots & \Sigma_{KK} \end{pmatrix} \bullet \\
&\quad \begin{pmatrix} (1-e^{-i\lambda})^{-d_1}(1-e^{i\lambda})^{-d_1} & (1-e^{-i\lambda})^{-d_1}(1-e^{i\lambda})^{-d_2} & \cdots & (1-e^{-i\lambda})^{-d_1}(1-e^{i\lambda})^{-d_K} \\ (1-e^{-i\lambda})^{-d_2}(1-e^{i\lambda})^{-d_2} & (1-e^{-i\lambda})^{-d_2}(1-e^{i\lambda})^{-d_2} & \ddots & (1-e^{-i\lambda})^{-d_2}(1-e^{i\lambda})^{-d_K} \\ \vdots & \vdots & \ddots & \vdots \\ (1-e^{-i\lambda})^{-d_K}(1-e^{i\lambda})^{-d_1} & (1-e^{-i\lambda})^{-d_K}(1-e^{i\lambda})^{-d_2} & \cdots & (1-e^{-i\lambda})^{-d_K}(1-e^{i\lambda})^{-d_K} \end{pmatrix} \times \\
&\quad (A_1^*)^s e^{i\lambda(s-r)}
\end{aligned}
$$

where $\bullet$ denotes the Hadamard (element-wise) matrix product. Let $A_1 = V_A \Lambda V_A^{-1}$ be an eigen-

value decomposition of A_1. For notational convenience, define:

$$\begin{aligned} Q(\lambda) &= V_A^{-1} \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1K} \\ \Sigma_{21} & \Sigma_{22} & \ddots & \Sigma_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{K1} & \Sigma_{K2} & \cdots & \Sigma_{KK} \end{pmatrix} \bullet \\ & \begin{pmatrix} (1-e^{-i\lambda})^{-d_1}(1-e^{i\lambda})^{-d_1} & (1-e^{-i\lambda})^{-d_1}(1-e^{i\lambda})^{-d_2} & \cdots & (1-e^{-i\lambda})^{-d_1}(1-e^{i\lambda})^{-d_K} \\ (1-e^{-i\lambda})^{-d_2}(1-e^{i\lambda})^{-d_2} & (1-e^{-i\lambda})^{-d_2}(1-e^{i\lambda})^{-d_2} & \ddots & (1-e^{-i\lambda})^{-d_2}(1-e^{i\lambda})^{-d_K} \\ \vdots & \vdots & \ddots & \vdots \\ (1-e^{-i\lambda})^{-d_K}(1-e^{i\lambda})^{-d_1} & (1-e^{-i\lambda})^{-d_K}(1-e^{i\lambda})^{-d_2} & \cdots & (1-e^{-i\lambda})^{-d_K}(1-e^{i\lambda})^{-d_K} \end{pmatrix} \times \\ & (V_A^*)^{-1} \end{aligned}$$

Using this notation, we describe the autocovariances of X_t:

$$\begin{aligned} \omega(h) &= \int_{-\pi}^{\pi} f_X(\lambda) e^{ih\lambda} d\lambda \\ &= \int_{-\pi}^{\pi} \sum_{r=0}^{\infty}\sum_{s=0}^{\infty} V_A \Lambda^r Q(\lambda) (\Lambda^*)^s V_A^* e^{-i\lambda(r-s)} e^{ih\lambda} d\lambda \\ &= V_A \left(\sum_{r=0}^{\infty}\sum_{s=0}^{\infty} \Lambda^r \left(\int_{-\pi}^{\pi} Q(\lambda) e^{-i\lambda(r-s-h)} d\lambda \right) (\Lambda^*)^s \right) V_A^* \end{aligned}$$

Notice that $\int_{-\pi}^{\pi} Q(\lambda) e^{-i\lambda(r-s-h)} d\lambda$ is $V_A^{-1}(\Sigma \bullet \phi(r-s-h))(V_A^*)^{-1}$, where $(\Sigma \bullet \phi(r-s-h))$ is the r^{th} autocovariance of a $VARFI(0, \vec{d})$; this can be computed using the expression in equation (13) above. Let $H_{ij}(r)$ be the (i,j) element of $V_A^{-1}\phi(r-s-h)(V_A^*)^{-1}$. Let Λ_{ii} be the (i,i) element of Λ. Then, the (i,j) element of the inner sum is:

$$\sum_{r=0}^{\infty}\sum_{s=0}^{\infty} \Lambda_{ii}^r \bar{\Lambda}_{jj}^s H_{ij}(h+s-r) = \sum_{u=-\infty}^{\infty} L_{ij}(u) H_{ij}(h-u) \tag{14}$$

where $\bar{\Lambda}_{jj}$ is the complex conjugate of Λ_{jj} and

$$L_{ij}(u) = \begin{cases} \frac{\Lambda_{ii}^u}{1-\Lambda_{ii}\bar{\Lambda}_{jj}} & u \geq 0 \\ \frac{\bar{\Lambda}_{jj}^{|u|}}{1-\Lambda_{ii}\Lambda_{jj}} & u < 0 \end{cases}$$

After the sums in (14) have been calculated for each lag and each $i = 1, ..., K$ and $j = 1, ..., K$, the matrix of sums for each u must be multiplied by V_A and V_A^* to find the covariances of the original process.

As before, we want to approximate the sums above by sums with a finite number of terms. Since each $L_{ij}(u)$ decays exponentially quickly, we again choose M so that $\sum_{u=M+1}^{\infty} L_{ij}(u) < \delta$

for a given tolerance δ and all i, j. Let $G = \max V_A^{-1}\phi(0)(V_A^{-1})^*$, where the maximum is taken over all of the entries in the product. Let $|\Lambda_{j^*j^*}|$ be the absolute value of the largest eigenvalue. Then, $L_{ij}(u) \leq \frac{\Lambda^u_{j^*j^*}}{1-|\Lambda_{j^*j^*}|^2}$, and we may bound the sum of the omitted terms:

$$\begin{aligned}\sum_{u=M+1}^{\infty} H(h-u)L_{ij}(u) &\leq G\sum_{u=M+1}^{\infty}\frac{\Lambda^u_{j^*j^*}}{1-|\Lambda_{j^*j^*}|^2} \\ &= G\frac{\Lambda^{M+1}_{j^*j^*}}{(1-|\Lambda_{j^*j^*}|^2)(1-|\Lambda_{j^*j^*}|)}\end{aligned}$$

Thus, we may choose $M > \frac{\log\delta+2\log(1-|\Lambda_{j^*j^*}|)+\log(1+|\Lambda_{j^*j^*}|)}{|\log\Lambda_{j^*j^*}|-\log G}$ to ensure that the sum of the omitted terms is less than δ. As in the computation of the autocovariances of FIVAR processes, we suggest using the fast Fourier transform to compute the convolutions in the case where $M > \log T$. This yields the following algorithm:

Algorithm 5 Computing the Covariances of a VARFI process to tolerance, δ.

- *Compute the eigenvalue decomposition, $A_1 = V_A\Lambda V_A^{-1}$ and find j^* such that $\Lambda_{j^*j^*}$ is the largest eigenvalue.*
- *Set G to be the maximum entry of $V_A^{-1}\phi(0)(V_A^{-1})^*$.*
- *Set M to be the smallest power of two greater than $\frac{\log\delta+2\log(1-\Lambda_{j^*j^*})+\log(1+\Lambda_{j^*j^*})}{\log\Lambda_{j^*j^*}-\log G}$.*
- *For $i = 1, ..., K$, $j = 1, ..., K$, and $u = -M, ..., M$, compute $L_{ij}(u)$ using equation (15).*
- *Compute the cross-covariances, $\phi(r)$, for ARFIMA processes with differencing parameters $\vec{d}$ from lags $r = -(M+T), ..., M+T$.*
- *For $r = -(M+T), ..., M+T$, compute $H(r) = V_A^{-1}\phi(r)(V_A^{-1})^*$.*
- *If $M \geq \log T$, compute the convolution of L_{ij} with H_{ij} for $i = 1, ..., K$ and $j = 1, ..., K$ using the Fast Fourier Transform:*
  - *Append enough zeroes to L_{ij} and H_{ij} so that the total length is the smallest power of two which is greater than $(length(L_{ij}) + length(H_{ij}))$. Call these series $\tilde{L}$ and $\tilde{H}$.*
  - *Compute the inverse Fast Fourier Transforms of $\tilde{L}$ and $\tilde{H}$ and multiply them together element-by-element.*
  - *Compute the Fast Fourier Transform of the result.*
  - *Return the first $(length(L_{ij}) + length(H_{ij}) - 1)$ elements of the result.*
  - *Extract the middle covariances, from $-T$ to T, from the result.*
- *If $M < \log T$, then compute the convolutions by summing all the terms directly.*

- *Pre-multiply the matrix for each lag by V_A and post-multiply the matrix for each lag by V_A^*.*

Like the algorithm for FIVAR covariances, this algorithm runs in $O(\min(M^2 + MT, (M + T)\log(M + T))$. In Table 4, we compare the processing time needed for this method to the time needed to use the integral definition of the autocovariance sequence. As in the FIVAR case, using the integral definition of the covariances requires dramatically more computing time, despite the fact that it is $O(T)$.

One disadvantage to this computational method for VARFI covariances is that V_A must be inverted. While this is generally fast for small K, it makes the computed covariances sensitive to the condition number of V_A. In particular, we have found that when V_A is close to singular, many of the computed covariances are zero, even though their exact values are nowhere near 0. This can occur when A_1 differs by a minute amount from a multiple of the identity matrix. This consideration should inform the choice of initial values in VARFI maximum likelihood estimation.

To extend this method for computing covariances to a $VARFI(p, \vec{d})$ model, we rewrite that model as a $VARFI(1, \vec{d^{\#}})$ model. Suppose $A(L) = I - A_1 L - ... - A_p L^p$. Let $X_t^{\#} = vec(X_t, ..., X_{t-p+1})$, and

$$A_1^{\#} = \begin{pmatrix} A_1 & A_2 & ... & A_p \\ I_K & 0 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

$$d^{\#} = \begin{pmatrix} d \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$\Sigma^{\#} = \begin{pmatrix} \Sigma & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

Then, $D^{\#}(L)(I - A_1^{\#} L)X_t^{\#} = \epsilon_t$ is a $VARFI(1, \vec{d^{\#}}, 0)$ process, and the first K series follow the original $VARFI(p, \vec{d}, 0)$ model. As before $\Sigma^{\#}$ is not generally positive definite, but this does not pose a problem for computing autocovariances. Thus, the method presented above generalizes to any $VARFI(p, \vec{d}, 0)$ model with finite p, where we extract the relevant autocovariances as we did in section 5.2. We do not extend this method to models with moving average components.

T	Splitting	Integral-Based Method
4	0.031	8.362
8	0.031	18.025
16	0.032	37.511
32	0.035	85.087
64	0.040	216.706
128	0.050	623.426

Table 4: Time needed to compute the autocovariances of a VARFI process with $d = (0.1, 0.4)$, $A_1 = (0.7, 0.1, 0.2, 0.6)$, and $\Sigma = (1, .5, .5, 2)$ using the integral definition and using the splitting algorithm presented in section 5.3. Times are the mean processing time needed over 100 repetitions of the calculation.

5.4 Cointegrated Systems

Consider the cointegrated FIVAR model, $A(L)D(L)VX_t = \epsilon_t$. Define the process, Y_t, by $A(L)D(L)Y_t = \epsilon_t$. Then,

$$\begin{aligned} \text{Cov}(X_t, X_{t-j}) &= \text{Cov}(V^{-1}Y_t, V^{-1}Y_{t-j}) \\ &= V^{-1}\text{Cov}(Y_t, Y_{t-j})(V^{-1})' \end{aligned}$$

Since Y_t is a FIVAR process, its autocovariance sequence can be computed using Algorithm 4 above. Then, we may compute the autocovariances of X_t by multiplying each autocovariance by V^{-1} and $(V^{-1})'$, which takes $O(T)$ additional steps.

6 Computing the Quadratic Form

To compute the quadratic form, $X\Omega^{-1}X$, in the expression for the likelihood in equation (1), we apply the preconditioned conjugate gradient algorithm. The application of preconditioned conjugate gradient algorithms to univariate long memory time series began with Deo et al. (2006); related theoretical results are available in Chen et al. (2006). The algorithm which we present in this section was developed by Chan and Olkin (1994), but this is its first application to multivariate long memory time series.

We begin this section with some background on the PCG algorithm. We then describe how we can apply it most efficiently to multivariate time series, and finally discuss the computational cost of these methods.

6.1 The Preconditioned Conjugate Gradient Algorithm

Preconditioned conjugate gradient methods have been used extensively in solving systems of linear equations of the form $\Omega y = b$ where Ω is symmetric and positive definite (in this section, we rely heavily on Shewchuk (1994); see his write-up for more details). The conjugate gradient method and the preconditioned conjugate gradient method are based on using the residual error at at each iteration to choose a search direction and the optimal distance in that direction. These methods can be applied to any system in which Ω is symmetric and positive definite.

Algorithm 6 Conjugate Gradient Algorithm (Shewchuk, 1994, see, for example,). *Let a tolerance, δ, be given. Let the initial value, $y_{(0)}$, be a vector of zeroes. Initialize:*

$$\begin{aligned} d_{(0)} &= b - \Omega y_{(0)} \\ r_{(0)} &= b - \Omega y_{(0)} \end{aligned}$$

Iterate through the following steps until $\|r_{(i)}\| < \delta$.

$$\begin{aligned} \alpha_{(i)} &= \frac{r'_{(i)} r_{(i)}}{d'_{(i)} \Omega d_{(i)}} \\ y_{(i+1)} &= y_{(i)} + \alpha_{(i)} d_{(i)} \\ r_{(i+1)} &= r_{(i)} - \alpha_{(i)} \Omega d_{(i)} \\ \beta_{(i+1)} &= \frac{r'_{(i+1)} r_{(i+1)}}{r'_{(i)} r_{(i)}} \\ d_{(i+1)} &= r_{(i+1)} + \beta_{(i+1)} d_{(i)} \end{aligned}$$

This algorithm chooses a direction, $d_{(i)}$, which is conjugate, or Ω-orthogonal, to all the previous search directions; that is, $d'_{(i)} \Omega d_{(j)} = 0$ when $i \neq j$. The choice of direction is based on Gram-Schmidt conjugation. Each search direction, $d_{(i)}$, is linearly independent, and the distance chosen for each search direction, $\alpha_{(i)}$, is optimal. That is, the resulting residual is conjugate to the search direction used to compute it. Because of this, if there were no roundoff error, each search direction would be linearly independent and used at most once. Thus, with infinite precision, the algorithm would always converge to exactly the true solution in a number of steps at most the dimension of Ω.

Because computers have finite precision, we say that the algorithm has converged when the distance from the computed answer to the true value is less than some tolerance level. In this algorithm, the search directions with the largest steps are used first, so convergence in this sense takes fewer steps than would be required with infinite precision. This is an important property

when the dimension of Ω is large. To be precise, the error in the i^{th} iteration, $e_{(i)} = y_{(i)} - y$, satisfies (Shewchuk, 1994, page 36):

$$\sqrt{e'_{(i)}\Omega e_{(i)}} \leq 2\left(\frac{\sqrt{\kappa(\Omega)}-1}{\sqrt{\kappa(\Omega)}+1}\right)^i e'_{(0)}\Omega e_{(0)} = 2\left(1-\frac{2}{\sqrt{\kappa(\Omega)}+1}\right)^i e'_{(0)}\Omega e_{(0)}$$

where $\kappa(\Omega)$ is the condition number of the matrix Ω, defined as the ratio of the largest to the smallest eigenvalue of Ω. Given any tolerance level and initial error, we can solve for i to find an approximate number of iterations required for convergence within that tolerance level. Such an analysis shows that the required number of iterations is $O(\sqrt{\kappa(\Omega)})$. This shows the importance of the condition number of Ω to the computational complexity of this algorithm.

When the condition number is large, we can "precondition" the matrix in order to reduce the condition number. This is based on solving the system of linear equations, $C^{-1}\Omega y = C^{-1}b$, where C approximates Ω but has an inverse which is easy to compute. This method is effective when $\kappa(C^{-1}\Omega) << \kappa(\Omega)$. However, one does not simply apply the conjugate gradient method to the system $C^{-1}\Omega y = C^{-1}b$, since the product $C^{-1}\Omega$ is not generally symmetric or positive definite. Instead, consider the matrix E such that $EE' = C$. Then, $\kappa(C^{-1}\Omega) = \kappa(E^{-1}\Omega(E^{-1})')$, and the latter matrix is symmetric and positive definite. Thus, we could solve the system $E^{-1}\Omega(E^{-1})'\hat{y} = E^{-1}b$ for $\hat{y}$, and then compute $y = (E^{-1})'\hat{y}$; this is called the transformed preconditioned conjugate gradient algorithm.

Using this version of the algorithm would require computing E. Instead, we define $\hat{r}_{(i)} = E^{-1}r_{(i)}$ and $\hat{d}_{(i)} = E'd_{(i)}$. We can substitute these into the conjugate gradient algorithm above to arrive at the untransformed preconditioned conjugate gradient (PCG) algorithm.

Algorithm 7 Preconditioned Conjugate Gradient Algorithm (Shewchuk, 1994, see, for example,). *Let a tolerance, δ, be given. Let $x_{(0)}$ be a vector of zeroes. Initialize:*

$$\begin{aligned} r_{(0)} &= b - \Omega x_{(0)} \\ d_{(0)} &= C^{-1}r_{(0)} \end{aligned}$$

Iterate through the following steps until $\|r_{(i)}\| < \delta$.

$$\begin{aligned} \alpha_{(i)} &= \frac{r'_{(i)}C^{-1}r_{(i)}}{d'_{(i)}\Omega d_{(i)}} \\ x_{(i+1)} &= x_{(i)} + \alpha_{(i)}d_{(i)} \\ r_{(i+1)} &= r_{(i)} - \alpha_{(i)}\Omega d_{(i)} \\ \beta_{(i+1)} &= \frac{r'_{(i+1)}C^{-1}r_{(i+1)}}{r'_{(i)}C^{-1}r_{(i)}} \\ d_{(i+1)} &= C^{-1}r_{(i+1)} + \beta_{(i+1)}d_{(i)} \end{aligned}$$

Note that this algorithm requires multiplying vectors by C, which can be one of the more computationally intensive steps of the process. Therefore, we choose C to make the multiplication more tractable. In the case of multivariate time series with $K << T$, we choose C to be block circulant. This choice allows us to take advantage of all the computational methods designed for circulants described in section 3. For an extensive review of the PCG algorithm for Toeplitz and block-Toeplitz matrices, see Chan and Ng (1996).

6.2 The Choice of Preconditioner

A good preconditioner, C, must approximate Ω. In addition, its inverse must lend itself to efficient multiplication. We choose to use the "level 1" preconditioners of Chan and Olkin (1994). In this section, we describe the preconditioner and how to compute it.

We begin by writing our block Toeplitz matrix, Ω, in terms of its blocks:

$$\Omega = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1K} \\ A_{21} & A_{22} & \cdots & A_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ A_{K1} & A_{K2} & \cdots & A_{KK} \end{pmatrix}$$

Define $A_{ij}(r)$, for $r = -(T-1), ..., (T-1)$, to be the element along the r^{th} sub-diagonal away from the main diagonal, where a negative r corresponds to diagonals in the lower triangle of the matrix and a positive r corresponds to diagonals in the upper triangle of the matrix. That is, we write:

$$A_{ij} = \begin{pmatrix} A_{ij}(0) & A_{ij}(1) & \cdots & A_{ij}(T-1) \\ A_{ij}(-1) & A_{ij}(0) & \cdots & A_{ij}(T-2) \\ \vdots & \vdots & \ddots & \vdots \\ A_{ij}(-(T-1)) & A_{ij}(-(T-2)) & \cdots & A_{ij}(0) \end{pmatrix},$$

As mentioned in the section 3.1, $A_{ij}(r) = \omega(-r)$; this allows us to relate the elements of this matrix to the properties of the underlying time series. We approximate Ω by approximating its individual blocks. The approximation we use here is T. Chan's (1988) optimal circulant preconditioner, $circ(A_{ij})$. This preconditioner is the circulant matrix with first row consisting of entries $c_0 = A_{ij}(0)$ and $c_r = \frac{rA_{ij}(-(T-r))+(T-r)A_{ij}(r)}{T}$, $r = 1, ..., T-1$. Combining the preconditioners for all of the blocks yields the following block circulant matrix:

$$C = \begin{pmatrix} circ(A_{11}) & circ(A_{12}) & \cdots & circ(A_{1K}) \\ circ(A_{21}) & circ(A_{22}) & \cdots & circ(A_{2K}) \\ \vdots & \vdots & \ddots & \vdots \\ circ(A_{K1}) & circ(A_{K2}) & \cdots & circ(A_{KK}) \end{pmatrix}.$$

With this theoretical preconditioner, we can now apply the methods we discussed in section 3. First, we store only the first row of each block of the preconditioner. Second, we find a

representation for C^{-1} using the inversion method for block circulant matrices described in Algorithm 1. Finally, we multiply by Ω and by C^{-1} using the fast multiplication methods discussed in 3.3.

6.3 Computational Cost

Chan and Olkin (1994) show that the algorithm described in the previous two sections has a set-up cost of $O(K^2T\log T + K^3T)$ to compute the preconditioner and a cost of $O(K^2T + KT\log T)$ per iteration. In most multivariate time series applications, K is generally fixed and much smaller than $\log T$, so the relevant costs are $O(K^2T\log T)$ and $O(KT\log T)$.

As we mentioned in 6.1, the number of iterations required for convergence depends on the condition number of the matrix. By preconditioning, we hope to reduce that ratio so that convergence is faster. In the case of a covariance matrix based on a univariate long memory model, Chen et al. (2006) show that the condition number of the preconditioned matrix grows as $O(\log^3 T)$, which implies that the overall algorithm with $K = 1$ runs in $O(T\log^{5/2} T)$ time. Chan and Olkin (1994) run numerical experiments in which the r^{th} diagonal (for $r = -(T-1), ..., T-1$) of the j^{th} block has element $\frac{1}{(j+1)^{1.1}+(|r|+1)^{1.1}}$ or $\frac{1}{(j+1)^{2.1}+(|r|+1)^{2.1}}$. In their experiment, they find that their preconditioner dramatically reduces the number of iterations, but sometimes increases the number of operations because of the additional multiplications.

In Tables 5 and 6, we report the condition number, before and after pre-conditioning, for the covariance matrices associated with FIVAR and VARFI processes. Preconditioning dramatically reduces the condition number in both cases. A simple regression of $\log(\kappa(C^{-1}\Omega))$ on $\log(\log(T))$ using the data in those tables produces slope estimates of 1.238 (standard error 0.0492) and 1.259 (standard error 0.0404) for FIVAR and VARFI processes, respectively. We also plot $\log(\kappa(C^{-1}\Omega))$ versus $\log(\log(T))$ in Figures 4 and 5; these plots show that the relationship is approximately linear. Based on the slope estimate and the linear relationship in the plots, the conditioned number of the preconditioned matrix appears to grow approximately as $O(\log^{5/4} T)$.

In the case of cointegrated FIVAR series, we may bound the condition number of Ω in terms of the cointegrating matrix and the properties of the underlying FIVAR series. Let Ω_0 be the covariance matrix of the series before they are cointegrated; this is the covariance matrix associated with the FIVAR process, Y_t, described in section 5.4. Let V be the cointegrating matrix as before. Then, $\Omega = (V^{-1} \otimes I)\Omega_0((V')^{-1} \otimes I)$. Applying the definition of a condition

T	$\kappa(\Omega)$	$\kappa(C^{-1}\Omega)$	$\log(\kappa(\Omega))$	$\log(\kappa(C^{-1}\Omega))$
4	782.7286	11.5169	6.6628	2.4438
8	1749.7115	22.7916	7.4672	3.1264
16	3322.2824	32.5125	8.1084	3.4816
32	5952.1906	38.2324	8.6915	3.6437
64	10454.6722	42.2234	9.2548	3.7430
128	18250.7736	56.7711	9.8120	4.0390
256	31801.4260	71.3439	10.3673	4.2675
512	55382.3246	83.8753	10.9220	4.4293

Table 5: Condition number of autocovariance matrices for a $FIVAR(1, \vec{d})$ process with parameters $d = (0.1, 0.4)$, $\Sigma = (1, 0.5, 0.5, 2)$, and $A_1 = (0.6, -0.1, 0.2, 0.8)$, for a range of T.

T	$\kappa(\Omega)$	$\kappa(C^{-1}\Omega)$	$\log(\kappa(\Omega))$	$\log(\kappa(C^{-1}\Omega))$
4	688.5823	8.2331	6.5346	2.1082
8	1537.3445	16.4618	7.3378	2.8010
16	2892.7798	23.2247	7.9700	3.1452
32	5123.9246	26.9049	8.5417	3.2923
64	8907.3649	32.7076	9.0946	3.4876
128	15417.6091	42.8939	9.6433	3.7587
256	26681.6308	52.1514	10.1917	3.9542
512	46214.2378	59.8457	10.7410	4.0918

Table 6: Condition number of autocovariance matrices for a $VARFI(1, \vec{d})$ process with parameters $d = (0.1, 0.4)$, $\Sigma = (1, 0.5, 0.5, 2)$, and $A_1 = (0.6, -0.1, 0.2, 0.8)$, for a range of T.

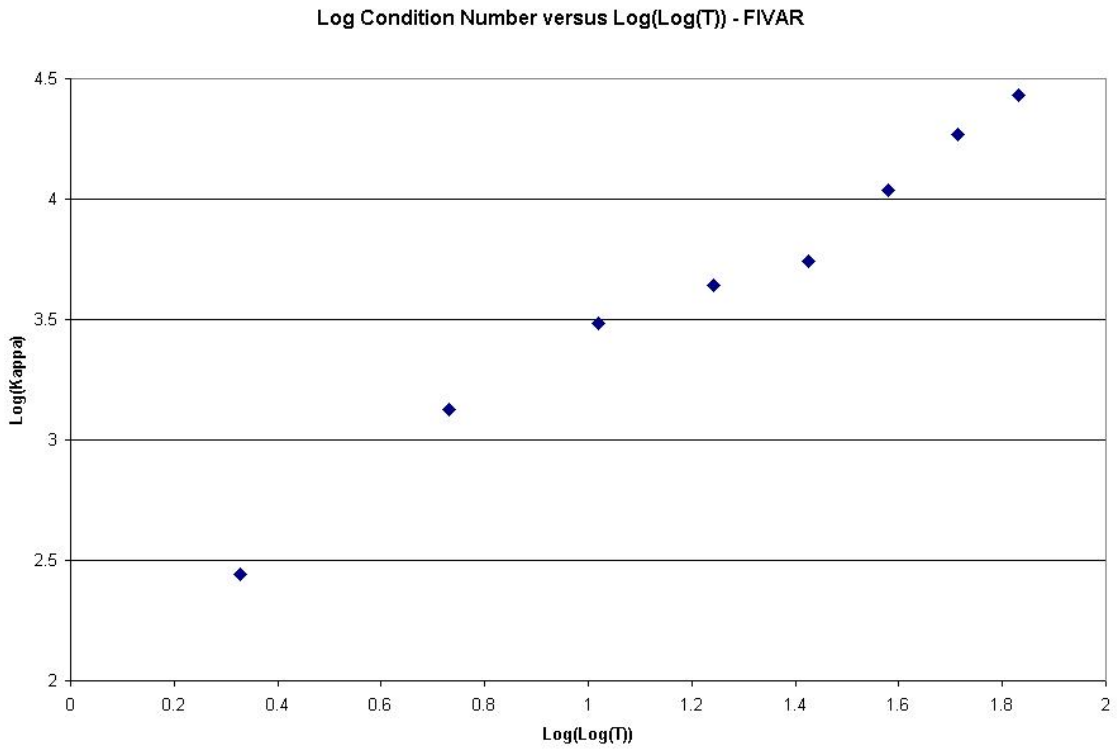

Figure 4: Plot of the logged condition number versus $\log(\log(T))$ for a $FIVAR(1, \vec{d})$ process with parameters $d = (0.1, 0.4)$, $\Sigma = (1, 0.5, 0.5, 2)$, and $A_1 = (0.6, -0.1, 0.2, 0.8)$.

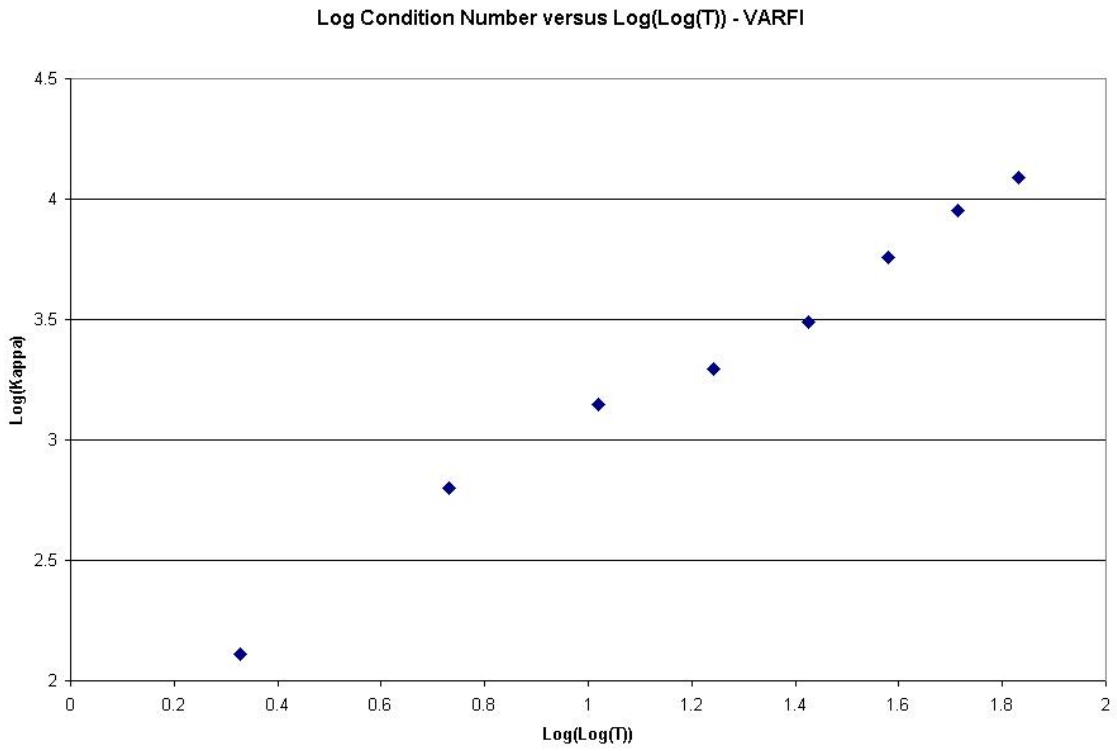

Figure 5: Plot of the logged condition number versus $\log(\log(T))$ for a $VARFI(1, \vec{d})$ process with parameters $d = (0.1, 0.4)$, $\Sigma = (1, 0.5, 0.5, 2)$, and $A_1 = (0.6, -0.1, 0.2, 0.8)$.

number,

$$\begin{aligned}
\kappa(\Omega) &= \max_{x\in\mathbb{R}^{KT}} \frac{\|\Omega x\|}{\|x\|} \\
&= \max_{x\in\mathbb{R}^{KT}} \left(\frac{\|\Omega x\|}{\|\Omega_0((V')^{-1}\otimes I)x\|} \times \frac{\|\Omega_0((V')^{-1}\otimes I)x\|}{\|((V')^{-1}\otimes I)x\|} \times \frac{\|((V')^{-1}\otimes I)x\|}{\|x\|} \right) \\
&\leq \max_{x\in\mathbb{R}^{KT}} \left(\frac{\|\Omega x\|}{\|\Omega_0((V')^{-1}\otimes I)x\|} \right) \times \max_{x\in\mathbb{R}^{KT}} \left(\frac{\|\Omega_0((V')^{-1}\otimes I)x\|}{\|((V')^{-1}\otimes I)x\|} \right) \times \max_{x\in\mathbb{R}^{KT}} \left(\frac{\|((V')^{-1}\otimes I)x\|}{\|x\|} \right) \\
&= \kappa(V^{-1}\otimes I)\kappa(\Omega_0)\kappa((V')^{-1}\otimes I) \\
&= \kappa(V)^2\kappa(\Omega_0)
\end{aligned}$$

Using Sowell's (1989b) representation of bivariate cointegration with $V = \begin{pmatrix} 1 & 0 \\ \rho & 1 \end{pmatrix}$, $\kappa(V)$ is larger for larger values of $|\rho|$. This shows that both the cointegrating matrix and the autocovariance sequence of the associated FIVAR process affect the condition number of the resulting covariance matrix.

In addition to looking at the condition numbers, we can compare the processing time of this algorithm to the processing time needed to compute the quadratic form using the method of Sowell (1989a). We time the Sowell method in two parts. First, the sequence of matrices, $v(n), d(n), A(n,k)$, must be computed. Then, those matrices must be used to compute the quadratic form itself. In Table 7, we present the processing time needed to compute the quadratic form using the PCG algorithm and Sowell's method. While the two methods are comparable for very small samples, we see that the PCG algorithm is almost ten times faster than Sowell's method at a sample size as small as 64. In Figure 6, we can also see that the time needed to use Sowell's method dwarfs the time needed for PCG; this figure also confirms that the processing time needed for Sowell's method grows quadratically with the sample size. In Figure 7, we plot only the processing time needed for the PCG method. As we expect from the discussion of condition numbers above, the PCG processing time seems to grow at less than a quadratic rate.

6.4 Relationship to Periodogram

In this section, we extend the analysis of Chen et al. (2006) to show that the block-circulant preconditioner is related to the expected value of the cross-periodogram of the multivariate time series, X_t. Let $I(\nu_s)$ be the $K \times K$ cross-periodogram of the vector X_t, where $\nu_s = \frac{2\pi s}{T}$ is the s^{th} Fourier frequency. Then, we may write $I(\nu_s)$ and its expectation in terms of the sample

T	Sowell Setup	Sowell Quadratic Form	PCG
8	0.007	0.009	0.007
16	0.021	0.021	0.010
32	0.079	0.051	0.015
64	0.276	0.141	0.026
128	1.041	0.417	0.049
256	4.076	1.350	0.090
512	16.149	4.686	0.176
1024	64.07194	17.758	0.351
2048	255.430	67.672	0.730

Table 7: Processing time used to compute $X\Omega^{-1}X$ where X is a vector of ones and Ω is the autocovariance matrix for the $FIVAR(0, \vec{d})$ with $d = (0.1, 0.4)$ and $\Sigma = (1, 0.5, 0.5, 2)$. All times are the mean processing time as measured by R, over 100 repetitions.

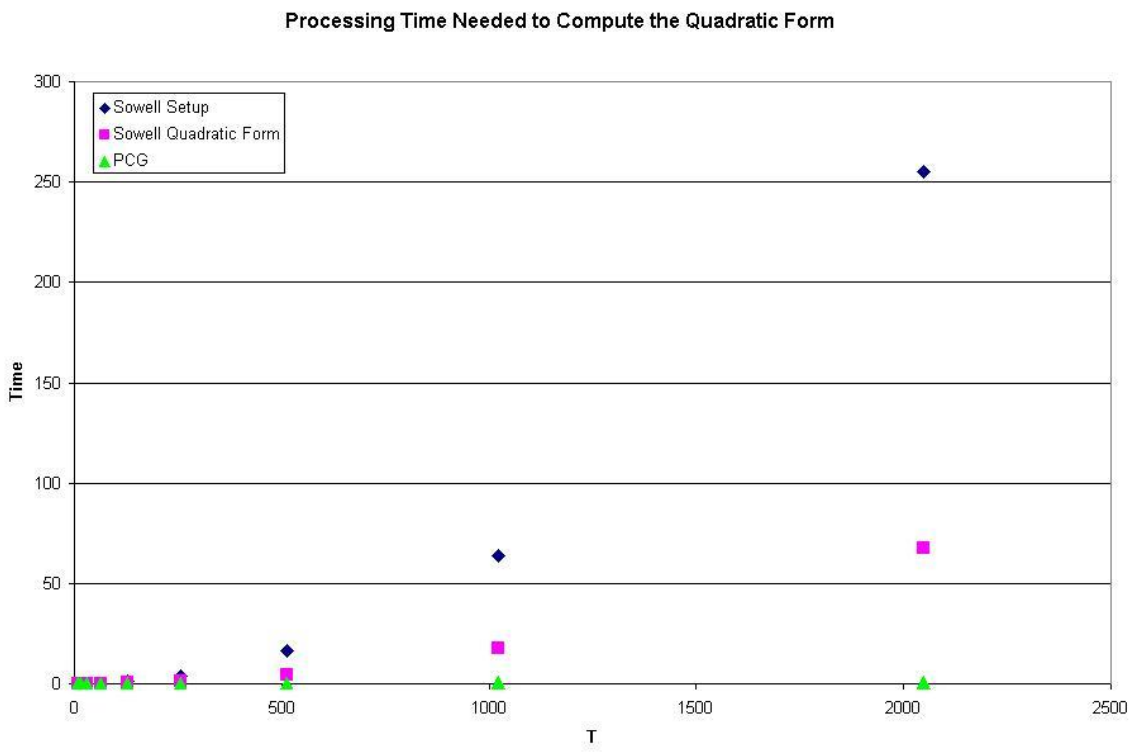

Figure 6: Processing time needed for quadratic form computation methods for various sample sizes.

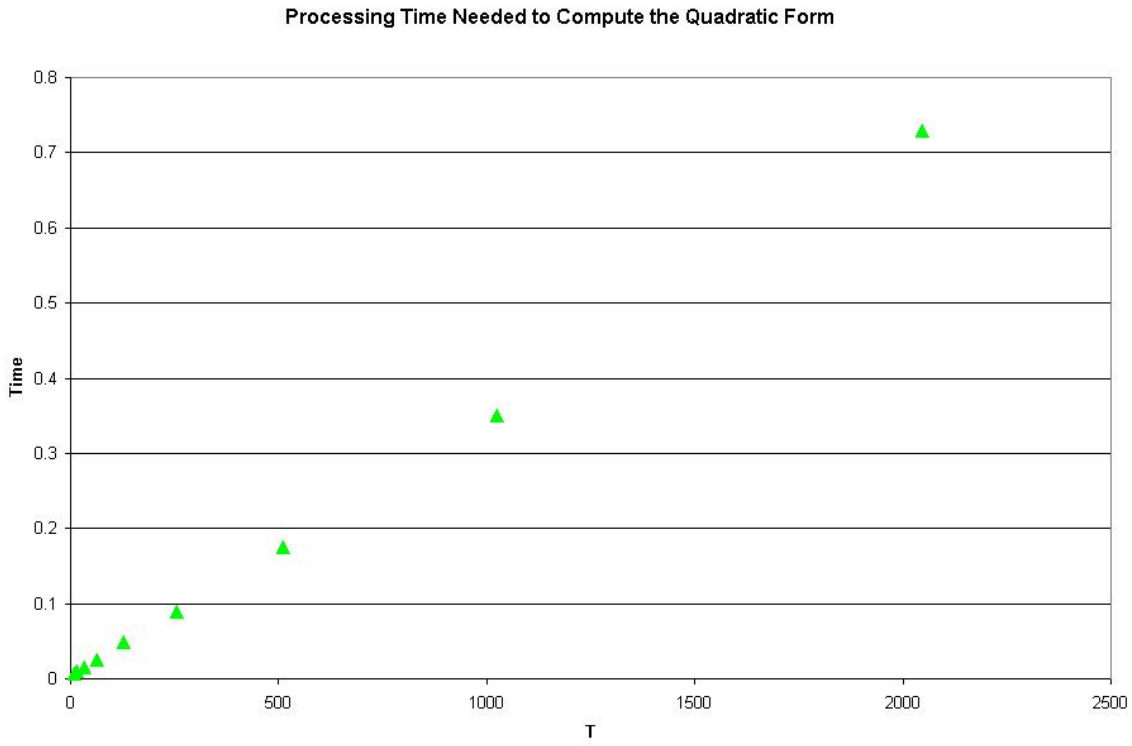

Figure 7: Processing time needed for quadratic form computation using the PCG algorithm.

cross-covariance and its expectation (for example Brockwell and Davis, 1993, p. 443):

$$\begin{aligned} I(\nu_s) &= \sum_{r=-(T-1)}^{T-1} \hat{\omega}(r) \exp(-ir\nu_s) \\ E(I(\nu_s)) &= \sum_{r=-(T-1)}^{T-1} E(\hat{\omega}(r)) \exp(-ir\nu_s) \end{aligned}$$

where $\hat{\omega}(r)$ is the sample cross-covariance of x_t at lag r, defined as:

$$\hat{\omega}(r) = \begin{cases} \frac{1}{T} \sum_{t=1}^{T-r} X_{t+r} X_t' & 0 \leq r \leq T-1 \\ \frac{1}{T} \sum_{t=-r+1}^{T} X_{t+r} X_t' & -T+1 \leq r < 0 \end{cases}$$

(This definition differs slightly from Brockwell and Davis (1993, page 407) because we do not subtract off the sample mean.) Note that $E(\hat{\omega}(r)) = \frac{T-|r|}{T}\omega(r)$. Meanwhile, we may compute the elements of Chan and Olkin's preconditioner for the (i,j) block and its eigenvalues, $\lambda_{ij}(s)$, in terms of the covariances, $\omega_{i,j}(r)$:

$$\begin{aligned} c_r &= \frac{1}{T}\left(r\omega_{ij}(-(T-r)) + (T-r)\omega_{ij}(r)\right) \\ \lambda_{ij}(s) &= \sum_{r=0}^{T-1} \frac{r}{T}\omega_{ij}(-(T-r)) \exp(-ir\nu_s) + \sum_{r=0}^{T-1} \frac{T-r}{T}\omega_{ij}(r) \exp(-ir\nu_s) \\ &= \sum_{q=-(T-1)}^{-1} \frac{T-|q|}{T}\omega_{ij}(q) \exp(-iq\nu_s) + \sum_{r=0}^{T-1} \frac{T-r}{T}\omega_{ij}(r) \exp(-ir\nu_s) \end{aligned}$$

where the last line follows from the substitution $q = -(T-r)$. Notice that the last line equals the (i,j) element of $E(I(\nu_s))$. Thus, the s^{th} eigenvalue of the (i,j) block of the preconditioner equals the expected value of the cross-periodogram of X_{it} and X_{jt} at ν_s. This corresponds to the results found in the univariate case, given in Chen et al. (2006, section 4).

6.5 Prediction

The preconditioned conjugate gradient algorithm which we have discussed can also be applied to efficiently compute the best linear predictor of multivariate processes. Notice that, for any Gaussian time series and lead time $h > 0$,

$$E(X_{T+h}|X) = E(X_{T+h}) + Cov(X, X_{T+h})Cov(X)^{-1}(X - E(X))$$

The preconditioned conjugate gradient algorithm can be used to compute $Cov(X)^{-1}(X - E(X))$, and the remaining multiplication can be computed in $O(TK)$ time. This gives an efficient prediction computation based on the full sample and known covariance structure, which allows us to avoid computing an autoregressive approximation.

7 Computing the Determinant

Let $\Omega(T)$ be the covariance matrix of T observations of any multivariate process, X_t, that has an infinite moving average representation driven by innovations, ϵ_t, that have covariance matrix $\Sigma = E(\epsilon_t \epsilon_t')$. As before, let the autocovariance matrix of X_t at lag r be $\omega(r)$. According to Sowell (1989a), we may write:

$$\begin{aligned} |\Omega(T)| &= \prod_{r=0}^{T-1} |v(r)| \\ v(r) &= v(0) - \Upsilon(r)' \Omega(r)^{-1} \Upsilon(r) \\ \Upsilon(r) &= \begin{pmatrix} \omega(-1) \\ \vdots \\ \omega(-r) \end{pmatrix} \end{aligned}$$

$v(r)$ is the prediction variance of X_t given $X_{t-1}, ..., X_{t-r}$. Notice that we may use the PCG algorithm presented in section 6 K times to compute $\Omega(r)^{-1}\Upsilon(r)$, by using PCG on each column of $\Upsilon(r)$ separately. This means that $v(r)$ can be computed efficiently for any particular value of r. However, computing all of the $v(r)$ using the PCG algorithm would be slower than the $O(T^2)$ time required by Sowell's (1989a) method presented in section 4.2. Instead, we use our knowledge of $|v(r)|$ as a function of r and consider a variety of methods which may allow us to approximate the determinant in less processing time.

We begin by noting a few facts about $|v(r)|$ as a function of r. These facts hold for any multivariate time series with a moving average representation. First, $|v(r)|$ is a non-increasing function, since

$$\begin{aligned} |v(r)| &= |\text{Var}(X_t | X_{t-1}, ..., X_{t-r})| \\ &\geq |\text{Var}(X_t | X_{t-1}, ..., X_{t-r}, X_{t-r-1})| \\ &= |v(r+1)| \end{aligned}$$

Second, $|v(r)|$ is bounded below by $|\Sigma|$, since ϵ_t is uncorrelated with all past observations. Third,

we can use the equations given in Sowell's algorithm to find that:

$$\begin{aligned}
v(r) - v(r-1) &= -\sum_{j=1}^{r} A(r,j)\omega(-j) + \sum_{j=1}^{r-1} A(r-1,j)\omega(-j) \\
&= \left(\sum_{j=1}^{r-1}(A(r-1,j) - A(r,j))\omega(-j)\right) - A(r,r)\omega(-j) \\
&= \left(\sum_{j=1}^{r-1} A(r,r)\bar{A}(r-1,r-j)\omega(-j)\right) - A(r,r)\omega(-j) \\
&= A(r,r)\bar{D}(r) \\
&= A(r,r)\bar{A}(r,r)v(r-1) \\
\frac{|v(r)|}{|v(r-1)|} &= |I - A(r,r)\bar{A}(r,r)|
\end{aligned}$$

This result is the analog of the result in the univariate case that $\frac{v(r)}{v(r-1)} = 1 - \phi_r^2$, where ϕ_r is the r^{th} partial autocorrelation (for example, Brockwell and Davis, 1993).

In addition, we have found empirically for both FIVAR and VARFI models that $|v(r)|$ is quite smooth as a function of r, when $r > 0$. This smoothness does not hold as well at $|v(0)|$, since the inclusion of the first lagged value in predictions reduces the prediction variance quite dramatically because of the long memory. (See Figure 8 for an example.) This observation, together with the theoretical facts about $|v(r)|$, inform our choice of methods to compute the determinant.

In the univariate case, Chen et al. (2006) suggest using an asymptotic approximation given by Bottcher and Silbermann (1999). Also in the univariate case, Rohit Deo (private communication) has proposed an exact computational method, which generalizes to VARFI processes but not to FIVAR processes; we discuss the generalization in section 7.3. Sowell's (1989a) decomposition can be used to compute the exact determinant for univariate and multivariate processes, but it requires $O(T^2)$ time. In this section, we will discuss two alternatives to Sowell's method. First, we describe asymptotic approximations. Second, we discuss approximations that use curve-fitting and show the effectiveness of this method. In the final two sections, we describe ways to compute VARFI determinants and the determinants of cointegrated systems; these two methods would be exact if we had exact expressions for the determinants of the covariance matrices associated with $VARFI(0, \vec{d})$ or FIVAR processes respectively. None of the possible alternatives is exact. However, as we will see in section 9, using the approximation we describe instead of Sowell's exact determinant does not change parameter estimates by very much while it does speed up computation.

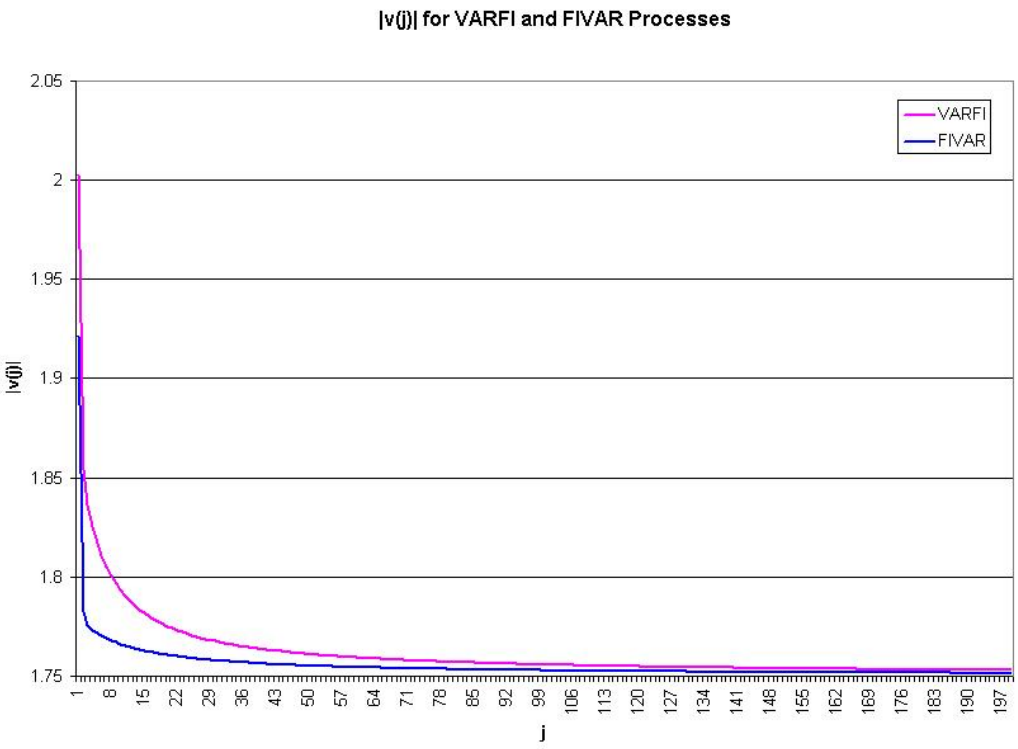

Figure 8: $|v(r)|$ for VARFI and FIVAR processes for r ranging from 1 to 199.

7.1 Asymptotic approximations to determinants

In this section, we will discuss two asymptotic approximations to the determinant. In our presentation, we will use two different notations for different types of asymptotic formulas. First, we write $f(T) \sim g(T)$ if $\lim_{T\to\infty} \frac{f(T)}{g(T)} = 1$. In some cases, we will also consider $\lim_{t\to\infty} \frac{f(T)}{f(T-1)}$. Notice that, if $f(T) \sim g(T)$ and $\lim_{T\to\infty} g(T) \neq 0$:

$$\begin{aligned}
\lim_{T\to\infty} \frac{f(T)/f(T-1)}{g(T)/g(T-1)} &= \lim_{T\to\infty} \left(\frac{f(T)}{g(T)}\right)\left(\frac{g(T-1)}{f(T-1)}\right) \\
&= \lim_{T\to\infty} \left(\frac{f(T)}{g(T)}\right) \lim_{T\to\infty} \left(\frac{g(T-1)}{f(T-1)}\right) \\
&= 1
\end{aligned}$$

and $\frac{f(T)}{f(T-1)} \sim \frac{g(T)}{g(T-1)}$. If we take the logarithm of $\frac{f(T)}{g(T)}$, we have $\log f(T) = \log g(T) + o(1)$.

We now consider asymptotic approximations to either the overall determinant, $|\Omega(T)|$, or to the individual $|v(r)|$. In the univariate case, Bottcher and Silbermann (1999) give an asymptotic formula for the overall determinant of a univariate ARFIMA process . Taking the ratio of the approximations for r and $r-1$ yields an approximation for $|v(r)|$ in the univariate case. This approximation is:

$$\begin{aligned}
|v(r)| &\sim |\Sigma| \exp\left(\frac{d^2}{r-1}\right) \\
\log|v(r)| &= \log|\Sigma| + \frac{d^2}{r-1} + o(1)
\end{aligned}$$

Torsten Ehrhardt (private communication) found that this asymptotically correct formula can be extended to the multivariate case by replacing d^2 by a different constant. In the case of a $VARFI(0,\vec{d},0)$ or $FIVAR(0,\vec{d},0)$ model where Σ is invertible, he has worked out the expression for this constant. Let δ be the $K\times K$ diagonal matrix with $e^{-2\pi i d_1}, ..., e^{-2\pi i d_K}$ along the diagonal. Define

$$U = \Sigma\delta^*\Sigma^{-1}\delta$$

Let $e^{2\pi i u_1}, ..., e^{2\pi i u_K}$ be the eigenvalues of U. Since $|U| = |\Sigma||\delta^{-1}||\Sigma^{-1}||\delta| = 1$, we must have $1 = \prod_{k=1}^K e^{2\pi i u_k} = e^{2\pi i \sum_{k=1}^K u_k}$. While it is always possible to choose the u_k so that $\sum_{k=1}^K u_k = 0$, Ehrhardt's expression might not hold if the u_k chosen are not the principal branch logarithms. However, Ehrhardt conjectures that for any choice where $|Re(u_k) - Re(u_j)| < 1$, the method will continue to hold. Given choices for u_k which obey this condition and which sum to 0, we have $|v(r)| \sim |\Sigma| \exp\left(\frac{\sum_{k=1}^K d_k^2 - \frac{1}{2}\sum_{k=1}^K u_k^2}{r-1}\right)$. We find that, when there is no short memory component, Ehrhardt's approximation improves monotonically as n increases, as seen in Figures 9 and 10. In addition, the error in the approximation is always of the same sign. The assumption that the error is monotonically decreasing allows us to bound the error in the approximation beyond a certain point. However, Ehrhardt's approximation does not give us a way to reduce the error

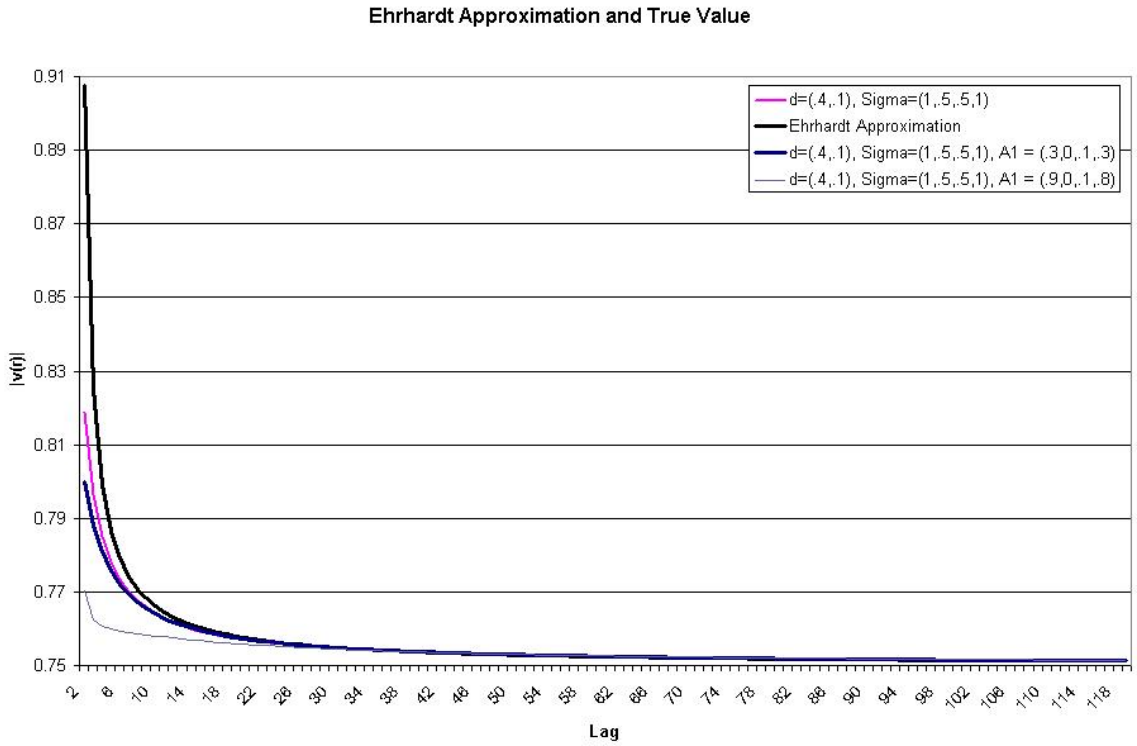

Figure 9: The Ehrhardt approximation to $|v(r)|$ and the true values for $|v(r)|$ for a variety of FIVAR processes.

beyond the initial approximation because there are no higher order terms. In fact, even if the term in the exponent is not correct, the approximation will eventually be close, since both the approximation and the true values tend toward $|\Sigma|$; in this case, the errors might not decrease monotonically and the approximation might not be as accurate for small n; we can see in this in Figure 10, looking at the two lines with $A_1 \neq 0$. However, as we see in Figure 10, Ehrhardt's approximation does not work well for small values of r.

We also consider the simpler approximation of the determinant of the covariance matrix by $|\Sigma|^T$, as suggested by Dunsmuir and Hannan (1976) and others. This is equivalent to approximating $|v(r)|$ by Σ for all r. This approximation ignores the $\exp\left(\frac{\sum_{k=1}^{K} d_k^2 - \frac{1}{2}\sum_{k=1}^{K} u_k^2}{r-1}\right)$ term of Ehrhardt's approximation. This term does not exist in short memory cases, since each $d_k = u_k = 0$. Furthermore, in the case of a $VAR(p)$ process, $v(r) = \Sigma$ for all $r \geq p$, because the prediction error based on the previous p observations is simply the next innovation, ϵ_t. In that

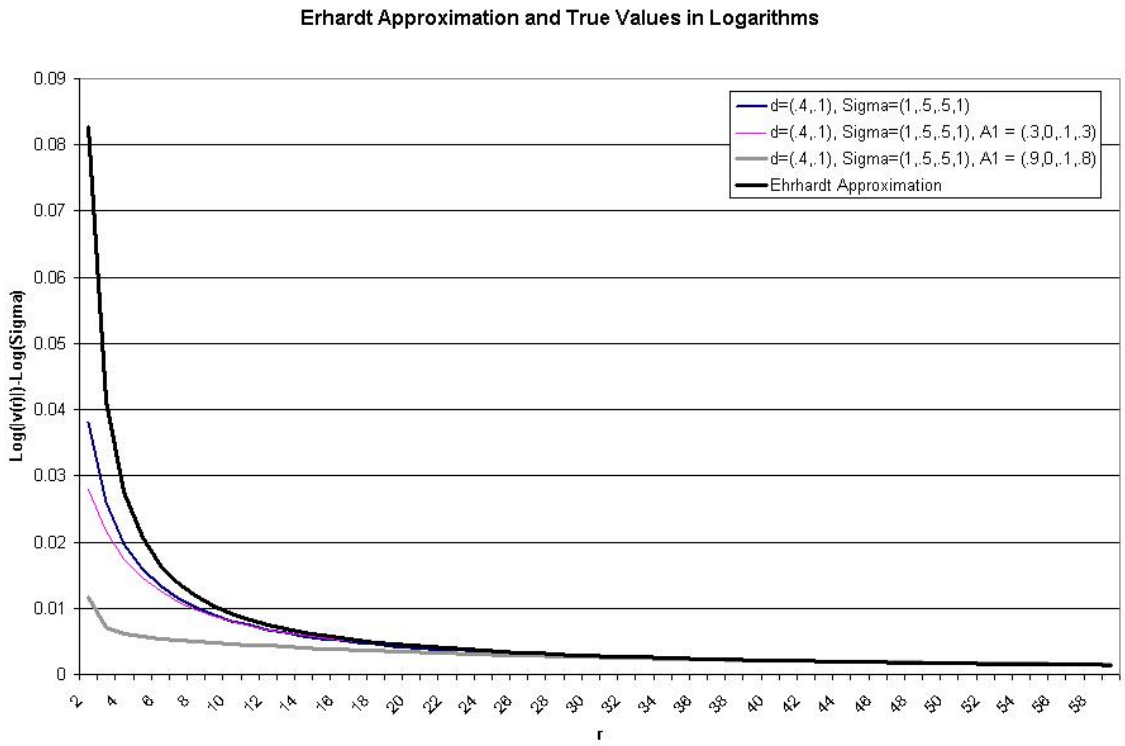

Figure 10: The Ehrhardt approximation to $|v(r)|$ and the true values for $|v(r)|$ for a variety of FIVAR processes, divided by $|\Sigma|$ and then logged.

case,

$$\begin{aligned} |\Omega(T)| &= \prod_{r=0}^{T-1} |v(r)| \\ &= |\Sigma|^{T-p} \prod_{r=0}^{p-1} |v(r)| \end{aligned}$$

Thus, for vector autoregressions, computation of the exact determinant requires the computation of only a fixed number of initial $|v(r)|$. Furthermore,

$$\begin{aligned} \lim_{T\to\infty} \frac{1}{T} \log |\Omega(T)| &= \lim_{T\to\infty} \frac{1}{T} \left((T-p) \log |\Sigma| + \sum_{r=0}^{p-1} \log |v(r)| \right) \\ &= \log |\Sigma| \end{aligned}$$

For this reason, it seems reasonable to approximate $|\Omega(T)|$ by $|\Sigma|^T$ for vector autoregressive models, even though the term containing $\log |\Omega(T)|$ is not divided by T in the expression for the likelihood. In a more general univariate case, under the conditions of a theorem of Grenander and Szego (1958, page 76), we must have:

$$\lim_{T\to\infty} \frac{\Omega(T)}{|\Sigma|^T} = C$$

where C is a constant that depends on the moving average representation of X_t. Even in this simple case, the assumption that $C = 1$ will not be accurate. This approximation has additional problems in the long memory case. One of the conditions of Grenander and Szego's theorem is that the spectral density, f, is differentiable and that the derivative, f', obeys:

$$|f'(x_1) - f'(x_2)| < K|x_1 - x_2|^\alpha$$

with $K > 0$ and $0 < \alpha < 1$; this excludes the case of long memory. The approximations offered by Dunsmuir and Hannan (1976) and Luceno (1996) for more general models assume that this limit continues to hold. However, the Ehrhardt approximation shows that $\log |v(r)| = \log |\Sigma| + \frac{\sum_{k=1}^K d_k^2 - \frac{1}{2}\sum_{k=1}^K u_k^2}{r-1} + o(1)$. Since $\sum_{r=0}^T \frac{1}{r-1}$ diverges as $T \to \infty$, the approximation of $|\Omega(T)|$ by $|\Sigma|^T$ may not be good enough for estimation. Our results in section 9.2 confirm this.

7.2 Determinant approximations using curve-fitting

Instead of using an asymptotic approximation, we consider using regression and curve-fitting as a way to interpolate between a few computed values of $|v(r)|$. We expect that the best fits will come from functions which are decreasing and have a finite asymptotic value, so that they can mimic the known behavior of $|v(r)|$. For such a method to be feasible, we must be able to find

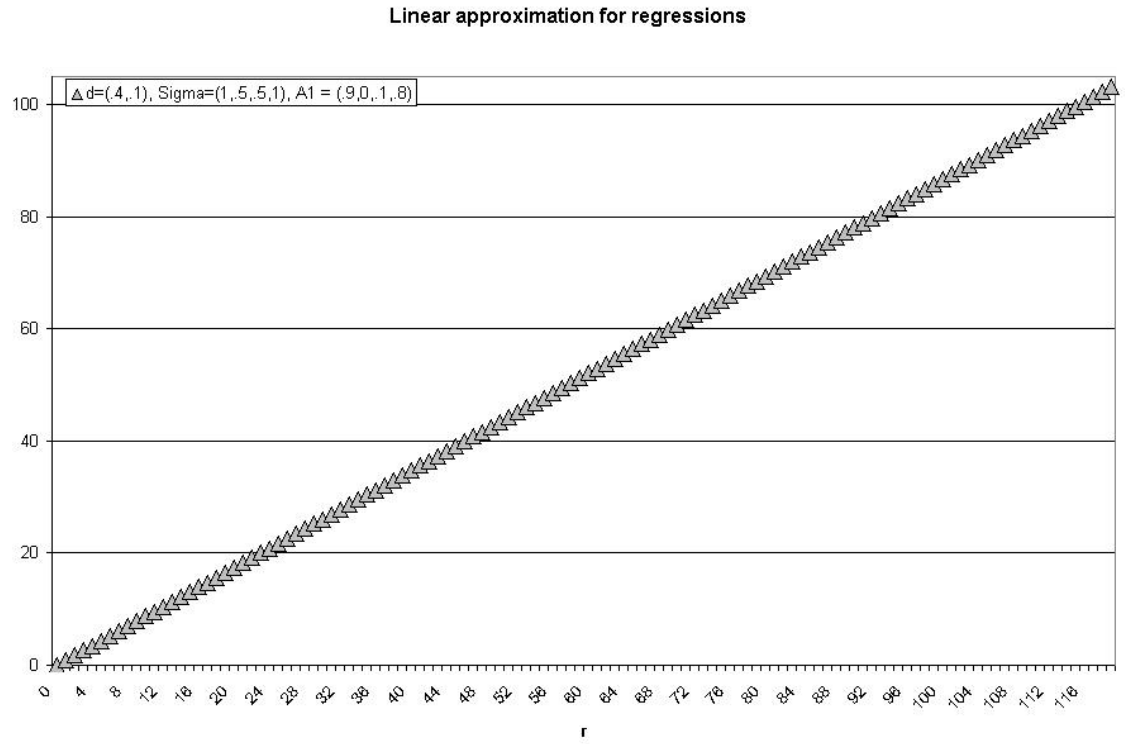

Figure 11: A plot of r versus $r\sqrt{|v(r)|}$ for the FIVAR process with $d = (0.1, 0.4)$, $\Sigma = (1, 0.5, 0.5, 2)$ and $A_1 = (0.7, 0.1, 0.2, 0.9)$.

a fit that is reasonably accurate based on computing only a subset of the $|v(r)|$ exactly, using either Sowell's method or PCG. We focus on fitting:

$$r\sqrt{|v(r)|} \quad = \quad \alpha + \beta r$$

This relationship is equivalent to:

$$|v(r)| \quad = \quad \beta^2 + \frac{2\alpha\beta}{r} + \frac{\alpha^2}{r^2}$$

which is decreasing and smooth in r. In this formulation, β^2 is able to adjust to match the asymptotic value of $|v(r)|$.

We will combine curve-fitting with the application of Sowell's method to an initial set of points. Though Sowell's method is too slow to use to compute $|v(r)|$ for all $r = 0, ..., T-1$ when T is large, it can be used for some of the initial points, $r = 0, ..., S$, where the curve may be hardest to fit and the approximation is least accurate. As long as the initial segment of points

used with Sowell's method grows more slowly than T, we can use this method to compute some of the determinants of the prediction variances exactly without much additional computational cost.

Our current method combines a regression with the application of Sowell's method. First, we apply Sowell's method to compute $|v(r)|$ for $r = 0, ..., S$, for some S (we use 32 in our program). Then, we use PCG to compute $|v(T-1)|$. We then regress $r\sqrt{|v(r)|}$ on r for $r = 1, .., S, T-1$. Using the fitted line, we estimate $|v(r)|$ for all the points where $|v(r)|$ is unknown.

Algorithm 8 *Approximating $|\Omega(T)|$ through curve-fitting.*

1. *Use Sowell's algorithm (Algorithm 3) to compute $|v(r)|$ for $r = 0, ..., S$.*
2. *Compute $|v(T-1)|$ using the PCG algorithm:*
   (a) *Set Υ to be the $KT \times K$ matrix which stacks the autocovariance matrices, $\omega(-1), ..., \omega(-r)$.*
   (b) *Set G to be a $KT \times K$ matrix.*
   (c) *For $i = 1, ..., K$, compute the i^{th} column of G as $\Omega^{-1}\Upsilon(\cdot, i)$ using the PCG algorithm, where $\Upsilon(\cdot, i)$ is the i^{th} column of Υ.*
   (d) *Compute $v(T-1) = \Upsilon' G$.*
   (e) *Compute $|v(T-1)|$.*
3. *Regress $r\sqrt{|v(r)|}$ on r for the points $r = 1, .., S, T-1$.*
4. *Compute the fitted values, $\widehat{|v(r)|}$ for $r = S+1, ..., T-2$ based on the fitted values from the regression.*
5. *Sum the logarithms of $|v(0)|, ...|v(S)|, |\widehat{v(S+1)}|, ..., |\widehat{v(T-2)}|, |v(T-1)|$ to find the approximate log determinant.*

While this method is ad hoc, Tables 8 and 9 show that it performs well for both FIVAR and VARFI models. The approximation is closest when A_1 is far from the unit circle, but our approximate log determinant is within 0.5 of Sowell's exact log determinant even in the case where $A_1 = \begin{pmatrix} .7 & .2 \\ .1 & .9 \end{pmatrix}$, which has one eigenvalue greater than 0.97. The approximation is better for VARFI than for FIVAR models. The difference in computing time between Sowell's exact method and our regression-based approximation is quite large; when $T = 1000$, Sowell's algorithm takes almost 70 times longer than our approximation. Furthermore, we will see in Section 9.2 that the maximum likelihood estimates for the parameters based on using this determinant are close to those from Sowell.

Ideally, we also wish to move from this approximation to an approximation which can be made as close as desired with some additional computations; to accomplish this, we must find

T	A_1	d	Sowell Time	Sowell Value	Regression Time	Regression Value	Naive Approximation
250	(0,0,0,0)	(.4,.1)	3.966	141.7575	0.292	141.7568	139.9039
250	(.4,.2,.1,.6)	(.4,.1)	3.950	143.6495	0.311	143.6363	139.9039
250	(.7,.2,.1,.9)	(.4,.1)	3.978	151.4243	0.395	151.2217	139.9039
500	(0,0,0,0)	(.4,.1)	18.359	281.7858	0.683	281.7827	279.8079
500	(.4,.2,.1,.6)	(.4,.1)	18.302	283.7176	0.751	283.6769	279.8079
500	(.7,.2,.1,.9)	(.4,.1)	18.184	291.8804	1.215	291.4227	279.8079
1000	(0,0,0,0)	(.4,.1)	74.211	561.7179	0.798	561.7127	559.6158
1000	(.4,.2,.1,.6)	(.4,.1)	73.016	563.6902	0.864	563.623	559.6158
1000	(.7,.2,.1,.9)	(.4,.1)	74.146	572.2505	1.146	571.5228	559.6158
250	(0,0,0,0)	(.4,.49)	3.883	145.9179	0.313	145.9187	139.9039
250	(.4,.2,.1,.6)	(.4,.49)	3.895	148.6055	0.320	148.5785	139.9039
250	(.7,.2,.1,.9)	(.4,.49)	3.880	157.7377	0.552	157.6283	139.9039
500	(0,0,0,0)	(.4,.49)	18.134	286.1003	0.734	286.1026	279.8079
500	(.4,.2,.1,.6)	(.4,.49)	18.167	288.7922	0.797	288.7112	279.8079
500	(.7,.2,.1,.9)	(.4,.49)	18.198	298.052	1.633	297.8051	279.8079
1000	(0,0,0,0)	(.4,.49)	73.729	566.18648	0.858	566.19019	559.6158
1000	(.4,.2,.1,.6)	(.4,.49)	72.877	568.88358	0.901	568.75156	559.6158
1000	(.7,.2,.1,.9)	(.4,.49)	74.253	578.28725	1.267	577.86903	559.6158

Table 8: The computed value of the log determinant and the processing time required to do the computation using Sowell's algorithm and using the regression-based approximation. The naive approximation is $\log |\Sigma|^T$. All models are FIVAR processes with $\Sigma = (1, .5, .5, 2)$. Times are the mean time taken over 100 repetitions of the calculation.

T	A_1	d	Sowell Time	Sowell Value	Regression Time	Regression Value	Naive Approximation
250	(0,0,0,0)	(.4,.1)	3.930	141.75751	0.294	141.75678	139.9039
250	(.4,.2,.1,.6)	(.4,.1)	4.004	143.06590	0.320	143.05746	139.9039
250	(.7,.2,.1,.9)	(.4,.1)	3.919	147.48359	0.346	147.44006	139.9039
500	(0,0,0,0)	(.4,.1)	17.905	281.78576	0.679	281.78269	279.8079
500	(.4,.2,.1,.6)	(.4,.1)	17.998	283.09378	0.738	283.06462	279.8079
500	(.7,.2,.1,.9)	(.4,.1)	17.919	287.50407	0.868	287.40505	279.8079
1000	(0,0,0,0)	(.4,.1)	73.589	561.71790	0.797	561.71271	559.6158
1000	(.4,.2,.1,.6)	(.4,.1)	72.954	563.02573	0.841	562.97756	559.6158
1000	(.7,.2,.1,.9)	(.4,.1)	73.227	567.43262	0.905	567.27000	559.6158
250	(0,0,0,0)	(.4,.49)	3.949	145.91789	0.304	145.91866	139.9039
250	(.4,.2,.1,.6)	(.4,.49)	3.922	148.03271	0.320	148.00202	139.9039
250	(.7,.2,.1,.9)	(.4,.49)	3.894	153.65466	0.538	153.57598	139.9039
500	(0,0,0,0)	(.4,.49)	17.932	286.10030	0.737	286.10259	279.8079
500	(.4,.2,.1,.6)	(.4,.49)	17.871	288.21319	0.795	288.12364	279.8079
500	(.7,.2,.1,.9)	(.4,.49)	18.020	293.81212	1.531	293.64902	279.8079
1000	(0,0,0,0)	(.4,.49)	72.464	566.18648	0.853	566.19019	559.6158
1000	(.4,.2,.1,.6)	(.4,.49)	72.054	568.29840	0.889	568.15291	559.6158
1000	(.7,.2,.1,.9)	(.4,.49)	72.932	573.88486	1.213	573.61398	559.6158

Table 9: The computed value of the log determinant and the processing time required to do the computation using Sowell's method and using the regression-based approximation. All models used $\Sigma = (1, .5, .5, 2)$ and a VARFI process.

a way to bound the approximation error and reduce the error if desired. The approximation method we will present does not do this.

7.3 An alternative way to compute the determinant of a VARFI process

We now consider an alternative way to compute the covariances of the $VARFI(1, \vec{d})$ process, $X_.$. This algorithm for computing the determinant is a generalization of a univariate algorithm given by Rohit Deo (private communication). This method would be exact if we could compute the determinant of a $VARFI(0, \vec{d})$ process exactly. Since our approximation is close for such processes, we expect this approximation will also be close.

As before, let Ω be the covariance matrix of X, and $\omega(h) = \mathrm{Cov}(X_t, X_{t-h})$ be the $K \times K$ autocovariance matrix at lag h. Define a new process, W_t, by:

$$\begin{aligned} W_1 &= X_1 \\ W_t &= X_t - A_1 X_{t-1} \end{aligned}$$

Then, $W = (W_1', ..., W_T')'$ can be written as $W = BX$, where $|B| = 1$. Thus, $|\mathrm{Var}(W)| = |B'\Omega B| = |\Omega|$, and it is sufficient to compute $|\mathrm{Var}(W)|$. Notice that:

$$\mathrm{Var}(W) = \begin{pmatrix} \omega(0) & C' \\ C & \Phi(T-1) \end{pmatrix}$$

where $\omega(h)$ is the autocovariance of the original process, $\Phi(T-1)$ is the covariance matrix of a $VARFI(0, \vec{d}, 0)$ process of length $T-1$ and C is the $K(T-1) \times K$ matrix given by:

$$C = \begin{pmatrix} \omega(1) - A_1\omega(0) \\ \vdots \\ \omega(T-1) - A_1\omega(T-2) \end{pmatrix}$$

Using a formula for the determinant of a partitioned matrix (Sowell, 1989a), we compute:

$$|\mathrm{Var}(W)| = |\Phi(T-1)| \cdot |\omega(0) - C'\Phi(T-1)^{-1}C|$$

The first term must be computed using the method given in the previous section. The product $\Phi(T-1)^{-1}C$ can be computed using the PCG algorithm K times, once for each column of C. Then, since $\omega(0) - C'\Phi(T-1)C$ is a $K \times K$ matrix, computation of the determinant can be done quickly using standard methods.

7.4 Determinants of Cointegrated Systems

Let $\gamma(j)$ be the autocovariance sequence of a cointegrated system. Using the results from section 5.4, we know that $\gamma(j) = V^{-1}\omega(j)(V^{-1})'$, where $\omega(j)$ is the autocovariance sequence of

the corresponding FIVAR process. Let

$$\begin{aligned} \Gamma(T) &= (V^{-1} \otimes I)\tilde{\Omega}(T)((V^{-1})' \otimes I) && (15) \\ |\Gamma(T)| &= |V|^{-2T}|\Omega(T)| && (16) \end{aligned}$$

If we use a lower triangular representation with ones along the diagonal for the cointegrating relationship, then $|V| = 1$, and the determinant of the covariance matrix of a cointegrated system equals the determinant of the covariance matrix of the system before it is cointegrated. Even if we do not impose a restriction that implies that $|V| = 1$, this computation in equation (16) takes $O(1)$ time once $|\Omega(T)|$ is known.

8 Efficient Simulation

In this section, we present an efficient algorithm for simulating from a vector ARFIMA process with normally distributed innovations. Our approach extends the method proposed by Davies and Harte (1987). Wood and Chan (1994) described the algorithm for a univariate time series in more detail and extended the algorithm to spatial time series in multiple dimensions but not to multivariate time series. The algorithm described in this section may be applied to other stationary multivariate time series, assuming that the conditions described are met.

As before, let Ω be the covariance matrix of the vector containing T periods of a stationary K-variate time series, where the data is grouped by series. The underlying idea of this algorithm is to embed Ω in a covariance matrix for a random vector which it is easy to simulate.

Recall that Ω is a block Toeplitz matrix, with K^2 blocks of size $T \times T$. Let $C(\Omega)$ be a block circulant embedding of Ω, where each block, $C_{ij}(\Omega)$, is of dimension M, with $M \geq 2T - 1$ and odd. We set the first row of $C(\Omega)$ equal to $\omega_{ij}(0), ..., \omega_{ij}(\frac{M-1}{2}), \omega_{ij}(-\frac{M-1}{2}), ..., \omega_{ij}(-1)$. Unlike the circulant embedding used in section 3.3, this embedding does not include a second diagonal with $\omega_{ij}(0)$.

Because $C(\Omega)$ is a block circulant matrix, we can apply the results of section 3.2 to write it as:

$$C(\Omega) = (I \otimes F^*)PB(\Omega)P'(I \otimes F)$$

where $B(\Omega)$ is a matrix with M blocks, $B_1, ..., B_M$, of size $K \times K$ along the diagonal. Using this representation, $C(\Omega)^{1/2}$ is straightforward to compute, using either the eigenvalue decomposition of each B_r or the algorithm of Denman and Beavers (1976). Notice, however, that $C(\Omega)$ need not be a positive definite matrix. If it is not, the algorithm below will not apply, since $C(\omega)$ must be a covariance matrix for simulation. However, Wood and Chan (1994, proposition 2) notes that, in the cases they consider, there is always a sufficiently large M such that the circulant embedding of size $M \times M$ will be positive definite. We have also found that omitting the second diagonal of $\omega_{ij}(0)$ generally results in a matrix that is positive definite. Because we do not repeat

$\omega_{ij}(0)$ but we do repeat $\omega_{ij}(r)$ for every other r, M must be odd. For the efficiency of the fast Fourier transform, we recommend choosing M such that it has many small factors; choosing M to be a power of three allows it to be odd and have many factors. All of these considerations yield the following algorithm for computing B, which is a specialization of Algorithm 1 to this case:

Algorithm 9 Preparation for simulation using block circulant embedding.

1. *Choose $M = 3^R$, where $3^{R-1} < 2T - 1 \leq 3^R$.*
2. *Compute the $K \times K$ autocovariances, γ, at lags $-\frac{M-1}{2}, ..., 0, ..., \frac{M-1}{2}$.*
3. *Set the first row of each block of the circulant embedding equal to $\omega_{ij}(0), ..., \omega_{ij}(\frac{M-1}{2}), \omega_{ij}(-\frac{M-1}{2}), ..., \omega_{ij}(-1)$.*
4. *Compute the inverse fast Fourier transform of each first block's first row. This yields $B_r(i, j)$.*
5. *For each $r = 1, .., M$, compute the eigenvalue decomposition of B_r. If any of the eigenvalues are negative, set M to $3M$ and return to step 2. Otherwise, compute $B_r^{1/2}$ and store the result.*

Given $C(\Omega)$, we require an algorithm to simulate a random vector with that covariance matrix. We extend the univariate algorithm of Wood and Chan (1994, section 5.1.2) to simulation from a block-circulant covariance matrix. Consider the random variable, $U \sim Normal(0, I_{M \times M})$. As long as $C(\Omega)$ is positive definite, $C(\Omega)^{1/2}U$ exists and has covariance matrix $C(\Omega)$. The subvector of $C(\Omega)^{1/2}U$ defined by the first T elements of each of the K series has covariance matrix Ω. Thus, a fast method for simulating $C(\Omega)^{1/2}U$ yields a simulation method for the original multivariate time series. This suggests the following algorithm, in which we describe each step in terms of the spectral decomposition of $C^{1/2}$ given in section 3.2:

Algorithm 10 Simulation.

- *$(I \otimes F)X$: Compute vectors $Y_{1\cdot}, ..., Y_{K\cdot}$ of length M with $Var(Y_{k,\cdot}) = FF^*$, using the method given in Wood and Chan (1994, section 5.1.2).*
- *$P'(I \otimes F)X$: Combine these vectors in the order $(Y_{11}, ..., Y_{K1}, ..., Y_{1M}, ..., Y_{KM})'$.*
- *$B_\Omega^{1/2}P'(I \otimes F)X$: Compute $B_r^{1/2}Y_r$ for each $r = 1, ..., M$.*
- *$PB_\Omega^{1/2}P'(I \otimes F)X$: Re-sort the vector to group the observations by series instead of by time.*
- *$(I \otimes F^*)PB_\Omega^{1/2}P'(I \otimes F)X$: Take the fast Fourier transform of each $Y_{k,\cdot}$ for $k = 1, ..., K$.*

T	Sowell Setup	Sowell Simulation	Circulant Setup	Circulant Simulation	M
4	0.003	0.005	0.017	0.001	9
8	0.007	0.009	0.047	0.003	27
16	0.021	0.021	0.132	0.008	81
32	0.078	0.052	0.130	0.008	81
64	0.273	0.142	0.375	0.023	243
128	1.039	0.414	1.091	0.070	729
256	4.074	1.365	1.090	0.070	729
512	16.248	4.764	3.257	0.205	2187
1024	63.666	17.698	3.260	0.210	2187

Table 10: Processing time needed to set up for simulation and simulate from a $FIVAR(0, \vec{d})$ with $d = (0.1, 0.4)$ and $\Sigma = (1, 0.5, 0.5, 2)$. Estimates for the setup time are based on 100 repetitions; estimates for the simulation times are based on 1000 repetitions.

- *Return the first T observations from each vector, $Y_{k,\cdot}$.*

Consider the time requirements of this method. The initialization algorithm is run once. Given a choice of M, the computation of the autocovariances and fast Fourier transforms takes $O(M \log_3 M)$ time, while the eigenvalue calculations take $O(M)$ time; as in the other algorithms we have presented, larger values of K will slow these steps down. The required M is unknown, but we found in our experiments that it needed to be increased from the initial value given in Step 1 of Algorithm 9 when $T = 4$ (see Tables 10, 11, and 12). In that case, $M = O(T)$. The simulation step also uses Fast Fourier Transforms, so that it also runs in $O(M \log_3 M)$ time. In contrast, the method of Sowell (1989a) requires an initial computation of his matrix decomposition, which takes $O(T^2)$ steps; each simulation takes another $O(T^2)$ steps, since the computation of:

$$X_t = \sum_{j=1}^{t-1} \bar{A}(t-1, t-j) X_j + \bar{v}(t-1)^{1/2} u_t$$

for $t = 1, ..., T$ will require $\frac{T(T-1)}{2}$ summations.

In Table 10, we show the processing time required for initialization of the algorithm and for each simulation for both Sowell and the block circulant embedding algorithm. In this test, the processing time required to compute the covariances for Sowell's simulation method is not included in the setup, but it is included in the setup for circulant embedding, since there was the change that M would need to be increased. Despite this disadvantage, our method always faster for simulation and is faster for the initialization except for small values of T.

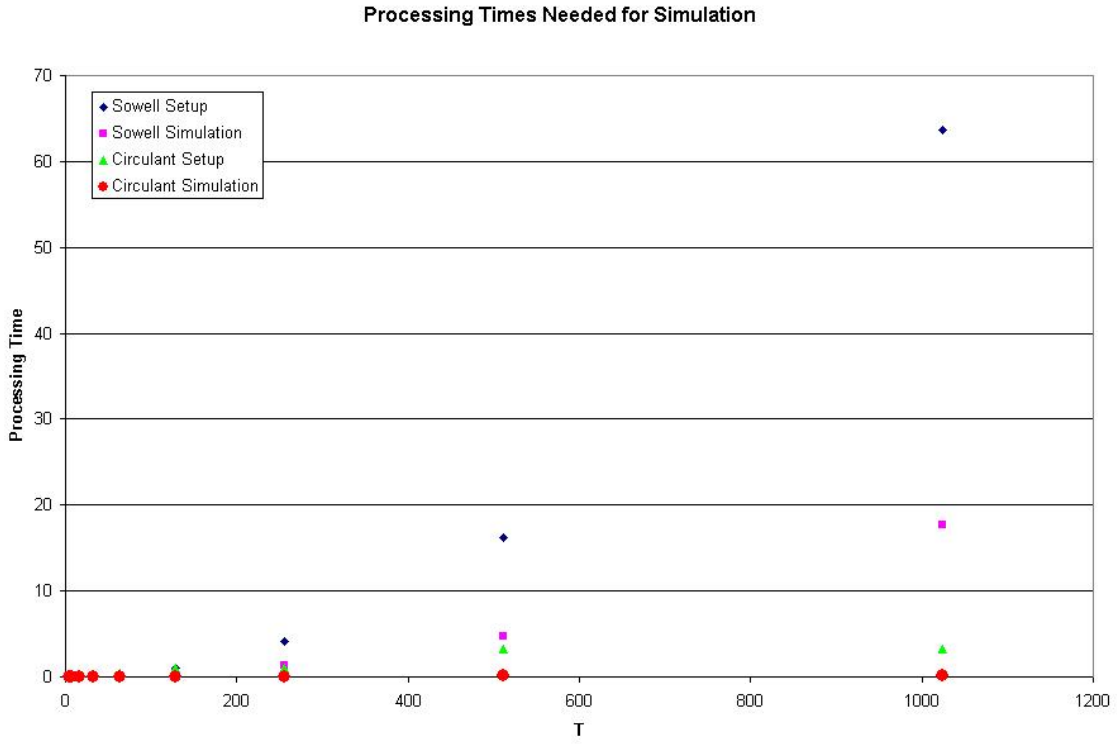

Figure 12: Processing time needed to set up for simulation and simulate from a $FIVAR(0, \vec{d})$ with $d = (0.1, 0.4)$ and $\Sigma = (1, 0.5, 0.5, 2)$. Estimates for the setup time are based on 100 repetitions; estimates for the simulation times are based on 1000 repetitions.

T	Sowell Setup	Sowell Simulation	Circulant Setup	Circulant Simulation	M
4	0.005	0.012	0.222	0.005	27
8	0.013	0.025	0.128	0.005	27
16	0.041	0.058	0.293	0.013	81
32	0.131	0.130	0.275	0.012	81
64	0.450	0.302	0.793	0.039	243
128	1.753	0.824	2.198	0.117	729
256	6.670	2.544	2.118	0.113	729
512	27.383	9.062	6.559	0.352	2187
1024	111.766	36.267	6.741	0.397	2187

Table 11: Processing time needed to set up for simulation and simulate from a $FIVAR(1, \vec{d})$ with $d = (0.1, 0.4)$, $\Sigma = (1, 0.5, 0.5, 2)$, and $A_1 = (0.6, -0.1, 0.2, 0.8)$. Estimates for the setup time are based on 100 repetitions; estimates for the simulation times are based on 1000 repetitions.

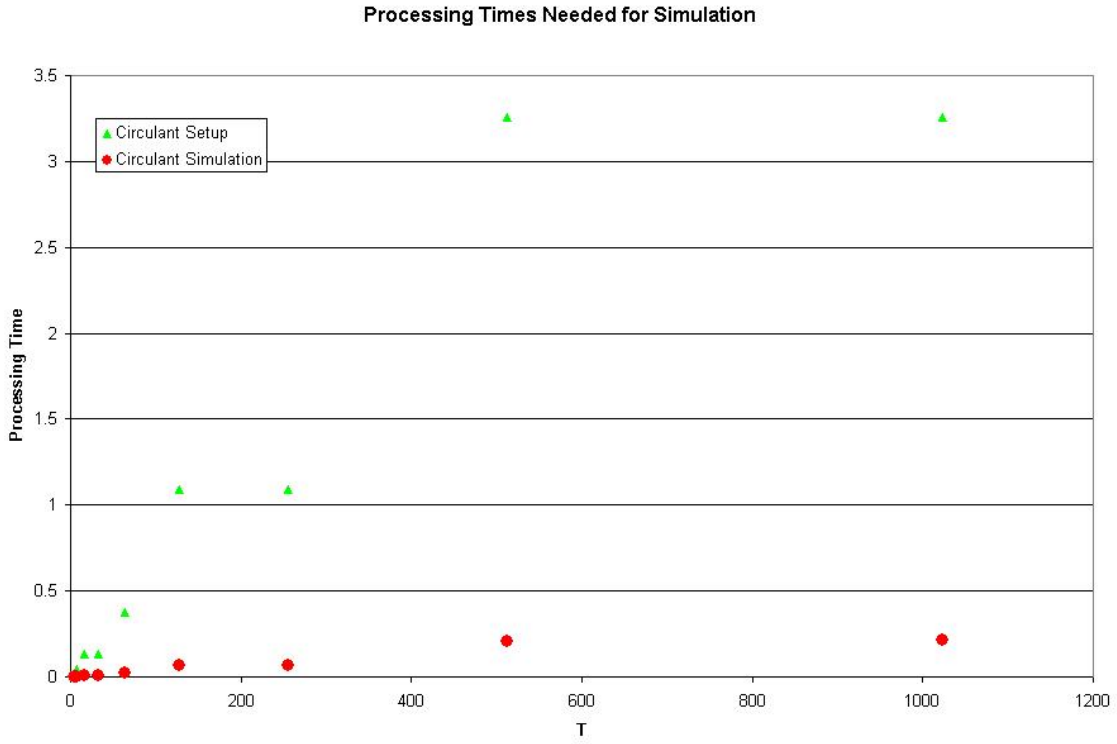

Figure 13: Processing time needed for the circulant embedding method to set up for simulation and simulate from a $FIVAR(0, \vec{d})$ with $d = (0.1, 0.4)$ and $\Sigma = (1, 0.5, 0.5, 2)$. Estimates for the setup time are based on 100 repetitions; estimates for the simulation times are based on 1000 repetitions.

T	Sowell Setup	Sowell Simulation	Circulant Setup	Circulant Simulation	M
4	0.005	0.010	0.230	0.006	27
8	0.013	0.023	0.153	0.006	27
16	0.052	0.053	0.414	0.016	81
32	0.168	0.107	0.367	0.013	81
64	0.470	0.285	0.762	0.040	243
128	1.599	0.713	2.125	0.109	729
256	7.424	2.425	2.529	0.107	729
512	24.251	7.763	6.096	0.310	2187

Table 12: Processing time needed to set up for simulation and simulate from a $VARFI(1, \vec{d})$ with $d = (0.1, 0.4)$, $\Sigma = (1, 0.5, 0.5, 2)$, and $A_1 = (0.6, -0.1, 0.2, 0.8)$. Estimates for the setup time are based on 100 repetitions; estimates for the simulation times are based on 1000 repetitions.

9 Maximum Likelihood Estimation and Monte Carlo

We now combine all of the computational methods we have discussed so far to run Monte Carlo experiments using the various estimation methods. We first discuss how we parameterize our models to ensure stationarity and invertibility. Second, we use Monte Carlo experiments to describe the effect of approximating the determinant using the methods discusses in section 7. Finally, we compare maximum likelihood estimation methods to the Whittle estimator for a variety of sample sizes.

9.1 Useful parameterizations for maximum likelihood estimation

To ensure that our parameter estimates are associated with a stationary and invertible model, we must ensure that $|d| < 0.5$, that Σ is positive definite, and that $A(L)$ has all of its roots outside the unit circle. The constraints on d can be implemented directly with box constraints. To ensure that Σ is positive definite, we follow the standard practice of constraining the diagonal element of its Cholesky decomposition to be positive. In the case where $A(L) = I - A_1 L$, $A(L)$ has all of its roots outside the unit circle if and only if all of the singular values of A_1 are less than one. In order for the covariance computation methods described in section 5 to work, we must bound the singular values away from one; if they approach one too closely, M in Algorithm 4 or 5 will tend towards infinity. In order to constrain the singular values of A_1, we use a modified version of the parameterization of Ansley and Kohn (1986), in which we ensure that the singular values of A_1 never exceed a given $\sigma < 1$ (0.99 in our algorithm). This parameterization results in a matrix, P, which is unconstrained and which can be mapped one-to-one onto the space of matrices with singular values less than σ. The algorithms to reparameterize A_1 and to return it to its original form are given below:

Algorithm 11 Conversion of a matrix, A_1, to the Ansley-Kohn parameterization, with maximum singular value, σ.

1. *Compute* $\tilde{A} = \frac{1}{\sigma} A_1$.
2. *Set* B *equal to the Cholesky decomposition of* $I_K - \tilde{A}\tilde{A}^T$*, where* I_K *is the identity matrix of size* K.
3. *Return* $(B^{-1})^T P_1$.

Algorithm 12 Conversion of a matrix, P, from the Ansley-Kohn parameterization with maximum singular value σ to its original form.

1. *Set* B *equal to the Cholesky decomposition of* $I_K + PP^T$*, where* I_K *is the identity matrix of size* K.

2. Set $\tilde{A} = (B^{-1})^T P$.

3. Return σP.

In Ansley and Kohn's original paper, they set $\sigma = 1$; Algorithms 11 and 12 reduce to their algorithm in that case. Given these parameterizations, we may implement maximum likelihood using simple box constraints.

9.2 The effects of the determinant approximation

We begin by studying the effects of the determinant approximation on the computed parameter estimates. In this section, we will compare three estimation methods: exact maximum likelihood using Sowell's algorithm, maximum likelihood in which the determinant is approximated in the most naive way by $|\Sigma|^T$, and maximum likelihood using the regression approximation to the determinant presented in Algorithm 8. To compare the three estimation methods, we simulate datasets of length $T = 100$ and 200 from FIVAR and VARFI processes with parameters $d = (0.1, 0.4)$, $\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix}$, and $A_1 = \begin{pmatrix} 0.6 & -0.1 \\ 0.2 & 0.8 \end{pmatrix}$. Because of the processing time required to compute the maximum likelihood estimates using Sowell's algorithm, all of our results are based on 100 simulated datasets.

In Tables 13, 14, 15, 16, and 17, we report the mean and standard deviation of the difference between the estimated parameter values using each approximation method and the estimated values using exact maximum likelihood. If our approximations were exact, then all of the means and standard deviations would be 0. For the regression approximation for FIVAR models, the mean difference never exceeds 0.003 in absolute value, and the standard deviation of the difference exceeds 0.01 only once. In contrast, the means and standard deviations of the differences between the parameter estimates using the naive approximation and the parameter estimates from exact maximum likelihood are quite large, especially for the estimates of the elements of Σ. For a more graphical illustration, in Figure 14, we show boxplots of the differences in the estimates of d for a FIVAR process with $T = 100$. The boxplots confirm that the regression approximation estimates deviate slightly from the estimates from exact maximum likelihood, while the naive approximation estimates often differ dramatically from the exact maximum likelihood estimates. For VARFI models, our regression approximation again does well, though some of the standard deviations of the differences are higher for the estimates of the elements of Σ. As before, the naive approximation is a much less successful approximation, though its problems in estimating Σ are less marked than for FIVAR models. These results provide further evidence that our regression approximation to the determinant works well and that the traditional approximation of $|\Omega|$ by $|\Sigma|^T$ is not a close enough approximation.

We also propose running future Monte Carlo experiments with alternative parameters configurations and models with $K > 2$.

Parameter	Regression Approximation	Naive Approximation
A_{11}	-0.0018 (0.0042)	-0.1221 (0.3771)
A_{21}	-0.0001 (0.0021)	-0.0869 (0.4036)
A_{12}	0.0022 (0.0068)	0.0674 (0.3987)
A_{22}	0.0010 (0.0057)	-0.2130 (0.4312)
Σ_{11}	0.0002 (0.0015)	656.6 (2189.6)
Σ_{12}	-0.0002 (0.0078)	-334.9 (37447.6)
Σ_{22}	0.0007 (0.0078)	531204.2 (1885517)
d_1	0.0016 (0.0107)	-0.1328 (0.2876)
d_2	-0.0008 (0.0069)	0.0797 (0.0560)
Log likelihood	-0.0213 (0.1114)	38.65 (60.47)

Table 13: Mean and standard deviation of the difference between the parameter estimate using the determinant approximation and the parameter estimate from exact maximum likelihood for a FIVAR model with $T = 100$. Standard deviations are given in parentheses. Estimates based on 100 repetitions.

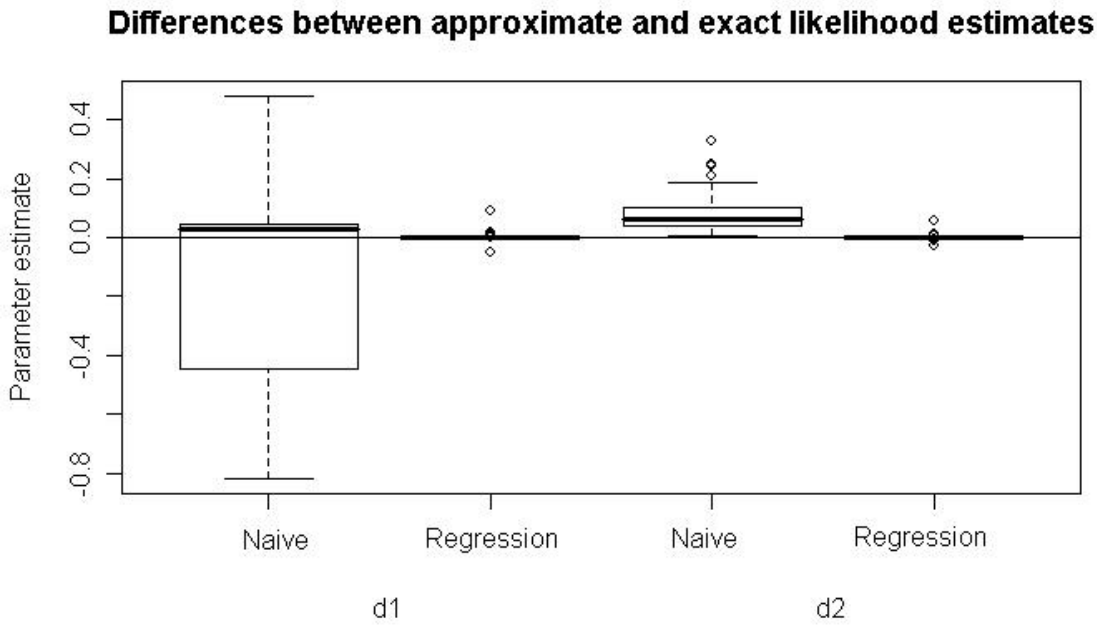

Figure 14: Boxplots of the differences between Sowell's exact maximum likelihood estimates and the two approximations in the estimates for d.

Parameter	Regression Approximation	Naive Approximation
A_{11}	-0.0016 (0.0029)	-0.1268 (0.3713)
A_{21}	0.0000 (0.0002)	-0.1202 (0.3367)
A_{12}	0.0017 (0.0020)	0.0290 (0.3319)
A_{22}	0.0012 (0.0026)	-0.1676 (0.3951)
Σ_{11}	0.0001 (0.0002)	294.635 (1306.914)
Σ_{12}	-0.0001 (0.0003)	8157.117 (129832.9)
Σ_{22}	0.0000 (0.0004)	3884703 (24760393)
d_1	0.0001 (0.0024)	-0.1213 (0.2880)
d_2	-0.0010 (0.0022)	0.0596 (0.0436)

Table 14: Mean and standard deviation of the difference between the parameter estimate using the determinant approximation and the parameter estimate from exact maximum likelihood for a FIVAR model with $T = 200$. Standard deviations are given in parentheses. Estimates based on 100 repetitions.

Parameter	Regression Approximation	Naive Approximation
A_{11}	-0.0017 (0.0122)	-0.1924 (0.1458)
A_{21}	-0.0000 (0.0055)	0.0858 (0.1220)
A_{12}	0.0001 (0.0033)	0.0934 (0.1280)
A_{22}	0.0003 (0.0038)	-0.3252 (0.1331)
Σ_{11}	0.0018 (0.0238)	-1.4827 (0.5943)
Σ_{12}	-0.00058 (0.0266)	0.8976 (0.4249)
Σ_{22}	0.0012 (0.0158)	99.05134 (1000.237)
d_1	-0.0014 (0.0257)	-0.0571 (0.1656)
d_2	-0.0004 (0.0081)	-0.0297 (0.1449)

Table 15: Mean and standard deviation of the difference between the parameter estimate using the determinant approximation and the parameter estimate from exact maximum likelihood for a VARFI model with $T = 100$. Standard deviations are given in parentheses. Estimates based on 100 repetitions.

Parameter	Regression Approximation	Naive Approximation
A_{11}	0.0004 (0.0078)	-0.1904 (0.1349)
A_{21}	-0.0003 (0.0023)	0.0912 (0.0804)
A_{12}	0.0002 (0.0025)	0.0910 (0.1099)
A_{22}	-0.0003 (0.0069)	-0.3219 (0.1300)
Σ_{11}	-0.0008 (0.0091)	1.5684 (30.0022)
Σ_{12}	0.0054 (0.1151)	-13.1160 (137.6588)
Σ_{22}	-0.0036 (0.0861)	63.4455 (629.2669)
d_1	-0.0005 (0.0090)	-0.0870 (0.1863)
d_2	0.0009 (0.0054)	-0.0768 (0.1283)

Table 16: Mean and standard deviation of the difference between the parameter estimate using the determinant approximation and the parameter estimate from exact maximum likelihood for a VARFI model with $T = 200$. Standard deviations are given in parentheses. Estimates based on 100 repetitions.

Parameter	Regression Approximation	Naive Approximation
A_{11}	-0.0016 (0.0221)	-0.1999 (0.0897)
A_{21}	0.0018 (0.0119)	0.0918 (0.0434)
A_{12}	-0.0006 (0.0120)	0.0817 (0.1209)
A_{22}	0.0003 (0.0125)	-0.3481 (0.0971)
Σ_{11}	0.0044 (0.0450)	-1.5867 (0.2511)
Σ_{12}	-0.0010 (0.0332)	0.9767 (0.2225)
Σ_{22}	0.0005 (0.0191)	4.2831 (52.8383)
d_1	0.0013 (0.0197)	-0.1048 (0.1238)
d_2	-0.0000 (0.0077)	-0.0905 (0.1100)

Table 17: Mean and standard deviation of the difference between the parameter estimate using the determinant approximation and the parameter estimate from exact maximum likelihood for a VARFI model with $T = 400$. Standard deviations are given in parentheses. Estimates based on 100 repetitions.

Model	Sowell Time	Regression Approximation Time	Naive Approximation Time
VARFI, $T = 100$	599.386	204.838	43.614
VARFI, $T = 200$	1977.928	390.172	77.807
VARFI, $T = 400$	8694.996	694.9787	131.9069

Table 18: Average processing time needed to compute the maximum likelihood estimators for each algorithm, for a variety of models.

T	MLE With Regression Approximation	Whittle
50	(0.156, 0.106)	(0.318, 0.152)
100	(0.150, 0.076)	(0.235, 0.135)
200	(0.149, 0.086)	(0.234, 0.135)

Table 19: Root mean squared errors of d estimates from a FIVAR model, based on 500 replications.

T	MLE With Regression Approximation	Whittle
50	(0.213, 0.522, 0.522, 0.537)	(0.840, 0.423, 0.423, 1.494)
100	(0.158, 0.507, 0.507, 0.459)	(0.843, 0.423, 0.423, 1.584)
200	(0.158, 0.508, 0.508, 0.459)	(0.840, 0.422, 0.422, 1.580)

Table 20: Root mean squared errors of Σ estimates from a FIVAR model, based on 500 replications.

9.3 Comparing maximum likelihood estimation to the Whittle estimator

We now run a larger Monte Carlo in which we compare the performance of our maximum likelihood estimates with the determinant approximation to the performance of the Whittle estimator. We will test these methods on both FIVAR and VARFI processes with a variety of sample sizes and parameter configurations. Here, we report preliminary results from simulations with $T = 50, 100$, and 200, $K = 2$, and parameters $d = (0.1, 0.4)$, $\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix}$, and $A_1 = \begin{pmatrix} 0.6 & -0.1 \\ 0.2 & 0.8 \end{pmatrix}$. All of these estimates are based on 500 simulated datasets.

In Tables 19, 20, and 21, we report the root mean squared error of each parameter estimate for each estimation method. These results show that maximum likelihood using the regression approximation performs the best in estimating both d and Σ, but the Whittle estimator does better in estimating the element of A_1, particularly the off-diagonal elements. Furthermore, we see from these results that the root mean squared error seems to be decreasing slowly as the sample size increases.

We now repeat the experiment with VARFI models. The results are given in Tables 23, 24, and 25. As in the FIVAR models, the Whittle estimator has the lower root mean squared error for some parameter estimates while our maximum likelihood estimator has lower root mean

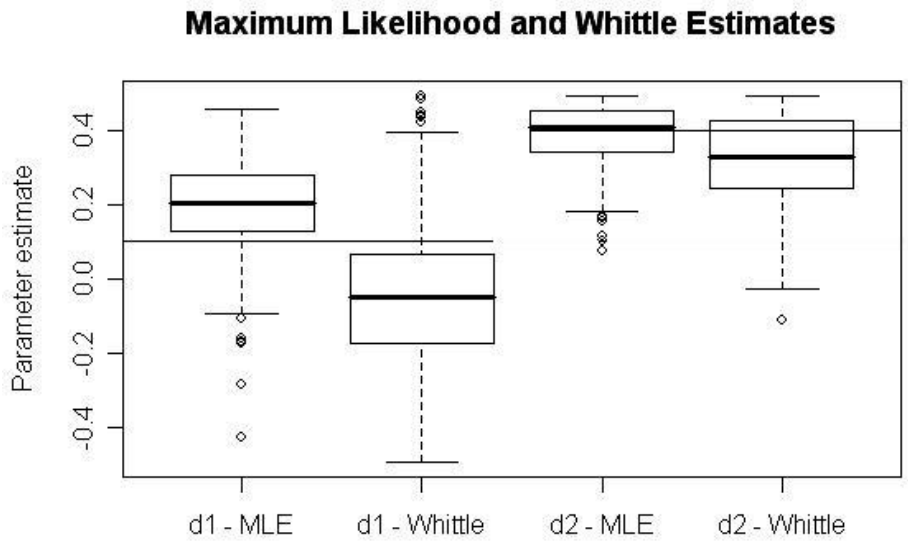

Figure 15: Boxplot of the estimated values of d, using maximum likelihood with the regression approximation and the Whittle estimator. The true values are 0.1 for d_1 and 0.4 for d_2.

T	MLE With Regression Approximation	Whittle
50	(0.214, 0.159, 0.717, 0.127)	(0.160, 0.069, 0.199, 0.098)
100	(0.186, 0.146, 0.713, 0.093)	(0.158, 0.055, 0.137, 0.096)
200	(0.185, 0.145, 0.713, 0.092)	(0.158, 0.053, 0.138, 0.097)

Table 21: Root mean squared errors of A_1 estimates from a FIVAR model, based on 500 replications.

T	MLE with Regression Approximation	Whittle
50	137.1381	33.971
100	209.4684	73.24466
200	416.3902	163.9263

Table 22: Average processing time needed for estimation of a VARFI model over 500 repetitions.

T	MLE With Regression Approximation	Whittle
50	(0.217, 0.228)	(0.197, 0.167)
100	(0.210, 0.096)	(0.190, 0.132)
200	(0.194, 0.059)	(0.211, 0.106)

Table 23: Root mean squared errors of d estimates from a VARFI model, based on 500 replications.

squared errors for others. For a number of parameters, such as the elements of d and A_1, the Whittle estimator performs better in the smallest sample, but the maximum likelihood estimator has a smaller RMSE for larger samples. Oddly, the Whittle estimator is dramatically better for one of the diagonal entries of Σ while the maximum likelihood estimator is dramatically better for the other. Examination of the mean estimates (not reported) shows that the Whittle estimates of the elements of Σ are biased toward zero, while the maximum likelihood estimates of the diagonal elements have an upward bias. Thus, the Whittle estimator fares better when for the smaller diagonal element, while the maximum likelihood estimator is more successful for the larger diagonal element.

Using a more extensive set of simulations, with a variety of parameter values for d and A_1, we find that the estimates of d using maximum likelihood with the regression approximation

T	MLE With Regression Approximation	Whittle
50	(1.467, 1.481, 1.481, 0.430)	(0.539, 0.779, 0.779, 1.505)
100	(1.521, 1.505, 1.505, 0.303)	(0.567, 0.715, 0.715, 1.588)
200	(1.520, 1.501, 1.501, 0.224)	(0.581, 0.690, 0.690, 1.626)

Table 24: Root mean squared errors of Σ estimates from a VARFI model, based on 500 replications.

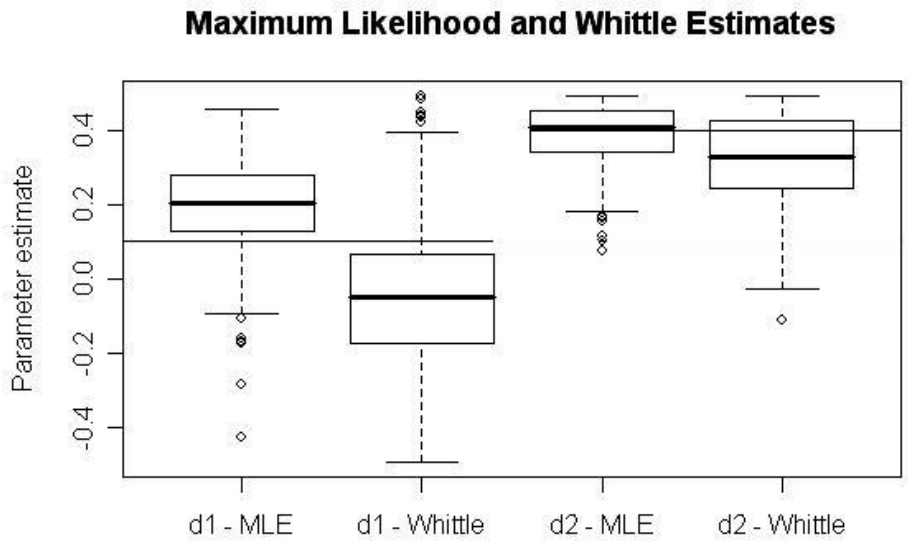

Figure 16: Boxplot of the estimated values of d, using maximum likelihood with the regression approximation and the Whittle estimator. The true values are 0.1 for d_1 and 0.4 for d_2.

T	MLE With Regression Approximation	Whittle
50	(0.162, 0.188, 0.250, 0.113)	(0.149, 0.134, 0.256, 0.105)
100	(0.134, 0.092, 0.221, 0.086)	(0.143, 0.093, 0.225, 0.097)
200	(0.099, 0.070, 0.211, 0.063)	(0.127, 0.069, 0.205, 0.080)

Table 25: Root mean squared errors of A_1 estimates from a VARFI model, based on 500 replications.

generally have smaller root mean squared errors than those from the Whittle estimator. As before, we found that the estimates of Σ from the Whittle estimator were biased toward 0, with estimates of the diagonal elements of Σ equal to 18% of their true values on average. In contrast, the estimates using maximum likelihood with the regression approximation had bias under 0.1 in most cases and root mean squared errors under 0.2. Results for estimates of A_1 were mixed in terms of bias and RMSE. The Whittle estimator generally had lower bias and RMSE for the off-diagonal elements of A_1, while the two estimators were evenly matched on the diagonal elements. Overall, we find that maximum likelihood with the regression approximation performs better, though computing estimates from both estimators could be helpful in some applications.

10 Data Analysis

In this section, we apply FIVAR and VARFI models to three different datasets. First, we apply our models to the components of inflation. Second, we discuss an application to a macroeconomic model of unemployment and inflation. Finally, we discuss an application in meteorology.

10.1 Goods and Services Inflation

We now consider a model for inflation in the goods and services sectors. While inflation is often considered as a single number, it is actually composed of the price changes across all goods and services produced in the economy. The relationship of the inflation rates across different sectors can be helpful for predicting inflation and for understanding how price changes in one sector affect price changes in other parts of the economy. Peach et al. (2004) modeled inflation in the goods and services sectors, excluding food and energy, as cointegrated time series, without allowing for fractional differencing. In this section, we estimate FIVAR and VARFI models based on overall goods and services inflation, as measured by the Consumer Price Index, for the period February 1956 through January 2008. The data are available online from the Bureau of Labor Statistics. The data are show in Figure 17.

We first fit univariate $ARFIMA(1, d, 0)$ models to the two series using maximum likelihood. The estimates are given in Table 26. According to these estimates, both series are fractionally integrated. Goods inflation is estimated to have a differencing parameter of 0.2265, while the differencing parameter of services inflation is estimated to be 0.4837, making it almost non-stationary. We use these estimates as starting values for our estimation of FIVAR models, setting all initial off-diagonal elements of A_1 and Σ to 0.

We estimate a FIVAR model based on the demeaned data, using both maximum likelihood and the Whittle estimator. Results are reported in Tables 27. As we found in the Monte Carlo simulations, the estimates of the covariance matrix based on Whittle estimator are much closer to 0 than the estimates from maximum likelihood are. Both estimators find that services

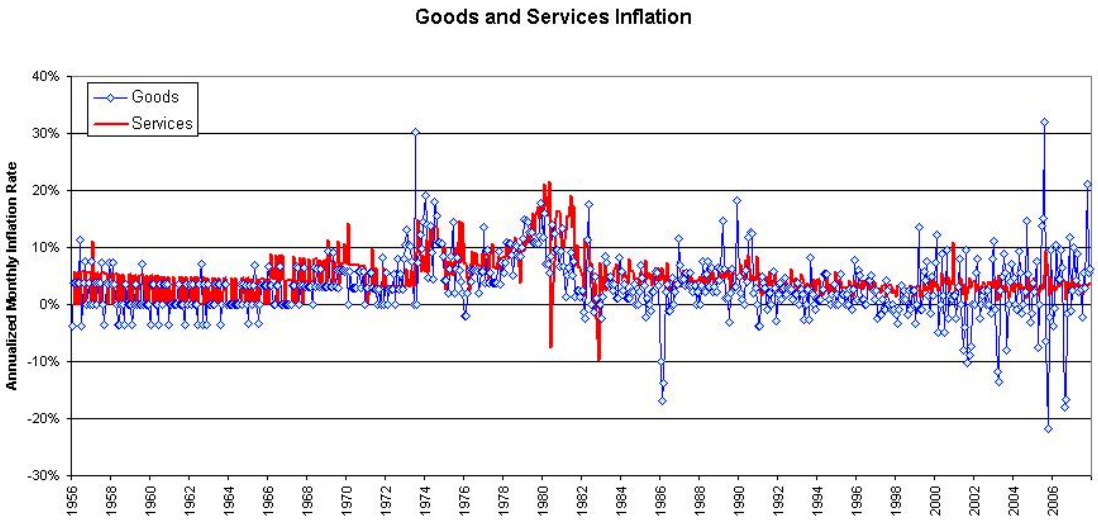

Figure 17: Annualized goods and services inflation rates, February 1956-January 2008.

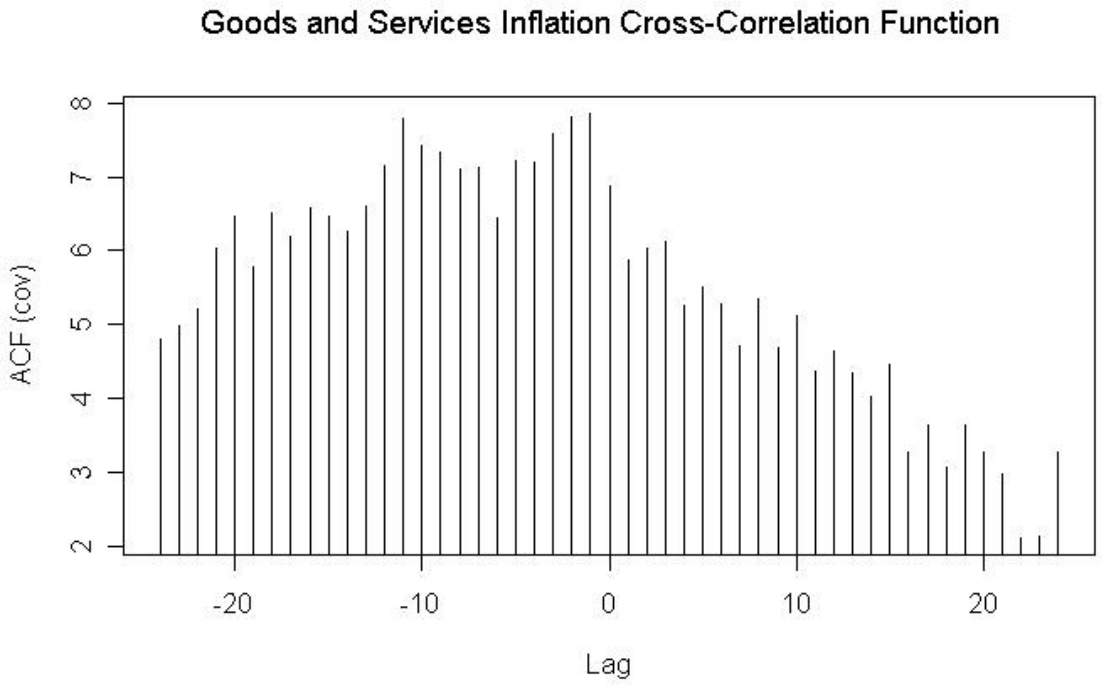

Figure 18: Empirical cross-correlation function of goods and services inflation rates.

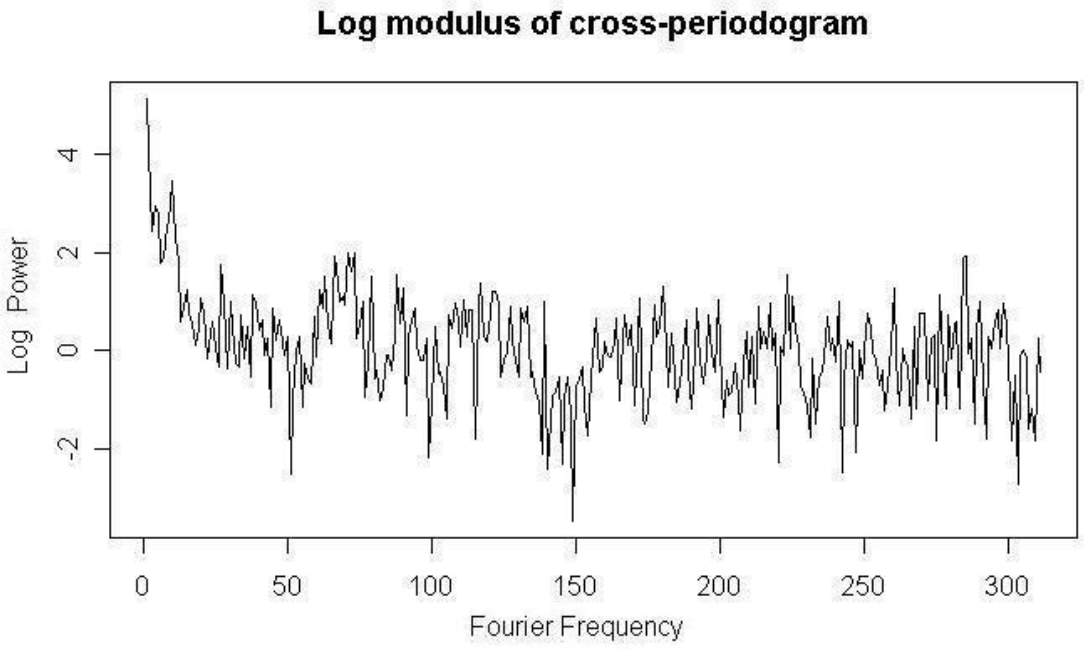

Figure 19: Log modulus of the cross-periodogram of goods and services inflation rates.

	Goods	Services
A_1	0.1053 (0.0032)	-0.3165 (0.0013)
Σ	21.2703 (1.4529)	7.0842 (0.1523)
d	0.2265 (0.0016)	0.4837 (0.0000)
Log Likelihood	-1266.140	-924.7657

Table 26: Maximum likelihood estimates for goods and services inflation, as univariate series. Approximate asymptotic standard errors in parentheses.

	Maximum Likelihood with Regression Approximation	Exact Maximum Likelihood	Whittle
A_{11}	0.1024 (0.0034)	0.1023	0.186
A_{21}	-0.0204 (0.0006)	-0.0204	0.099
A_{12}	0.1510 (0.0104)	0.1509	-0.00
A_{22}	-0.3101 (0.0022)	-0.3103	-0.33
Σ_{11}	21.0912 (0.0003)	21.0909	3.386
Σ_{12}	0.6260 (0.3120)	0.6257	0.135
Σ_{22}	7.0812 (0.0763)	7.0804	1.094
d_1	0.2281 (0.0017)	0.2282	0.141
d_2	0.4770 (0.0006)	0.4771	0.487
Log likelihood	-2187.109	-21887.095	-4

Table 27: FIVAR estimates for goods and services inflation data. The Whittle log likelihood is the regression approximation to the likelihood at those parameter values. Approximate asymptotic standard errors are given in parentheses for all estimators except for Sowell's exact estimator.

inflation has a larger differencing parameter than goods inflation, with the services differencing parameter quite close to 0.5. In Figures 20 and 21, we plot the logged modulus of the cross-periodogram and the logged modulus of the implied cross-spectral densities based on the two estimators. The spectral density based on the maximum likelihood estimates seems to fit the cross-periodogram more closely.

We now fit a VARFI model to this data, again using both estimators. The estimated covariance matrices are similar, but the maximum likelihood estimate of the smaller differencing parameter has dropped from 0.22 to 0. This does not mean that goods inflation is now estimated to have short memory; on the contrary, under the VARFI model the two series are estimated to have the same memory parameter. In contrast, the Whittle estimates of d are almost unchanged. As before, we compare the cross-periodogram to the implied cross-spectral densities from the two estimates in Figures 22 and 23.

Since the $FIVAR(1, \vec{d})$ and $VARFI(1, \vec{d})$ have the same number of parameters, we may compare their log likelihoods to choose between them. In this case, the VARFI model has a higher log likelihood. We may write the VARFI model in a form analogous to a VAR, where the errors driving the VAR are no longer white noise:

$$\begin{aligned} goods_t &= 0.3027 goods_{t-1} + 0.4245 services_{t-1} + u_{1t} \\ services_t &= -0.0237 goods_{t-1} - 0.3085 services_{t-1} + u_{2t} \\ \begin{pmatrix} u_{1t} \\ (1-L)^{0.4835} u_{2t} \end{pmatrix} &\sim Normal\left(0, \begin{pmatrix} 20.2342 & 0.4605 \\ 0.4605 & 7.0783 \end{pmatrix}\right) \end{aligned}$$

Though the goods equation is driven by shocks that have short memory, long memory in goods inflation is induced by the lagged services inflation. In the services equation, lagged services

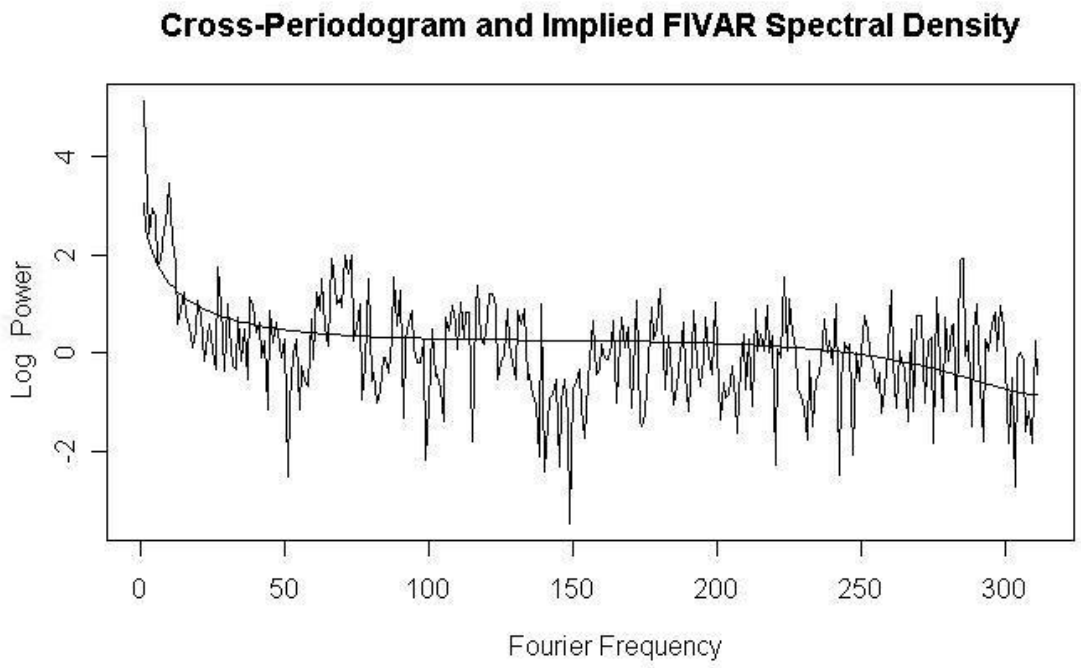

Figure 20: Log modulus of the cross-periodogram of goods and services inflation rates and of the impled cross-spectral density of the estimated FIVAR model.

	Maximum Likelihood with Regression Approximation	Exact Maximum Likelihood	Whittle
A_{11}	0.3027 (0.0014)	0.3027	0.161
A_{21}	-0.0237 (0.0005)	-0.0237	0.054
A_{12}	0.4245 (0.0027)	0.4245	0.088
A_{22}	-0.3085 (0.0018)	-0.3085	-0.32
Σ_{11}	20.2342 (0.8669)	20.2342	3.373
Σ_{12}	0.4605 (0.2275)	0.4605	0.131
Σ_{22}	7.0783 (0.1619)	7.0783	1.117
d_1	0.0000 (0.0000)	0.0000	0.151
d_2	0.4835 (0.0004)	0.4835	0.489
Log likelihood	-2174.263	-2174.249	-4

Table 28: VARFI estimates for goods and services inflation data. The Whittle log likelihood is the regression approximation to the likelihood at those parameter values. Approximate asymptotic standard errors are given in parentheses for all estimators except for Sowell's exact estimator.

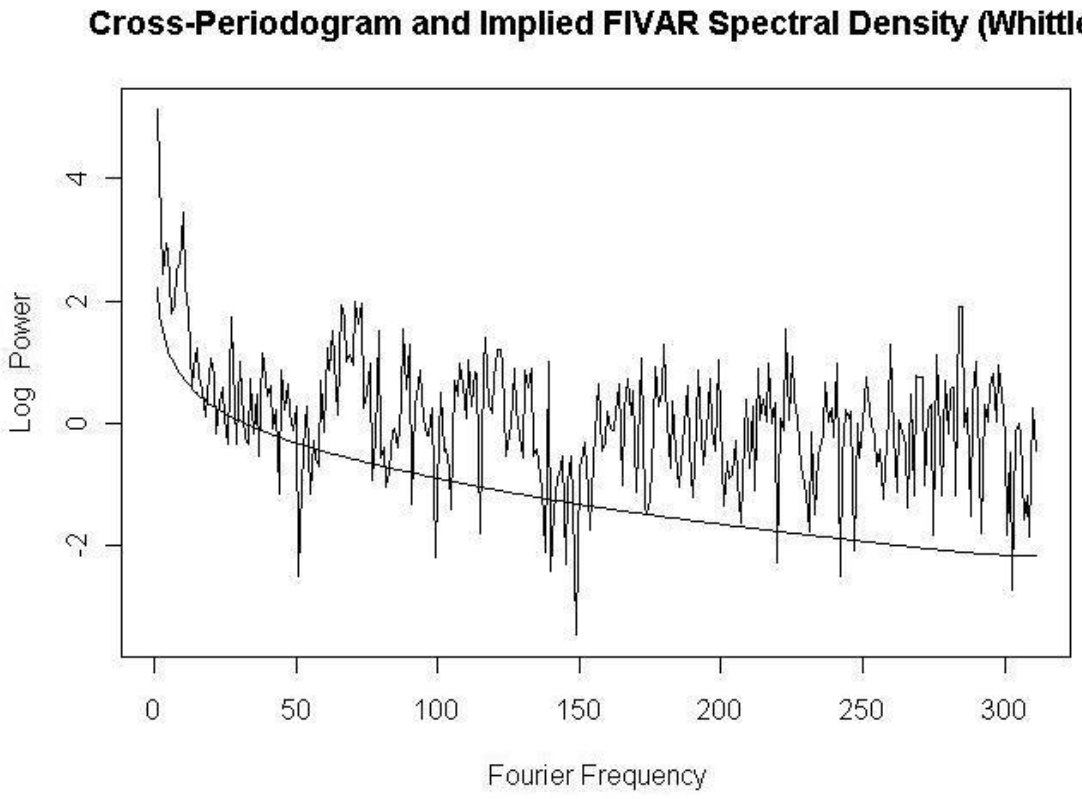

Figure 21: Log modulus of the cross-periodogram of goods and services inflation rates and of the impled cross-spectral density of the estimated FIVAR model, using the Whittle estimator.

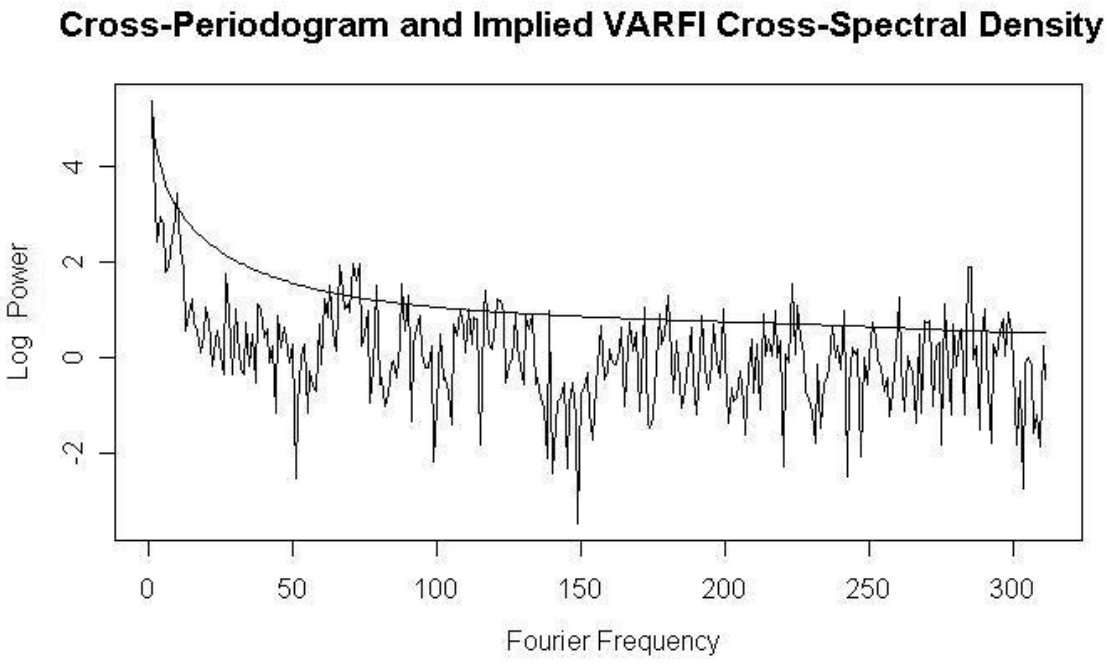

Figure 22: Log modulus of the cross-periodogram of goods and services inflation rates and of the impled cross-spectral density of the estimated VARFI model.

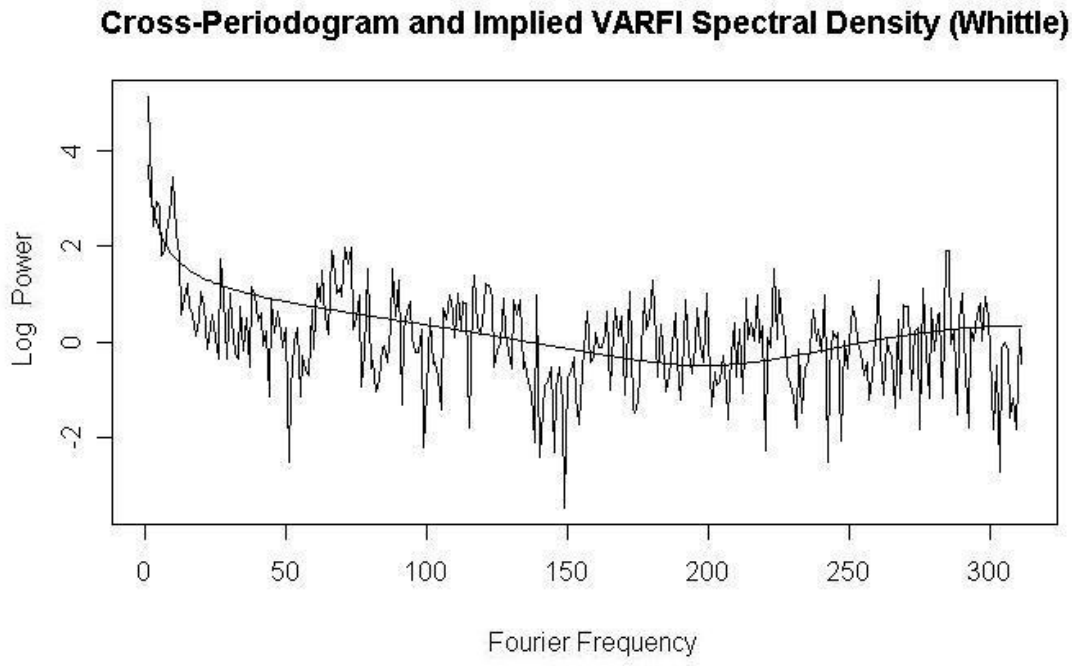

Figure 23: Log modulus of the cross-periodogram of goods and services inflation rates and of the impled cross-spectral density of the estimated VARFI model.

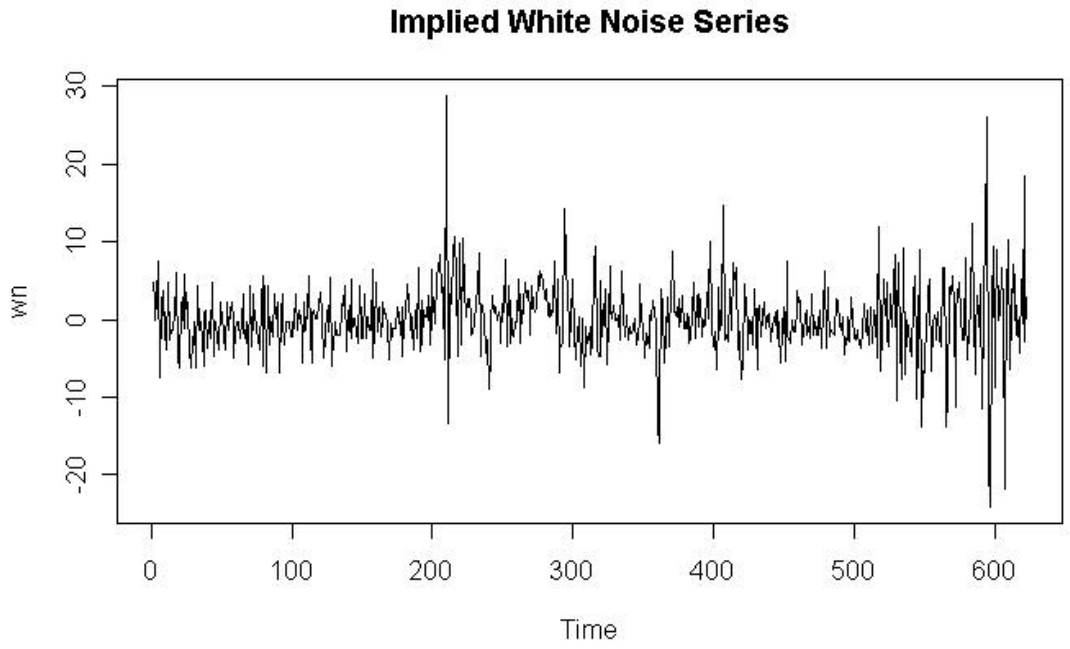

Figure 24: Time series of the linear combination of lagged goods and services inflation that the VARFI model implies is white noise.

inflation has a negative coefficient; however, services inflation is persistent because of the persistence in the shock process. While lagged services inflation has a significant influence on goods inflation, lagged goods inflation has little effect on services inflation.

Rewriting the first equation in the VARFI model, we find that $w_t = goods_t - 0.3027 goods_{t-1} - 0.4245 services_{t-1}$ is estimated to be white noise. To confirm this, we compute w_t over the sample period and plot it in Figure 24. This series appears to be approximately white noise, though there are some periods of increased volatility, particularly near the end of the sample period. The log periodogram, shown in Figure 25, confirms that all long memory has been removed by this linear combination.

For comparison, we also fit a short memory vector autoregressive model to the data. We consider two lag lengths. First, we use a $VAR(2)$, since that model has only two more parameters than a $FIVAR(1, \vec{d})$ or $VARFI(1, \vec{d})$ model does. Second, we use the AIC to choose a lag length, and a vector autoregressive model with 10 lags is chosen. We plot the log modulus of the cross-periodogram and the log modulus of the spectral density implied by the estimates of these two models in Figures 26 and 27. When only two lags are included, the fact that the spectral density is finite at 0 is quite evident; the model cannot match the peak in the periodogram at 0. When

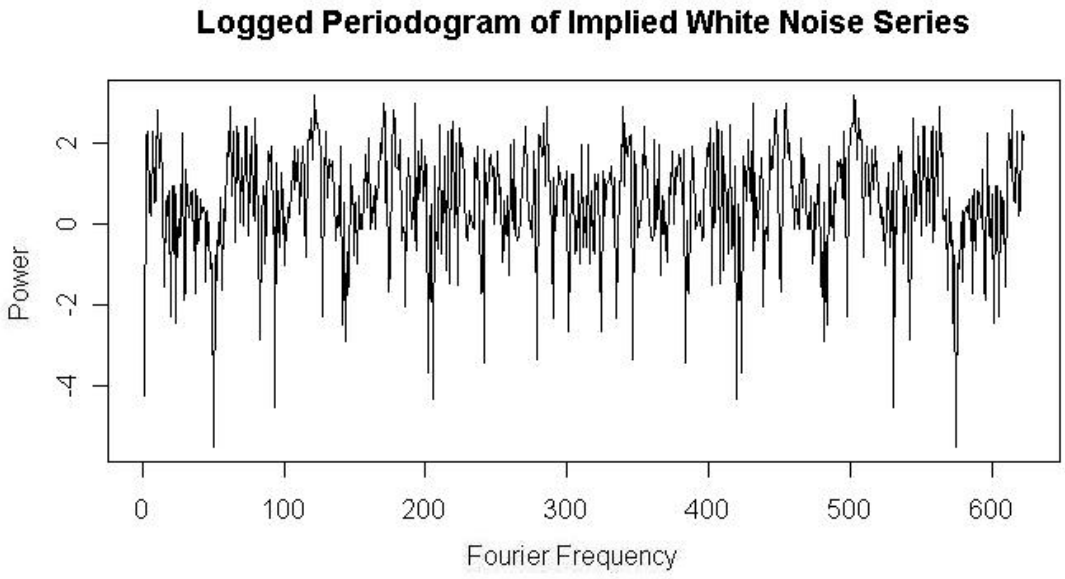

Figure 25: Log periodogram of the linear combination of lagged goods and services inflation that the VARFI model implies is white noise.

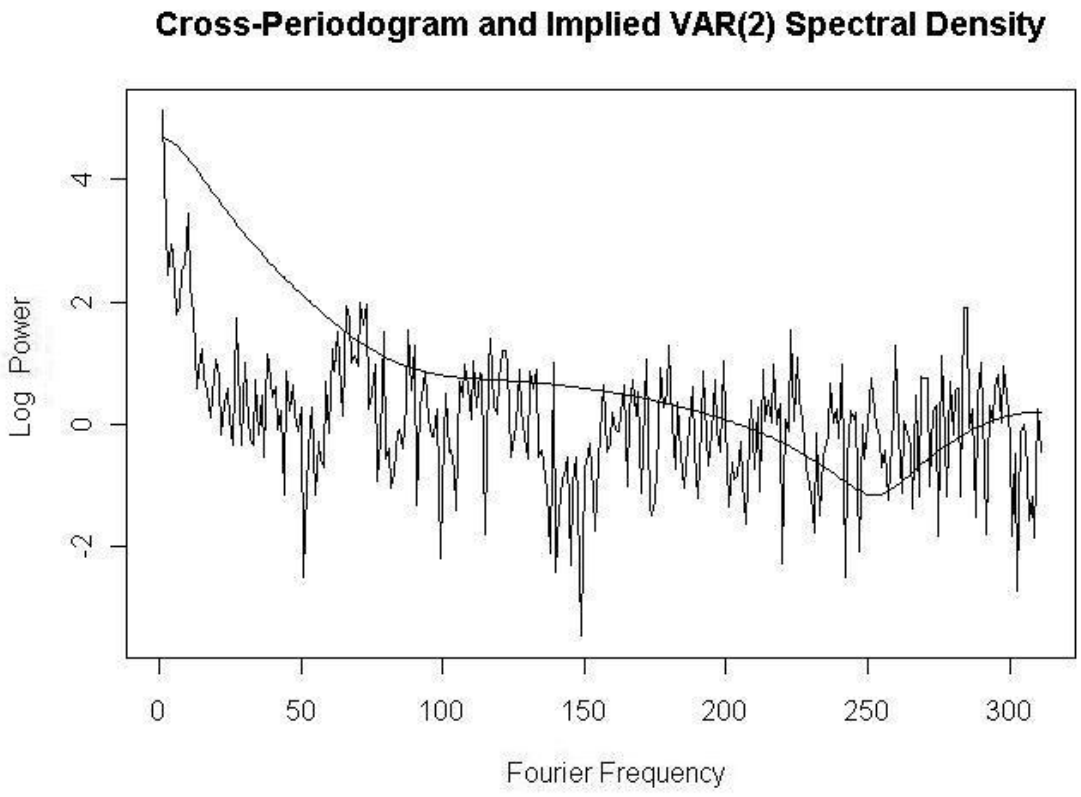

Figure 26: Log modulus of the cross-periodogram of goods and services inflation rates and of the impled cross-spectral density of the estimated VAR(2) model.

10 lags are included, the model fits the peak, but the spectral density is less smooth, suggesting overfitting.

As a final comparison among the models, we compute out-of-sample predictions for February through May 2008. For goods inflation, the $VAR(2)$ performed best; for services inflation, the VARFI model with the maximum likelihood estimates performs best. In both cases, the $VAR(10)$ was by far the worst performer. In Figure 28, we plot the forecasts and realization of services inflation. At the end of the sample, services inflation had been below its mean for 26 consecutive months. The long memory structure of the VARFI and FIVAR models could model this persistence, and predicted that inflation would move very slowly toward its mean. In contrast, the predictions based on the $VAR(2)$ returned to the mean at an exponential rate. This difference accounts for the improved performance of the long memory models for services inflation.

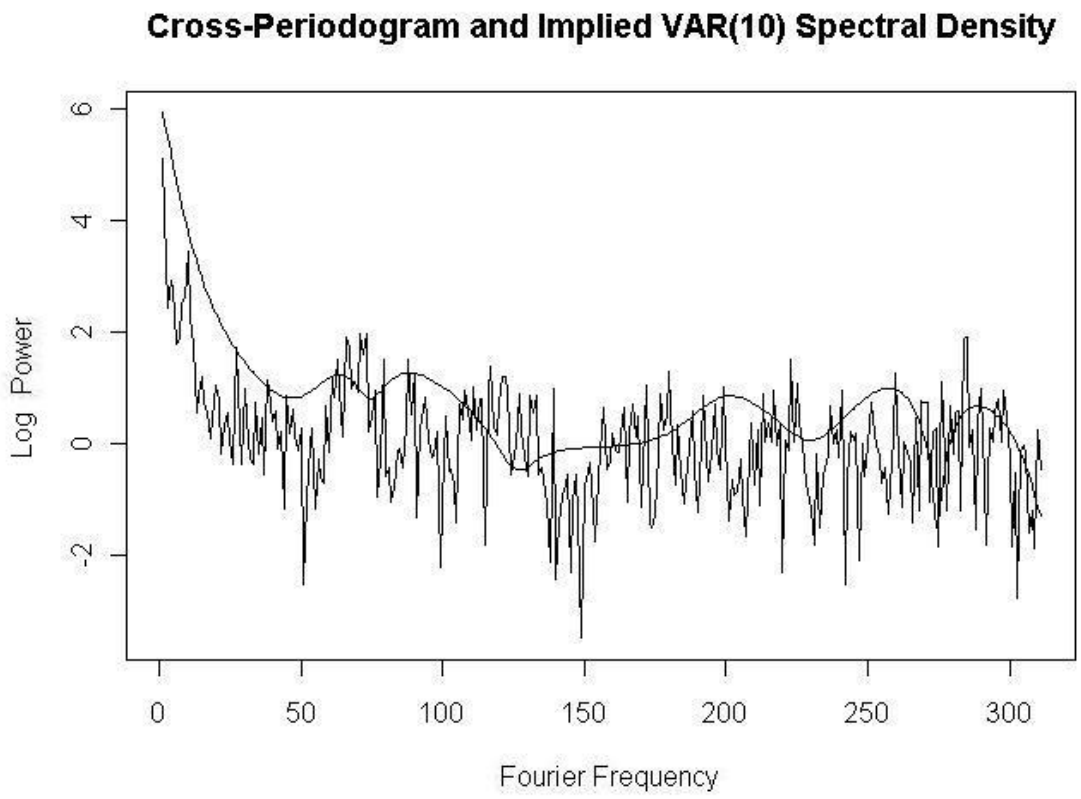

Figure 27: Log modulus of the cross-periodogram of goods and services inflation rates and of the impled cross-spectral density of the estimated VAR(10) model.

	Goods	Services
FIVAR-MLE	5.264	1.236
FIVAR-Whittle	5.189	1.256
VARFI-MLE	5.211	1.215
VARFI-Whittle	5.194	1.280
VAR(2)	5.008	1.327
VAR(10)	6.263	2.232

Table 29: Root mean squared errors for out-of-sample from February to May 2008.

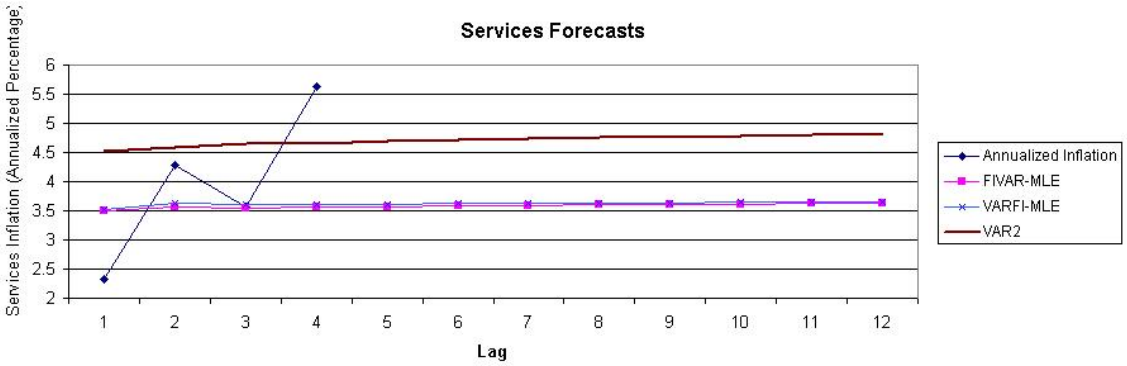

Figure 28: Realized out-of-sample services inflation and forecasts from the $VAR(2)$, VARFI and FIVAR models.

10.2 Phillips Curve Data

One of the most basic models in macroeconomics is the Phillips curve, which relates the unemployment rate to inflation. (See a macroeconomics textbook, such as Hall and Taylor (1997) for more background.) The simplest form of the Phillips curve states that an increase in the slack in the economy, as measured by the unemployment rate, leads to a decrease in inflation. Empirically, we see that inflation is generally persistent (see Figure 29); this is often explained in models by assuming that people have expectations about inflation, and that the effect of the unemployment rate on inflation is relative to the expectations. The simplest form of inflation expectations sets the expectation for tomorrow equal to today's inflation (for example, Wooldridge, 2000, example 11.5). Such a model implies a relationship between the level of the unemployment rate and the first difference of the inflation rate; if the unemployment rate were constant, this would imply a unit root in inflation. However, as we see Figure 29, the unemployment rate is also persistent, while inflation is persistent but is also likely to be mean-reverting; this suggests that a multivariate long memory model might be a better description of the data. We will not justify the use of a FIVAR or VARFI model using economic theory, but only as a useful description of the data. In this estimation, we use annual data on the unemployment rate and the inflation rate from 1948 to 1996.[1] The estimated cross-correlation function for this data is given in Figure 30. This figure shows that past inflation is strongly correlated with the future unemployment rate, which runs counter to the usual understanding of the Phillips curve, in which the slack in the economy, as measured by the unemployment rate, would affect future inflation.

We first fit FIVAR models to these data using maximum likelihood with the regression

[1]This dataset is available from the website of Jeffrey Wooldridge at `http://www.msu.edu/ ec/faculty/wooldridge/book2.htm`, as `Phillips.RAW`.

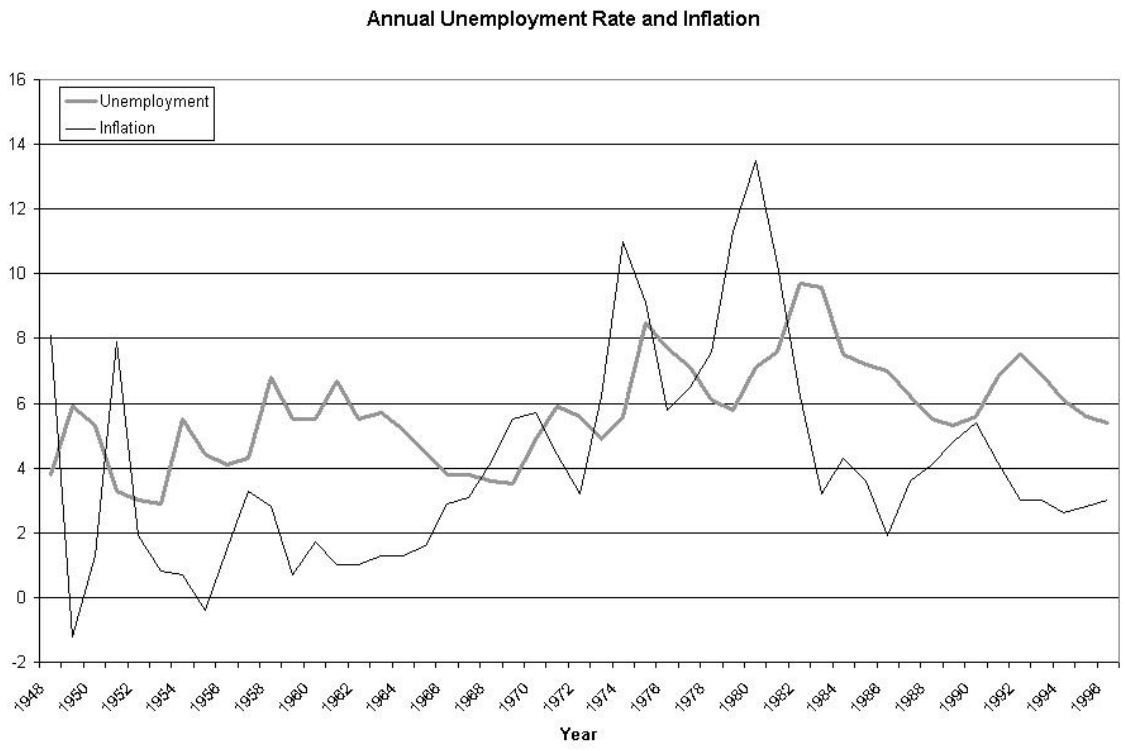

Figure 29: Annual unemployment rate and inflation rate used for estimating the Phillips curve.

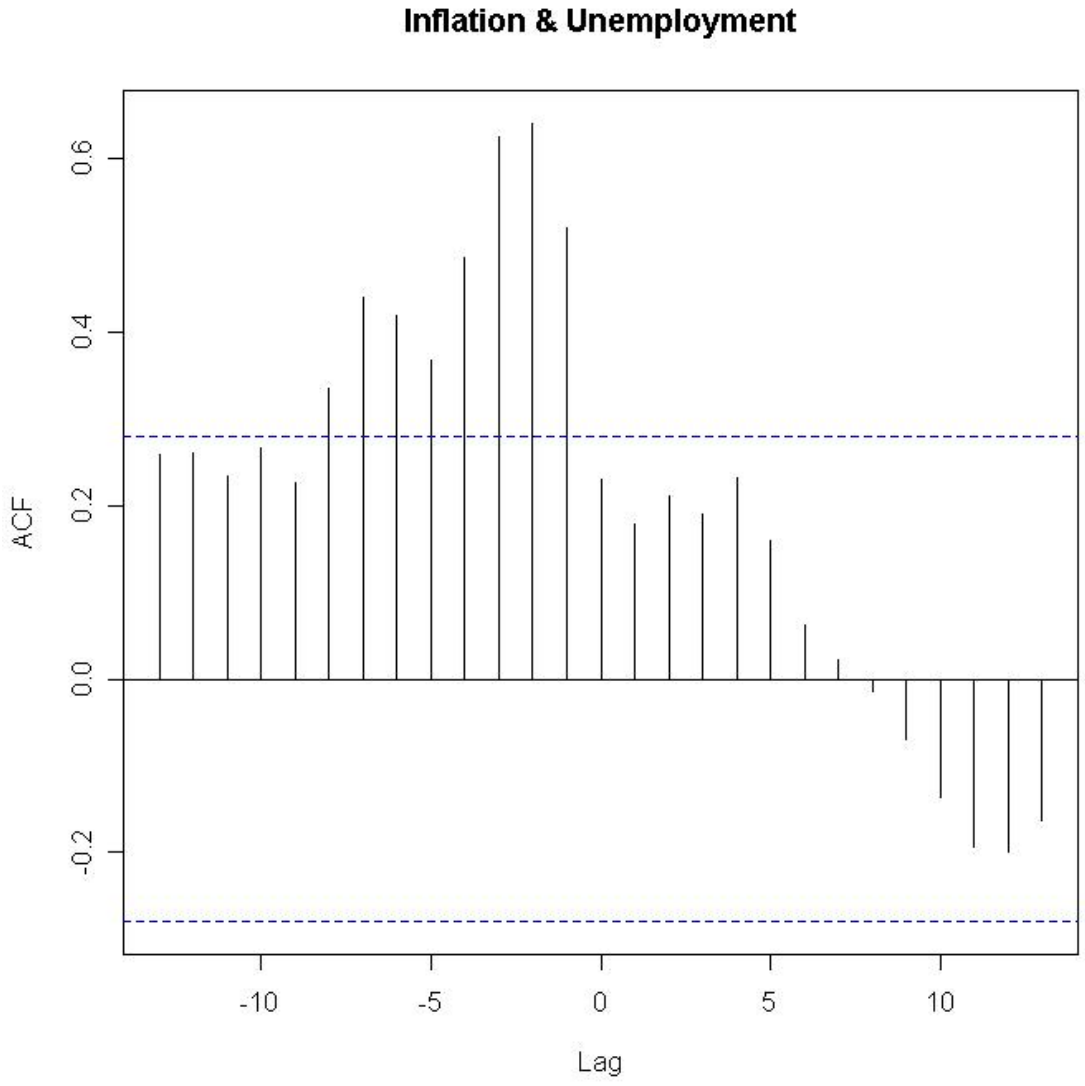

Figure 30: The empirical cross-correlation function of the unemployment rate and the inflation rate.

	Maximum Likelihood with Regression Approximation	Exact Maximum Likelihood	Whittle Approximation
A_1	(-0.1085, 0.0668, 0.9360, 0.3120)	(-0.1075, 0.0670, 0.9361, 0.3119)	(-0.1788, 0.2524, 0.3441, 0.4774)
Σ	(2.3052, -1.3910, -1.3910, 4.9402)	(2.3056, -1.3912, -1.3912, 4.9398)	(0.3000, -0.2941, -0.2941, 1.1282)
d	(0.3601, 0.3364)	(0.3595, 0.3365)	(0.4900, 0.0137)
Log likelihood	-105.3	-105.2991	-357.2535

Table 30: FIVAR estimates for Phillips curve data. The Whittle log likelihood is the regression approximation to the likelihood at those parameter values.

approximation, exact maximum likelihood using Sowell's method, and Whittle's approximation to the likelihood. The estimated parameter values are given in Table 30. Using our default initial values, the estimates from Sowell's method failed to converge; when we used the FIVAR estimates as the initial values, the estimates converged to the values that we report. While the exact maximum likelihood estimate and the estimate from the regression approximation match quite closely, as we would expect from section 9.2, the Whittle estimate is very different, and the Whittle estimate for d_1 is on the boundary of the parameter space. In the maximum likelihood estimates, the estimated differencing parameters are quite close. Since the FIVAR and VARFI models are identical when the differencing parameters are equal, this suggests that the VARFI model will have similar parameter estimates.

To check this hypothesis, we estimate a VARFI model with the same data. The estimates are reported in Table 31; the estimates for both maximum likelihood methods use the FIVAR estimates as initial values. In this case, the Whittle estimates of the parameters are somewhat closer to the maximum likelihood estimates, though the estimates of the elements of Σ are still much closer to 0 than the maximum likelihood estimates of Σ.

We have estimated two distinct models based on the same data. Comparing the maximum likelihood estimates from the two models, we see that the VARFI estimates of the differencing parameters differ by more than the FIVAR estimates do, but that the averages of the estimated differencing parameters are almost identical (0.344 for the VARFI model and 0.348 for the FIVAR model). The estimates of the innovation variances match closely, while the estimates of the autoregressive parameters are of the same signs and similar magnitudes. Because these two models have the same number of parameters, we may choose between them based on the log likelihoods. Using this criterion, we prefer the VARFI model to describe the relationship between the unemployment rate and inflation. In Figure 31, we plot the implied cross-covariances based on the VARFI model. The asymmetric pattern of slowly decaying cross-covariances is captured nicely by the VARFI model.

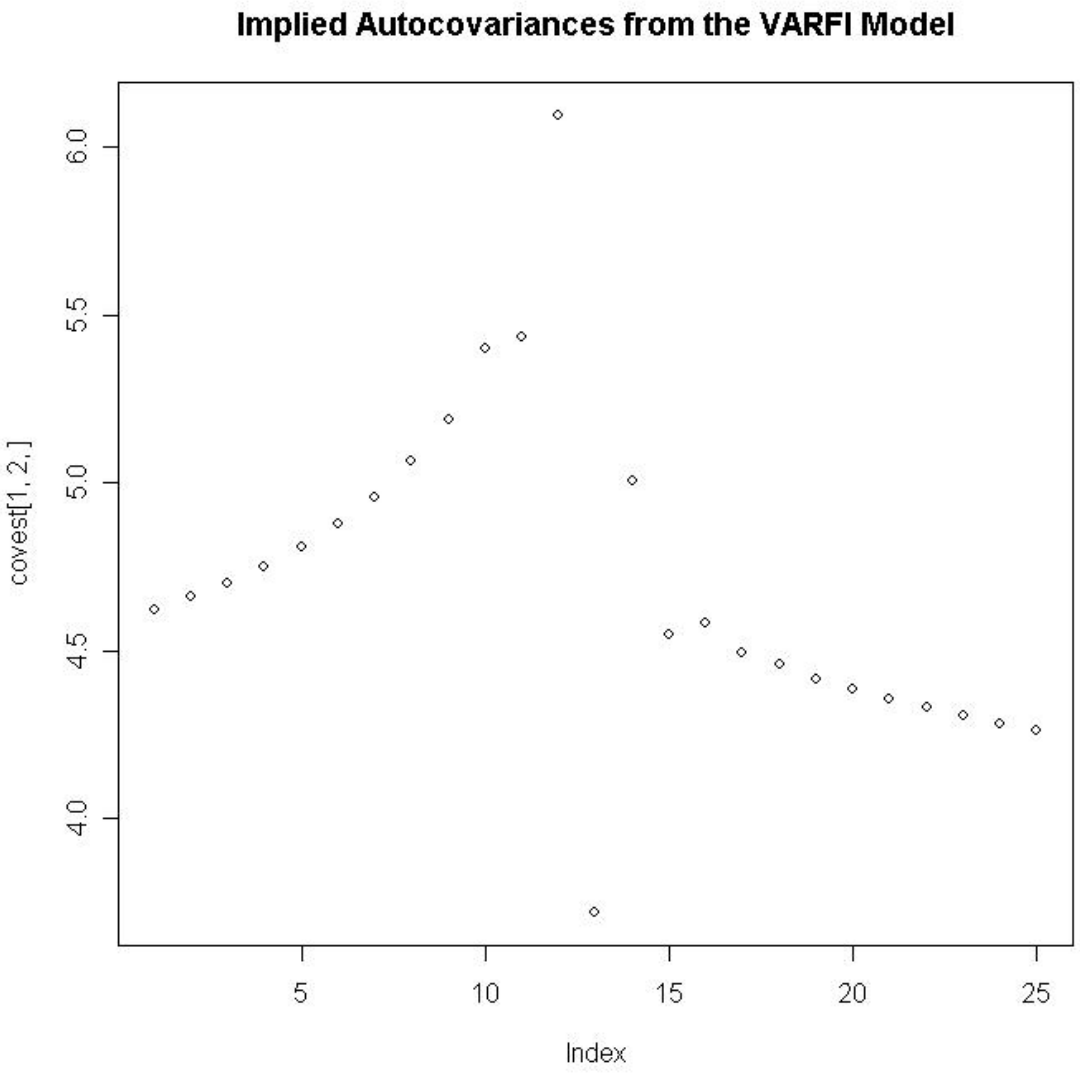

Figure 31: The cross-correlation function of the unemployment rate and the inflation rate implied by the VARFI model.

	Maximum Likelihood with Regression Approximation	Exact Maximum Likelihood	Whittle Approximation
A_1	(-0.2228, 0.0449, 0.9020, 0.3601)	(-0.2226, 0.0449, 0.9020, 0.3601)	(-0.0110, 0.2807, 0.2600, 0.1255)
Σ	(2.2248, -1.4736, -1.4736, 4.9647)	(2.2252, -1.4741, -1.4741, 4.9643)	(0.3195, -0.3231, -0.3231, 1.1791)
d	(0.4480, 0.2402)	(0.4480, 0.2411)	(0.4677, 0.3831)
Log likelihood	-104.0927	-104.0907	-315.5599

Table 31: VARFI estimates for Phillips curve data. The Whittle log likelihood is the regression approximation to the likelihood at those parameter values.

As we discussed in Section 2.2, a VARFI model is a vector autoregression driven by fractionally integrated white noise. That means that we may write this VARFI model as:

$$unemp(t) = 0.223unemp(t-1) - 0.045infl(t-1) + u_{1t} \quad (17)$$

$$infl(t) = -0.902unemp(t-1) - 0.360infl(t-1) + u_{2t} \quad (18)$$

where (u_{1t}, u_{2t}) are distributed as fractionally integrated white noise with covariance matrix $\begin{pmatrix} 2.225 & -1.473 \\ -1.473 & 4.965 \end{pmatrix}$ and differencing parameters $(0.448, 0.240)$. Equation 18 matches the traditional intuition about the Phillips curve: an increase in the unemployment rate is associated with a decrease in the inflation rate in the next period. We also find that an increase in inflation is associated with a decrease in the unemployment rate in the next period. In this model, though, the "shocks" are correlated across time, which leads to more persistence in both inflation and the unemployment rate, despite the negative coefficients on the AR(1) parameter in Equation 18. Thus, the VARFI model matches the basic economic theory of the Phillips curve but can also match the empirical persistence in the cross-covariances.

For comparison, we also fit a vector autoregressive model to this data. The Akaike Information Criterion suggests a lag length of 2 for this data. The parameter estimates for a vector autoregressive model of order 2 are given in Table 32. The estimated covariance matrix for the innovations in this model is $\begin{pmatrix} 0.7929 & -0.3482 \\ -0.3482 & 4.0797 \end{pmatrix}$. Notice that the estimated covariance matrix entries lie between the maximum likelihood estimates and the Whittle estimates of Σ. While we cannot compare the VAR coefficients to the VARFI coefficients in the same way, we can compare the implied autocovariance functions. The cross-covariance function implied by the VAR(2) is given in Figure 32. In contrast to the covariances from the VARFI model, the covariances from the VAR model start lower and decay to 0 quite quickly. We also note that the conditional log likelihood of the VAR(2) is -154.783. Since this VAR has been estimated conditional on the first two periods, we must add on the likelihood of the initial observations in order to make the likelihood comparable to the unconditional likelihood given in Table 31.

	Unemployment Rate	Inflation
Unemployment Rate - Lag 1	0.67779 (0.15544)	-0.5224 (0.3526)
Inflation - Lag 1	0.14147 (0.05766)	0.7737 (0.1308)
Unemployment Rate - Lag 2	-0.07806) (0.13205	0.5458 (0.2995)
Inflation - Lag 2	0.05758 (0.06735)	-0.0297 (0.1528)

Table 32: Parameter estimates from a VAR(2) model for the Phillips curve data. Standard errors are given in parentheses.

	Maximum Likelihood with Regression Approximation	Exact Maximum Likelihood	Whittle Approximation
A_1	(-0.0299, -0.0600, -0.0096, 0.1672, -0.3953, 0.1866, -0.2486, 0.1054, 0.1785)	(-0.0298, -0.0596, -0.0092, 0.1667, -0.3956, 0.1864, -0.2488, 0.1051, 0.1784)	(0.0257, 0.2750, -0.1787, -0.3346, 0.6549, 0.2596, -0.3561, 0.3588, 0.3096)
Σ	(9.8310, 5.6902, 6.7674, 5.6902, 10.0516, 5.5276, 6.7674, 5.5276, 9.6809)	(9.8317, 5.6902, 6.7678, 5.6902, 10.0506, 5.5285, 6.7678, 5.5285, 9.6820)	(1.3921, 0.7775, 0.8253, 0.7775, 1.6941, 0.7477, 0.8253, 0.7477, 1.2596)
d	(0.0004, 0.2464, 0.0982)	(0.0000, 0.2460, 0.0980)	(-0.1832, -0.4900, -0.2468)
Log likelihood	-380.3562	-380.3556	-1159.296

Table 33: FIVAR estimates for the precipitation data. All log likelihoods are the exact log likelihoods computed using Sowell's algorithm at the estimated parameter values.

Using the unconditional covariances of a VAR(2), we find that the likelihood of the first two observations is -22.5925. Summing the two parts of the log likelihood, we find that the VAR model has a log likelihood of -177.3755, which is lower than the likelihood of the VARFI model, despite including two more estimated parameters. Thus, the VARFI model is a better fit to these data than a vector autoregression is.

10.3 Great Lakes Precipitation

We now model data on precipitation in the Great Lakes. This data, from Hipel and McLeod[2], measures the annual precipitation, in inches, on Lakes Huron, Michigan, and Superior from 1900 to 1986. The autocorrelation functions of the three series, in Figure 33, suggest that the series for Lakes Huron and Superior have some long memory, while Lake Michigan's series has short memory or a differencing parameter very close to zero. Furthermore, the cross-correlation function of Lakes Huron and Superior, shown in Figure 33 seems to decay slowly. The two cross-correlation functions with Lake Michigan decay more quickly.

[2]These data are available online from `http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/` in the meteorology section.

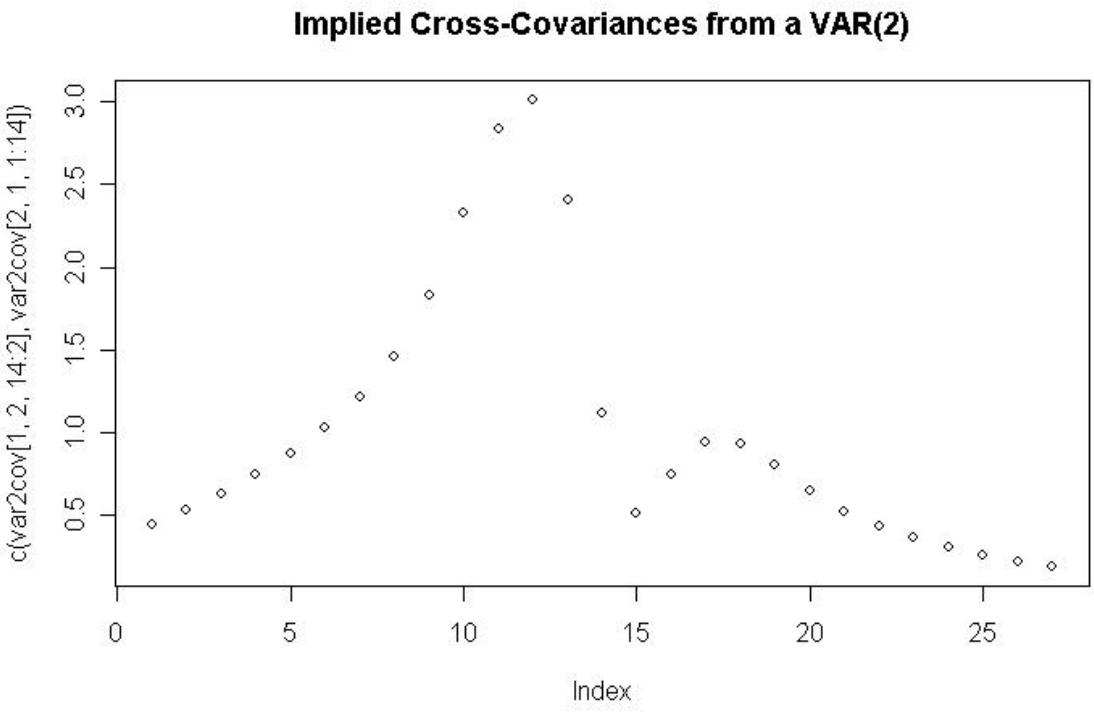

Figure 32: The cross-correlation function of the unemployment rate and the inflation rate implied by the VAR(2) model.

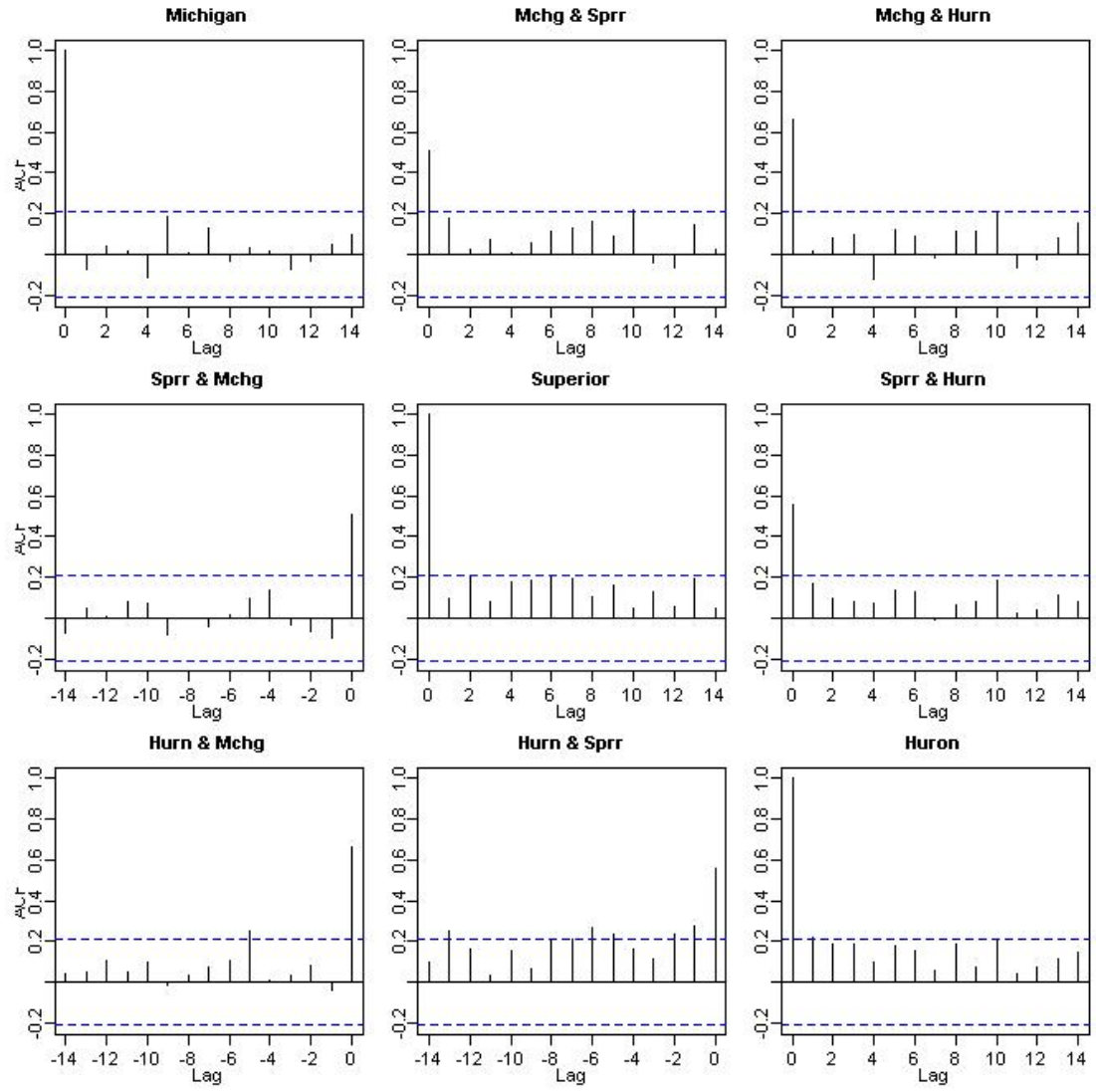

Figure 33: The empirical auto-correlation and cross-correlations function of the annual precipitation at Lakes Superior, Huron, and Michigan.

	Maximum Likelihood with Regression Approximation	Whittle Approximation
A_1	(-0.2081, -0.3401, -0.8997, -0.7509, -0.4554, 0.3307, -0.4317, 0.4014, -0.0602)	(-0.2798, 0.3268, -0.0757, -0.4089, 0.2689, 0.3370, -0.4194 0.3236, 0.1464)
Σ	(2.8295, 3.3341, 3.4696, 3.3341, 3.9288, 4.1146, 3.4696, 4.1146, 11.0970)	(1.4674, 0.8552, 0.9218, 0.8552, 1.6725, 0.8563, 0.9218, 0.8563, 1.4106)
d	(-0.4900, -0.3196, 0.1498)	(0.0533, -0.2027, 0.0698)
Log likelihood	-583.5166	-926.7704

Table 34: VARFI estimates for the precipitation data. All log likelihoods are the exact log likelihoods computed using Sowell's algorithm at the estimated parameter values.

In Tables 33 and 34, we report the estimated parameter values for the FIVAR and VARFI models. Because the likelihood of the FIVAR model is dramatically higher than that of the VARFI model, we focus on the FIVAR model as the better description of the data. According to the maximum likelihood estimates of the FIVAR model, the precipitation at Lake Superior has the largest differencing parameter, while the differencing parameter of the precipitation at Lake Michigan is almost 0. The cross-covariances between Lake Huron and Lake Superior are plotted in Figure 34.

11 Conclusion

This paper has discussed two multivariate generalizations of fractionally integrated autoregressive models. While the two models appear similar at first glance, their implications differ dramatically. One model leads to series with different orders of integration, while the other can lead to series which have the same order of integration but a relationship like cointegration among them. We have also described computationally efficient methods for using these two models. The algorithms for simulation and computing the quadratic form can be applied to any multivariate model, not just FIVAR and VARFI models. Finally, we have fit these models to data.

Much research remains to be done, because these models are relatively new. There are likely to be theoretical results on the growth of the condition number of Ω, just as there are in the univariate case. It make also be possible to prove whether it is always possible to simulate if sufficiently many covariances are used. It is also unknown whether there is a more elegant algorithm for computing the determinant. Work also remains to be done on cointegration in these models. We hope that finding algorithms which make computation with these models faster will allow them to enter wider use, so that long memory can be addressed in a multivariate context.

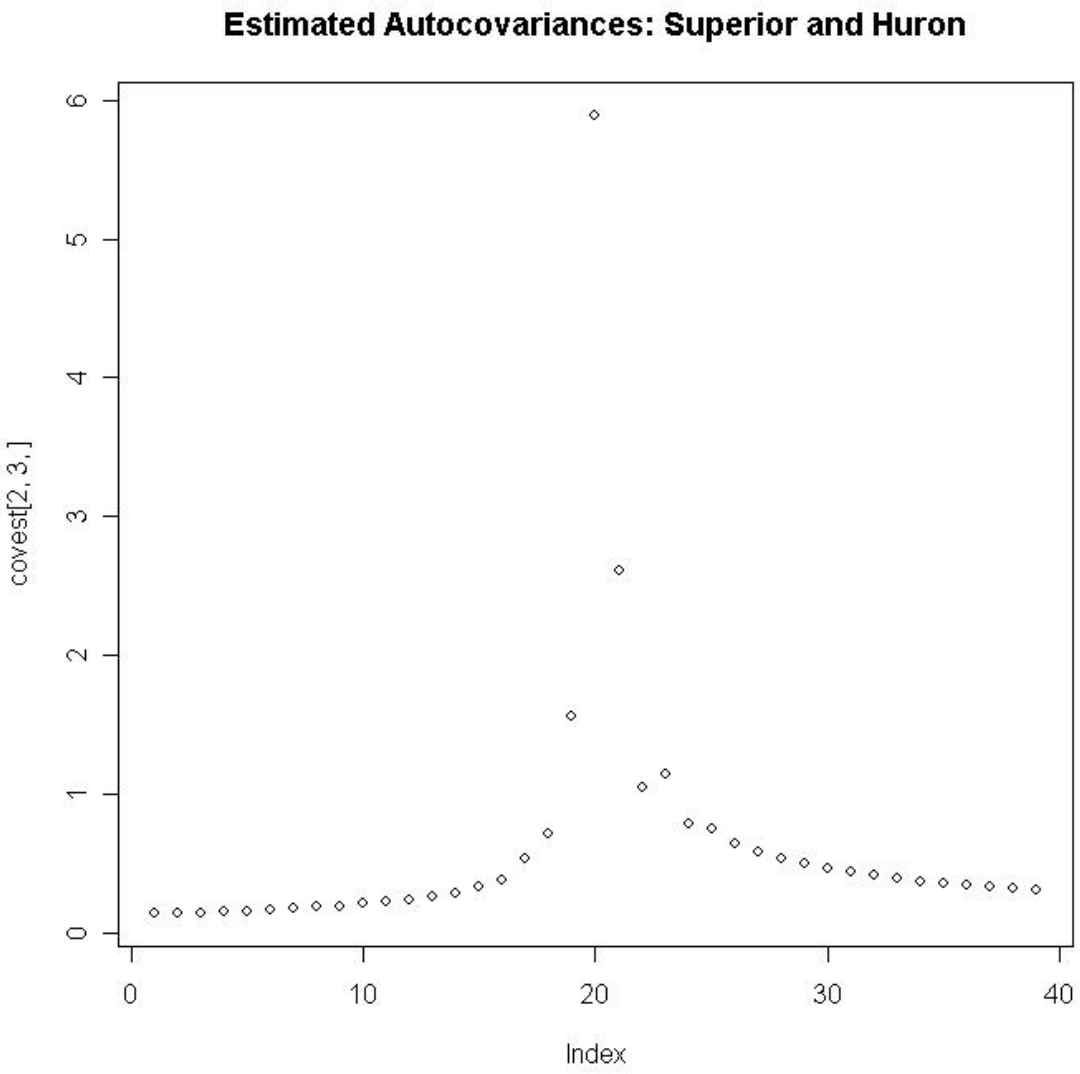

Figure 34: The cross-covariance function between Lake Huron and Lake Superior implied by the maximum likelihood estimate of the FIVAR model.

References

Ansley, C. F. and Kohn, R. "A note on reparameterizing a vector autoregressive moving average model to enforce stationarity." *Journal of Statistical Computation and Simulation*, 24:99–106 (1986).

Baillie, R. T. "Long memory processes and fractional integration in econometrics." *Journal of Econometrics*, 73:5–59 (1996).

Bertelli, S. and Caporin, M. "A note on calculating autocovariances of long-memory processes." *Journal of Time Series Analysis*, 23(5) (2002).

Bottcher, A. and Silbermann, B. *Introduction to large truncated Toeplitz matrices*. Springer (1999).

Brockwell, P. J. and Davis, R. A. *Time Series: Theory and Methods*. Springer-Verlag (1993).

Chan, R. H. and Ng, M. K. "Conjugate gradient methods for Toeplitz systems." *SIAM Review*, 38(3):427–482 (1996).

Chan, T. F. "An optimal circulant preconditioner for Toeplitz systems." *SIAM Journal of Statistical Computation*, 9:766–771 (1988).

Chan, T. F. and Olkin, J. A. "Circulant preconditioners for Toeplitz-block matrices." *Numerical Algorithms*, 6:89–101 (1994).

Chen, W., Hurvich, C. M., and Lu, Y. "On the correlation matrix of the discrete Fourier transform and the fast solution of large Toeplitz systems for long-memory time series." *Journal of the American Statistical Association*, 101(474):812–822 (2006).

Chung, C.-F. "Calculating and analyzing impulse responses for the vector ARFIMA model." *Economics Letters*, 71:17–25 (2001).

Davies, R. and Harte, D. "Tests for the Hurst effect." *Biometrika*, 74:95–101 (1987).

Dempster, A. P., Laird, N., and Rubin, D. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society: Series B*, 39:1–38 (1977).

Denman, E. D. and Beavers, A. N. "The matrix sign function and computations in systems." *Applied Mathematics and Computation*, 2:63–94 (1976).

Deo, R., Hurvich, C., and Lu, Y. "Forecasting realized volatility using a long-memory stochastic volatility model: estimation, prediction and seasonal adjustment." *Journal of Econometrics*, 131(1-2):29–58 (2006).

Doornik, J. A. and Ooms, M. "Computational aspects of maximum likelihood estimation of autoregressive fractionally integrated moving average models." *Computational Statistics and Data Analysis*, 42(3):333–348 (2003).

Dunsmuir, W. and Hannan, E. "Vector linear time series models." *Advances in Applied Probability*, 8(2):339–364 (1976).

Granger, C. W. J. and Joyeux, R. "An introduction to long memory time series models and fractional differencing." *Journal of Time Series Analysis*, 1 (1980).

Grenander, U. and Szego, G. *Toeplitz Forms and Their Applications*. University of California Press (1958).

Hall, R. E. and Taylor, J. B. *Macroeconomics*. W. W. Norton and Company (1997).

Hamilton, J. D. *Time Series Analysis*. Princeton University Press (1994).

Hannan, E. *Multiple Time Series*. John Wiley and Sons, Inc. (1970).

Harville, D. A. "Maximum likelihood approaches to variance component estimation and to related problems." *Journal of the American Statistical Association*, 72:320–338 (1977).

Heath, M. T. *Scientific Computing: an Introductory Survey*. McGraw-Hill (2002).

Hosking, J. "Fractional differencing." *Biometrika*, 68(1):165–176 (1981).

Hosoya, Y. "The quasi-likelihood approach to statistical inference on multiple time-series with long-range dependence." *Journal of Econometrics*, 73 (1996).

Hualde, J. and Robinson, P. "Root-n-consistent estimation of weak fractional cointegrations." *Journal of Econometrics*, 140:450–484 (2007).

Lobato, I. N. "Consistency of the averaged cross-periodogram in long memory time series." *Journal of Time Series Analysis*, 18(2):137–155 (1997).

Luceno, A. "A fast likelihood approximation for vector general linear processes with long series: Application to fractional differencing." *Biometrika*, 83(3):603–614 (1996).

Martin, V. L. and Wilkins, N. P. "Indirect estimation of ARFIMA and VARFIMA models." *Journal of Econometrics*, 93:149–175 (1999).

Peach, R. W., Rich, R. W., and Anoniades, A. "The Historical and Recent Behavior of Goods and Services Inflation." *Economic Policy Review*, 10:19–31 (2004).

Ravishanker, N. and Ray, B. K. "Bayesian analysis of vector ARFIMA processes." *Australian Journal of Statistics*, 39:295–312 (1997).

—. "Bayesian prediction for vector ARFIMA processes." *International Journal of Forecasting*, 18(2):207–214 (2002).

Robinson, P. and Hualde, J. "Cointegration in fractional systems with unknown integration orders." *Econometrica*, 71(6):1727–1766 (2003).

Shewchuk, J. R. "An introduction to the conjugate gradient method without the agonizing pain." (1994). Unpublished manuscript.

Sowell, F. "A decomposition of block Toeplitz matrices with applications to vector time series." (1989a). Unpublished manuscript.

—. "Maximum likelihood estimation of fractionally integrated time series models." (1989b). Unpublished manuscript.

Tsay, W.-J. "Maximum likelihood estimation of stationary multivariate ARFIMA processes." (2007). Unpublished manuscript.

Weisstein, E. "Hypergeometric function." (2008). `http://mathworld.wolfram.com/HypergeometricFunction.ht`

Whittle, P. "On the fitting of multivariate autoregressions, and the approximate canonical factorization of a spectral density matrix." *Biometrika*, 50(1-2):129–134 (1963).

Wood, A. T. and Chan, G. "Simulation of stationary Gaussian processes in $[0, 1]^d$." *Journal of Computational and Graphical Statistics*, 3(4):409–432 (1994).