Nielsen, Morten Ørregaard; Noël, Antoine

Working Paper

To infinity and beyond: Efficient computation of ARCH(\infty) models

Queen's Economics Department Working Paper, No. 1425

Provided in Cooperation with:
Queen's University, Department of Economics (QED)

Suggested Citation: Nielsen, Morten Ørregaard; Noël, Antoine (2020) : To infinity and beyond: Efficient computation of ARCH(\infty) models, Queen's Economics Department Working Paper, No. 1425, Queen's University, Department of Economics, Kingston (Ontario)

This Version is available at:
https://hdl.handle.net/10419/230578

Queen's Economics Department Working Paper No. 1425

To infinity and beyond: Efficient computation of ARCH(\infty) models

Morten Ørregaard Nielsen
Queen's University and CREATES

Antoine Noël

Department of Economics
Queen's University
94 University Avenue
Kingston, Ontario, Canada
K7L 3N6

2-2020

To infinity and beyond: Efficient computation of ARCH(∞) models*

Morten Ørregaard Nielsen†
Queen's University and CREATES
mon@econ.queensu.ca

Antoine L. Noël
Queen's University
noela@econ.queensu.ca

February 18, 2020

Abstract

This paper provides an exact algorithm for efficient computation of the time series of conditional variances, and hence the likelihood function, of models that have an ARCH(∞) representation. This class of models includes, e.g., the fractionally integrated generalized autoregressive conditional heteroskedasticity (FIGARCH) model. Our algorithm is a variation of the fast fractional difference algorithm of Jensen and Nielsen (2014). It takes advantage of the fast Fourier transform (FFT) to achieve an order of magnitude improvement in computational speed. The efficiency of the algorithm allows estimation (and simulation/bootstrapping) of ARCH(∞) models, even with very large data sets and without the truncation of the filter commonly applied in the literature. We also show that the elimination of the truncation of the filter substantially reduces the bias of the quasi-maximum-likelihood estimators. Our results are illustrated in two empirical examples.

JEL codes: C22, C58, C63, C87.

Keywords: Circular convolution theorem, conditional heteroskedasticity, fast Fourier transform, FIGARCH, truncation.

*We are grateful to participants at the annual Canadian Econometric Study Group meeting 2019 for comments. Nielsen thanks the Canada Research Chairs program, the Social Sciences and Humanities Research Council of Canada, and the Center for Research in Econometric Analysis of Time Series (CREATES, funded by the Danish National Research Foundation, DNRF78) for financial support. Noël thanks the Social Sciences and Humanities Research Council of Canada for financial support. Data and computer programs to replicate results are available on the authors' websites.

†Corresponding author. Address: Department of Economics, 94 University Avenue, Queen's University, Kingston, Ontario K7L 3N6, Canada. Email: mon@econ.queensu.ca. Tel.: 613-533-2262. Fax: 613-533-6668.

1 Introduction

Many autoregressive conditional heteroskedasticity (ARCH) models have a so-called ARCH(∞) representation, which is similar to the linear representation for time series models for the conditional mean. The ARCH(∞) representation, or model, appears to be introduced by Bollerslev (1986). An example is the very popular fractionally integrated GARCH (FIGARCH) model of Baillie, Bollerslev, and Mikkelsen (1996); see below for additional examples.

In ARCH(∞) models, the sequence of conditional variances is a linear combination of all previous squared innovations, specifically a linear convolution, where the weights are simple functions of the parameters of the model. By standard methods, the calculation of the sequence of conditional variances, and hence of the likelihood function, requires $O(T^2)$ arithmetic operations. Evaluating the likelihood function of an ARCH(∞) model requires calculation of the sequence of conditional variances, and optimizing the likelihood for parameter estimation in the model typically requires many iterations and hence many calculations of the sequence of conditional variances based on different parameter values. Thus, for large sample sizes, the $O(T^2)$ computational cost can be prohibitive for estimation of the model. Even if it is not prohibitive for a given sample size, it may render bootstrap inference or simulation methods infeasible.

To speed up computation it has been standard in the literature, at least since Baillie et al. (1996), to truncate the convolution at a fixed truncation number such as $1,000$. That is, only a part of the time series of innovations is used to calculate the entire sequence of conditional variances. Of course, this is an approximation which implies that the calculation is no longer exact.

In this paper, we discuss efficient computation of the sequence of conditional variances, and hence the likelihood function, for any model that has an ARCH(∞) representation. We make two separate contributions.

First, we show how to apply the fast Fourier transform (FFT) of Cooley and Tukey (1965) and the circular convolution theorem to reduce the required number of arithmetic operations from $O(T^2)$ to $O(T \log T)$. Our proposed algorithm is a variation of an algorithm recently proposed by Jensen and Nielsen (2014) for the calculation of fractional differences. The algorithm is exact and does not rely on any approximation device. The increase in computational speed can easily be a factor of 10 or even much more for sample sizes that can be encountered in practical applications, and is sufficient to make bootstrap and simulation methods for ARCH(∞) models feasible, even with very large sample sizes.

Second, in a small Monte Carlo simulation, we quantify the estimation bias introduced by using a fixed truncation number. We show that, for truncation numbers typically applied

in the literature, the bias can be substantial. However, because our algorithm is exact and sufficiently fast to eliminate the need for truncation, it does not suffer from this bias.

In an empirical application we apply our algorithm to two data sets of exchange rates. These highlight the differences obtained with and without the truncation number, and also emphasize the very different computational time achieved with the new FFT-based algorithm compared with standard linear convolution.

Recently, Klein and Walther (2017) also proposed an application of the FFT and circular convolution theorem, based on Jensen and Nielsen (2014), to calculate the sequence of conditional variances for FIGARCH models. Their method and algorithm is closely related to ours. However, Klein and Walther (2017) maintain the fixed truncation number throughout and do not consider exact algorithms. They also do not consider the bias associated with the application of a fixed truncation number. We include some comparisons with their algorithm, which show that, in the absence of the fixed truncation number, it is not much faster than the standard linear convolution algorithm.

Many conditional heteroskedasticity models proposed in the literature have an ARCH(∞) representation, going back to the GARCH model of Bollerslev (1986). However, the conditional variance of the GARCH model is a sum of a fixed number of terms (in the same way that an ARMA model is a sum of a fixed number of terms, but it has an MA(∞) representation), and consequently it will not benefit noticeably from our method. In contrast, our method will be of great benefit to models where the conditional variance is not a sum of a *fixed* number of terms. In particular, this covers all the long-memory-type conditional variance models in the ARCH(∞) class, e.g., the FIGARCH model of Baillie et al. (1996), the long-memory GARCH model of Karanasos, Psaradakis, and Sola (2004), and the hyperbolic GARCH model of Davidson (2004). Finally, our method also applies to some models that are not even in the ARCH(∞) class. For example, it applies to the fractionally integrated asymmetric power ARCH model of Tse (1998) and the linear ARCH model of Robinson (1991) and Giraitis, Robinson, and Surgailis (2000), and it partly applies to the fractionally integrated exponential GARCH model of Bollerslev and Mikkelsen (1996); see Section 2.2 for details.

The next section defines the class of ARCH(∞) models and explains our proposed FFT-based algorithm for calculation of the sequence of conditional variances. Section 3 presents results on the computational time for the new algorithm and compares with the standard linear convolution algorithm and the Klein and Walther (2017) algorithm. An analysis of the effects on computational time of using a fixed truncation number is also given. In Section 4 we present the results of a small simulation study that examines the estimation bias that results from truncation. Section 5 illustrates our results with two empirical applications. Finally, in Section 6 we give some concluding remarks.

2 FFT algorithm for ARCH(∞) models

We consider the following generic conditional variance model for the series $(\epsilon_t)_{t=1}^T$, which is either an observable series (such as asset returns) or the error term from a (regression) model,

$$\epsilon_t = \sigma_t z_t, \quad \mathbb{E}_{t-1}(\epsilon_t^2) = \sigma_t^2, \quad \text{and} \quad z_t \sim \text{i.i.d.}(0,1). \tag{1}$$

Here, $(z_t)_{t=1}^T$ is an independently and identically distributed (i.i.d.) innovation series with mean zero and variance normalized to one, $(\sigma_t^2)_{t=1}^T$ is the conditional variance series, and $\mathbb{E}_{t-1}$ represents the expectation conditional on the information available at time $t-1$.

The ARCH(∞) model (or representation) for the conditional variance is

$$\sigma_t^2 = c + \sum_{j=0}^{\infty} \lambda_j \epsilon_{t-j}^2, \tag{2}$$

where c and λ_j are the corresponding constant and coefficients, respectively. Of course, in practical applications, the series (ϵ_t) will be available only for $t = 1, \ldots, T$, where T denotes the sample size. Thus, the summation in (2) will need to be truncated at $j = t-1$, thus setting pre-sample values of ϵ_t equal to their (conditional) mean of zero. This results in the feasible ARCH(∞) model,

$$\sigma_t^2 = c + \sum_{j=0}^{t-1} \lambda_j \epsilon_{t-j}^2. \tag{3}$$

The summation in (3) is a linear convolution. By a standard linear convolution algorithm, for given series $(\epsilon_t)_{t=1}^T$ and $(\lambda_j)_{j=0}^{T-1}$, and a given value of t, the calculation (3) requires $2t$ arithmetic operations (t multiplications and t additions). Thus, the calculation of the series $(\sigma_t^2)_{t=1}^T$ requires $\sum_{t=1}^T 2t = T^2 + T$ arithmetic operations.

When the sample size, T, is large, the computational burden of T^2 arithmetic operations can make the calculation of the conditional variance series $(\sigma_t^2)_{t=1}^T$ very slow. This can render optimization of a likelihood function, and certainly simulations and bootstrapping, infeasible. To overcome the computational burden of the calculation in (3), Baillie et al. (1996) suggested truncating the summation in (3) at a fixed number, say n, which is typically chosen to be $1,000$. This truncation has become the standard in the literature. The resulting model is thus

$$\sigma_t^2 = c + \sum_{j=0}^{\min\{t-1,n\}} \lambda_j \epsilon_{t-j}^2. \tag{4}$$

The required number of arithmetic operations to calculate the series $(\sigma_t^2)_{t-1}^T$ in (4) is only of order nT. Importantly, for a fixed truncation number n, the number of operations grows only linearly with the sample size and hence avoids the squared growth required for the calculation in (3).

While the introduction of the truncation number, n, in (4) allows for much faster calcu-

lation of the conditional variance series $(\sigma_t^2)_{t=1}^T$, it also introduces an approximation. It is no longer an exact algorithm to calculate the desired series given in (3). In Section 4 we present the results of some Monte Carlo simulations to illustrate and quantify the bias in parameter estimation resulting from this approximation.

We now introduce an alternative method to calculate the conditional variance series $(\sigma_t^2)_{t=1}^T$ in (3) without truncation using frequency domain techniques. Specifically, we will apply the fast Fourier transform (FFT) combined with the so-called circular convolution theorem. As discussed in Jensen and Nielsen (2014) in the context of calculating fractional differences, this method reduces the number of arithmetic operations to order $T \log T$. To describe this method, we will need the following two definitions.

Definition 1. The discrete Fourier transform (DFT), $f = (f_j)_{j=1}^T$, of a series $a = (a_t)_{t=1}^T$ is the solution to

$$a = T^{-1} F f,$$

where F is the Fourier matrix given by $F_{jk} = w_T^{(j-1)(k-1)}$ with $w_T = e^{2\pi \mathsf{i}/T}$ and $\mathsf{i} = \sqrt{-1}$. □

Definition 2. Let a_j and b_j be two periodic sequences, meaning that $a_{j+NT} = a_j$ and $b_{j+NT} = b_j$ for $N = \pm 0, \pm 1, \pm 2, \ldots$. Then the circular convolution of $(a_j)_{j=1}^T$ and $(b_j)_{j=1}^T$ is defined as

$$(a \circledast b)_t = \sum_{j=1}^{T} a_j b_{t-j+1} = \sum_{j=1}^{t} a_j b_{t-j+1} + \sum_{j=t+1}^{T} a_j b_{T+t-j+1}, \quad t = 1, \ldots, T. \qquad \square$$

We next present two theorems. The first is a finite version of the circular convolution theorem, which shows how the circular convolution in Definition 2 can be calculated using the DFT in Definition 1. For periodic integrable functions, this result can be found in Zygmund (2003, Thm. 1.5, p. 36). The finite version has appeared in the early engineering literature as an application of the FFT; e.g. Stockham (1966, p. 230) and Cooley, Lewis, and Welch (1969, p. 32). Our version, presented in Theorem 1, is taken from Jensen and Nielsen (2014) and the proof can be found there.[1]

Theorem 1. *Let $a = (a_t)_{t=1}^T$ and $b = (b_t)_{t=1}^T$ be two sequences. Then*

$$a \circledast b = T^{-1} F(\bar{F} a \circ \bar{F} b), \tag{5}$$

where $\circ$ denotes element-wise multiplication.

To apply the result in Theorem 1 in our context, we need to transform the linear convolutions in (3) and (4) into circular convolutions. To economize on notation, we consider only (4), but allow the possibility that $n = T - 1$, in which case we obtain (3).

[1] In Jensen and Nielsen (2014) there is a small typo in the fifth and sixth lines of the proof, where $(t-1)(s-1)$ should be $(t-1)(u-1)$ in the exponents. However, since the last equality sets $s = u$ the result is the same.

Our second theorem shows that the sequence of conditional variances in (3) or (4) (where the former is obtained by setting $n = T-1$ in the latter) can be calculated by extending the sequences with zeros and applying the result in Theorem 1. Specifically, we extend the vector of errors, $(\epsilon_t)_{t=1}^T$, and the vector of coefficients, $(\lambda_j)_{j=0}^n$, with zeros to length $2T-1$. Thus, let the extended vectors be denoted $\tilde{\epsilon} = [\epsilon', O'_{T-1}]$ and $\tilde{\lambda} = [\lambda', O'_{2T-1-(n+1)}]$, respectively, where O_m denotes an $m \times 1$ vector of zeros. We then prove in Theorem 2 that the linear convolution of $(\epsilon_t)_{t=1}^T$ and $(\lambda_j)_{j=0}^n$ in (4) can be found as the first T elements of the circular convolution of $(\tilde{\epsilon}_t)_{t=1}^{2T-1}$ and $(\tilde{\lambda}_j)_{j=0}^{2T-2}$.

Theorem 2. *Let $\tilde{\epsilon}$ and $\tilde{\lambda}$ denote the $(2T-1)\times 1$ extended vectors of errors and coefficients, respectively. Then the vector of conditional variances, $(\sigma_t^2)_{t=1}^T$, in (3) or (4) can be calculated as the first T elements of the $(2T-1)\times 1$ vector $c + T^{-1}F(\bar{F}\tilde{\lambda} \circ \bar{F}\tilde{\epsilon})$.*

Because of the truncation number, n, the result in Theorem 2 is different from the corresponding result in Theorem 2 of Jensen and Nielsen (2014), which deals with fractional differencing without a truncation number. We therefore give a short proof of Theorem 2.

Proof. From Theorem 1 we know that $\tilde{\lambda} \circledast \tilde{\epsilon} = T^{-1}F(\bar{F}\tilde{\lambda} \circ \bar{F}\tilde{\epsilon})$. Therefore, it only remains to be shown that $(\tilde{\lambda} \circledast \tilde{\epsilon})_t = \sum_{j=0}^{\min\{t-1,n\}} \lambda_j \epsilon_{t-j}^2$ for $t = 1, \ldots, T$. From Definition 2, noting that the extended sequences have $2T-1$ elements and that the $\tilde{\lambda}_j$ sequence starts at index $j = 0$, we find that

$$(\tilde{\lambda} \circledast \tilde{\epsilon})_t = \sum_{j=1}^{t} \tilde{\lambda}_{j-1}\tilde{\epsilon}_{t-j+1}^2 + \sum_{j=t+1}^{T+t-1} \tilde{\lambda}_{j-1}\tilde{\epsilon}_{2T+t-j}^2 + \sum_{j=T+t-1}^{2T-1} \tilde{\lambda}_{j-1}\tilde{\epsilon}_{2T+t-j}^2. \tag{6}$$

When $j = t+1, \ldots, T+t-1$ we have $2T+t-j = T+1, \ldots, 2T-1$, so that $\tilde{\epsilon}_{2T+t-j}^2 = 0$ by definition. Similarly, when $j = T+t+1, \ldots, 2T-1$ and $t = 1, \ldots, T$ we have $\tilde{\lambda}_{j-1} = 0$ by definition. This leaves only the first term on the right-hand side of (6). When $t-1 \leq n$, this term is equal to $\sum_{j=0}^{\min\{t-1,n\}} \tilde{\lambda}_j \tilde{\epsilon}_{t-j}^2$, which is what we needed to show. When $t-1 > n$, we split the first term on the right-hand side of (6) as

$$\sum_{j=1}^{t} \tilde{\lambda}_{j-1}\tilde{\epsilon}_{t-j+1}^2 = \sum_{j=0}^{n} \tilde{\lambda}_j \tilde{\epsilon}_{t-j}^2 + \sum_{j=n+1}^{t-1} \tilde{\lambda}_j \tilde{\epsilon}_{t-j}^2.$$

For $j = n+1, \ldots, t-1$ and $t = 1, \ldots, T$ we have $\tilde{\lambda}_j = 0$ by definition, so that the last summation is zero, which proves the desired result. □

Theorem 2 shows that the linear convolution of $(\epsilon_t)_{t=1}^T$ and $(\lambda_j)_{j=0}^n$ in (4) can be found as the first T elements of the circular convolution of $(\tilde{\epsilon}_t)_{t=1}^{2T-1}$ and $(\tilde{\lambda}_j)_{j=0}^{2T-2}$, which in turn can be calculated using the DFT. The power of this result lies in the fact that the required DFTs can be calculated extremely efficiently by the FFT. Indeed, the latter requires only an order

Listing 1: Matlab code to calculate ARCH(∞) conditional variances by LC method

```
function [sigma2_arch] = arch_lc(cst, epsilon, lambda)
     sigma2_arch = cst + filter(lambda, 1, epsilon.^2);
end
```

$T \log T$ arithmetic operations, as shown by Cooley and Tukey (1965). This implies that, not only can the conditional variances be calculated an order of magnitude faster than using the standard linear convolution method, but also that the order of magnitude is the same with and without truncation. That is, there is no need to introduce a truncation number to speed up computation as in (4).

Most standard implementations of the FFT requires the length of the vectors to be a power of two.[2] This is easily accommodated by further extending the vectors with zeros. Thus, to apply the FFT in the calculations, the vectors ϵ and λ must be extended with zeros such that the number of elements in the vectors is equal to the smallest power of two that is at least $2T-1$. This ensures that both Theorem 2 and the FFT can be applied.

The implementation described in the previous paragraph implies that conditional variance series for a range of sample sizes can be calculated in the same amount of time. For example, consider $T = 1,025$ and $T = 2,048$. In both cases, the smallest power of two that is at least $2T-1$ is 11, and consequently one must extend with $3,071$ zeros when $T = 1,025$ and $2,048$ zeros when $T = 2,048$. In both situations, the extended vectors have $4,096$ elements, and thus it will take the same amount of time to compute the conditional variances using the FFT-based method in Theorem 2.

In Listings 1 and 2 we present our Matlab codes to implement the standard linear convolution algorithm and the FFT-based algorithm in Theorem 2, respectively. The former is a simple application of Matlab's `filter` function. Implementations of the FFT-based method in Theorem 2 for R and Ox can be found in Listings 3 and 4, respectively. All codes are downloadable from the authors' websites.

2.1 Example: FIGARCH model

We now consider an example that we will apply extensively in the remainder. The most well-known ARCH(∞) model for the conditional variance is probably the FIGARCH(p,d,q) model of Baillie et al. (1996). For that model, the conditional variance specification is

$$\beta(L)\sigma_t^2 = \omega + \Big(\beta(L) - \phi(L)(1-L)^d\Big)\epsilon_t^2, \tag{7}$$

[2]Some modern implementations require only that the length is a product of powers of small prime numbers, such as $2^k 3^l 5^m$, and the subsequent discussion is easily adapted to such cases.

Listing 2: Matlab code to calculate ARCH(∞) conditional variances by FFT method

```
function [sigma2_arch] = arch_fft(cst, epsilon, lambda)
    T = size(epsilon, 1);
    np2 = 2.^nextpow2(2*T-1);
    sigma2_arch = ifft(fft(epsilon.^2, np2).*fft(lambda, np2));
    sigma2_arch = cst + sigma2_arch(1:T);
end
```

Listing 3: R code to calculate ARCH(∞) conditional variances by FFT method

```
arch_fft <- function(cst, epsilon, lambda){
    iT <- length(epsilon)
    np2 <- nextn(2*iT-1, 2)
    sigma2_arch <- fft(fft(c(lambda, rep(0, np2-iT))) * fft(c(epsilon^2,
        rep(0, np2-iT))), inverse = T) / np2;
    sigma2_arch <- cst + sigma2_arch[1:iT]
    return(Re(sigma2_arch))
}
```

where $\beta(L) = 1-\sum_{i=1}^{p} \beta_i L^i$ and $\phi(L) = 1-\sum_{i=1}^{\max\{p,q\}} \phi_i L^i$ are polynomials in the lag operator, L, whose roots all lie outside the complex unit circle. The fractional difference operator $(1-L)^d$ is defined in terms of the fractional coefficients $\pi_j(u) = u(u+1)\dots(u+j-1)/j!$ from the binomial expansion of $(1-z)^{-u}$, so that, for a generic series $(x_t)_{t=1}^{T}$, we have $(1-L)^d x_t = \sum_{j=0}^{t-1} \pi_j(-d) x_{t-j}$.

The specification (7) can be rearranged to get an ARCH(∞) representation. For simplicity, suppose $p = q = 1$, so that $\beta(L) = 1-\beta L$ and $\phi(L) = 1-\phi L$. This is the most commonly applied variant of the FIGARCH model. Then (7) can be written in the form (2)

Listing 4: Ox code to calculate ARCH(∞) conditional variances by FFT method

```
arch_fft(const cst, const epsilon, const lambda)
{
    decl T, sigma2_arch;
    T = rows(epsilon);
    sigma2_arch = fft(cmul(fft(lambda'~zeros(1,T)), fft((epsilon.^2)'~zeros
        (1,T)), 2);
    return cst + sigma2_arch'[0:T-1];
}
```

Listing 5: Matlab code to calculate λ_j coefficients of FIGARCH$(1,d,1)$ by LC method

```
function [lambda] = lambda_lc(phi, beta, d, nTrunc)
    lambda = zeros(nTrunc+1,1);
    k = (1:nTrunc)';
    pij = cumprod((k(1:end)-d-1)./(k(1:end)));
    g = zeros(nTrunc,1);
    g(1) = phi - beta + d;
    g(2:end) = phi*pij(1:end-1) - pij(2:end);
    lambda(1) = 0;
    lambda(2:end) = filter(beta.^(0:nTrunc-1), 1, g);
end
```

with $c = \omega/(1-\beta)$ and

$$\lambda_j = \begin{cases} 0 & \text{for } j = 0, \\ \phi - \beta + d & \text{for } j = 1, \\ \beta\lambda_{j-1} + \phi\pi_{j-1}(-d) - \pi_j(-d) & \text{for } j \geq 2; \end{cases} \tag{8}$$

see Baillie et al. (1996) for details. Let $g_1 = \phi - \beta + d$ and $g_j = \phi\pi_{j-1}(-d) - \pi_j(-d)$ for $j \geq 2$. Then the λ_j coefficients in (8) can be rewritten as

$$\lambda_j = \begin{cases} 0 & \text{for } j = 0, \\ \sum_{i=0}^{j-1} \beta^i g_{j-i} & \text{for } j \geq 1. \end{cases} \tag{9}$$

The last term in (9) is clearly a linear convolution. To speed up computation, the series of λ_j coefficients should also be calculated using our FFT-based algorithm.

It follows that efficient calculation of the conditional variances in the FIGARCH$(1,d,1)$ model should use our FFT-based algorithm twice: once to calculate the conditional variances given the λ_j coefficients, and once to calculate the λ_j coefficients given the model parameters. The former calculation is described in detail in Theorem 2. Similarly, $(\lambda_j)_{j=1}^n$ can be found as the first n elements of the $(2n-1)\times 1$ vector $n^{-1}F(\bar{F}\tilde{\beta} \circ \tilde{g})$, where $\tilde{\beta}$ and $\tilde{g}$ denote the vectors $(\beta^j)_{j=0}^{n-1}$ and $(g_j)_{j=1}^n$ extended with zeros to length of the smallest power of two that is at least $2n-1$. Matlab implementations of the calculation of $(\lambda_j)_{j=0}^n$ are shown in Listings 5 and 6 for the linear convolution algorithm and the FFT-based algorithm, respectively.

2.2 Application to non-ARCH(∞) models

Our FFT-based methodology can also be applied with great benefit to some models that are not even in the class of ARCH(∞) models. In particular, the linear ARCH model of Robinson (1991) and Giraitis et al. (2000) is given by $\sigma_t = c + \sum_{j=1}^{\infty} \lambda_j \epsilon_{t-j}$, c.f. (2). Clearly, our FFT-

Listing 6: Matlab code to calculate λ_j coefficients of FIGARCH$(1, d, 1)$ by FFT method

```
function [lambda] = lambda_fft(phi, beta, d, nTrunc)
    lambda = zeros(nTrunc+1,1);
    k = (1:nTrunc)';
    pij = cumprod((k(1:end)-d-1)./(k(1:end)));
    g = zeros(nTrunc,1);
    g(1) = phi - beta + d;
    g(2:end) = phi*pij(1:end-1) - pij(2:end);
    np2 = 2.^nextpow2(2*nTrunc-1);
    lambda(1) = 0;
    tmp = ifft(fft(beta.^(0:nTrunc-1)', np2).*fft(g, np2));
    lambda(2:end) = tmp(1:nTrunc);
end
```

based algorithms described above and presented in Listings 2 and 6 can easily be adopted to this model by simply replacing σ_t^2 and ϵ_{t-j}^2 with σ_t and ϵ_{t-j}, respectively. Similarly, the fractionally integrated asymmetric power ARCH model of Tse (1998) is given as in (7), but with σ_t^2 and ϵ_t^2 replaced by σ_t^δ and $h(\epsilon_t)$, respectively, where $h(\epsilon) = (|\epsilon| - \gamma\epsilon)^\delta$ and $\delta > 0$. Again, our FFT-based algorithms apply with minimal changes to this model.

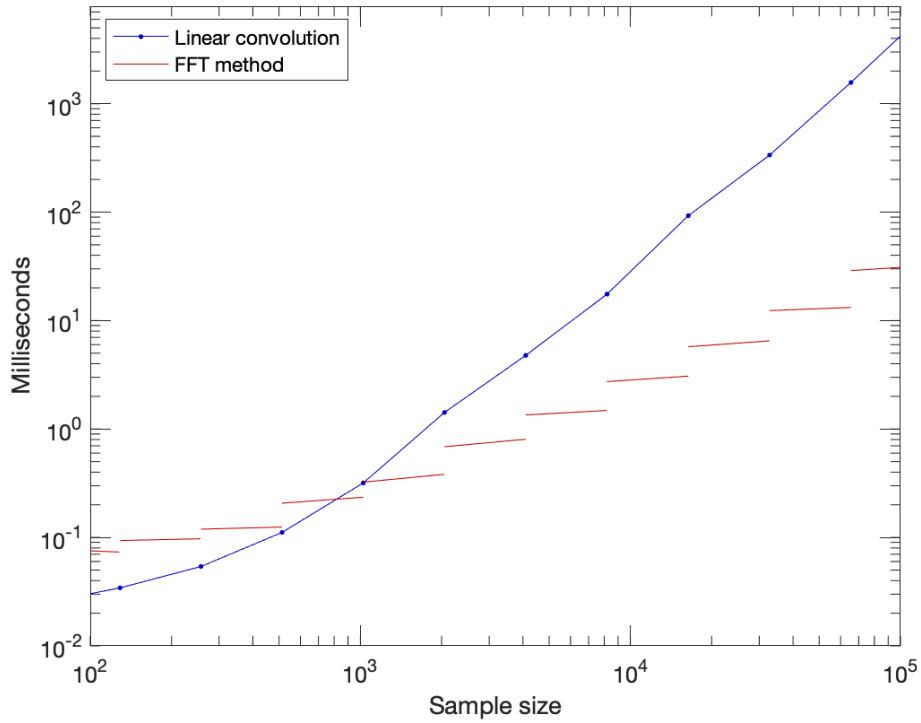
Finally, the popular fractionally integrated exponential GARCH (FIEGARCH) model of Bollerslev and Mikkelsen (1996) is given as in (7), but with σ_t^2 replaced by $\log(\sigma_t^2)$ and ϵ_t^2 replaced by $h(\epsilon_t/\sigma_t)$, where $h(z) = \gamma_1 z + \gamma_2(|z| - \mathbb{E}|z|)$. Thus, because σ_t enters non-linearly on the right-hand side of the conditional variance equation, the sequence of conditional variances for the FIEGARCH model cannot be written as a convolution and our methodology does not apply to the calculation of the conditional variance sequence. However, the λ_j coefficients for this model are identical to those for the FIGARCH model, and the calculation of these can therefore take advantage of our FFT-method as in Listing 6. At least this will reduce the computational time of calculating the λ_j coefficients for this model from $O(T^2)$ to $O(T \log T)$ compared with standard linear convolution.

3 Computational time

In this section, we compare the computational time of our new algorithm in Theorem 2 and Listings 2,6 with the algorithm in Klein and Walther (2017) as well as the standard linear convolution implementation in Listings 1,5. All computations were performed in Matlab on a desktop with an Intel Core i5 (5250U) 1.6 GHz processor running macOS Mojave 10.14.6.

In our first numerical experiment, we compare the difference in computational time for standard linear convolution and our FFT-based algorithm as a function of sample size.

Figure 1: Computational time (logarithmic scale)



Specifically, for a given vector of coefficients, $[\beta, \phi, d, \omega]'$, we calculate the conditional variances for the FIGARCH$(1, d, 1)$ model using the method in Listings 1,5 (blue line) and the FFT-based algorithm in Listings 2,6 (red line). That is, we calculate the λ_j coefficients in (9) followed by the conditional variances in (3), i.e. without truncation, with random numbers for ϵ_t. This is done a large number of times ($10,000$ for $T < 4,096$ and $1,000$ for $T \geq 4,096$), and the median time in milliseconds across repetitions is reported in Figure 1 as a function of sample size. Note that the axes are logarithmic. In particular, the figure clearly shows the different orders of magnitude of the computational time for the two methods (T^2 vs. $T \log T$). The figure also demonstrates how the computational time for the FFT-based algorithm is essentially a step function of the sample size, because of the padding to the smallest power of two that is at least $2T - 1$.

We also notice from Figure 1 that for $T = 500$ to $T = 1,000$, the two methods perform equally well, while linear convolution does better for smaller sample sizes and the FFT-based algorithm does better for larger sample sizes. Thus, to obtain the fastest possible algorithm across all sample sizes, we recommend implementing the linear convolution method for smaller sample sizes and the FFT-based algorithm for larger sample sizes. This can easily be achieved by combining the implementations in Listings 1 and 2 using an `if` statement, and similarly for Listings 5 and 6.

Table 1: Computational time for different sample sizes and truncation numbers

	$n = 1,000$			$n = 2,000$		
T	KW	LC	FFT	KW	LC	FFT
5,000	0.76	0.98	**0.66**	1.58	2.41	**0.81**
10,000	1.45	1.84	**1.27**	2.26	4.17	**1.43**
15,000	1.52	2.69	**1.26**	3.89	5.96	**1.45**
20,000	3.07	3.54	**2.79**	4.00	7.72	**3.01**
30,000	3.19	5.25	**2.81**	4.14	11.2	**3.08**
40,000	6.80	6.97	**6.37**	7.70	14.8	**6.49**
100,000	16.6	17.5	**15.6**	17.6	36.0	**15.3**
	$n = 3,000$			$n = 4,000$		
T	KW	LC	FFT	KW	LC	FFT
5,000	4.00	7.02	**0.98**	7.59	10.5	**1.04**
10,000	4.68	12.9	**1.61**	7.67	18.3	**1.70**
15,000	6.45	18.8	**1.73**	9.31	26.1	**1.71**
20,000	6.39	24.6	**3.23**	9.38	33.9	**3.17**
30,000	10.4	36.4	**3.27**	12.8	49.5	**3.23**
40,000	10.6	48.1	**7.20**	13.1	65.2	**6.80**
100,000	19.9	119	**15.5**	22.3	159	**15.8**
	$n = 5,000$			$n = T - 1$		
T	KW	LC	FFT	KW	LC	FFT
5,000	11.0	14.0	**1.35**	11.0	14.1	**1.45**
10,000	11.0	23.9	**2.04**	35.3	51.1	**2.79**
15,000	12.5	33.7	**2.09**	70.0	110	**3.10**
20,000	12.6	43.6	**3.56**	122	201	**6.40**
30,000	16.3	63.2	**3.59**	294	530	**7.09**
40,000	16.4	82.8	**7.08**	550	1,005	**15.4**
100,000	26.2	201	**15.9**	3,420	6,612	**36.2**

Notes: Median computational time in milliseconds across 10,000 repetitions (fixed n) or 1,000 repetitions ($n = T - 1$) for different sample sizes and truncation numbers. KW is the algorithm of Klein and Walther (2017), LC is linear convolution (Listings 1,5), and FFT is our proposed algorithm (Listings 2,6). The fastest algorithm for each (T, n) pair is highlighted in bold.

In our next experiment, we focus on the effect of truncation, and compare linear convolution (LC), our FFT-based algorithm, and the algorithm of Klein and Walther (2017, henceforth KW).[3] The setup is the same as in Figure 1, except for the introduction of the truncation number. We consider a range of sample sizes from $T = 5,000$ to $T = 100,000$ and truncation numbers from $n = 1,000$ to $n = 5,000$, as well as no truncation, $n = T - 1$. For

[3]The Klein and Walther (2017) Matlab code was downloaded from the publisher's website.

each (T, n) pair, we repeated the calculation $10,000$ times (for fixed n) or $1,000$ times (for $n = T-1$). In Table 1 we report the resulting median time across repetitions in milliseconds.

In Table 1 we first note that our FFT-based method is faster than the other methods for all sample sizes and truncation numbers considered. While the computational times of both the KW and LC algorithms are very sensitive to the choice of truncation number, the FFT-based method is much less so. For example, from $n = 1,000$ to $n = T-1$, the computational time of the KW and LC algorithms increase by a factor of 15 for $T = 5,000$ and several hundred for $T = 100,000$. On the other hand, the computational time of the FFT-based method only slightly more than doubles. Hence, with our FFT-based algorithm, there is really no need for the truncation number and the associated approximation.

Using the exact filter to calculate the conditional variances, i.e. without truncation ($n = T-1$), the difference in computational time between the KW and LC algorithms on the one hand and our FFT-based algorithm on the other hand is very substantial. For $T = 5,000$, which is a rather common sample size in finance, the FFT-based algorithm is nearly 10 times faster than the KW and LC algorithms. For the largest sample size considered, $T = 100,000$, the FFT-based algorithm is nearly 100 times faster than the KW algorithm and over 180 times faster than the standard LC algorithm.

4 Estimation bias

Since we are able to compute the sequence of conditional variances, and hence the likelihood function, of any model with an ARCH(∞) representation very quickly using our FFT-based algorithm, we can explore the effects of approximating the calculations on the estimators. That is, we can simulate the estimation bias that results from using a fixed truncation number in the calculation of the conditional variances in (4). Specifically, in this section we simulate the bias of the estimated coefficients in the baseline FIGARCH$(1, d, 1)$ model for a range of sample sizes and truncation numbers, including no truncation.

We consider sample sizes $T \in \{10,000; 25,000; 50,000\}$ and truncation numbers $n \in \{100; 1,000; t-1\}$. For each sample size, we simulate $10,000$ samples from the FIGARCH$(1, d, 1)$ model given by (1) and (7) with $d = 0.4$, $\phi = 0.2$, $\beta = 0.6$, $\omega = 0.0001$, and $z_t \sim \mathcal{N}(0, 1)$.

We collect the parameters in the vector $\theta = [d, \phi, \beta, \omega]'$. Given observations $(\epsilon_t)_{t=1}^T$ and the truncation number n, the quasi-maximum-likelihood estimator is

$$\hat{\theta} = \arg\max_{\theta} \ \log L(\theta) \quad \text{with} \quad \log L(\theta) = -\frac{1}{2}\sum_{t=1}^{T}\left(\log(\sigma_t^2) + \frac{\epsilon_t^2}{\sigma_t^2}\right), \tag{10}$$

where σ_t^2 is calculated from (4) and (9), and where the maximization is subject to the parameter constraints $\omega > 0, 0 \leq d \leq 1 - 2\phi, 0 \leq \beta \leq d + \phi$ to guarantee non-negativity of the conditional variances; see Baillie et al. (1996, footnote 19).

Table 2: Simulations for a FIGARCH$(1, d, 1)$ model

$T = 10,000$	$n = 100$		$n = 1,000$		$n = T - 1$	
Parameter	Bias	MCSE	Bias	MCSE	Bias	MCSE
d	0.1391	0.0124	0.0122	0.0001	-0.0018	0.0011
ϕ	-0.0327	0.0028	0.0006	0.0005	-0.0022	0.0006
β	0.1014	0.0043	0.0125	0.0012	-0.0035	0.0014
ω	5.2×10^{-5}	2.9×10^{-9}	2.0×10^{-5}	6.5×10^{-10}	5.6×10^{-6}	5.1×10^{-10}
$T = 25,000$	$n = 100$		$n = 1,000$		$n = T - 1$	
Parameter	Bias	MCSE	Bias	MCSE	Bias	MCSE
d	0.1379	0.0121	0.0119	0.0001	-0.0020	0.0011
ϕ	-0.0321	0.0027	0.0008	0.0005	-0.0019	0.0006
β	0.1008	0.0042	0.0123	0.0012	-0.0036	0.0013
ω	5.2×10^{-5}	2.9×10^{-9}	2.0×10^{-5}	6.6×10^{-10}	5.7×10^{-6}	5.0×10^{-10}
$T = 50,000$	$n = 100$		$n = 1,000$		$n = T - 1$	
Parameter	Bias	MCSE	Bias	MCSE	Bias	MCSE
d	0.1562	0.0088	0.0244	0.0003	-0.0003	0.0002
ϕ	-0.0353	0.0020	0.0019	0.0001	-0.0006	0.0001
β	0.1146	0.0026	0.0250	0.0003	-0.0008	0.0003
ω	1.4×10^{-4}	4.7×10^{-9}	6.9×10^{-5}	6.2×10^{-10}	2.5×10^{-6}	1.5×10^{-10}

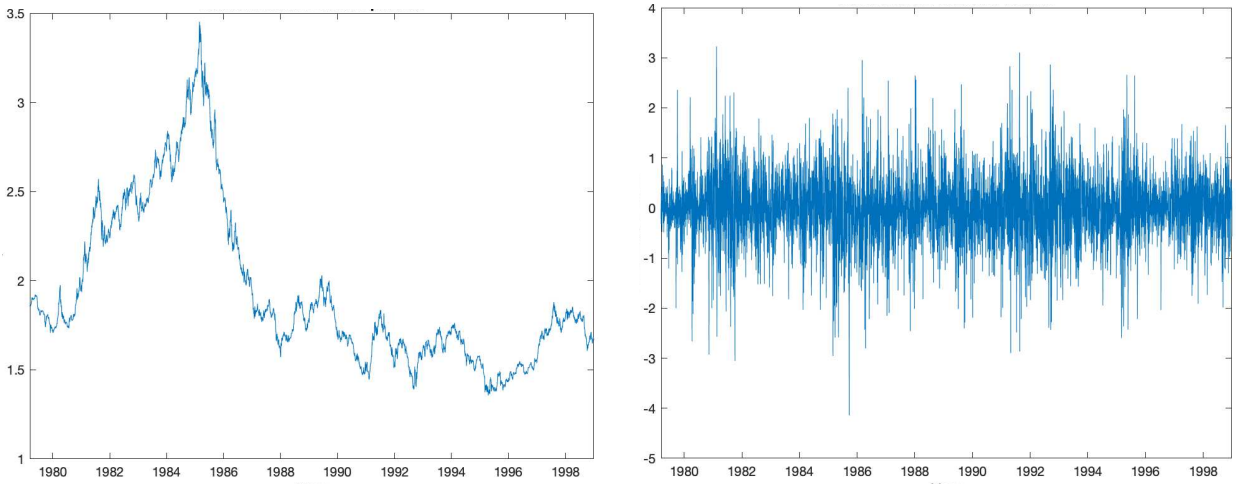
Notes: Simulated bias and Monte Carlo standard error (MCSE) of the parameters of a FIGARCH$(1, d, 1)$ model for different sample sizes and truncation numbers.

The Monte Carlo simulation results can be found in Table 2. For each sample size and truncation number, we report the bias and Monte Carlo standard error (MCSE) for the four parameters.

We can see from Table 2 that for any sample size, the bias of the estimated coefficients decreases as the truncation number increases. This is expected since a larger truncation number implies a smaller approximation. The parameters d and β seem particularly affected by the truncation. The biases in these estimators are about 0.10 to 0.15 for $n = 100$ and about 0.012 to 0.025 when $n = 1,000$, whereas the untruncated estimators are essentially unbiased (the largest bias for $n = T - 1$ in the table is only 0.0036 in absolute value).

The biases when $n = 100$ are obviously very substantial, and such a small truncation number is rarely used in practice, except maybe for some bootstrap or simulation purposes. It is clear from Table 2 that using $n = 100$ is a very bad idea. Although the bias when $n = 1,000$ is much smaller than when $n = 100$, it is still substantial in the former case. In particular, it is likely several parameter standard errors in magnitude. Moreover, the bias does not diminish when the sample size increases. In fact, it doubles from $T = 25,000$ to

Figure 2: Daily DEM-USD exchange rate 3/15/1979–12/31/1998



(a) spot rate (b) return

$T = 50,000$ for the estimators of both d and β.

In conclusion, the results of our small Monte Carlo simulation in Table 2 clearly illustrate the consequences of the approximation implied by truncating the convolution as in (4). The resulting bias in the estimators is substantial, and it is worse for larger sample sizes. Importantly, however, this bias is easily avoidable by application of our proposed FFT-based algorithm, which is exact and does not rely on any approximation or truncation.

5 Empirical illustrations

The presence of persistence in the volatility of nominal exchange rates has been well-documented in the literature, and exchange rates are often modeled using FIGARCH-type models. Indeed, a daily time series of Deutsche Mark-US dollar (DEM-USD) exchange rates was used as the empirical example in Baillie et al. (1996). An updated data set, covering until the end of the DEM era and start of the Euro, was subsequently analyzed in Baillie, Cecen, and Han (2000). In our first empirical illustration, we consider this data set. As a second empirical illustration, we consider a longer time series of daily US Dollar-British Pound (USD-GBP) exchange rates.

5.1 Deutsche Mark-US Dollar exchange rate from 1979 to 1998

In this subsection, we analyze the daily DEM-USD exchange rate from March 14, 1979 to December 31, 1998, for a total of $4,974$ spot rate observations.[4] In Figure 2(a) we plot

[4]The Euro was introduced on January 1, 1999. Our data set was downloaded from the Federal Reserve H.10 historical data website, and covers the same time period as that analyzed in Baillie et al. (2000), but has a slightly different number of observations.

Table 3: FIGARCH models for DEM-USD exchange rate (1979–1998)

	$n = 100$		$n = 1,000$		$n = T - 1$	
(p, d, q)	$(1, d, 1)$	$(1, d, 0)$	$(1, d, 1)$	$(1, d, 0)$	$(1, d, 1)$	$(1, d, 0)$
μ	0.0034	0.0033	0.0037	0.0033	0.0039	0.0035
	(0.0086)	(0.0087)	(0.0086)	(0.0086)	(0.0086)	(0.0086)
ω	0.0318	0.0957	0.0190	0.0613	0.0162	0.0511
	(0.0035)	(0.0078)	(0.0043)	(0.0090)	(0.0047)	(0.0100)
β	0.6467	0.2224	0.6367	0.2026	0.6499	0.2130
	(0.0149)	(0.0231)	(0.0143)	(0.0224)	(0.0143)	(0.0221)
ϕ	0.2792	—	0.2882	—	0.2660	—
	(0.0203)		(0.0191)		(0.0187)	
d	0.4340	0.2904	0.4230	0.2741	0.4586	0.2854
	(0.0166)	(0.0155)	(0.0150)	(0.0142)	(0.0150)	(0.0136)
b_3	−0.10	−0.10	−0.11	−0.10	−0.11	−0.10
b_4	4.44	4.45	4.59	4.57	4.64	4.63
$Q(20)$	53.58	51.28	53.86	51.62	54.64	52.50
$Q^2(20)$	7.98	29.63	11.25	29.81	12.06	31.37
$\log L$	−4,867	−4,890	−4,867	−4,886	−4,870	−4,889
Time FFT	218	277	308	268	319	362
Time LC	163	219	304	285	1,617	1,422

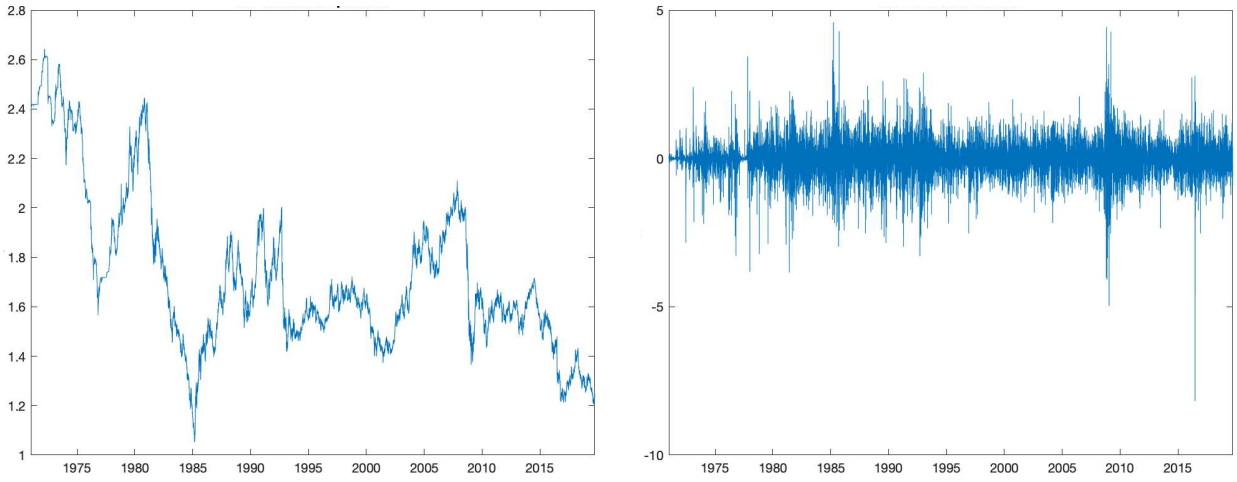
Notes: Quasi-maximum-likelihood estimates for FIGARCH(p, d, q) models for the DEM-USD percentage returns from March 15, 1979 to December 31, 1998 for different truncation numbers and $T = 4,973$ observations. Standard errors are reported in parentheses, and b_3, b_4 are the skewness and kurtosis of the standardized residuals, respectively. The Ljung-Box portmanteau tests for up to 20th-order serial correlation are reported for the standardized residuals, $Q(20)$, and the squared standardized residuals, $Q^2(20)$. Finally, $\log L$ is the log-likelihood value at the maximum, and the computing time is reported in milliseconds using the FFT-based method and the linear convolution (LC) method.

the spot rate and in Figure 2(b) we plot the continuously compounded percentage returns. It is clear from the figure that the spot exchange rate is nonstationary. Indeed, standard arbitrage arguments would dictate that the spot rate should be a Martingale, which justifies modeling the returns as serially uncorrelated as in (1).

We estimate two different FIGARCH(p, d, q) specifications, namely $(1, d, 1)$ and $(1, d, 0)$, for different truncation numbers. The results for these models can be found in Table 3. For all models, the estimates are obtained as in (10) using the relevant conditional variance specification and $\epsilon_t = r_t - \mu$, where r_t is the observed return series and the parameter μ is the mean return (which is included in the vector θ).

Table 3 reports the quasi-maximum-likelihood estimates for the FIGARCH$(1, d, 1)$ and FIGARCH$(1, d, 0)$ models and truncation numbers $n \in \{100; 1,000; t-1\}$. Several interesting

Figure 3: Daily USD-GBP exchange rate 1/5/1971–9/27/2019



(a) spot rate (b) return

findings appear from these results. First of all, regardless of the truncation number, the additional parameter in the $(1, d, 1)$ specification compared with the $(1, d, 0)$ specification is highly significant, whether measured via a t-test or likelihood ratio test. Thus, we consider mainly the FIGARCH$(1, d, 1)$ model. Secondly, for this model, we note that the parameter estimates for d do vary with the truncation number. In particular, d changes from 0.43 to 0.46 for $n = 100$ to $n = T - 1$, which is about two standard errors for this parameter. Of course, this is expected based on the simulation results in Section 4. Thirdly, the computing times for the FFT-based method and LC method are about the same with $n = 100$ and $n = 1,000$, but without truncation ($n = T - 1$) the computing time for our FFT-based method is about five times faster than for the LC method.

5.2 US Dollar-British Pound exchange rate from 1971 to 2019

In this subsection, we analyze the daily USD-GBP spot exchange rate from January 4, 1971 to September 27, 2019, for a total of 12,229 observations. The data series was downloaded from the FRED database and is plotted in Figure 3. As in the previous subsection, the spot exchange rate is nonstationary and we model the continuously compounded returns.

The estimation results for the USD-GBP data set are presented in Table 4, which is laid out precisely as Table 3. The results are qualitatively quite similar, except they are more pronounced in Table 4 because of the longer sample.

First of all, the FIGARCH$(1, d, 1)$ specification is again statistically much superior to the FIGARCH$(1, d, 0)$ specification, regardless of truncation number. Secondly, the estimates for several parameters are now dramatically different with and without truncation. With truncation at $n = 100$, the difference in most parameter estimates are many standard errors

Table 4: FIGARCH models for USD-GBP exchange rate (1971–2019)

	$n = 100$		$n = 1,000$		$n = T - 1$	
(p, d, q)	$(1, d, 1)$	$(1, d, 0)$	$(1, d, 1)$	$(1, d, 0)$	$(1, d, 1)$	$(1, d, 0)$
μ	0.0010	0.0022	0.0034	0.0038	0.0046	0.0049
	(0.0049)	(0.0044)	(0.0056)	(0.0050)	(0.0066)	(0.0059)
ω	0.0115	0.0374	0.0077	0.0211	0.0058	0.0137
	(0.0033)	(0.0064)	(0.0033)	(0.0080)	(0.0038)	(0.0107)
β	0.7131	0.3430	0.6300	0.2699	0.5956	0.2458
	(0.0155)	(0.0159)	(0.0139)	(0.0161)	(0.0142)	(0.0172)
ϕ	0.2076	—	0.2753	—	0.2755	—
	(0.0148)		(0.0163)		(0.0180)	
d	0.5844	0.4028	0.4433	0.3362	0.4124	0.3183
	(0.0135)	(0.0108)	(0.0191)	(0.0133)	(0.0228)	(0.0160)
b_3	−0.12	−0.14	−0.15	−0.15	−0.15	−0.14
b_4	9.56	8.00	10.95	9.30	12.32	10.89
$Q(20)$	67.47	68.29	68.96	71.15	72.66	75.92
$Q^2(20)$	12.31	25.83	9.32	18.76	8.97	15.71
$\log L$	−9,584	−9,663	−9,541	−9,594	−9,528	−9,570
Time FFT	390	319	459	429	926	711
Time LC	251	272	601	500	13,546	9,604

Notes: Quasi-maximum-likelihood estimates for FIGARCH(p, d, q) models for the USD-GBP percentage returns from January 5, 1971 to September 27, 2019 for different truncation numbers and $T = 12,228$ observations. Standard errors are reported in parentheses, and b_3, b_4 are the skewness and kurtosis of the standardized residuals, respectively. The Ljung-Box portmanteau tests for up to 20th-order serial correlation are reported for the standardized residuals, $Q(20)$, and the squared standardized residuals, $Q^2(20)$. Finally, $\log L$ is the log-likelihood value at the maximum, and the computing time is reported in milliseconds using the FFT-based method and the linear convolution (LC) method.

compared with no truncation. Even comparing truncation at $n = 1,000$, which is standard in the literature, with no truncation, we find that the estimates of d and β both differ by 1–2 standard errors. The parameters μ and ω, which are often not of primary importance, differ even more. Again, these results are expected based on the simulation findings in Section 4.

For the computing time using the FFT-based method and the LC method, we find that the FFT-based method is now somewhat faster than the LC method even with truncation at $n = 1,000$. Without truncation, the FFT-based method is about 14 times faster than the LC method. As an order of magnitude, 999 bootstrap or Monte Carlo replications for the FIGARCH$(1, d, 1)$ model would require nearly 4 hours for the LC method and only 15 minutes for our FFT-based algorithm. This result confirms those found in Section 3, and could certainly be the difference between bootstrap inference being feasible or not.

6 Conclusion

In this paper we have considered the computation of the time series of conditional variances for models in the ARCH(∞) class. This class is very large and contains many commonly applied models in empirical finance such as the FIGARCH model. For models in the ARCH(∞) class, we have provided an exact algorithm for efficient computation of the conditional variances, and hence of the likelihood function. Our algorithm is based on the fast Fourier transform (FFT) and achieves an order of magnitude improvement in computational speed from $O(T^2)$ to $O(T \log T)$. The efficiency of the algorithm allows estimation, as well as simulation and/or bootstrapping of ARCH(∞) models, even with very large data sets.

Furthermore, to speed up computation, it has been completely standard in the literature to resort to a truncation of the ARCH(∞) model at a fixed truncation lag. With our proposed algorithm, this is not needed. As a second contribution, we also showed that the elimination of the truncation substantially reduces the bias of the quasi-maximum-likelihood estimators.

We illustrated our results with two empirical examples that highlighted both the differences in the computational speed and in the parameter estimates for our FFT-based algorithm and the standard algorithm with and without truncation.

As sample sizes in economics and finance have become longer in recent years, efficient computation has become increasingly more important. As we have shown, this is certainly true for daily time series, and when the models discussed in this paper are applied to high-frequency data this becomes crucially important. Furthermore, the gaining popularity of computationally intensive methods of inference, based on simulation, bootstrapping, or machine-learning techniques, that require estimation of (variations of) each model a large number of times, emphasizes these points even more.

References

Baillie, R. T., T. Bollerslev, and H. O. Mikkelsen (1996). Fractionally integrated generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics 74*, 3–30.

Baillie, R. T., A. Cecen, and Y. W. Han (2000). High frequency Deutsche Mark-US Dollar returns: FIGARCH representations and non linearities. *Multinational Finance Journal 4*, 247–267.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics 31*, 307–327.

Bollerslev, T. and H. O. Mikkelsen (1996). Modeling and pricing long memory in stock market volatility. *Journal of Econometrics 73*, 151–184.

Cooley, J. W., P. A. W. Lewis, and P. D. Welch (1969). The fast Fourier transform and its applications. *IEEE Transactions on Education 12*, 27–34.

Cooley, J. W. and J. W. Tukey (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation 19*, 297–301.

Davidson, J. (2004). Moment and memory properties of linear conditional heteroscedasticity models, and a new model. *Journal of Business & Economic Statistics 22*, 16–29.

Giraitis, L., P. M. Robinson, and D. Surgailis (2000). A model for long memory conditional heteroscedasticity. *Annals of Applied Probability 10*, 1002–1024.

Jensen, A. N. and M. Ø. Nielsen (2014). A fast fractional difference algorithm. *Journal of Time Series Analysis 35*, 428–436.

Karanasos, M., Z. Psaradakis, and M. Sola (2004). On the autocorrelation properties of long-memory GARCH processes. *Journal of Time Series Analysis 25*, 265–282.

Klein, T. and T. Walther (2017). Fast fractional differencing in modeling long memory of conditional variance for high-frequency data. *Finance Research Letters 22*, 274–279.

Robinson, P. M. (1991). Testing for strong serial correlation and dynamic conditional heteroskedasticity in multiple regression. *Journal of Econometrics 47*, 67–84.

Stockham, T. G. (1966). High-speed convolution and correlation. *Proceedings of the Spring Joint Computer Conference 28*, 229–233.

Tse, Y. K. (1998). The conditional heteroscedasticity of the yen-dollar exchange rate. *Journal of Applied Econometrics 13*, 49–55.

Zygmund, A. (2003). *Trigonometric Series, vol. I and II, 3rd rev. ed.* Cambridge, UK: Cambridge University Press.