A FAST ALGORITHM FOR MAXIMUM LIKELIHOOD-BASED FUNDAMENTAL FREQUENCY ESTIMATION

Jesper Kjær Nielsen[1,2], Tobias Lindstrøm Jensen[1], Jesper Rindom Jensen[3], Mads Græsbøll Christensen[3], and Søren Holdt Jensen[1]

[1]Aalborg University, Denmark
Dept. of Electronic Systems
{jkn,tlj,shj}@es.aau.dk

[2]Bang & Olufsen A/S
Struer, Denmark

[3]Aalborg University, Denmark
Audio Analysis Lab, AD:MT
{jrj,mgc}@create.aau.dk

ABSTRACT

Periodic signals are encountered in many applications. Such signals can be modelled by a weighted sum of sinusoidal components whose frequencies are integer multiples of a fundamental frequency. Given a data set, the fundamental frequency can be estimated in many ways including a maximum likelihood (ML) approach. Unfortunately, the ML estimator has a very high computational complexity, and the more inaccurate, but faster correlation-based estimators are therefore often used instead. In this paper, we propose a fast algorithm for the evaluation of the ML cost function for complex-valued data over all frequencies on a Fourier grid and up to a maximum model order. The proposed algorithm significantly reduces the computational complexity to a level not far from the complexity of the popular harmonic summation method which is an approximate ML estimator.

Index Terms— Fundamental frequency estimation, Levinson algorithm, Durbin algorithm, non-linear least squares, fast implementation, MATLAB, C++

1. INTRODUCTION

Many real-world signals are either periodic or approximately so, and they can, therefore, be modelled by a weighted sum of sinusoids whose frequencies are integer multiples of a fundamental frequency. Such periodic signals are encountered in a wide range of applications such as music and speech processing [1, 2], sonar [3, 4], and electrocardiography [5]. Although the signals in these applications are all real-valued, these are often transformed into down-sampled analytic signals [6] for two reasons. First, the complex-valued representation is easier to work with from an analytical point of view, and, second, the complex-valued representation allows for computationally more efficient algorithms as only half as many samples and sinusoids are in the model [7, 8]. In this paper, we add further weight to the latter argument.

The complex-valued signal model for a periodic signal in additive noise $e(n)$ is given by

$$x(n) = \sum_{i=1}^{l} \alpha_i \exp(j\omega_0 i n) + e(n) \tag{1}$$

where $j = \sqrt{-1}$, α_i, and ω_0 are the imaginary unit, the complex amplitude of the i'th harmonic component, and the fundamental frequency in radians/sample, respectively. If we assume a finite set of N data points $\{x(n)\}_{n=n_0}^{n_0+N-1}$ are observed, the signal model (1) can be written on the vector form

$$\boldsymbol{x} = \boldsymbol{Z}_l(\omega_0)\boldsymbol{\alpha}_l + \boldsymbol{e} \tag{2}$$

where we have defined

$$\boldsymbol{x} = \begin{bmatrix} x(n_0) & \cdots & x(n_0+N-1) \end{bmatrix}^T \tag{3}$$

$$\boldsymbol{e} = \begin{bmatrix} e(n_0) & \cdots & e(n_0+N-1) \end{bmatrix}^T \tag{4}$$

$$\boldsymbol{z}(\omega) = \begin{bmatrix} \exp(j\omega n_0) & \cdots & \exp(j\omega(n_0+N-1)) \end{bmatrix}^T \tag{5}$$

$$\boldsymbol{Z}_l(\omega_0) = \begin{bmatrix} \boldsymbol{z}(\omega_0) & \cdots & \boldsymbol{z}(l\omega_0) \end{bmatrix} \tag{6}$$

$$\boldsymbol{\alpha}_l = \begin{bmatrix} \alpha_1 & \cdots & \alpha_l \end{bmatrix}^T. \tag{7}$$

The start index n_0 can be selected in an arbitrary way and is usually set to $n_0 = 0$. As we show in this paper, however, it is advantageous to select $n_0 = -(N-1)/2$ to lower the computational complexity.

Numerous fundamental frequency estimation methods have been proposed in the scientific literature. These range from simple correlation-based estimators [9] to high resolution parametric estimators [8]. In particular the non-linear least squares (NLS) estimator have proven to work very well in practice, but also the harmonic MUSIC estimator has been demonstrated to perform well [8, 10]. Provided that the noise term $\boldsymbol{e}$ is white and Gaussian, the maximum likelihood estimator is given by (see, e.g., [11, pp. 157–158])

$$\hat{\omega}_0 = \underset{\omega_0}{\operatorname{argmax}}\, \boldsymbol{x}^H \boldsymbol{P}_l(\omega_0)\boldsymbol{x} \tag{8}$$

The work by J.K. Nielsen was supported by the Danish Innovations-Fonden. The work by T.L. Jensen and J.R. Jensen was partly supported by the Independent Research Council for Technology and Production 4005-00122 and 1337-00084, respectively. The work by M.G. Christensen was supported by the Villum Foundation

where

$$\boldsymbol{P}_l(\omega_0) = \boldsymbol{Z}_l(\omega_0)\left[\boldsymbol{Z}_l^H(\omega_0)\boldsymbol{Z}_l(\omega_0)\right]^{-1}\boldsymbol{Z}_l^H(\omega_0) \quad (9)$$

is the projection matrix for the column space of $\boldsymbol{Z}_l(\omega_0)$. If the assumption on the noise is not fulfilled, then the estimator is known as the NLS estimator. Since the maximum likelihood estimator is a special case of the NLS estimator, we will use the term NLS estimator in the rest of the paper. Similarly, the harmonic MUSIC estimator is the solution to

$$\hat{\omega}_0 = \underset{\omega_0}{\operatorname{argmax}} \sum_{i=1}^{l} \boldsymbol{s}_i^H \boldsymbol{P}_l(\omega_0)\boldsymbol{s}_i \quad (10)$$

where $\{\boldsymbol{s}_i\}_{i=1}^l$ are l data-dependent vectors spanning the so-called signal subspace [12]. Other fundamental frequency estimators, however, involve cost-functions on slightly different forms, such as the filtering methods in [13]. Consequently, the fast implementations herein does not apply directly to these, but similar ideas can be used to obtain fast implementations as explained in [14]. The major disadvantage of the abovementioned estimators are that the cost functions in the above optimisation problems have a very oscillatory behaviour and can therefore only be maximised using a computationally complex search over a fine grid. Consequently, faster, but less accurate methods such as the harmonic summation (HS) method [15] based on the asymptotic result that $\lim_{N\to\infty} N^{-1}\boldsymbol{Z}_l^H(\omega_0)\boldsymbol{Z}_l(\omega_0) = \boldsymbol{I}_l$ are used instead. However, especially for low fundamental frequencies, these methods perform quite poorly [12]. Moreover, for joint fundamental frequency and model order estimation as, e.g., suggested in [16], the value of the exact NLS cost function is required. We note that fundamental frequency estimation is often used in multi fundamental frequency estimation [8].

For fundamental frequency estimation, model order comparison is very important to, e.g., avoid the problem of pitch halving and doubling. Nevertheless, the model order l is often assumed to be known a priori, and we believe that the computational complexity is a major reason for this. For a given model l, the complexity for computing the NLS cost function in (8) for a candidate fundamental frequency is $\mathcal{O}(l^2N)$+$\mathcal{O}(l^3)$. Thus, if the NLS cost function is evaluated naïvely over a grid of F/l candidate fundamental frequencies up to a model order of L, the order of complexity is $\mathcal{O}(F\log F)$+$\mathcal{O}(FL^3)$. To the best of our knowledge, no NLS- or MUSIC-based methods have been published for joint fundamental frequency and model order estimation which has lower than this cubic complexity in L.

In this paper, we propose a fast algorithm for the evaluation of the exact cost function of the NLS method in (8) over the Fourier grid for all candidate model orders up to L. The proposed algorithm has only linear complexity in L as it reduces the complexity from $\mathcal{O}(F\log F)$+$\mathcal{O}(FL^3)$ to just $\mathcal{O}(F\log F)$+$\mathcal{O}(FL)$ which is the same order as the HS method. Moreover, the algorithm can also be vectorised so that the cost function for a candidate model order is computed for all candidate fundamental frequencies at once. This gives a huge boost in speed for a MATLAB implementation. In addition to a MATLAB implementation, we also provide an even faster C++-implementation. The results can also be used in the evaluation of the harmonic MUSIC cost function in (10) which has a very similar structure to the NLS cost function. The huge speed up is obtained by using the particular structure in the complex-valued projection matrix (9) which allow us to use the Levinson algorithm in a clever way. Unfortunately, the same structure does not exist in the real-valued case, but we believe that the results presented here is also a first step in developing a fast algorithm for the real-valued NLS estimator.

2. FAST EVALUATION OF THE NLS COST FUNCTION

In this section, we derive the fast algorithm we advertised in the introduction. We assume that we have to compute the cost function over a uniform grid $\Omega_l = \{2\pi(f-1)/F\}_{f=1}^{\lceil F/l\rceil}$ consisting of $\lceil F/l\rceil$ points for all model orders $l \in \{1,\ldots,L\}$. The upper limit depends on l since $\omega_0 \in (0, 2\pi/l)$ to ensure that $l\omega_0 \in (0, 2\pi)$. Before we present the fast algorithm, however, we first briefly outline the standard algorithm for computing this cost function.

2.1. The Standard Algorithm

The NLS cost function is typically evaluated as outlined in algorithm 1. The cost in line 6 is set to one since $\boldsymbol{b}_l$ is only a subset of the elements in $\boldsymbol{f}$ described via the selection matrix $\boldsymbol{S}_l$. Depending on the size of F and L, the major contributions are either the computations in line 1 or line 7. Since

$$\sum_{l=1}^{L}\sum_{f=1}^{\lceil F/l\rceil} \mathcal{O}(l^3) = \mathcal{O}(FL^3)\,, \quad (11)$$

the time complexity of the standard implementation is therefore $\mathcal{O}(F\log F)$+$\mathcal{O}(FL^3)$. The HS method is algorithm 1 with line 7 approximated as $N\boldsymbol{I}_l\boldsymbol{\alpha}_l = \boldsymbol{b}_l$.

2.2. A Fast Algorithm

We are now ready to present the fast algorithm. Before presenting the details of the algorithm, however, we first highlight the key facts that lead to the algorithm.

1. In order to compute the cost function efficiently, we have to solve line 7 of algorithm 1

$$\boldsymbol{A}_l\boldsymbol{\alpha}_l = \boldsymbol{b}_l \quad (12)$$

for $\boldsymbol{\alpha}_l$ in an efficient way where

$$\boldsymbol{A}_l = \boldsymbol{Z}_l^H(\omega_0)\boldsymbol{Z}_l(\omega_0) \quad (13)$$

$$\boldsymbol{b}_l = \boldsymbol{Z}_l^H(\omega_0)\boldsymbol{x}\,. \quad (14)$$

Algorithm 1 The standard algorithm for computing the NLS cost function for L model orders and F fundamental frequencies on the Fourier grid. The notation $\odot$ and $[\cdot]_{i,k}$ denotes element-wise multiplication and element (i,k), respectively.

1: $\boldsymbol{f} = \text{fft}(\boldsymbol{x})$ ▷ $\mathcal{O}(F \log F)$
2: $[\boldsymbol{J}]_{1,1:F} = N^{-1}(\boldsymbol{f}^* \odot \boldsymbol{f})^T$ ▷ $\mathcal{O}(F)$
3: **for** $l \in 2, 3, \ldots, L$ **do**
4: **for** $f \in \{1, 2, \ldots, \lceil F/l \rceil\}$ **do**
5: $\omega_0 = 2\pi(f-1)/F$ ▷ $\mathcal{O}(1)$
6: $\boldsymbol{b}_l = \boldsymbol{S}_l \boldsymbol{f}$ ▷ $\mathcal{O}(1)$
7: Solve $\boldsymbol{Z}_l^H(\omega_0)\boldsymbol{Z}_l(\omega_0)\boldsymbol{\alpha}_l = \boldsymbol{b}_l$ for $\boldsymbol{\alpha}_l$ ▷ $\mathcal{O}(l^3)$
8: $[\boldsymbol{J}]_{l,f} = \boldsymbol{b}_l^H \boldsymbol{\alpha}_l$ ▷ $\mathcal{O}(l)$

2. The (i,k)'th element of the matrix $\boldsymbol{A}_l$ is given by

$$[\boldsymbol{A}_l]_{i,k} = \boldsymbol{z}^H(i\omega_0)\boldsymbol{z}(k\omega_0) \tag{15}$$

$$= \begin{cases} N & \text{for } i = k \\ \exp(j\omega_0[i-k]\left[n_0 - \frac{N-1}{2}\right]) & \\ \quad \times \frac{\sin(N[i-k]\omega_0/2])}{\sin([i-k]\omega_0/2])} & \text{for } i \neq k \end{cases} \tag{16}$$

3. As seen from (16), the matrix $\boldsymbol{A}_l$ is a Hermitian and *Toeplitz* matrix. Therefore, (12) can be solved efficiently using the *Levinson* algorithm [17]. We note in passing that the Toeplitz structure does not exist in the real-valued case (not even a block-Toeplitz structure).
4. The Levinson algorithm computes the solution to (12) iteratively for all model orders. Thus, we can compute the solution to (12) for all models orders by just running the Levinson algorithm once.
5. As alluded to earlier, the value of the NLS cost function does not depend on how we select the time index n_0. We therefore select it as $n_0 = -(N-1)/2$ since the matrix $\boldsymbol{A}_l$ is then a *real-valued* matrix (see (16)). Thus, the Durbin step inside the Levinson algorithm only involves real-valued algebra.

Although both the Levinson and the Durbin algorithms are well-known, we briefly describe them here for the sake of completeness. See a much more complete treatment in [17].

2.2.1. The Levinson Algorithm

Suppose we have computed the solution to $\boldsymbol{A}_{l-1}\boldsymbol{\alpha}_{l-1} = \boldsymbol{b}_{l-1}$ and wish to solve $\boldsymbol{A}_l\boldsymbol{\alpha}_l = \boldsymbol{b}_l$ which we write as

$$\begin{bmatrix} \boldsymbol{A}_{l-1} & \boldsymbol{J}_{l-1}\boldsymbol{a}_{l-1} \\ \boldsymbol{a}_{l-1}^T\boldsymbol{J}_{l-1} & N \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_l \\ \mu_l \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_{l-1} \\ b_l \end{bmatrix} \tag{17}$$

where $\boldsymbol{J}_{l-1}$ is the $l-1 \times l-1$ exchange matrix and $\boldsymbol{a}_{l-1}$ is a vector containing the first $l-1$ elements of the last column of $\boldsymbol{A}_l$ in reverse order. It is not hard to show that the solution $\boldsymbol{\alpha}_l$ can be computed with a linear complexity as

$$\mu_l = \gamma_l(b_l - \boldsymbol{a}_{l-1}^T\boldsymbol{J}_{l-1}\boldsymbol{\alpha}_{l-1}) \tag{18}$$

$$\boldsymbol{v}_l = \boldsymbol{\alpha}_{l-1} + \mu_l\boldsymbol{J}_{l-1}\boldsymbol{y}_{l-1} \tag{19}$$

$$\boldsymbol{\alpha}_l = \begin{bmatrix} \boldsymbol{v}_l^T & \mu_l \end{bmatrix}^T \tag{20}$$

where γ_l and $\boldsymbol{y}_{l-1}$ are computed in the Durbin algorithm. Note that all operations require complex-valued algebra.

2.2.2. The Real-valued Durbin Algorithm

Suppose we have computed the solution to $\boldsymbol{A}_{l-1}\boldsymbol{y}_{l-1} = -\boldsymbol{a}_{l-1}$. We now wish to solve $\boldsymbol{A}_l\boldsymbol{y}_l = -\boldsymbol{a}_l$ written as

$$\begin{bmatrix} \boldsymbol{A}_{l-1} & \boldsymbol{J}_{l-1}\boldsymbol{a}_{l-1} \\ \boldsymbol{a}_{l-1}^T\boldsymbol{J}_{l-1} & N \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_l \\ \beta_l \end{bmatrix} = -\begin{bmatrix} \boldsymbol{a}_{l-1} \\ a_l \end{bmatrix} \tag{21}$$

where a_l is the lower left element of $\boldsymbol{A}_{l+1}$. As in the Levinson algorithm, the solution $\boldsymbol{y}_l$ can be computed with a linear complexity as

$$\gamma_l = (N + \boldsymbol{a}_{l-1}^T\boldsymbol{y}_{l-1})^{-1} = \gamma_{l-1}(1-\beta_{l-1}^2)^{-1} \tag{22}$$

$$\beta_l = -\gamma_l(a_l + \boldsymbol{a}_{l-1}^T\boldsymbol{J}_{l-1}\boldsymbol{y}_{l-1}) \tag{23}$$

$$\boldsymbol{z}_l = \boldsymbol{y}_{l-1} + \beta_l\boldsymbol{J}_{l-1}\boldsymbol{y}_{l-1} \tag{24}$$

$$\boldsymbol{y}_l = \begin{bmatrix} \boldsymbol{z}_l^T & \beta_l \end{bmatrix}^T . \tag{25}$$

Note that the Durbin step does not depend on the data $\boldsymbol{x}$. The quantities $\boldsymbol{y}_l$ and γ_l can therefore be computed offline for all candidate model orders and fundamental frequencies and be stored in a look-up table in memory. This is also much more memory efficient than storing $\boldsymbol{A}_l^{-1}$ in memory for all candidate model orders and fundamental frequencies.

2.2.3. The Algorithm

By using the Levinson algorithm to solve the problem in (12), we therefore get a significant speed-up in computing the exact NLS cost function for all model orders up to L and for all frequencies on the Fourier grid. The algorithm is outlined in algorithm 2, and its computational complexity can be shown to be $\mathcal{O}(F \log F)$+$\mathcal{O}(FL)$. Moreover, we note the following about the proposed algorithm.

1. The outer for-loop over all the frequencies is used in algorithm 2 to keep a simpler notation. In practice, everything inside the outer loop can be vectorised, and this is done in our MATLAB implementation.
2. In the algorithm, the fundamental frequency is running in the range $(0, 2\pi/l)$. However, in both our MATLAB and C++-implementations, a lower and an upper bound can be specified inside this interval, and the cost function is not calculated outside these bounds.
3. In line 12 of the algorithm, a complex exponential p_l is multiplied onto the DFT element computed in line 1 to compensate for the start index $n_0 = -(N-1)/2$ compared to a standard FFT algorithm using $n_0 = 0$.

Algorithm 2 A fast algorithm for computing the NLS cost function for L model orders and F fundamental frequencies on the Fourier grid. The notation $\odot$ and $[\cdot]_{i,k}$ denotes element-wise multiplication and element (i,k), respectively.

1: $\boldsymbol{f} = \text{fft}(\boldsymbol{x})$ ▷ $\mathcal{O}(F \log F)$
2: $[\boldsymbol{J}]_{1,1:F} = N^{-1}(\boldsymbol{f}^* \odot \boldsymbol{f})^T$ ▷ $\mathcal{O}(F)$
3: **for** $f \in \{2, 3, \ldots, F\}$ **do**
4: $\quad \omega_0 = 2\pi(f-1)/F$ ▷ $\mathcal{O}(1)$
5: $\quad \boldsymbol{a}_1 = \sin(N\omega_0/2)/\sin(\omega_0/2)$ ▷ $\mathcal{O}(1)$
6: $\quad \gamma_1 = N^{-1}$ ▷ $\mathcal{O}(1)$
7: $\quad \beta_1 = \boldsymbol{y}_1 = -\gamma_1 \boldsymbol{a}_1$ ▷ $\mathcal{O}(1)$
8: $\quad \boldsymbol{b}_1 = [\boldsymbol{f}]_f \exp(j\omega_0 \frac{N-1}{2})$ ▷ $\mathcal{O}(1)$
9: $\quad \boldsymbol{\alpha}_1 = \gamma_1 \boldsymbol{b}_1$ ▷ $\mathcal{O}(1)$
10: $\quad$ **for** $l \in 2, 3, \ldots, L$ **do**
11: $\quad\quad$ **if** $f < \lceil F/l \rceil$ **then** ▷ Ensure that $\omega_0 \in (0, 2\pi/l)$
12: $\quad\quad\quad \boldsymbol{b}_l = \begin{bmatrix} \boldsymbol{b}_{l-1} \\ [\boldsymbol{f}]_{l(f-1)+1} \exp\left(j\omega_0 \frac{l(N-1)}{2}\right) \end{bmatrix}$ ▷ $\mathcal{O}(1)$
13: $\quad\quad\quad \gamma_l = \gamma_{l-1}(1 - \beta_{l-1}^2)^{-1}$ ▷ $\mathcal{O}(1)$
14: $\quad\quad\quad \mu_l = \gamma_l \left([\boldsymbol{b}_l]_l - \boldsymbol{a}_{l-1}^T \boldsymbol{J}_{l-1} \boldsymbol{\alpha}_{l-1}\right)$ ▷ $\mathcal{O}(l)$
15: $\quad\quad\quad \boldsymbol{v}_l = \boldsymbol{\alpha}_{l-1} + \mu_l \boldsymbol{J}_{l-1} \boldsymbol{y}_{l-1}$ ▷ $\mathcal{O}(l)$
16: $\quad\quad\quad \boldsymbol{\alpha}_l = \begin{bmatrix} \boldsymbol{v}_l^T & \mu_l \end{bmatrix}^T$ ▷ $\mathcal{O}(1)$
17: $\quad\quad\quad [\boldsymbol{J}]_{l,f} = \boldsymbol{b}_l^H \boldsymbol{\alpha}_l$ ▷ $\mathcal{O}(l)$
18: $\quad\quad\quad$ **if** $l < L$ **then**
19: $\quad\quad\quad\quad \boldsymbol{a}_l = \begin{bmatrix} \boldsymbol{a}_{l-1} \\ \sin(Nl\omega_0/2)/\sin(l\omega_0/2) \end{bmatrix}$ ▷ $\mathcal{O}(1)$
20: $\quad\quad\quad\quad \beta_l = -\gamma_l \left([\boldsymbol{a}_l]_l + \boldsymbol{a}_{l-1}^T \boldsymbol{J}_{l-1} \boldsymbol{y}_{l-1}\right)$ ▷ $\mathcal{O}(l)$
21: $\quad\quad\quad\quad \boldsymbol{z}_l = \boldsymbol{y}_{l-1} + \beta_l \boldsymbol{J}_{l-1} \boldsymbol{y}_{l-1}$ ▷ $\mathcal{O}(l)$
22: $\quad\quad\quad\quad \boldsymbol{y}_l = \begin{bmatrix} \boldsymbol{z}_l^T & \beta_l \end{bmatrix}^T$ ▷ $\mathcal{O}(1)$

4. Line 1, 2, 9, 8, 12, 14, 15, and 17 involve complex-valued algebra. The remaining lines only involve real-valued algebra.
5. Although the algorithm is presented for a uniform grid of frequencies, it can easily be modified to compute the value of the NLS cost function for just a single and arbitrary frequency and model order. However, the computational saving are only in the order of L (instead of L^2) in this case since we cannot exploit that the solutions to all models orders up to a certain candidate order are computed in the same recursion.

3. RESULTS

In this section, we evaluate the computational saving of our proposed algorithm. Specifically, we compare the computation speed of computing the exact NLS cost function to the HS method. As described earlier, the latter is a very fast, but approximate way of computing the NLS cost function. As we show via the example in Fig. 1, the approximation is good unless the fundamental frequency is low.

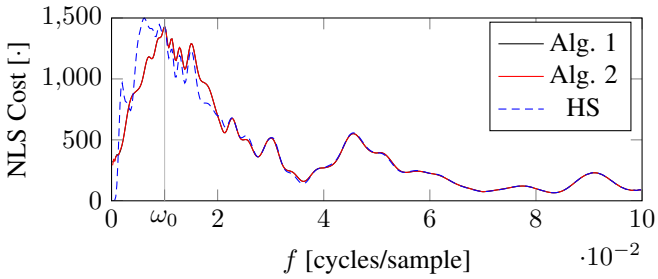

Fig. 1. An example of the exact (for both alg. 1 and 2) and HS cost functions for $N = 100$ and $L = 10$.

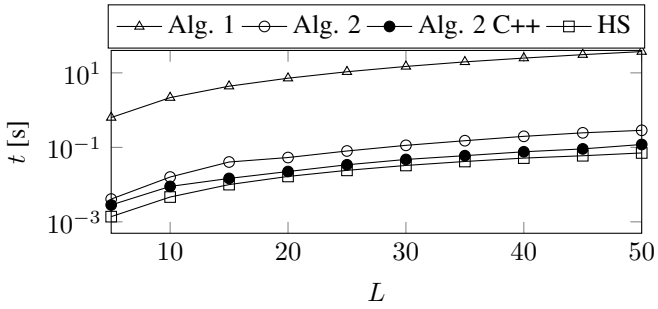

Fig. 2. The computation time for four different ways of computing either the exact or the approximate NLS cost function for $N = 200$ and $F = 5NL$.

The algorithms have been implemented in MATLAB and line 2-25 of algorithm 2 have also been implemented in C++ with a MATLAB interface using mex. Specifically, the C++ implementation uses BLAS level 1 from the MATLAB library `lwblas` and element-wise operations are implemented as simple for-loops. To compare the computation speed of our implementations, we run a benchmark procedure. Timings are obtained by computing 10^i solutions for the smallest $i \in \{0, 1, 2, \cdots\}$ such that the execution time $t_0 \geq 0.2\,\text{s}$ and then reporting the minimum (average) time $t = \min(t_1, t_2, t_3)/10^i$ over 3 repetitions (same as iPython's magic function `timeit` [18]). All timings are executed on an Intel(R) Dual Core(TM) i5-2410M CPU at 2.3 GHz with Ubuntu Linux kernel 3.13.0-24-generic and Matlab 8.4.0. All these implementations and the code for generating the results presented here are available from `http://kom.aau.dk/~jkn/publications/publications.php`.

In Fig. 2, we have compared the computation speed for $N = 200$ data points and a variable model order from $L = 5$ to $L = 50$. The number of grid points F is set to $5NL$ since we have found that this resolution is sufficient not to miss the true peak in the cost function. The figure clearly shows that both the MATLAB and the C++-implementations of algorithm 2 are much faster than the implementation of algorithm 1. Moreover and more interestingly, the MATLAB and the C++-implementations are only approximately 1.5 and 4 times slower, respectively, than the MATLAB implementa-

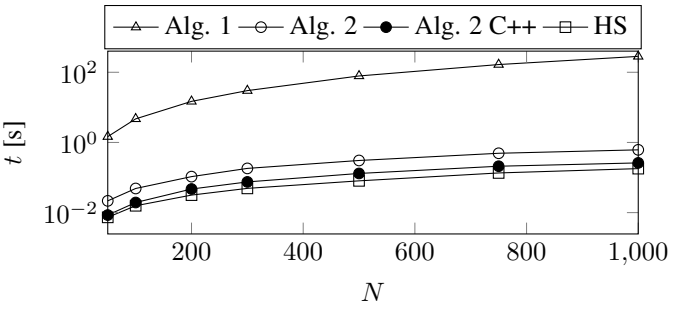

Fig. 3. The computation time for four different ways of computing either the exact or the approximate NLS cost function for $L = 30$ and $F = 5NL$.

tion of the HS method whereas the standard implementation is approximately 450 times slower. The same trend is shown in Fig. 3. Here, the model order is fixed to $L = 30$ and the number of data points is varied from $N = 50$ to 1000. Again, the MATLAB and the C++-implementations of algorithm 2 are much faster than the implementation of algorithm 1 and only slightly slower than the implementation of the HS method. Although not shown here, we have observed the same trend when we fixed the ratio between N and L and varied N.

4. CONCLUSION

For the problem of joint fundamental frequency and model order estimation as suggested in [16], the main computational cost is the evaluation of the non-linear least squares (NLS) cost function over a Fourier grid of candidate frequencies for all model orders up to a maximum model order. In this paper, we have proposed a new and fast algorithm based on the Levinson algorithm for evaluating this cost function from complex-valued data. The proposed algorithm reduces the computational complexity from $\mathcal{O}(F \log F)$+$\mathcal{O}(FL^3)$ to just $\mathcal{O}(F \log F)$+$\mathcal{O}(FL)$ where F is the number of points in the Fourier grid and L is the maximum number of sinusoidal components. Via simulations, we have shown that both a MATLAB and a C++-implementation of the proposed algorithm reduces the computation time to a level not far from the complexity of the popular harmonic summation method which is an approximate NLS estimator. Moreover, the reduction in computation time also makes the exact NLS estimator much more attractive in comparison to the inaccurate, but fast and popular correlation-based methods.

5. REFERENCES

[1] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*, 2nd ed. Springer, Jun. 1998.

[2] R. J. Sluijter, "The development of speech coding and the first standard coder for public mobile telephony," Ph.D. dissertation, Techniche Universiteit Eindhoven, 2005.

[3] G. Ogden, L. Zurk, M. Siderius, E. Sorensen, J. Meyers, S. Matzner, and M. Jones, "Frequency domain tracking of passive vessel harmonics," *J. Acoust. Soc. Am.*, vol. 126, no. 4, pp. 2249–2249, 2009.

[4] G. L. Ogden, L. M. Zurk, M. E. Jones, and M. E. Peterson, "Extraction of small boat harmonic signatures from passive sonar," *J. Acoust. Soc. Am.*, vol. 129, no. 6, pp. 3768–3776, Jun. 2011.

[5] V. K. Murthy, L. J. Haywood, J. Richardson, R. Kalaba, S. Salzberg, G. Harvey, and D. Vereeke, "Analysis of power spectral densities of electrocardiograms," *Mathematical Biosciences*, vol. 12, no. 1–2, pp. 41–51, Oct. 1971.

[6] S. L. Marple, Jr., "Computing the discrete-time "analytic" signal via FFT," *IEEE Trans. Signal Process.*, vol. 47, no. 9, pp. 2600–2603, Sep. 1999.

[7] M. G. Christensen, A. Jakobsson, and S. H. Jensen, "Joint high-resolution fundamental frequency and order estimation," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 5, pp. 1635–1644, Jul. 2007.

[8] M. G. Christensen and A. Jakobsson, *Multi-Pitch Estimation*, B. H. Juang, Ed. San Rafael, CA, USA: Morgan & Claypool, 2009.

[9] L. R. Rabiner, "On the use of autocorrelation analysis for pitch detection," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 1, pp. 24–33, Feb. 1977.

[10] J. Tabrikian, S. Dubnov, and Y. Dickalov, "Maximum a-posteriori probability pitch tracking in noisy environments using harmonic model," *IEEE Trans. Speech Audio Process.*, vol. 12, no. 1, pp. 76–87, 2004.

[11] P. Stoica and R. L. Moses, *Spectral Analysis of Signals*. Englewood Cliffs, NJ, USA: Prentice Hall, May 2005.

[12] M. G. Christensen, "Accurate estimation of low fundamental frequencies from real-valued measurements," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 10, pp. 2042–2056, 2013.

[13] M. G. Christensen, J. H. Jensen, A. Jakobsson, and S. H. Jensen, "On optimal filter designs for fundamental frequency estimation," *IEEE Signal Process. Lett.*, vol. 15, pp. 745–748, 2008.

[14] J. R. Jensen, G.-O. Glentis, M. G. Christensen, A. Jakobsson, and S. H. Jensen, "Fast LCMV-based methods for fundamental frequency estimation," *IEEE Trans. Signal Process.*, vol. 61, no. 12, pp. 3159–3172, Jun. 2013.

[15] A. M. Noll, "Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate," in *Proc. of the symposium on computer process. commun.*, vol. 779, 1969.

[16] J. K. Nielsen, M. G. Christensen, and S. H. Jensen, "Default Bayesian estimation of the fundamental frequency," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 3, pp. 598–610, Mar. 2013.

[17] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. The Johns Hopkins University Press, Oct. 1996.

[18] *IPython Documentation*, The IPython Development Team, 2014, v. 1.2.1, http://ipython.org/ipython-doc/1/index.html.