FFT formulations of adaptive Fourier decomposition

Article · April 2017

DOI: 10.1016/j.cam.2017.04.029

4 authors:

Gao You
University of Macau
4 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Min Ku
Tsinghua University
33 PUBLICATIONS **183** CITATIONS
SEE PROFILE

Tao Qian
University of Macau
198 PUBLICATIONS **1,756** CITATIONS
SEE PROFILE

Jianzhong Wang
Sam Houston State University
117 PUBLICATIONS **1,420** CITATIONS
SEE PROFILE

Journal of Computational and Applied Mathematics 324 (2017) 204–215

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

FFT formulations of adaptive Fourier decomposition

You Gao [a,*], Min Ku [b], Tao Qian [a], Jianzhong Wang [c]

[a] Department of Mathematics, University of Macau, Macao

[b] CIDMA, Department of Mathematics, University of Aveiro, Portugal

[c] Department of Mathematics and Statistics, Sam Houston State University, United States

ARTICLE INFO

Article history:
Received 22 December 2016

MSC:
42A50
32A30
32A35
46J15

Keywords:
Fast Fourier transform
Computational complexity
Adaptive decomposition
Greedy algorithm
Reproducing kernel Hilbert space

ABSTRACT

Adaptive Fourier decomposition (AFD) has been found to be among the most effective greedy algorithms. AFD shows an outstanding performance in signal analysis and system identification. As compensation of effectiveness, the computation complexity is great, that is especially due to maximal selections of the parameters. In this paper, we explore the discretization of the 1-D AFD integration via with discrete Fourier transform (DFT), incorporating fast Fourier transform (FFT). We show that the new algorithm, called FFT-AFD, reduces the computational complexity from $\mathcal{O}(MN^2)$ to $\mathcal{O}(MN\log N)$, the latter being the same as FFT. Through experiments, we verify the effectiveness, accuracy, and robustness of the proposed algorithm. The proposed FFT-based algorithm for AFD lays a foundation for its practical applications.

1. Introduction

Adaptive Fourier Decomposition, being abbreviated as AFD, has recently been developed and proved to be among the most effective greedy algorithms [1,2]. AFD is originally developed for the contexts of the unit disc and of the upper-half plane, being precisely called 1-D AFD, or Core-AFD. AFD was originated with the purpose of positive frequency decomposition of signals. The method is based on complex analysis of the unit disc, and of the upper half plane, and in particular related to Möbius transformations and Blaschke products of those contexts. A very closely related functional approximation method, called unwinding Fourier expansion with the positive frequency nature as well, being promoted by a recent paper of Coifman and Steinerberger based on the Nevanlinna factorization of the Hardy space functions, was found extremely effective in signal analysis practice, see [3–8]. AFD is sometimes also called Core-AFD, being due to the reason that it is the constructive block of the lately developed algorithms of a few variations of 1-D AFD, including Unwinding AFD and Cyclic AFD [4,9], etc. Most recently, the concept of AFD is generalized to approximations by linear combinations of shifted Szegö kernels and their derivatives in various contexts, including those by using methods in learning theory, such as SVM [10], and regularization [11], etc. In the present article, we focus on the Core-AFD algorithm, with direct applications to system identification and signal analysis [12–15], as well as being building blocks of the other types of AFD algorithms and greedy algorithms in general. Core-AFD was lately found to correspond to Pre-OGA. In fact, Pre-OGA in the unit disc Hardy H^2 space is identical with Core-AFD. Pre-OGA has been theoretically proved to be more effective than the other known greedy algorithms, including the general greedy and the orthogonal types [16–18]. General greedy algorithms

* Corresponding author.
E-mail addresses: map2gao@gmail.com (Y. Gao), kumin0844@163.com (M. Ku), fsttq@umac.mo (T. Qian), jzwang@shsu.edu (J. Wang).

http://dx.doi.org/10.1016/j.cam.2017.04.029
0377-0427/

are approximations by linear combinations of dictionary elements that are reproducing kernels of the underlying Hilbert space [19]. In contrast, AFDs, or more generally Pre-OGAs, are approximations by linear combinations of reproducing kernels as well as their directional derivatives of arbitrary order, constituting the so-called complete dictionary. It is the restriction of using only an incomplete dictionary that causes the low efficiency of ordinary greedy algorithms. In fact, many functions in the Hardy space have multiple poles, corresponding to directional directives of certain orders, and thus AFD, more generally Pre-OGA, provides more promising approximations.

AFD involves maximal selections of the parameters that cause great computational complexity. In [20] it is shown that the computational complexity of 1-D AFD is $\mathcal{O}(MN^2)$, where N is the number of the discretization points on the unit circle and M is the number of points in [0, 1), on which maximal values are selected. The quantity M cannot be reduced as it is related to the accuracy of the method. It is, however, of a great practical value to have an algorithm with low computational complexity in relation to N, the discretization scale, for AFD. The computation encounters a similar situation as to DFT in the discretization of the integral. As the result of this paper, FFT, as a fast and effective method for DFT, is built into AFD to significantly reduce the computational complexity of 1-D AFD to $\mathcal{O}(MN\log N)$, which is as the same as that of FFT to DFT, indicating that the complexity cannot be further reduced. The same strategy is applicable to Pre-OGA, for general reproducing kernel Hilbert spaces with a mild boundary condition. In particular, in the 1-D classical Hardy spaces case, Pre-OGA reduces to AFD.

This paper is organized as follows. In Section 2, we briefly introduce 1-D AFD and propose the discretization scheme. In Section 3, we perform a FFT formulation for 1-D AFD. The FFT-based algorithm for 1-D AFD is described in this part as well as the computational complexity analysis of the proposed algorithm. In Section 4, applicability and efficiency of our method are illustrated by numerical experiments with comparisons with 1-D AFD without FFT formulation. For simplification of notation and terminology we call the initial and direct algorithm of AFD without FFT formulation [20,21] as Direct-AFD; and the proposed one adopting an FFT formulation as FFT-AFD.

2. Preliminaries

In this section, we will recall the basic theory of 1-D AFD.

Denote by $H^2 = H^2(\mathbb{D})$ the Hardy space of holomorphic functions on the unit disc $\mathbb{D} = \{z \in \mathbb{C} : |z| < 1\}$:

$$H^2(\mathbb{D}) = \left\{ f : \mathbb{D} \to \mathbb{C} | f \text{ is holomorphic in } \mathbb{D} \text{ and } \sup_{0\le r<1} \frac{1}{2\pi}\int |f(re^{it})|^2 dt < \infty \right\}. \tag{1}$$

Note that there holds the following relation:

$$H^2(\mathbb{D}) = \left\{ f : \mathbb{D} \to \mathbb{C} | f(z) = \sum_{k=0}^{\infty} c_k z^k, \ \sum_{k=0}^{\infty} |c_k|^2 < \infty \right\}.$$

Since Hardy space functions have boundary limits [5], for all $f, g \in H^2$, the inner product of H^2 is defined as

$$\langle f, g\rangle = \frac{1}{2\pi}\int_0^{2\pi} f(e^{it})\overline{g}(e^{it})dt,$$

where $\overline{a}$ denotes the complex conjugate of $a \in \mathbb{C}$.

We will use the Szegö dictionary of the Hardy space H^2 that consists of the normalized Szegö kernels:

$$\mathcal{D} = \left\{ e_a := \frac{\sqrt{1-|a|^2}}{1-\bar{a}z}, a \in \mathbb{D} \right\}.$$

1-D AFD, or Core-AFD, is formulated as follows (cf. [1,4,20]). Let G be an arbitrary function in the Hardy space H^2. With $G_1 = G$, we first have

$$G(z) = \langle G_1, e_{a_1}\rangle e_{a_1}(z) + G_2(z)\frac{z-a_1}{1-\overline{a_1}z},$$

where

$$G_2(z) = \left(G_1(z) - \langle G_1, e_{a_1}\rangle e_{a_1}(z)\right)\frac{1-\overline{a_1}z}{z-a_1}$$

and

$$a_1 = \arg\max_{a\in\mathbb{D}} \left\{ |\langle G_1, e_a\rangle|^2 \right\}.$$

The selection and the possibility of the selection of such $a_1 \in \mathbb{D}$ are called the maximal selection principle (in Szegö dictionary $\mathcal{D}$). It is shown in [1] that such a_1, at which the maximal value exists and is attained, can only exist inside the open unit disc. We note that G_2 is again in the Hardy space H^2. This fact is crucial for the following recursive steps.

The above backward shift procedure for defining a new Hardy space function G_2 from G_1 can be repeated. After n iterative steps, we get

$$G(z) = S_n(z) + G_{n+1}(z) \prod_{k=1}^{n} \frac{z - a_k}{1 - \overline{a_k} z}, \tag{2}$$

where

$$S_n(z) = \sum_{k=1}^{n} \left\langle G_k, e_{a_k} \right\rangle e_{a_k}(z) \prod_{j=1}^{k-1} \frac{z - a_j}{1 - \overline{a_j} z}, \tag{3}$$

the reduced reminder G_{k+1} is obtained through the recursive formula

$$G_{k+1}(z) = \left(G_k(z) - \left\langle G_k, e_{a_k} \right\rangle e_{a_k}(z) \right) \frac{1 - \overline{a_k} z}{z - a_k},$$

and

$$a_k = \arg\max_{a \in \mathbb{D}} \left\{ |\langle G_k, e_a \rangle|^2 \right\}.$$

Write

$$B_k(z) = e_{a_k}(z) \prod_{j=1}^{k-1} \frac{z - a_j}{1 - \overline{a_j} z} = \frac{\sqrt{1 - |a_k|^2}}{1 - \overline{a_k} z} \prod_{j=1}^{k-1} \frac{z - a_j}{1 - \overline{a_j} z}.$$

Each B_k is the product of a dictionary element, as normalized Szegö kernel, and a Blaschke product with $a_1, a_2, \ldots, a_{k-1}$ as its zeros. Then $\{B_n\}_{n=1}^{+\infty}$ is a rational orthonormal system, also called Takenaka–Malmquist system [6].

Furthermore, due to the orthogonality, for each n there holds

$$\left\| G - \sum_{k=1}^{n} \langle G_k, e_{a_k} \rangle B_k \right\|^2 = \|G\|^2 - \sum_{k=1}^{n} |\langle G_k, e_{a_k} \rangle|^2 = \|G_{n+1}\|^2. \tag{4}$$

The last equality relation is due to the unimodule property of Blaschke products. In [1] we show

$$\lim_{k \to \infty} \|G_{k+1}\| = 0,$$

that implies the convergence and

$$G = \sum_{k=1}^{\infty} \langle G_k, e_{a_k} \rangle B_k. \tag{5}$$

The above is called 1-D AFD or Core AFD. Cyclic AFD and unwinding AFD are further developments of Core AFD. Under Cyclic AFD, for a fixed n, one seeks a simultaneous selection of all n parameters $a_1, \ldots, a_n$ that gives rise to the minimal remainder in the TM-system expansion [9]. It gives a partial solution of the long open problem of best rational approximations to a Hardy space function. With Unwinding AFD, at each recursive step, one first performs the Nevanlinna factorization to factorize out the inner function part and then uses Core AFD to decompose the outer function part [4].

3. FFT formulation

In our case, the Hilbert space is $H^2(\mathbb{D})$. As the non-tangential boundary limit of $f(z) \in H^2(\mathbb{D})$ is $f(e^{it}) \in H^2(\partial\mathbb{D})$, where $\partial\mathbb{D}$ is the unit circle, it is an isometric map from $H^2(\mathbb{D})$ to $H^2(\partial\mathbb{D})$. 1-D AFD mentioned in Section 1 is applied to functions in $H^2(\partial\mathbb{D})$.

The main computation step is to compute the coefficients in (5). At the kth step, the kth coefficient is of the maximum absolute value among the absolute value of the inner product, which is the integration of the product of the reduced remainder and the normalized Szegö kernel in $\mathcal{D}$. Due to the similarity between a Szegö kernel and a general monomial term in the Fourier system, the integration is analogous to the Fourier transform of a periodic function. In what follows, we apply the polar coordinate and use the grid mesh

$$C_{M,N} = \left\{ a_{l,j} = r_l e^{i\frac{2j}{2N}},\ 0 < r_1 < \cdots < r_M < 1,\ l = 1, 2, \ldots,\ M, j = 0, 1, 2, \ldots, 2N - 1 \right\}$$

in searching over $\mathbb{D}$, instead of the usual rectangular grid. This is the starting point of our algorithm proposed in the context.

Specially the grid points on the circle of radius r_l constitute a subset of $C_{M,N}$ as

$$C_{l,N} = \left\{ a_{l,j} = r_l e^{i\frac{2j}{2N}},\ 0 < r_l < 1, M,\ j = 0, 1, 2, \ldots, 2N - 1 \right\},$$

which is our FFT formulation involved.

3.1. FFT formulation of 1-D AFD

In 1-D AFD, the kth coefficient in (5) is the inner product of the kth reduced remainder G_k and the Szegö kernel in $\mathcal{D}$. As mentioned, due to the similarity between a Szegö kernel and a monomial, the discretization of the integral for computing the inner product gives rise to a DFT like series that can incorporate the FFT idea.

To simplify the notation we denote G_k as G in this part. Under the maximal selection principle (MSP), it needs to find the parameter $a_k \in \mathbb{D}$ that satisfies

$$\left|\langle G, e_{a_k}\rangle\right|^2 = \max_{a\in\mathbb{D}} |\langle G, e_a\rangle|^2 .$$

The kth coefficients of 1-D AFD is then given by $\langle G, e_{a_k}\rangle$ and

$$\langle G, e_a\rangle = \frac{1}{2\pi}\int_0^{2\pi} G(e^{it})\frac{\sqrt{1-r^2}}{1-ae^{-it}}dt. \tag{6}$$

Numerical model is established as follows. Suppose the interval $[0, 2\pi)$ is evenly divided into $0 = t_0 < \cdots < t_{2N-1} < t_{2^K} = t_{2N} = 2\pi$ with a large integer K. Then the inner product (6) can be discretized as

$$\langle G, e_a\rangle \approx \sum_{m=0}^{2N-1} \frac{\sqrt{1-|a|^2}}{2N} G\left(e^{i\frac{2\pi m}{2N}}\right)\frac{1}{1-ae^{-i\frac{2\pi m}{2N}}}. \tag{7}$$

Under the maximal selection principle, by adopting grid mesh $C_{M,N}$ the proposed algorithm works on finding a point $a_{l',j'} \in C_{M,N}$ satisfying

$$\left|\left\langle G, e_{a_{l',j'}}\right\rangle\right|^2 \approx \max_{a_{l,j}\in C_{M,N}} \left|\langle G, e_{a_{l,j}}\rangle\right|^2 .$$

Let W_{2N}^m denote the exponential term $W_{2N}^m = e^{-i\frac{2\pi}{2N}m}$. Then for discrete data $G\left(W_{2N}^{-m}\right)$ and $a_{l,j}$ in the polar coordinate form $a_{l,j} = r_l e^{i\frac{2\pi}{2N}j}$, (7) can be approximated by a series

$$\langle G, e_a\rangle \approx \sum_{m=0}^{2N-1} \frac{\sqrt{1-r_l^2}}{2N}\frac{G\left(W_{2N}^{-m}\right)}{1-r_l W_{2N}^{m-j}} \overset{\text{def}}{=} \widetilde{\langle G, e_{a_{l,j}}\rangle}. \tag{8}$$

When $j = 1, 2, \ldots, 2N-1$, it is clear that (8) is a DFT like series. It is feasible that the computation of these series is realized on the principle of FFT.

Theorem 3.1. *For fixed l_0, $0 < r_{l_0} < 1$, $a_j = r_{l_0}e^{i\frac{2\pi}{2N}j}$, the right side of (8) is equivalent to the case*

$$\begin{cases} \widetilde{\langle G, e_{a_{l_0,j}}\rangle} = H_j^{(1)} + r_{l_0}W_{2N}^{-j}G_j^{(1)}, \\ \widetilde{\langle G, e_{a_{l_0,j+N}}\rangle} = H_j^{(1)} - r_{l_0}W_{2N}^{-j}G_j^{(1)}, \end{cases} \quad j = 0, 1, 2, \ldots, N-1, \tag{9}$$

where

$$\begin{aligned} H_j^{(1)} &= \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\left(\frac{G\left(W_{2N}^{-2m}\right)}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2} + \frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2}\right), \\ G_j^{(1)} &= \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\left(\frac{G\left(W_{2N}^{-2m}\right)W_{2N}^{2m}}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2} + \frac{G\left(W_{2N}^{-(2m+1)}\right)W_{2N}^{2m+1}}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2}\right). \end{aligned} \tag{10}$$

Proof. Noticing that for a fixed $0 < r_{l_0} < 1$, it is sufficient for us to turn the expression (8) of length $2N$ into a summation of two expressions of length N as

$$\begin{aligned} \widetilde{\langle G, e_{a_{l_0,j}}\rangle} &= \sum_{m=0}^{2N-1}\frac{\sqrt{1-r_{l_0}^2}}{2N}\frac{G\left(W_{2N}^{-m}\right)}{1-r_{l_0}W_{2N}^{m-j}} = \sum_{2m=0}^{2N-2}\frac{\sqrt{1-r_{l_0}^2}}{2N}\frac{G\left(W_{2N}^{-2m}\right)}{1-r_{l_0}W_{2N}^{2m-j}} + \sum_{2m+1=1}^{2N-1}\frac{\sqrt{1-r_{l_0}^2}}{2N}\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-r_{l_0}W_{2N}^{2m-j}W_{2N}} \\ &= \frac{\sqrt{1-r_{l_0}^2}}{2N}\left(\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)}{1-r_{l_0}W_{2N}^{2m-j}} + \sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-r_{l_0}W_{2N}^{2m-j}W_{2N}}\right), \quad j = 0, 1, 2, \ldots, 2N-1, \end{aligned}$$

which can be simplified as:

$$\langle G, e_{a_{l_0,j}}\rangle\tilde{} = H_j + G_j, \tag{11}$$

by denoting

$$H_j = \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)}{1-r_{l_0}W_{2N}^{2m-j}}, \qquad G_j = \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-r_{l_0}W_{2N}^{2m+1-j}}.$$

Observing that $e^{i\frac{2\pi}{2N}N} = -1$, associating with (11), we get

$$\langle G, e_{a_{l_0,j+N}}\rangle\tilde{} = H_{j+N} + G_{j+N}, \tag{12}$$

where

$$H_{j+N} = \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)}{1+r_{l_0}W_{2N}^{2m-j}}, \qquad G_{j+N} = \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1+r_{l_0}W_{2N}^{2m+1-j}}, \quad j = 0, 1, 2, \ldots, N-1.$$

It is clear that the denominators of H_j and H_{j+N} are different, while the situation is the same to G_j and G_{j+N}. Following the idea of FFT, generating two common terms of length N by the periodicity of (8) is the key point to reducing computation. Meanwhile these two common terms should be in a similar form of (8) for the following recursive steps. Then we rearrange (11) and (12) to a common denominator and factor the common twiddle factor $r_{l_0}W_{2N}^{-j}$ out. Namely we have

$$\begin{aligned}
\langle G, e_{a_{l_0,j}}\rangle\tilde{} &= \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2} + r_{l_0}W_{2N}^{-j}\frac{\sqrt{1-r^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)W_{2N}^{2m}}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2} \\
&\quad + \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2} + r_{l_0}W_{2N}^{-j}\frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)W_{2N}^{2m+1}}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2} \\
&= \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2} + \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2} \\
&\quad + r_{l_0}W_{2N}^{-j}\frac{\sqrt{1-r_{l_0}^2}}{2N}\left(\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)W_{2N}^{2m}}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2} + \sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)W_{2N}^{2m+1}}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2}\right),
\end{aligned}$$

and

$$\begin{aligned}
\langle G, e_{a_{l_0,j+N}}\rangle\tilde{} &= \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)}{1+r_{l_0}W_{2N}^{2m-j}} + \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1+r_{l_0}W_{2N}^{2m+1-j}}, \\
&= \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2} - r_{l_0}W_{2N}^{-j}\frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)W_{2N}^{2m}}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2} \\
&\quad + \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2} - r_{l_0}W_{2N}^{-j}\frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)W_{2N}^{2m+1}}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2} \\
&= \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2} + \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2} \\
&\quad - r_{l_0}W_{2N}^{-j}\frac{\sqrt{1-r_{l_0}^2}}{2N}\left(\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)W_{2N}^{2m}}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2} + \sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)W_{2N}^{2m+1}}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2}\right).
\end{aligned}$$

So, to compute (11) and (12) is equivalent to calculate both of (13) and (14) as follows

$$\langle G, e_{a_{l_0,j}}\rangle^{\sim} = \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\left(\frac{G\left(W_{2N}^{-2m}\right)}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2}+\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2}\right) + r_{l_0}W_{2N}^{-j}\frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\left(\frac{G\left(W_{2N}^{-2m}\right)W_{2N}^{2m}}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2}+\frac{G\left(W_{2N}^{-(2m+1)}\right)W_{2N}^{2m+1}}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2}\right), \tag{13}$$

and

$$\langle G, e_{a_{l_0,j+N}}\rangle^{\sim} = \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\left(\frac{G\left(W_{2N}^{-2m}\right)}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2}+\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2}\right) - r_{l_0}W_{2N}^{-j}\frac{\sqrt{1-r^2}}{2N}\sum_{m=0}^{N-1}\left(\frac{G\left(W_{2N}^{-2m}\right)W_{2N}^{2m}}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2}+\frac{G\left(W_{2N}^{-(2m+1)}\right)W_{2N}^{2m+1}}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2}\right). \tag{14}$$

Denote two common terms respectively as

$$H_j^{(1)} = \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\left(\frac{G\left(W_{2N}^{-2m}\right)}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2}+\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2}\right),$$
$$G_j^{(1)} = \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\left(\frac{G\left(W_{2N}^{-2m}\right)W_{2N}^{2m}}{1-\left(r_{l_0}W_{2N}^{2m-j}\right)^2}+\frac{G\left(W_{2N}^{-(2m+1)}\right)W_{2N}^{2m+1}}{1-\left(r_{l_0}W_{2N}^{2m+1-j}\right)^2}\right). \tag{15}$$

Then we get the recursive relationship

$$\begin{cases}\langle G, e_{a_{l_0,j}}\rangle^{\sim} = H_j^{(1)} + r_{l_0}W_{2N}^{-j}G_j^{(1)},\\ \langle G, e_{a_{l_0,j+N}}\rangle^{\sim} = H_j^{(1)} - r_{l_0}W_{2N}^{-j}G_j^{(1)},\end{cases} \quad j = 0, 1, 2, \ldots, N-1. \quad \square \tag{16}$$

3.2. Numerical method

Here we describe our FFT formulation to the inner product of the given function and the normalized Szegö kernel in detail. The computation of the inner product part contributes the major part of the total computation algorithm. The rest computations in 1-D AFD are mainly discretization of the corresponding algebraic expressions. The computation complexity of our method will be shown in next part.

Step 1. Compute $\langle G, e_{a_{1,j}}\rangle$ for $a_{1,j} \in C_{1,N}$.

We have to compute and store the quantities as follows

$$W_{2N}^m,\ r_l W_{2N}^{-m},\ G(W_{2N}^{-m})W_{2N}^m, \quad m = 0, 1, 2, \ldots, 2N-1,\ l = 1, 2, \ldots, M. \tag{17}$$

Next, compute $\langle G, e_{a_{1,j}}\rangle^{\sim}$, $j = 0, 1, 2, \ldots, 2N-1$ with r_1. First of all, it is necessary for us to have the values of the input components to obtain $\{\langle G, e_{a_{1,j}}\rangle^{\sim}\}_{j=0}^{2N-1}$. Note that the integral variable here is m, with the binary representation as

$$m = 2^{K-1}m_{K-1} + \cdots + 2^2 m_2 + 2^1 m_1 + m_0 =: (m_{K-1}, \ldots, m_2, m_1, m_0), \quad m_i = 0, 1.\ i = 0, 1, \ldots, K-1.$$

By adopting Theorem 3.1 into binary representation, the input components $f(h_{K-1}, \ldots, h_1, h_0)$ can be written as

$$f(h_{K-1}, \ldots, h_1, h_0) = \sum_{m_0=0}^{1}\sum_{m_1=0}^{1}\cdots\sum_{m_{K-1}=0}^{1}\frac{G(W_{2N}^{(m_0,m_1,\ldots,m_{K-1})})W_{2N}^{(m_0,m_1,\ldots,m_{K-1})(h_{K-1},\ldots,h_1,h_0)}}{1-r^{2N}}. \tag{18}$$

For $m = (m_{K-1}, \ldots, m_1, m_0)$, denote the bit-reversal permutations of size $2N = 2^K$ as $s =: (m_0, m_1, \ldots, m_{K-1})$ and $h =: (h_{K-1}, \ldots, h_1, h_0)$. Then (18) becomes

$$f(h) = \frac{1}{1-r^{2N}}\sum_{s=0}^{2N-1}G(W_{2N}^s)W_{2N}^{hs}, \tag{19}$$

the DFT of $G(W_{2N}^s)$ multiplied by a constant, which can be directly obtained by FFT.

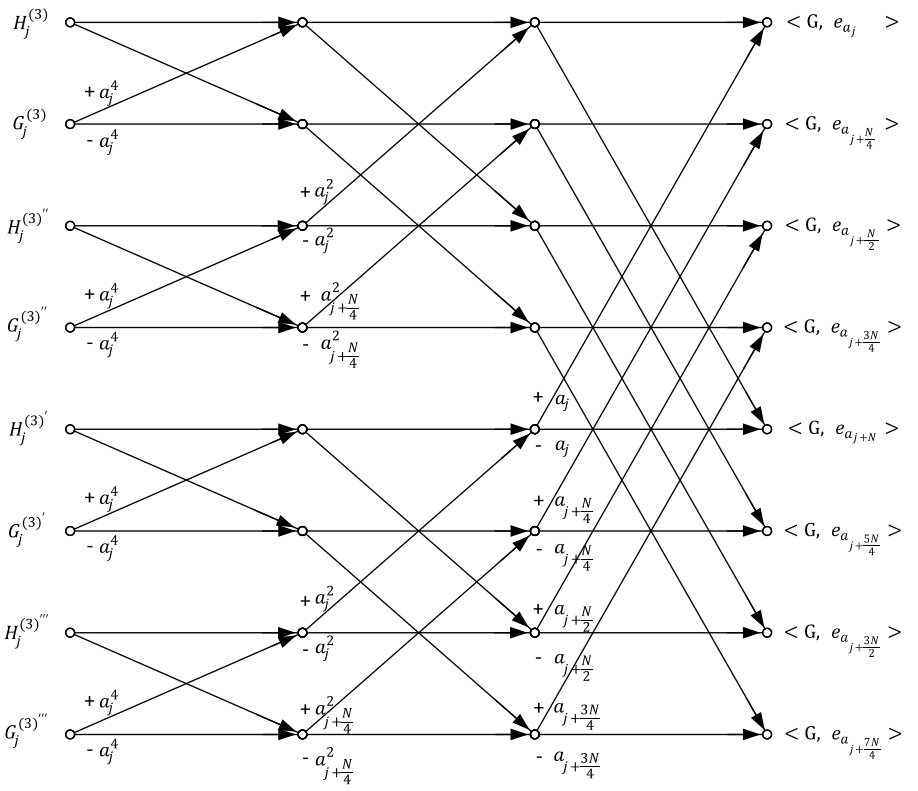


Fig. 1. Obtain the inner product from the original input components.

Secondly, making full use of (9), it is sufficient to compute $H_j, G_j, j = 0, 1, 2, \ldots, N-1$. Finally, repeating the same procedure, we get all the values $\langle G, e_{a_{1,j}}\rangle\tilde{}, j = 0, 1, 2, \ldots, 2N-1$.

To explain in detail how to compute $\langle G, e_{a_{1,j}}\rangle\tilde{}, j = 0, 1, 2, \ldots, 2N-1$, we set an example for $2N = 8$. We denote the elements in $f(h)$ as $f(h) = \left[H_j^{(3)}, H_j^{(3)'}, H_j^{(3)''}, H_j^{(3)'''}, G_j^{(3)}, G_j^{(3)'}, G_j^{(3)''}, G_j^{(3)'''}\right]$. The detailed data flow diagram for $2N = 8$ is shown in Fig. 1.

Step 2. Find a point $a_{l',j'} \in \mathbb{D}$, satisfying $\left|\langle G, e_{a_{l',j'}}\rangle\tilde{}\right|^2 = \max_{a_{l,j}\in C_{M,N}} \left|\langle G, e_{a_{l,j}}\rangle\right|^2$.

First, for all $r_l, l = 1, 2, \ldots, M$, we get all of the values of $\langle G, e_{a_{l,j}}\rangle\tilde{}$ by repeating Step 1, where $|a_{l,j}| = r_l, j = 0, 1, 2, \ldots, 2N-1$.

Second, apply the search method for all of $\langle G, e_{a_{l,j}}\rangle\tilde{}, a_{l,j} = r_l e^{i\frac{2\pi}{2N}j}, j = 0, 1, 2, \ldots, 2N-1, l = 1, 2, \ldots, M$ to find a point $a_{l',j'} \in \mathbb{D}$, satisfying $\left|\langle G, e_{a_{l',j'}}\rangle\tilde{}\right|^2 = \max_{a_{l,j}\in C_{M,N}} \left|\langle G, e_{a_{l,j}}\rangle\tilde{}\right|^2$.

3.3. Computational complexity

In what follows, we refer to the argument of the classical FFT to derive the computational complexity of our method. Let us start with

$$\begin{aligned}\langle G, e_{a_{l_0,j}}\rangle\tilde{} = H_j + G_j &= \frac{\sqrt{1-r_{l_0}^2}}{2N}\left(\sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-2m}\right)}{1-r_{l_0}W_{2N}^{2m-j}} + \sum_{m=0}^{N-1}\frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-r_{l_0}W_{2N}^{2m+1-j}}\right)\\ &= \frac{\sqrt{1-r_{l_0}^2}}{2N}\sum_{m=0}^{N-1}\left(\frac{G\left(W_{2N}^{-2m}\right)}{1-r_{l_0}W_{2N}^{2m-j}} + \frac{G\left(W_{2N}^{-(2m+1)}\right)}{1-r_{l_0}W_{2N}^{2m+1-j}}\right), \quad j = 0, 1, 2, \ldots, 2N-1. \end{aligned} \tag{20}$$

Table 1
Running time (s) in Case 1.

FFT-AFD	0.2145	0.2150	0.2138	0.2143	0.2136	0.2257
Direct-AFD	4.6202	4.6896	4.6081	4.5976	4.7307	4.7403

Hereby, for convenience, we do not consider the coefficient of $\frac{\sqrt{1-r_{l_0}^2}}{2N}$ because it does not influence the computational complexity of $\langle G, e_{a_{l_0,j}}\rangle\tilde{}, j = 0, 1, 2, \ldots, 2N-1$, and the rest is still denoted by $\langle G, e_{a_{l_0,j}}\rangle\tilde{}, j = 0, 1, 2, \ldots, 2N-1$ without confusion.

In Step 1, the computational complexity of the three terms of (17) is $\mathcal{O}(N)$. Under this setting, the computational complexity of getting $H_j^{(1)}$ and $G_j^{(1)}, j = 0, 1, 2, \ldots, N-1$, respectively, is the same, denoted by $F(N)$. Associating (11) with (13), (14), in order to obtain $\langle G, e_{a_{l_0,j}}\rangle\tilde{}, j = 0, 1, 2, \ldots, 2N-1$, it is sufficient to compute $\langle G, e_{a_{l_0,j}}\rangle\tilde{}, \langle G, e_{a_{l_0,j+N}}\rangle\tilde{}, j = 0, 1, 2, \ldots, N-1$, whose computational complexity is denoted by $F(2N)$. Observing the recursive relationship (9), which reduces $2N$ computation of $\langle G, e_{a_{l_0,j}}\rangle\tilde{}, j = 0, 1, \ldots, 2N-1$ to N computation of $\langle G, e_{a_{l_0,j}}\rangle\tilde{}, j = 0, 1, \ldots, N-1$. Following the argument of the classical FFT, we derive

$$F(2N) = 2F(N) + N, \tag{21}$$

which leads to the computational complexity $\mathcal{O}(N \log N)$ to compute all of $\langle G, e_{a_{1,j}}\rangle\tilde{}, j = 0, 1, 2, \ldots, 2N-1$.

So the total computational complexity of Step 1 is $\mathcal{O}(N \log N)$.

In Step 2, for different $r_1, r_2, \ldots, r_M$, we need to repeat the procedure of Step 1 M times. The computational complexity of this sub-step is $\mathcal{O}(MN \log N)$. In the second sub-step of Step 2, there at most are MN points to be searched. This leads to the computational complexity $\mathcal{O}(MN)$.

Summarizing, the total computational complexity of Step 1 and 2 is $\mathcal{O}(MN \log N)$.

Remark 3.2. For a fixed $r_{l_0} : 0 < r_{l_0} < 1$, all $\langle G, e_{a_{l_0,j}}\rangle\tilde{}$ for $|a_{l_0,j}| = r_{l_0}, j = 0, 1, 2, \ldots, 2N-1$ could be computed directly, starting with (8). It can be found that its computational complexity is $\mathcal{O}\left(N^2\right)$. Hence the computational complexity of the direct computation is higher than that of Step 1.

4. Numerical experiments

In Ref. [20], although the realizable method was proposed, the computational complexity is $\mathcal{O}\left(MN^2\right)$. However, this is not a fast method to realize Core-AFD. In this context, following the fundamental principal of the classical FFT, the proposed method is of the computational complexity $\mathcal{O}(MN \log N)$. This results in that the proposed method should be much more efficient and practical than that in [20]. In this section, we give the numerical experiment results by using our method FFT-AFD and one of the Direct-AFD, the method in [20].

The analysis of our methods is given from three different aspects, corresponding to three subsections respectively [22]. The first one is to list the approximated results, the running time results, the parameters a_k and relative errors in two cases, while original functions sampled by 2048 points. The second one is to make the running time comparison for the functions sampled by the different sample rates. The last one is to conduct the experiments on the original signals with a 20 dB Gaussian noise. The numerical experiments show that FFT-AFD obtains almost the same results of the Direct-AFD in less time.

In all of the following experiments, the radius of the unit disc $r = 0, 0.1, 0.2, \ldots, 0.8$ will be considered. Let G, S_n be defined as in (2). We define *relative error* by $\delta = \frac{\|G-S_n\|^2}{\|G\|^2}$. It must be noted that we set $a_1 = 0$ in every experiment to obtain a mono-components decomposition [1]. All codes in experiments are programmed in Matlab R2012b.

4.1. Basic experiments

Case 1. The original function is chosen as $f_1 = \frac{(0.07x^2-0.6)(0.2x-0.9)}{(x^2+2)} \in H^2(\mathbb{D})$.

The time consuming to run 4 steps is shown in Table 1. The experiments are repeated 6 times.

The real part of approximation results are shown in Table 2.

The parameters a_k and the relative errors are shown in Tables 3 and 4.

Analysis 1. From Table 1, it is obvious that the numerical computation significantly accelerates. From Tables 3 and 4, the difference of the parameters a_k and the relative error generated in our method are limited to 0.0002. These data indicate our method actually realize the effects of AFD in less time.

Case 2. Let f_2 be given as an example of the step functions $f_2 = sgn(\sin t)$.

The time consuming to run 25 steps is shown in Table 5.

The approximation results are shown in Table 6.

Table 2
Comparison between the real part of approximation results in Case 1.

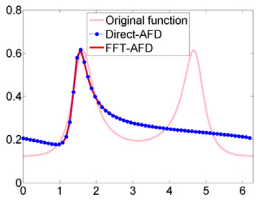
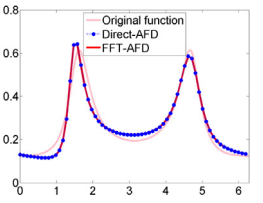
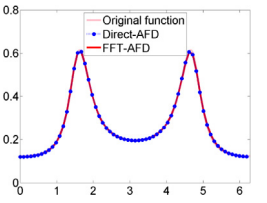
	S_2	S_3	S_4
1-D AFD			

Table 3
Parameters a_k in Case 1.

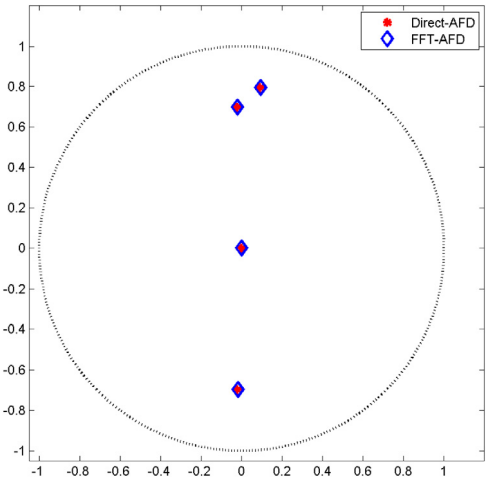


Table 4
Relative error δ in Case 1.

N	FFT-AFD	Direct-AFD
1	0.3676	0.3676
2	0.2140	0.2140
3	0.0216	0.0214
4	0.0002	0.0002

Table 5
Running time (s) in Case 2.

FFT-AFD	1.3218	1.3042	1.3254	1.3412	1.3070	1.3363
Direct-AFD	6.5031	6.7042	6.7122	6.5361	6.6030	6.6430

Table 6
Comparison between the approximation results in Case 2.

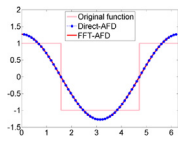
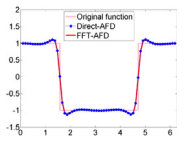
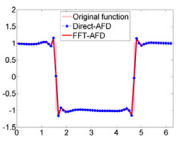
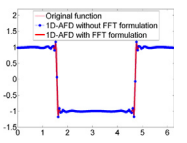
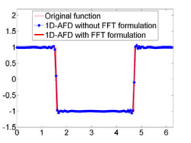
	S_2	S_4	S_9	S_{16}	S_{25}
1-D AFD					

Table 7
Parameters a_k in Case 2.

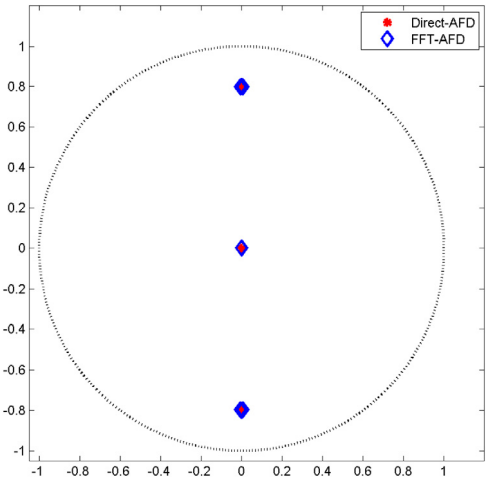


Table 8
Relative error δ in Case 2.

N	FFT-AFD	Direct-AFD
1	1.0000	1.0000
2	0.1895	0.1895
4	0.0267	0.0267
9	0.0121	0.0121
16	0.0061	0.0061
25	0.0038	0.0038

Table 9
Running time (s).

Sampled by	512	1024	2048	4096	8192
FFT-AFD	0.0516	0.1007	0.2163	0.4641	3.5796
Direct-AFD	0.3082	1.1677	4.7843	22.7593	6067.6351

The a_k and the relative errors are shown in Tables 7 and 8.

Analysis 2. The running time of our method is very stable. As shown in Table 6, the red curve from our method coincides with the blue curve from the method in [20]. The parameters and relative errors also have exactly the same values. Namely, FFT-AFD achieves the same effects of Direct-AFD with less computation cost.

4.2. Influence of sampling

In this part, the experiments devote to show the influence of the number of samples obtained from original signals. In this experiment, the discretization of 1-D AFD depends on the number of samples obtained from original signals. The results show that our method becomes more efficient when more samples are considered.

Our method in Section 3 has the $\mathcal{O}(MN \log N)$ computational complexity, which is used to approximately compute the inner product (7). Compared with the computational complexity $\mathcal{O}(MN^2)$ of the method [20], the running time should have an obvious improvement while the sampling points of the original function increase greatly. Hereby, the original signal f_1 is sampled by 512, 1024, 2048, 4096, 8192 points, and the method in Section 3 is run with the sampled signals, respectively. The running time to run 4 steps can be found in Table 9.

4.3. Precision stability under disturbance

In this part, the aim of the experiments is to observe the precision stability under disturbance of our method. Explicitly, our method works on f_1 with the 20 dB Gaussian noise.

Case 3. Let f_1 with 20 dB Gaussian noise be a signal disturbed by a noise with the random amplitude at any time. The real part of approximation results are shown in Table 10. The parameters a_k and the relative error are shown in Tables 11 and 12.

Table 10
Comparison between the real part of approximation results in Case 3.

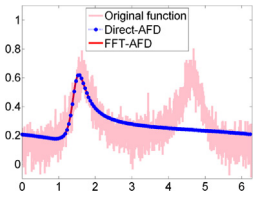
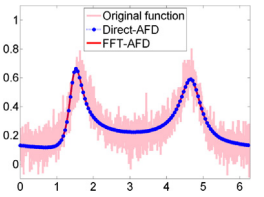
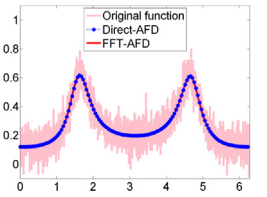
	S_2	S_3	S_4
1-D AFD			

Table 11
Parameters a_k in Case 3.

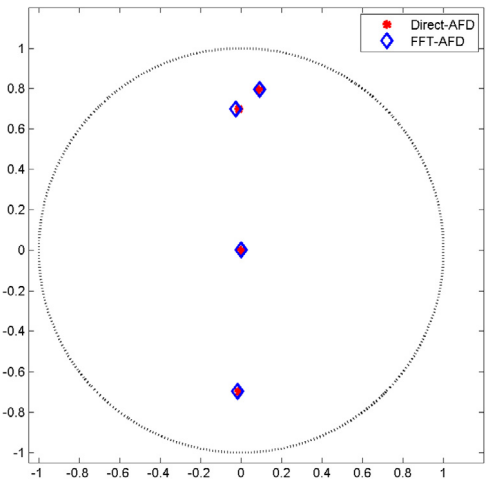


Table 12
Relative error δ in Case 3.

N	FFT-AFD	Direct-AFD
1	0.4185	0.4185
2	0.2753	0.2749
3	0.0973	0.0973
4	0.0845	0.0762

Analysis 3. Additive noise causes a less than 0.5% difference of 4th relative error between our method and the method in [20]. We think that additive noise has no influence on 1D-AFD.

Acknowledgments

We would like to thank Xiaoyin Wang, Weixiong Mai for comments on a previous version of the paper. This work was supported by the Multi-Year Research Grant of the University of Macau No. MYRG2016-00053-FST, the Annual Research Grant of the University of Macau No. CPG2015-00023-FST, the National Natural Science Foundation of China (No. 11571089), the Portuguese Foundation for Science and Technology (FCT—Fundao para a Cincia e a Tecnologia) through the CIDMA—Center for Research and Development in Mathematics and Applications, within project UID/MAT/04106/2013 and the Postdoctoral Foundation from FCT (Portugal) under Grant No. SFRH/BPD/74581/2010.

Conflict of interest. The authors declare that they have no conflict of interest.

References

[1] Tao Qian, Yan-Bo Wang, Adaptive Fourier seriesa variation of greedy algorithm, Adv. Comput. Math. 34 (3) (2011) 279–293.
[2] Tao Qian, Two-dimensional adaptive fourier decomposition, Math. Methods Appl. Sci. 39 (10) (2016) 2431–2448.
[3] Ronald R. Coifman, Stefan Steinerberger, Nonlinear phase unwinding of functions, J. Fourier Anal. Appl. (2015) 1–32.
[4] Tao Qian, Intrinsic mono-component decomposition of functions: An advance of fourier theory, Math. Methods Appl. Sci. 33 (7) (2010) 880–891.
[5] John B. Garnett, Bounded Analytic Functions, Vol. 96, Academic press, 1981.

[6] Adhemar Bultheel, Orthogonal Rational Functions, Cambridge University Press, 1999.
[7] Michel Rene Nahon, Phase Evaluation and Segmentation (Ph.D. thesis), Yale University, New Haven, CT, USA, 2000, pp. 1–162.
[8] Mary Weiss, Guido Weiss, A derivation of the main results of the theory of hp spaces, Rev. Un. Mat. Argentina 20 (1962) 63–71.
[9] T. Qian, Cyclic afd algorithm for best approximation by rational functions of given order, Math. Methods Appl. Sci. (2014).
[10] Yan Mo, Tao Qian, Support vector machine adapted tikhonov regularization method to solve dirichlet problem, Appl. Math. Comput. 245 (2014) 509–519.
[11] Tao Qian, Li-ming Zhang, Mathematical theory of signal analysis vs. complex analysis method of harmonic analysis, Appl. Math. J. Chinese Univ. 28 (4) (2013) 505–530.
[12] Wen Mi, Tao Qian, Frequency-domain identification: An algorithm based on an adaptive rational orthogonal system, Automatica 48 (6) (2012) 1154–1162.
[13] Wen Mi, Tao Qian, On backward shift algorithm for estimating poles of systems, Automatica (2014).
[14] Pei Dang, Tao Qian, Analytic phase derivatives, all-pass filters and signals of minimum phase, IEEE Trans. Signal Process. 59 (10) (2011) 4708–4718.
[15] Pei Dang, Guan-Tie Deng, Tao Qian, A tighter uncertainty principle for linear canonical transform in terms of phase derivative, IEEE Trans. Signal Process. 61 (21) (2013) 5153–5164.
[16] V.N. Temlyakov, Weak greedy algorithms, Adv. Comput. Math. 12 (2–3) (2000) 213–227.
[17] Stéphane G. Mallat, Zhifeng Zhang, Matching pursuits with time-frequency dictionaries, IEEE Trans. Signal Process. 41 (12) (1993) 3397–3415.
[18] Vladimir Temlyakov, Greedy Approximation, Vol. 20, Cambridge University Press, 2011.
[19] Geoffrey Davis, Adaptive nonlinear approximations (Ph.D. thesis), New York University, 1994.
[20] Tao Qian, Liming Zhang, Zhixiong Li, Algorithm of adaptive fourier decomposition, IEEE Trans. Signal Process. 59 (12) (2011) 5899–5906.
[21] Tao Qian, Yanbo Wang, Remarks on adaptive fourier decomposition, Int. J. Wavelets Multiresolut. Inf. Process. 11 (01) (2013).
[22] Nicholas J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, 2002.