Received: 19 December 2017

DOI: 10.1002/mma.4767

RESEARCH ARTICLE

WILEY

Fast algorithm of adaptive Fourier series

You Gao[1] | Min Ku[2] | Tao Qian[3]

[1]School of Mathematics (Zhuhai), Sun Yat-Sen University, Zhuhai, China

[2]Department of Computing Science, University of Radboud, 6525 EC Nijmegen, Netherlands

[3]Department of Mathematics, University of Macau, Macao (Via Hong Kong), China

Correspondence
Min Ku, CIDMA, Department of Computing Science, University of Radboud, 6525 EC Nijmegen, Netherlands.
Email: kumin0844@163.com

Communicated by: W. Sprößig

Funding information
Macao Government FDCT, Grant/Award Number: 079/2016/A2; Multi-Year Research Grant (MYRG), Grant/Award Number: MYRG2016-00053-FST; Macao Government FDCT, Grant/Award Number: 099/2014/A2

MSC Classification: 42A50; 32A30; 32A35; 46J15

Adaptive Fourier decomposition (AFD, precisely 1-D AFD or Core-AFD) was originated for the goal of positive frequency representations of signals. It achieved the goal and at the same time offered fast decompositions of signals. There then arose several types of AFDs. The AFD merged with the greedy algorithm idea, and in particular, motivated the so-called pre-orthogonal greedy algorithm (pre-OGA) that was proven to be the most efficient greedy algorithm. The cost of the advantages of the AFD-type decompositions is, however, the high computational complexity due to the involvement of maximal selections of the dictionary parameters. The present paper constructs one novel method to perform the 1-D AFD algorithm. We make use of the FFT algorithm to reduce the algorithm complexity, from the original $\mathcal{O}(MN^2)$ to $\mathcal{O}(MN\log_2 N)$, where N denotes the number of the discretization points on the unit circle and M denotes the number of points in $[0, 1)$. This greatly enhances the applicability of AFD. Experiments are performed to show the high efficiency of the proposed algorithm.

KEYWORDS

adaptive decomposition, analytic signals, computational complexity, Hilbert space

1 | INTRODUCTION

One-dimensional adaptive Fourier decomposition (abbreviated as 1-D AFD) has recently been proposed and proved to be among the most effective greedy algorithms.[1-3] The AFD is originally developed for the contexts of the unit disc and the upper-half plane and now is formally called 1-D AFD, or Core-AFD. The reason for the last terminology is because it becomes the constructive block of the lately developed variations of 1-D AFD, such as Unwinding AFD and Cyclic AFD, where the former is an algorithm for more effective frequency decompositions of signals,[4] and the latter is for finding solutions of n-best rational approximations of functions in the Hardy space.[5,6] Most recently, the concept of AFD is generalized to approximations of linear combinations of the Szegö kernels and their derivatives. In the later studies, such approximations are not necessarily obtained through greedy-type algorithms, viz, the maximal selection principle.[7,8] They can be obtained by any method, for instance, SVM in learning theory,[9] the regularizations in compressed sensing,[10] or the Tikhonov regularization, etc. In this article, we focus on an AFD algorithm of the greedy type using the maximal selection principle. The AFD-type decompositions all have promising applix cations to system identification and signal analysis[11,12] with proven effectiveness. Recently, 1-D AFD has been generalized in either the Clifford (Quaternionic) algebra,[13,14] or the several complex variables settings.[2] In the sequel, when we use the notion AFD, we will specify the context for clearness.

However, since 1-D AFD involves maximal selections of the parameters in the Szegö kernels, it has great computational complexity. For instance, the algorithm in Qian et al[15] is shown to be of the computational complexity $\mathcal{O}(MN^2)$, where N is the discretization of the unit circle and M is the number of samples in the radius of the unit disc, on which the maximal value is selected. We note that the quantity M cannot be reduced because it is independent on the discretization on the

unit circle. On the one hand, it has the necessity to reduce the computational complexity in order to make AFD more practical in applications. On the other hand, it is, in fact, feasible to build in FFT into the AFD algorithm. In the present paper, we provide one such algorithm reducing the complexity to $\mathcal{O}(MN\log_2 N)$ from the original $\mathcal{O}(MN^2)$, where N is for the discretization of the unit circle and M is the number of samples in the radius of the unit disc. In Section 2, we briefly recall 1-D AFD and its discretization scheme. In Section 3, we introduce the proposed algorithm and analyse its computational complexity. In Section 4, we give numerical examples to compare the precisions, the selected parameters, and the related errors between what we propose with the original 1-D AFD algorithm.

2 | PRELIMINARIES

Let L^2 be the Hilbert space of signals with finite energy on the closed interval $[0, 2\pi]$, equipped with the inner product

$$\langle G, F\rangle = \int_0^{2\pi} G(e^{it})\overline{F}(e^{it})dt, \tag{1}$$

where $G, F : [0, 2\pi] \to \mathbb{C}$, and $\overline{a}$ denotes the usual complex conjugate of $a \in \mathbb{C}$.[16] $H^2 = H^2(\mathbb{D})$ denotes the Hardy space on the unit disk $\mathbb{D} = \{z \in \mathbb{C} : |z| < 1\}$ of the complex plane $\mathbb{C}$. $\{B_k\}_{k=1}^{+\infty}$ is the Takenaka-Malmquist system or orthonormal rational function system, see, eg, previous works,[17-20] where

$$B_k(z) = B_{a_1,a_2,\dots,a_k}(z) = \frac{1}{\sqrt{2\pi}}\frac{\sqrt{1-|a_k|^2}}{1-\overline{a}_k z}\prod_{l=1}^{k-1}\frac{z-a_l}{1-\overline{a}_l z}, \tag{2}$$

$a_k \in \mathbb{D}, k \in \mathbb{N}$.

Based on $\{B_k\}_{k=1}^{+\infty}$, the core algorithm of AFD is constructed.[1,4,15] For a given analytic signal $G \in H^2$, with $G_1 = G$, there exits the decomposition

$$G(z) = \langle G_1, e_{a_1}\rangle e_{a_1} + G_2(z)\frac{z-a_1}{1-\overline{a_1}z}, \tag{3}$$

where $e_{a_1} = \frac{\sqrt{1-|a_1|^2}}{1-\overline{a}_1 z}$, $a_1 \in \mathbb{D}, z \in \partial\mathbb{D}$,

$$G_2(z) = \left(G_1 - \langle G_1, e_{a_1}\rangle e_{a_1}(z)\right)\frac{1-\overline{a_1}z}{z-a_1}, \tag{4}$$

and $a_1 \in \mathbb{D}$ is selected by according to the maximal selection principle. That is,

$$a_1 = \arg\max_{a\in\mathbb{D}}\left\{|\langle G_1, e_a\rangle|^2\right\}, \tag{5}$$

which is crucial for the core algorithm of AFD (cf., eg, Qian and Wang[1]). Repeating such process to the n-th step, we get

$$G(z) = \sum_{k=1}^{n}\langle G_k, e_{a_k}\rangle B_{a_1,\dots,a_k}(z) + G_{n+1}(z)\prod_{k=1}^{n}\frac{z-a_k}{1-\overline{a_k}z} = S_n + G_{n+1}(z)\prod_{k=1}^{n}\frac{z-a_k}{1-\overline{a_k}z}, \tag{6}$$

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

$$G_{k+1}(z) = \left(G_k(z) - \langle G_k, e_{a_k}\rangle e_{a_k}\right)\frac{1-\overline{a_k}z}{z-a_k}, \tag{7}$$

and

$$a_k = \arg\max_{a\in\mathbb{D}}\left\{|\langle G_k, e_a\rangle|^2\right\}. \tag{8}$$

Moreover, due to the orthogonality and the unimodular property of Blaschke products, for each n, there holds

$$\left\|G - \sum_{k=1}^{n}\langle G_k, e_{a_k}\rangle B_k\right\|^2 = \|G\|^2 - \sum_{k=1}^{n}\left|\langle G_k, e_{a_k}\rangle\right|^2. \tag{9}$$

As a Möbius transform is of norm 1 on the unit circle, the above equation is equal to $\|G_{n+1}\|^2$. The equation $\langle G_k, e_{a_k}\rangle = \langle G, B_k\rangle$ also holds because of the orthogonalization of $\{B_k\}_{k=1}^{n}$. Then

$$\lim_{n\to+\infty}\|G_{n+1}\|^2 = \lim_{n\to+\infty}\left\|G - \sum_{k=1}^{n}\langle G_k, e_{a_k}\rangle B_k\right\|^2 = \lim_{n\to+\infty}\left\|G - \sum_{k=1}^{n}\langle G, B_k\rangle B_k\right\|^2. \tag{10}$$

It has been proved that above limits convergent to zero when parameters a_k is selected by (8) (cf., eg, Qian and Wang[1]). Finally, we have

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

The above is called 1-D AFD or Core AFD. It can be shown that for $f(z) \in H^2(\mathbb{D})$, as $z \to e^{it}$ in the nontangential manner there exists the nontangential limits $f(e^{it})$ for almost all $e^{it} \in \partial\mathbb{D}$ (cf., eg, Garnett[17]). The mapping between $f(z)$ and its boundary limit function $f(e^{it})$ is an isometric isomorphism. Then $f(e^{it})$ could be processed by the approximation theory in $H^2(\mathbb{D})$. The biggest computation amount in 1-D AFD of $f(e^{it})$ is to find a point $a_k \in \mathbb{D}, k = 1, 2, \cdots$, satisfying $\left|\langle G_k, e_{a_k} \rangle\right|^2 = \max_{a \in \mathbb{D}} |\langle G_k, e_a \rangle|^2$, where $e_a = \frac{\sqrt{1-|a|^2}}{1-\bar{a}z}, a \in \mathbb{D}, z \in \partial\mathbb{D}$. The key step is to compute the following integral

$$\langle G_k, e_a \rangle = \frac{1}{2\pi} \int_0^{2\pi} G_k(e^{it}) \frac{\sqrt{1-|a|^2}}{1-ae^{-it}} dt, \forall a \in \mathbb{D}. \tag{11}$$

3 | FORMULATION, ALGORITHM, AND COMPLEXITY ANALYSIS

In this section, we will derive our approximation procedure of (11) incorporating FFT. We denote G_k as G for simplifying our notation.

3.1 | Formulation

We first give a discrete numerical model of (11). Suppose that the interval $[0, 2\pi)$ is evenly divided into $0 = t_0 < \cdots < t_m < \cdots < t_{2^K-1} < 2\pi$, $t_m = \frac{2\pi m}{2^K}$ with K being large. Then

$$\langle G, e_a \rangle = \frac{1}{2\pi} \int_0^{2\pi} G(e^{it}) \frac{\sqrt{1-|a|^2}}{1-ae^{-it}} dt \approx \sum_{m=0}^{2^K-1} \frac{\sqrt{1-|a|^2}}{2^K} G\left(e^{i\frac{2\pi m}{2^K}}\right) \frac{1}{1-ae^{-i\frac{2\pi m}{2^K}}}. \tag{12}$$

We index $a \in \mathbb{D}$ in polar coordinate. For a fixed $r\,(0 < r < 1)$, the circle of radius r is evenly divided by the 2^K points $e^{i\frac{2\pi}{2^K}j}, j = 0, 1, \ldots, N = 2^K-1$, which is the same segmentation step distance as (12). That is $a_j = re^{i\frac{2\pi}{2^K}j}, j = 0, 1, \ldots, 2^K-1$. The relation (12), therefore, can be rewritten by approximating a by a_j on a fixed circle as

$$\langle G, e_{a_j} \rangle \approx \sum_{m=0}^{2^K-1} \frac{\sqrt{1-r^2}}{2^K} G\left(e^{i\frac{2\pi m}{2^K}}\right) \frac{1}{1-a_j e^{-i\frac{2\pi m}{2^K}}}, j = 0, 1, 2, \ldots, 2^K-1. \tag{13}$$

To further reduce (12), we define the notation for the right side of (13) as

$$\langle G, e_{a_j} \rangle^{\%} = \sum_{m=0}^{2^K-1} \frac{\sqrt{1-r^2}}{2^K} G\left(e^{i\frac{2\pi m}{2^K}}\right) \frac{1}{1-a_j e^{-i\frac{2\pi m}{2^K}}}, j = 0, 1, 2, \ldots, 2^K-1. \tag{14}$$

Then a simple computation gives

$$\langle G, e_{a_j} \rangle^{\%} = \sum_{l=0}^{2^K-1} \frac{\sqrt{1-r^2}}{2\pi} \frac{a_j^l}{1-a_j^{2^K}} c_l, j = 0, 1, 2, \ldots, 2^K-1, \tag{15}$$

where the coefficients

$$c_l = \sum_{m=0}^{2^K-1} G\left(e^{i\frac{2\pi m}{2^K}}\right) e^{-il\frac{2\pi m}{2^K}}, l \in \mathbb{N} \cup \{0\}. \tag{16}$$

In fact, observing that function $e^{-i\frac{2\pi}{2^K}}$ has a period 2^K, then coefficient c_l is a periodic function of 2^K, ie, $c_l = c_{l+n2^K}, n \in \mathbb{N}$. Indeed, noticing $\left|a_j e^{-i\frac{2\pi m}{2^K}}\right| < 1, a_j \in \mathbb{D}$. This allows us to get another form of (14) as follows:

$$\begin{aligned}\langle G, e_{a_j}\rangle\tilde{} &= \sum_{m=0}^{2^K-1}\sum_{l=0}^{+\infty}\frac{\sqrt{1-r^2}}{2^K}a_j^l G\left(e^{i\frac{2\pi m}{2^K}}\right)e^{-il\frac{2\pi m}{2^K}} = \sum_{l=0}^{+\infty}\frac{\sqrt{1-r^2}}{2^K}a_j^l\sum_{m=0}^{2^K-1}G\left(e^{i\frac{2\pi m}{2^K}}\right)e^{-il\frac{2\pi m}{2^K}}\\ &= \sum_{s=0}^{+\infty}\sum_{l=s2^K}^{(s+1)2^K-1}\frac{\sqrt{1-r^2}}{2^K}a_j^l c_l = \sum_{l=0}^{2^K-1}\frac{\sqrt{1-r^2}}{2\pi}\frac{a_j^l}{1-a_j^{2^K}}c_l, a_j \in \mathbb{D}.\end{aligned}$$

Substituting $a_j = re^{i\frac{2\pi}{2^K}j} \in \mathbb{D}$ in (15), we have

$$\begin{aligned}\langle G, e_{a_j}\rangle\tilde{} &= \frac{\sqrt{1-r^2}}{2\pi\left(1-a_j^{2^K}\right)}\sum_{l=0}^{2^K-1}a_j^l c_l = \frac{\sqrt{1-r^2}}{2\pi\left(1-\left(re^{i\frac{2\pi}{2^K}j}\right)^{2^K}\right)}\sum_{l=0}^{2^K-1}\left(re^{i\frac{2\pi}{2^K}j}\right)^l c_l\\ &= \frac{\sqrt{1-r^2}}{2\pi\left(1-r^{2^K}e^{i\frac{2\pi}{2^K}j2^K}\right)}\sum_{l=0}^{2^K-1}r^l e^{i\frac{2\pi}{2^K}jl}c_l = \frac{\sqrt{1-r^2}}{2\pi\left(1-r^{2^K}\right)}\sum_{l=0}^{2^K-1}r^l c_l e^{i\frac{2\pi}{2^K}jl}, \quad j = 0, 1, \ldots, 2^K-1.\end{aligned} \tag{17}$$

To compute (17), we divide it into two steps. First, applying the FFT to all of $c_l, l = 0, 1, 2, \ldots, 2^K-1$. In fact, for arbitrary $0 \le l \le 2^K-1, l \in \mathbb{N}\cup\{0\}$, starting with (16), we have

$$\begin{aligned}c_l &= \sum_{m=0}^{2^K-1}G\left(e^{i\frac{2\pi m}{2^K}}\right)e^{-il\frac{2\pi m}{2^K}} = \sum_{2m=0}^{2^K-2}G\left(e^{i\frac{2\pi(2m)}{2^K}}\right)e^{-il\frac{2\pi(2m)}{2^K}} + \sum_{2m+1=1}^{2^K-1}G\left(e^{i\frac{2\pi(2m+1)}{2^K}}\right)e^{-il\frac{2\pi(2m+1)}{2^K}}\\ &= \sum_{m=0}^{2^{K-1}-1}G\left(e^{i\frac{2\pi(2m)}{2^K}}\right)e^{-i\frac{2\pi(2m)}{2^K}l} + \sum_{m=0}^{2^{K-1}-1}G\left(e^{i\frac{2\pi(2m+1)}{2^K}}\right)e^{-i\frac{2\pi(2m+1)}{2^K}l}.\end{aligned} \tag{18}$$

Let $W_{2^K} = e^{-i\frac{2\pi}{2^K}}$, we get

$$\begin{cases}c_l = \sum_{m=0}^{2^{K-1}-1}G\left(W_{2^K}^{-2m}\right)W_{2^K}^{2ml} + \sum_{m=0}^{2^{K-1}-1}G\left(W_{2^K}^{-(2m+1)}\right)W_{2^K}^{2ml}W_{2^K}^{l},\\ c_{l+2^{K-1}} = \sum_{m=0}^{2^{K-1}-1}G\left(W_{2^K}^{-2m}\right)W_{2^K}^{2ml} - \sum_{m=0}^{2^{K-1}-1}G\left(W_{2^K}^{-(2m+1)}\right)W_{2^K}^{2ml}W_{2^K}^{l}.\end{cases} \tag{19}$$

The second step is to use FFT formulation and to derive the following theorem.

Theorem 3.1. *The right-hand side of (14) is equal to the cases*

$$\begin{cases}\langle G, e_{a_j}\rangle\tilde{} = \frac{\sqrt{1-r^2}}{2^K\left(1-r^{2^K}\right)}\left(\sum_{l=0}^{2^{K-1}-1}r^{2l}c_{2l}W_{2^K}^{-2jl} + \sum_{l=0}^{2^{K-1}-1}r^{2l+1}c_{2l+1}W_{2^K}^{-2jl}W_{2^K}^{-j}\right),\\ \langle G, e_{a_{j+2^{K-1}}}\rangle\tilde{} = \frac{\sqrt{1-r^2}}{2\pi\left(1-r^{2^K}\right)}\left(\sum_{l=0}^{2^{K-1}-1}r^{2l}c_{2l}W_{2^K}^{-2jl} - \sum_{l=0}^{2^{K-1}-1}r^{2l+1}c_{2l+1}W_{2^K}^{-2jl}W_{2^K}^{-j}\right),\end{cases} \tag{20}$$

where $j = 0, 1, 2, \ldots, 2^{K-1}-1$ *and* $c_l, l = 0, 1, 2, \ldots, 2^{K-1}-1$, *is given by (19).*

Proof. Observing from (14), one gets

$$\langle G, e_{a_j}\rangle\tilde{} = \frac{\sqrt{1-r^2}}{2^K\left(1-a_j^{2^K}\right)}\sum_{l=0}^{2^K-1}a_j^l c_l = \frac{\sqrt{1-r^2}}{2^K\left(1-\left(rW_{2^K}^{-j}\right)^{2^K}\right)}\sum_{l=0}^{2^K-1}\left(rW_{2^K}^{-j}\right)^l c_l \tag{21}$$

$$
\begin{aligned}
&= \frac{\sqrt{1-r^2}}{2^K\left(1-r^{2^K}W_{2^K}^{-j2^K}\right)}\sum_{l=0}^{2^K-1} r^l W_{2^K}^{-jl}c_l = \frac{\sqrt{1-r^2}}{2^K\left(1-r^{2^K}\right)}\sum_{l=0}^{2^K-1} r^l c_l W_{2^K}^{-jl}\\
&= \frac{\sqrt{1-r^2}}{2^K\left(1-r^{2^K}\right)}\sum_{2l=0}^{2^K-2} r^{2l} W_{2^K}^{-2lj}c_{2l} + \frac{\sqrt{1-r^2}}{2^K\left(1-r^{2^K}\right)}\sum_{2l+1=1}^{2^K-1} r^{2l+1} W_{2^K}^{-2lj}W_{2^K}^{-j}c_{2l+1}\\
&= \frac{\sqrt{1-r^2}}{2^K\left(1-r^{2^K}\right)}\left(\sum_{l=0}^{2^{K-1}-1} r^{2l} W_{2^K}^{-2lj}c_{2l} + \sum_{l=0}^{2^{K-1}-1} r^{2l+1} W_{2^K}^{-2lj}W_{2^K}^{-j}c_{2l+1}\right), j = 0, 1, \ldots, 2^K-1.
\end{aligned}
\tag{22}
$$

Hence, we get

$$
\begin{aligned}
\langle G, e_{a_{j+2^{K-1}}}\rangle^{\check{}} &= \frac{\sqrt{1-r^2}}{2^K\left(1-r^{2^K}\right)}\left(\sum_{l=0}^{2^{K-1}-1} r^{2l}c_{2l}W_{2^K}^{-2(j+2^{K-1})l} + \sum_{l=0}^{2^{K-1}-1} r^{2l+1}c_{2l+1}W_{2^K}^{-2(j+2^{K-1})l}W_{2^K}^{-(j+2^{K-1})}\right)\\
&= \frac{\sqrt{1-r^2}}{2^K\left(1-r^{2^K}\right)}\left(\sum_{l=0}^{2^{K-1}-1} r^{2l}c_{2l}W_{2^K}^{-2jl} - \sum_{l=0}^{2^{K-1}-1} r^{2l+1}c_{2l+1}W_{2^K}^{-2jl}W_{2^K}^{-j}\right).
\end{aligned}
\tag{23}
$$

Therefore, for $0 \le j < 2^{K-1}-1$, we have

$$
\begin{cases}
\langle G, e_{a_j}\rangle^{\check{}} = \frac{\sqrt{1-r^2}}{2^K\left(1-r^{2^K}\right)}\left(\sum_{l=0}^{2^{K-1}-1} r^{2l}c_{2l}W_{2^K}^{-2jl} + \sum_{l=0}^{2^{K-1}-1} r^{2l+1}c_{2l+1}W_{2^K}^{-2jl}W_{2^K}^{-j}\right),\\
\langle G, e_{a_{j+2^{K-1}}}\rangle^{\check{}} = \frac{\sqrt{1-r^2}}{2^K\left(1-r^{2^K}\right)}\left(\sum_{l=0}^{2^{K-1}-1} r^{2l}c_{2l}W_{2^K}^{-2jl} - \sum_{l=0}^{2^{K-1}-1} r^{2l+1}c_{2l+1}W_{2^K}^{-2jl}W_{2^K}^{-j}\right).
\end{cases}
\tag{24}
$$

□

Remark 3.2. Since the computational complexity of directly computing (14) is $\mathcal{O}(N^2)$, it is unacceptable especially when N takes a large positive integer. Thus, what is the key point is to reduce the computational complexity of (14) from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log_2 N)$. We achieve this by constructively changing (14) into (15) with coefficients c_l given by (16), and then transferring (15) into (20). This is because through the technical observation, we can make full use of the FFT algorithm to compute (20) and coefficients c_l given by (16), whose computational complexity is $\mathcal{O}(N\log_2 N)$. These ideas are the starting of the following algorithm.

3.2 | Algorithm

In this section, we propose our fast algorithm for the 1-D AFD, making use of the FFT mechanism. Parameters are expressed in polar coordinate. Radius is sampled evenly by M points as $0 < r_1 < r_2 < \ldots < r_s < \ldots < r_{M-1} < r_M < 1$, where $r_s = \frac{s}{M+1}, s = 1, 2, \ldots, M$. Denote the grid mesh on the circle of radius r_s is

$$C_{s,2^K} = \{a_j = r_s W_{2^K}^{-j}, j = 0, 1, 2, \ldots, 2^K-1\}.$$

This set is in particular involved in the FFT formulation of the Section 3.1.

3.2.1 | Algorithm description

Step 1. Compute $\langle G, e_{a_j}\rangle^{\%}, j = 0, 1, 2, \ldots, 2^K-1$ for all of $a_j \in C_{s,2^K}, s = 1, 2, \ldots, M$.Applying (19), we compute all of $r_s^l, c_l, l = 0, 1, 2, \ldots, 2^K-1, s = 1, 2, \ldots, M$, and store them.Next, starting from the recursive relation (20) with a fixed r_s, we need to work backward K times to obtain the expression of the input data. Set the input of the recursive relation to be $f(l) = r_s^l c_l, l = 1, 2, \ldots, 2^K-1$. Then the twiddle factor of (20) is $W_{2^K}^l$, while that of FFT is $W_{2^K}^{-l}$. Because f is output in the order of l, the bit-reversal permutation of m is necessary. Similarly, we need to input f in the order of the bit-reversal permutation of l to obtain $\langle G, e_{a_j}\rangle^{\%}, j = 0, 1, 2, \ldots, 2^K-1$. We repeat the above procedure to obtain $\langle G, e_{a_j}\rangle^{\%}$ for all $s = 1, 2, \ldots, M$.

Step 2. Find a point $a_{s',j'} \in \cup_{s=1}^{M} C_{s,2^K}$ satisfying $\left|\langle G, e_{a_{s',j'}}\rangle^{\%}\right|^2 = \max_{\substack{a_j \in C_{s,2^K}\\ s=1,2,\ldots,M}} \left|\langle G, e_{a_j}\rangle^{\%}\right|^2$.We compare all $\langle G, e_{a_j}\rangle^{\%}, j = 0, 1, 2, \ldots, 2^K-1, s = 1, 2, \ldots, M$, to find the maximum value and the corresponding parameter $a_{s',j'} = r_{s'}W_{2^K}^{-j'}$.

Remark 3.3. In our algorithm proposed in Section 3.1, we select evenly the sampling. While the nonuniform/nonevenly sampling is allowed, our algorithm still holds by applying the nonuniform discrete Fourier transforms developed in, eg, Frigo[21]

3.3 | Computational complexity

In this section, we will analyse the computation complexity of the proposed algorithm in Section 3.2. Here let $N = 2^K$.

In Step 1, applying the argument of the classical FFT, the computational complexity of (19) is $\mathcal{O}(N \log_2 N)$. The computational complexity of the input signal $f(l) = r^l c_l, l = 0, 1, 2, \dots, N-1$ is $\mathcal{O}(N + N \log_2 N)$, which is also $\mathcal{O}\left(N \log_2 N\right)$. For $s = 1, 2, \dots, M$, the total computational complexity of $\langle G, e_{a_j}\rangle^{\%}, j = 0, 1, \dots, 2^K - 1, s = 1, 2, \dots, M$, is $\mathcal{O}\left(MN \log_2 N\right)$.

In Step 2, in order to find a point $a_{s',j'} \in \cup_{s=1}^{M} C_{s,2^K}$, satisfying $\left|\langle G, e_{a_{s',j'}}\rangle^{\%}\right|^2 = \max\limits_{\substack{a_j \in C_{s,2K} \\ s=1,2,\dots,M}} \left|\langle G, e_{a_j}\rangle^{\%}\right|^2$, we compare the absolute value of all $\langle G, e_{a_j}\rangle^{\%}$ obtained from step 1 one by one. As the length of the above absolute values is MN, the computational complexity to find the maximum is $\mathcal{O}(MN)$.

Totally, the computational complexity of Algorithm 1 is $\mathcal{O}\left(MN \log_2 N\right)$.

Remark 3.4. For a fixed $r : 0 < r < 1$, $\langle G, e_{a_j}\rangle\ \tilde{}\ , |a_j| = r, \quad j = 0, 1, 2, \dots, 2^K - 1$, could be computed directly from (14), the computational complexity is $\mathcal{O}(N^2)$, being considerably higher than that of (19). But if we do not chose evenly 2^{K-m} $(0 < m < K, m \in \mathbb{N})$ on the circle of radius r, the argument of the classical FFT cannot play a role in the computation of 1-D AFD. In such a case, $\langle G, e_{a_j}\rangle^{\%}$ can be obtained from (14) with the computational complexity $\mathcal{O}(N^2)$.

4 | NUMERICAL EXPERIMENTS

In this section, we temporarily call the proposed algorithm FFT-AFD. We also call the direct computation of AFD without FFT mechanism as direct-AFD. The efficiency of the proposed algorithm is analysed from two different aspects. The first one is to list the output of the proposed algorithm: the running times, the approximation results, the parameters a_k, and relative errors. The second one is to make a running time comparison when the original signal is discretized by a different length of samples. The comparisons are presented between the output of the proposed algorithm and the ones of the Direct-AFD algorithm in Qian et al.[15]

We define the relative error by

$$\delta = \frac{||G - S_n||^2}{||G||^2}, \tag{25}$$

where G is the original signal and S_n is the summation of n terms, ie, $S_n = \sum_{k=1}^{n} \langle G, B_k\rangle B_k$.

In all of the following experiments, the radius of the unit disc $r = 0, 0.1, 0.2, \dots 0.8$ will be considered. The CPU of the used computer is the Intel G540 under the default setting of a single thread. All experiments are conducted in Matlab 2012b.

4.1 | Basic experiments

All original functions are sampled by the 1024 points in this part.

Case 1. The original function is chosen as

$$f_1 = \frac{(0.0247e^{i3t} + 0.355e^{i2t})}{(1 - 0.3679e^{it})} \in H^2.$$

The time-consuming to run 10 steps is shown in Table 1 below. The experiments are repeated 6 times.

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

The a_k's and the relative errors obtained from 1-D AFD with our proposed algorithm are shown in the following Tables 3 and 4.

TABLE 1 Running time, s

FFT-AFD	0.2617	0.2609	0.2613	0.2610	0.2609	0.2614
Direct-AFD	1.3315	1.3297	1.3294	1.3323	1.3294	1.3329

Abbreviations: AFD, adaptive Fourier decomposition; FFT, fast Fourier transform.

TABLE 2 Comparison between the approximation results in case 1

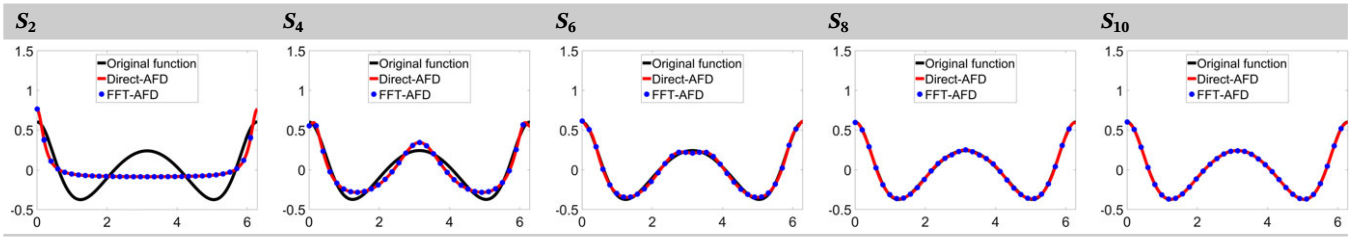

TABLE 3 Parameters a_k in case 1

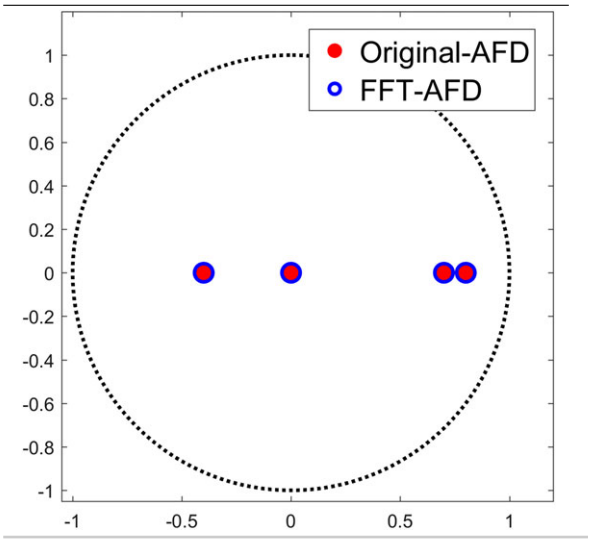

TABLE 4 Relative error, dB

N	FFT-AFD	Direct-AFD
1	1.0000	1.0000
2	0.5790	0.5790
3	0.2092	0.2093
4	0.0553	0.0553
5	0.0189	0.0189
6	0.0052	0.0052
7	0.0017	0.0017
8	0.0005	0.0005
9	0.0002	0.0002
10	0.0000	0.0001

Abbreviations: AFD, adaptive Fourier decomposition; FFT, fast Fourier transform.

Analysis 1. From Table 1, it is clear that the numerical computation is significantly accelerated. From Table 3, the difference of the parameters a_k and that of the relative error are limited in 0.0001. These data indicate that our proposed algorithm actually achieves the effects of 1-D AFD in less time.

Case 2. f_1 is a rational function being of good smoothness. Then f_2 is given as an example of the step functions as

$$f_2 = sgn(\sin t).$$

The running time, a_k, and the relative error are given in Table 5 as follows.
The approximation results are shown in Table 6.
The a_k's and the relative errors obtained from our proposed algorithm are shown in Tables 7 and 8.

Analysis 2. The running time of our proposed algorithm keeps a stable acceleration. The parameters a_k's have a little difference from those of the Direct-AFD, in which the value of the integral (inner product) is yielded by the Newton-Cotes rules. It does not cause any influence on the relative errors in this example.

TABLE 5 Running time, s

FFT-AFD	0.2618	0.2616	0.2593	0.2598	0.2598	0.2598
Direct-AFD	1.3110	1.3157	1.3294	1.3099	1.3150	1.3079

Abbreviations: AFD, adaptive Fourier decomposition; FFT, fast Fourier transform.

TABLE 6 Comparison between the approximation results in case 2

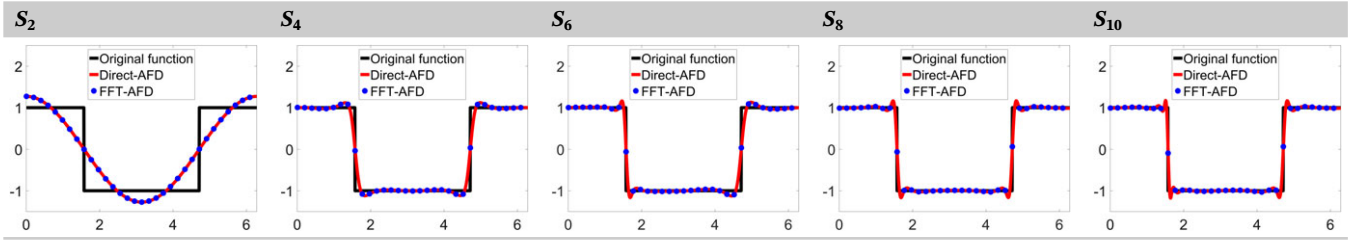

TABLE 7 Parameters a_k in case 1

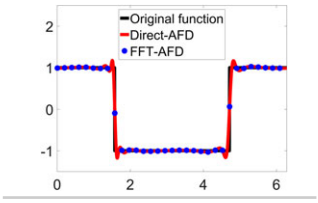

TABLE 8 Relative error, dB

N	FFT-AFD	Direct-AFD
1	1.0000	1.0000
2	0.1895	0.1895
3	0.1260	0.1260
4	0.0266	0.0266
5	0.0247	0.0247
6	0.0199	0.0199
7	0.0183	0.0183
8	0.0129	0.0129
9	0.0120	0.0120
10	0.0106	0.0105

Abbreviations: AFD, adaptive Fourier decomposition; FFT, fast Fourier transform.

TABLE 9 Running time, s

The Length of Samples	128	256	512	1024	2048	4096
FFT-AFD	0.0331	0.0634	0.1325	0.2662	0.5416	1.1243
Direct-AFD	0.0369	0.1049	0.3530	1.3087	5.1891	24.3431

Abbreviations: AFD, adaptive Fourier decomposition; FFT, fast Fourier transform.

4.2 | The influence of the length of samples

The proposed algorithm in Section 3 is of $\mathcal{O}\left(MN\log_2 N\right)$ computation complexity. Compared to Direct-AFD with the computational complexity $\mathcal{O}\left(MN^2\right)$, the running time should be obviously improved with an increased length of samples of the original function. Hereby, the original signal f_1 is sampled by 128, 256, 512, 1024, 2048 points respectively. We observe the running times of the proposed algorithm approximating 10 steps with the sampled signals. The running time can be found in Table 9.

The parameters have a little difference from those from Direct-AFD, when $N = 2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}$, respectively. Moreover, it does not cause any influence on the relative errors in this example.

Remark 4.1. Compared to the ideas contained in Gao et al,[3] our algorithm is much feasible and efficient to occupy much less ram and to cost much less time. This is because our algorithm can be executed by directly calling arbitrary software packages, which integrate FFT while they are being continuously updated and optimized in engineering.

Remark 4.2. In our algorithm presented in Section 3.2 and the numerical experiments in Section 4, we have considered the case of the dynamic parameter choice $N = 2^K$. Moreover, following the argument contained in Rader,[22] our algorithm is still valid when the dynamic parameter choice $N = p^K$ with p being an arbitrary prime is considered.

ACKNOWLEDGEMENTS

This work was supported in part by the Macao Government FDCT 079/2016/A2, by Multi-Year Research Grant (MYRG) MYRG2016-00053-FST, and by Macao Government FDCT 099/2014/A2. The authors cordially thank two anonymous referees for their valuable comments that lead to the improvement of this paper.

ORCID

You Gao http://orcid.org/0000-0001-6548-6433
Min Ku http://orcid.org/0000-0002-5580-8248

REFERENCES

1. Qian T, Wang YB. Adaptive Fourier series—a variation of greedy method. *Adv Comput Math*. 2010;34(3):279-293.
2. Qian T. Two-dimensional adaptive Fourier decomposition. *Math Methods Appl Sci*. 2015;39(10):2431-2448.
3. Gao Y, Ku M, Qian T, Wang JZ. FFT Formulations of adaptive Fourier decomposition. *J Comput Appl Math*. 2017;324:204-215.
4. Qian T. Intrinsic mono-component decomposition of functions: an advance of Fourier theory. *Math Methods Appl Sci*. 2010;33:880-891.
5. Qian T, Wegert E. Optimal approximation by Blaschke forms. *Complex Var Elliptic Eq*. 2013;58:123-133.
6. Qian T. Cyclic AFD algorithm for best approximation by rational functions of given order. *Math Methods Appl Sci*. 2013;37(6):846-859.
7. Davis G, Mallat S, Avellaneda M. Adaptive greedy approximations. *Constr Approx*. 1997;13(1):57-98.
8. Devore RA, Temlyakov VN. Some remarks on greedy algorithm. *Adv Comput Math*. 1996;5:173-187.
9. Mo Y, Qian T. Support vector machine adapted Tikhonov regularization method to solve Dirichlet problem. *Appl Math Comput*. 2014;245:509-519.
10. Li S, Qian T, Mai WX. Sparse reconstruction of Hardy signal and applications to time-frequency distribution. *Int J Wavelets Multiresolution Inf Process*. 2013;11:1350031.
11. Mi W, Qian T. Frequency-domain identification: an method based on an adaptive rational orthogonal system. *Automatica*. 2012;48(6):1154-1162.
12. Dang P, Qian T. Transient time-frequency distribution based on mono-component decomposition. *Int J Wavelets Multiresolution Inf Process*. 2013;11:1-24.
13. Qian T, Sprössig W, Wang JX. Adaptive Fourier decomposition of functions in quaternionic Hardy spaces. *Math Methods Appl Sci*. 2012;35:43-64.

14. Qian T, Wang JX, Yang Y. Matching pursuits among shifted Cauchy kernels in higher-dimensional spaces. *Acta Math Sci*. 2014;34(3):660-672.
15. Qian T, Zhang L, Li Z. Algorithms of adaptive Fourier decomposition. *IEEE Trans Signal Process*. 2011;59(12):5899-5906.
16. Gabor D. Theory of communication. Part 1: the analysis of information. *J Inst Electr Eng - Part III: Radio Commun Eng*. 1946;93(26):429-441.
17. Garnett J. *Bounded analytic functions, Graduate Texts in Mathematics 236*. San Francisco: Academic Press, New York; 1981.
18. Bultheel A, Carrette P. Takenaka-Malmquist basis and general Toeplitz matrices. In: IEEE Proceedings of the 42nd CDC Conference, Vol. 1; 2003; Maui, Hawaii. 486-491.
19. Akċay H, Ninness B. Orthonormal basis functions for modelling continuous time systems. *Signal Process*. 1999;77(3):261-274.
20. Wahlberg B. Orthogonal rational functions: a transformation analysis. *SIAM Rev*. 2003;45(4):689-705.
21. Frigo M, Johnson SG. The design and implementation of FFTW3. *Proc IEEE*. 2005;93(2):216-231.
22. Rader CM. Discrete Fourier transforms when the number of data samples is prime. *Proc IEEE*. 1968;56:1107-1108.

How to cite this article: Gao Y, Ku M, Qian T. Fast algorithm of adaptive Fourier series. *Math Meth Appl Sci*. 2018;41:2654–2663. https://doi.org/10.1002/mma.4767