symmetry

MDPI

Article

The Cauchy Conjugate Gradient Algorithm with Random Fourier Features

Xuewei Huang [1,2], Shiyuan Wang [1,2,*] and Kui Xiong [1,2]

[1] College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China; h870826158@email.swu.edu.cn (X.H.); xiongk@email.swu.edu.cn (K.X.)

[2] Chongqing Key Laboratory of Nonlinear Circuits and Intelligent Information Processing, Chongqing 400715, China

* Correspondence: wsy@swu.edu.cn

Received: 23 September 2019; Accepted: 17 October 2019; Published: 22 October 2019

Abstract: Random Fourier mapping (RFM) in kernel adaptive filters (KAFs) provides an efficient method to curb the linear growth of the dictionary by projecting the original input data into a finite-dimensional space. The commonly used measure in RFM-based KAFs is the minimum mean square error (MMSE), which causes performance deterioration in the presence of non-Gaussian noises. To address this issue, the minimum Cauchy loss (MCL) criterion has been successfully applied for combating non-Gaussian noises in KAFs. However, these KAFs using the well-known stochastic gradient descent (SGD) optimization method may suffer from slow convergence rate and low filtering accuracy. To this end, we propose a novel robust random Fourier features Cauchy conjugate gradient (RFFCCG) algorithm using the conjugate gradient (CG) optimization method in this paper. The proposed RFFCCG algorithm with low complexity can achieve better filtering performance than the KAFs with sparsification, such as the kernel recursive maximum correntropy algorithm with novelty criterion (KRMC-NC), in stationary and non-stationary environments. Monte Carlo simulations conducted in the time-series prediction and nonlinear system identification confirm the superiorities of the proposed algorithm.

Keywords: random Fourier mapping; minimum Cauchy loss; non-Gaussian noise; conjugate gradient; complexity

1. Introduction

Many applications in the real world, such as system identification, regression, and online kernel learning (OKL) [1], require complex nonlinear models. The kernel method using a Mercer kernel has attracted interests in tackling these complex nonlinear applications, which transforms nonlinear applications into linear ones in the reproducing kernel Hilbert space (RKHS) [2]. Developed in RKHS, a kernel adaptive filter (KAF) [2] is the most celebrated subfield of OKL algorithms. Using the simplest stochastic gradient descent (SGD) method for learning, KAFs including the kernel least mean square (KLMS) algorithm [3], kernel affine projection algorithm (KAPA) [4], and kernel recursive least squares (KRLS) algorithm [5] have been proposed.

However, allocating a new kernel unit as a radial basis function (RBF) center with the coming of new data, the linearly growing structure (called "dictionary" hereafter) will increase the computational and memory requirements in KAFs. To curb the growth of the dictionary, two categories are chosen for sparsification. The first category accepts only informative data as new dictionary centers by using a threshold, including the surprise criterion (SC) [6], the coherence criterion (CC) [7], and the vector quantization (VQ) [8]. However, these methods cannot fully address the growing problem and still introduce additional time consumption at each iteration. The fixed points methods as

the second category, including the fixed-budget (FB) [9], the sliding window (SW) [10], and the kernel approximation methods (e.g., the Nyström method [11] and random Fourier features (RFFs) method [12]), are used to overcome the sublinearly growing problem. However, the FB method and the SW method cannot guarantee a good performance in specific environments with a small amount of time [13]. Compared with the Nyström method, RFFs are drawn from a distribution that is randomly independent from the training data. Due to a data-independent vector representation, RFFs can provide a good solution to non-stationary circumstances. On the basis of RFFs, random Fourier mapping (RFM) is proposed by mapping input data into a finite-dimensional random Fourier features space (RFFS) using a randomized feature kernel's Fourier transform in a fixed network structure. The RFM alleviates the computational and storage burdens of KAFs, and ensures a satisfactory performance under non-stationary conditions. The examples for developing KAFs with RFM are the random Fourier features kernel least mean square (RFFKLMS) algorithm [13], random Fourier features maximum correntropy (RFFMC) algorithm [14], and random Fourier features conjugate gradient (RFFCG) algorithm [15].

For the loss function, due to their simplicity, smoothness, and mathematical tractability, the second-order statistical measures (e.g., minimum mean square error (MMSE) [2] and least squares [16]) are widely utilized in KAFs. However, KAFs based on the second-order statistical measures are sensitive to non-Gaussian noises including the sub-Gaussian and super-Gaussian noises, which means that their performance may be seriously degraded if the training data are contaminated by outliers. To handle this issue, robust statistical measures have therefore gained more attention, among which the lower-order error measure [17] and the higher-lower error measure [18] are two typical examples. However, the higher-order error measure is not suitable for the mixture of Gaussian and super-Gaussian noises (Laplace, α-stable, etc.) with poor stability and astringency, and the lower-order measure of error is usually more desirable in these noise environments with slow convergence rate. Recently, the information theoretic learning (ITL) [19] similarity measures, such as the maximum correntropy criterion (MCC) [20] and minimum error entropy criterion (MEE) [19], have been introduced to implement robust KAFs. The ITL similarity measures have been shown to have a strong robustness against non-Gaussian noises at the expense of increasing computational burden in training processing. In addition, minimizing the logarithmic moments of the error, the logarithmic error measure—including the Cauchy loss (CL) [21] with low computational complexity—is an appropriate measure of optimality. Using the Cauchy loss to penalize the noise term, some algorithms based on the minimum Cauchy loss (MCL) criterion are efficient for combating non-Gaussian noises, especially for heavy-tailed α- stable noises.

From the aspect of the optimization method, the stochastic gradient descent (SGD)-based algorithms cannot find the minimum using the negative gradient in some loss functions [20–22]. Toward this end, recursive-based algorithms [23] address these issues at the cost of increasing computational cost. In comparison with the SGD method and recursive method, the conjugate gradient (CG) method [24–26] and Newton's method as developments of SGD have become alternative optimization methods in KAFs. The inverse of matrix of Newton's method increases the computation and causes the divergence of algorithms in some cases [22]. However, the CG method gives a trade-off between convergence rate and computational complexity without the inverse computation, and has been successfully applied in various fields, including compressed sensing [27], neural networks [28], and large-scale optimization [29]. In addition, the kernel conjugate gradient (KCG) method is proposed [30] for adaptive filtering. KCG with low computational and space requirements can produce a better solution than KLMS, and has comparable accuracy to KRLS.

In this paper, to reduce the computational complexity, we apply the RFM in the MCL-based KAF to address the problem of linear growth and improve the robustness. Further, the CG optimization method is used to improve the filtering accuracy and convergence rate, developing a novel robust random Fourier features Cauchy conjugate gradient (RFFCCG) algorithm. The contributions of this paper are summarized as follows. 1) Inspired by the finite-dimensional RFM and MCL criterion,

a novel RFFCCG algorithm is derived by mapping the original input data into the fixed-dimensional RFFS, which can significantly solve the problem of the growth of network structure and improve robustness compared to other robust algorithms in the context of non-Gaussian noises. 2) By applying the CG method, RFFCCG with low computational and space complexities provides good filtering accuracy against non-Gaussian noises. The computational and space complexities of RFFCCG are also discussed. 3) The proposed algorithm can also achieve excellent tracking performance when a system has a sudden change.

The rest of this paper is structured as follows. The MCL criterion and its convexity are described in Section 2, and the online CG algorithm is also briefly reviewed in this section. In Section 3, we present the proposed RFFCCG algorithm and its complexity analysis. Illustrative simulations in the presence of non-Gaussian noises are presented to confirm the effectiveness of the proposed algorithm in Section 4. Finally, Section 5 gives the concluding remarks of this paper.

2. Background

In this section, we first briefly review the minimum Cauchy loss (MCL) criterion. The performance surfaces of Cauchy loss (CL) and mean square error (MSE) are also compared. Then, the conjugate gradient (CG) method and its online algorithm are introduced.

2.1. Minimum Cauchy Loss Criterion

Given two random variables X and Y, the Cauchy loss [21] is defined as:

$$\begin{aligned} V(X,Y) &= \mathrm{E}\left[\ln\left(1+\frac{(X-Y)^2}{\gamma^2}\right)\right] \\ &= \int \ln\left(1+\frac{(X-Y)^2}{\gamma^2}\right) dF_{XY}(x,y), \end{aligned} \tag{1}$$

where $\mathrm{E}[\cdot]$ denotes the mathematical expectation, $F_{XY}(x,y)$ denotes the joint distribution function of (X,Y), and $\gamma > 0$ is a constant. Since $F_{XY}(x,y)$ is usually unknown, it is difficult to calculate $V(X,Y)$ directly. In practice, given a finite number of samples $\{x_k, y_k\}_{k=1}^N$, (1) can be approximated as follows:

$$\hat{V}(X,Y) = \frac{1}{N}\sum_{k=1}^{N} \ln\left(1+\frac{(x_k-y_k)^2}{\gamma^2}\right), \tag{2}$$

which is called the empirical CL. In addition, (2) can also be regarded as a generalized error between two vectors $\mathbf{X} = [x_1, x_2, ..., x_N]^{\mathrm{T}}$ and $\mathbf{Y} = [y_1, y_2, ..., y_N]^{\mathrm{T}}$.

Let $\mathbf{e} = \mathbf{X} - \mathbf{Y} = [e_1, e_2, ..., e_N]^{\mathrm{T}}$, where $e_k = x_k - y_k$, $k = 1, 2, ...N$. Because of (2), the Hessian matrix of $\hat{V}(\mathbf{X}, \mathbf{Y})$ with respect to $\mathbf{e}$ is expressed as:

$$\mathbf{H}_{\hat{V}(\mathbf{X},\mathbf{Y})}(\mathbf{e}) = \left[\frac{\partial^2 \hat{V}(\mathbf{X},\mathbf{Y})}{\partial e_k \partial e_j}\right] = \mathrm{diag}[\xi_1, \xi_2, ..., \xi_N], \tag{3}$$

where

$$\xi_k = \frac{2\left(\gamma^2 - e_k^2\right)}{\left(\gamma^2 + e_k^2\right)^2}, k = 1, 2, ..., N. \tag{4}$$

We have that $\mathbf{H}_{\hat{V}(\mathbf{X},\mathbf{Y})}(\mathbf{e}) \geq \mathbf{0}$ when $|e_k| \leq \gamma$, that is, the empirical CL is convex at $|e_k| \leq \gamma$. A larger γ results in a larger convex range in general.

The optimal solution to the Cauchy loss function can be obtained by solving the following optimization problem:

$$\min \sum_{k=1}^{N} \ln \left(1 + \frac{(d_k - y_k)^2}{\gamma^2}\right), \tag{5}$$

which is called the minimum Cauchy loss (MCL) criterion. Figure 1 shows the comparison of MSE and CL functions with different γ. We can observe that: 1) compared with the MSE loss function, the CL function maintains the insensitivity to large errors. Thus, adaptive filters using the CL function will be robust against large outliers. 2) The value of γ can control the shape of the CL function, and a larger γ can generate a smoother curve when the error is smaller, which means that the CL function can provide good smoothness to the steady-state error. In practice, we choose an appropriate γ to guarantee the robustness and convexity of the CL function.

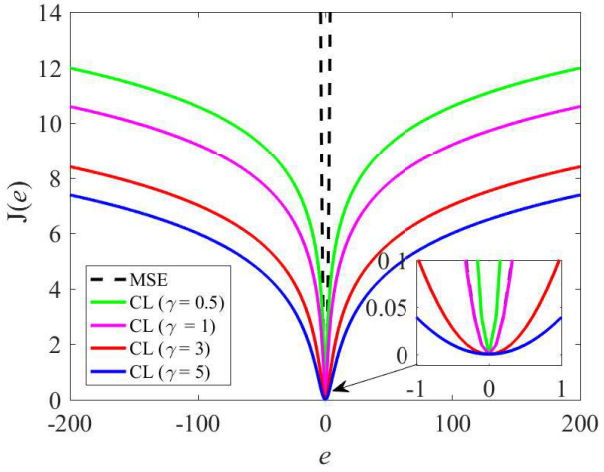

Figure 1. Comparison of mean square error (MSE) and Cauchy loss (CL) with different γ.

2.2. Conjugate Gradient Algorithm

The typical conjugate direction method is the conjugate gradient (CG) method, which is developed by selecting the successive direction vectors as conjugate versions of the successive gradients. The CG method is introduced to a linear adaptive filter in [24,26], which is used to solve the following linear equation:

$$\mathbf{R}\boldsymbol{\omega} = \mathbf{b} \tag{6}$$

and (6) is also equivalent to solve the following purely quadratic function:

$$f(\boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{\omega}^{\mathrm{T}}\mathbf{R}\boldsymbol{\omega} - \mathbf{b}^{\mathrm{T}}\boldsymbol{\omega}. \tag{7}$$

where $\boldsymbol{\omega} \in \mathbb{R}^n$ is the weight vector, $\mathbf{b} \in \mathbb{R}^n$ is the cross-correlation vector, and $\mathbf{R} \in \mathbb{R}^{n \times n}$ is a symmetric positive definite auto-correlation matrix. To find the optimal $\boldsymbol{\omega}$, the CG method—which is described as follows—provides an alternative method instead of estimating $\mathbf{R}^{-1}$. Beginning with any $\boldsymbol{\omega}_0 \in \mathbb{R}^n$, and direction $\mathbf{p}_0 = -\mathbf{g}_0 = \mathbf{b} - \mathbf{R}\boldsymbol{\omega}_0$, the global minimum to (6) can be derived by iteratively computing

$$\begin{cases} \alpha_k = -\frac{\mathbf{g}_k^{\mathrm{T}}\mathbf{p}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{R}\mathbf{p}_k} \\ \boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \alpha_k\mathbf{p}_k \\ \mathbf{g}_{k+1} = \mathbf{b} - \mathbf{R}\boldsymbol{\omega}_{k+1} \\ \beta_k = \frac{\mathbf{g}_{k+1}^{\mathrm{T}}\mathbf{g}_{k+1}}{\mathbf{g}_k^{\mathrm{T}}\mathbf{g}_k} \\ \mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \beta_k\mathbf{p}_k \end{cases}, \tag{8}$$

where $(\cdot)^{\mathrm{T}}$ denotes the transpose, $\mathbf{g}_k$ is the gradient of $f(\boldsymbol{\omega}_k)$ with respect to $\boldsymbol{\omega}$ at discrete time k, the step-size α_k is given by $\arg\min f(\boldsymbol{\omega}_k + \alpha\mathbf{p}_k)$, and constant β_k is selected to provide $\mathbf{R}$-conjugacy for vector $\mathbf{p}_k$ regarding the previous direction vectors $\mathbf{p}_{k-1}, \mathbf{p}_{k-2}, ..., \mathbf{p}_0$. In (8), note that the new conjugate direction $\mathbf{p}_k$ is formed by a linear combination of the current negative gradient $-\mathbf{g}_k$ and the previous direction vectors with a proper β_k to avoid manually resetting the direction vector.

The CG method can be regarded as a trade-off between the stochastic gradient descent (SGD) method and Newton's method in terms of convergence rate and the complexities of computation and storage. Especially, the inverse of the Hessian matrix—which will cause the divergence of the algorithm—is avoided in the CG method. Therefore, the CG method is widely applicable to address quadratic optimization problems. In addition, it can also be extended to approximate non-quadratic optimization problems.

2.3. Online Conjugate Gradient Algorithm

The aforementioned CG method is generally used for offline applications. In this section, the online CG algorithm is given for online learning, where training data arrive sequentially [1,30].

It is necessary to estimate $\mathbf{R}$ and $\mathbf{b}$ for online applications. Using the exponentially decaying data window [2], the following recursive form to update $\mathbf{R}$ and $\mathbf{b}$ is given by

$$\begin{aligned} \mathbf{R}_{k+1} &= \lambda\mathbf{R}_k + \mathbf{x}_{k+1}\mathbf{x}_{k+1}^{\mathrm{T}} \\ \mathbf{b}_{k+1} &= \lambda\mathbf{b}_k + d_{k+1}\mathbf{x}_{k+1}, \end{aligned} \tag{9}$$

where positive constant $0 < \lambda < 1$ is usually closest to one, $\mathbf{x}_k \in \mathbb{R}^n$ is the input data, and d_k is the desired output at iteration k.

Define a residual vector of normal equations as $\mathbf{s}_k = \mathbf{b}_k - \mathbf{R}_k\boldsymbol{\omega}_k$ at iteration k. To improve clarity, the online CG algorithm to estimate the weight vector is summarized as follows [15]. Given the initial conditions $\boldsymbol{\omega}_0 = 0$ and $\mathbf{p}_0 = \mathbf{s}_0$, when $\{\mathbf{x}_k, d_k\}$ is available, do

$$\begin{cases} \alpha_k = \frac{\mathbf{s}_k^{\mathrm{T}}\mathbf{p}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{R}_{k+1}\mathbf{p}_k} \\ \boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \alpha_k\mathbf{p}_k \\ \mathbf{R}_{k+1} = \lambda\mathbf{R}_k + \mathbf{x}_{k+1}\mathbf{x}_{k+1}^{\mathrm{T}} \\ \mathbf{s}_{k+1} = \lambda\mathbf{s}_k - \alpha_k\mathbf{R}_{k+1}\mathbf{p}_k + e_{k+1}\mathbf{x}_{k+1} \\ \beta_k = \frac{\mathbf{s}_{k+1}^{\mathrm{T}}\mathbf{s}_{k+1}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{s}_k} \\ \mathbf{p}_{k+1} = \mathbf{s}_{k+1} + \beta_k\mathbf{p}_k \end{cases}. \tag{10}$$

The online CG algorithm is only efficient in dealing with linear problems, but may cause degradation for nonlinear problems. To address this problem, the online kernel conjugate gradient (KCG) based on the least squares (LS) is presented in [30]. Because the LS criterion is used in KCG, it may be degraded considerably or even suffer from divergence in non-Gaussian noise environments. In addition, the linearly growing structure of KCG with each new sample poses both computational and memory issues. To address the complexity issue of KCG, RFFCG is proposed by using the CG

method. However, using the LS criterion, RFFCG is only appropriate for Gaussian noises. Therefore, we propose a novel online algorithm to deal with these issues.

3. Proposed Algorithm

In this section, we first describe a fixed dimensional mapping in random Fourier features space (RFFS). Then, the CG method can be applied to the MCL criterion to generate a robust algorithm—that is, the robust random Fourier features Cauchy conjugate gradient (RFFCCG) algorithm.

3.1. Random Fourier Mapping

Based on an available sequence of training data pairs $\{\mathbf{x}_k, d_k\}_{k=1}^{N}$, kernel adaptive filters are used to deal with the following nonlinear problem:

$$d_k = f^*(\mathbf{x}_k) + v_k, \tag{11}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the input data, $d_k \in \mathbb{R}$ is the desirable output at iteration k in the original data space $\mathbb{U}$, v_k is an additive noise signal, and $f^*(\mathbf{x}_k)$ is the optimal estimate of d_k with a nonlinear input–output mapping $f : \mathbb{R}^n \to \mathbb{R}$.

To construct the mapping relationship in the form of $f(\cdot) = \sum_{l=1}^{k} \eta_l \kappa(\cdot, \mathbf{x}_l)$, a kernel method [2] is utilized to map the original input data into a high-dimensional feature space. A continuous, symmetric, and positive definite Mercer kernel is used in the kernel method. Thus, using the kernel method, we can compute the filter output at discrete time k as

$$f(\mathbf{x}_k) = \sum_{l=1}^{k-1} \eta_l \kappa(\mathbf{x}_k, \mathbf{x}_l), \tag{12}$$

with coefficients η_l, $l = 1, 2, ...k-1$. We observe that the network increases linearly with the length of data to train in the filter output, which poses some challenges to the kernel method in practice. However, the random Fourier features (RFFs) present an efficient solution to solving this problem by transforming input $\mathbf{x}_k$ into $\mathbf{z}(\mathbf{x}_k)$, that is,

$$\mathbf{z} : \mathbf{x}_k \in \mathbb{R}^n \to \mathbf{z}(\mathbf{x}_k) \in \mathbb{R}^m, \tag{13}$$

where $m \gg n$. Considering a shift-invariant and positive definite kernel $\kappa(\mathbf{x}, \mathbf{y}) = \kappa(\mathbf{x} - \mathbf{y})$ on $\mathbb{R}^n$, Bochner's theorem [12] ensures that the Fourier transform of $\rho(\mathbf{w})$ corresponds to the kernel function given by

$$\kappa(\mathbf{x} - \mathbf{y}) = \int_{\mathbb{R}^n} \rho(\mathbf{w}) e^{j\mathbf{w}^{\mathrm{T}}(\mathbf{x}-\mathbf{y})} d\mathbf{w}, \tag{14}$$

where $\rho(\cdot)$ denotes the probability density function (PDF) and $\mathbf{w}$ is a Gaussian random vector drawn from $\rho(\mathbf{w}) \sim N(0, \sigma^2 \mathbf{I}_n)$ with an $n \times n$ identity matrix $\mathbf{I}_n$. In (14), the commonly used Mercer kernel is the following Gaussian kernel [2], that is,

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|_2^2}{2\sigma^2}\right), \tag{15}$$

where σ is the kernel bandwidth and $\|\cdot\|_2$ is the ℓ_2 norm.

Now, define $\mathbf{Z}_{\mathbf{w}}(\mathbf{x}) = e^{j\mathbf{w}^{\mathrm{T}}\mathbf{x}}$, and (14) is actually equivalent to the expectation operation of $\mathbf{Z}_{\mathbf{w}}^{\mathrm{H}}(\mathbf{x})\mathbf{Z}_{\mathbf{w}}(\mathbf{y})$. Then, we have the following expression for kernel function:

$$\kappa(\mathbf{x} - \mathbf{y}) = E_{\mathbf{w}}[\mathbf{Z}_{\mathbf{w}}^{\mathrm{H}}(\mathbf{x})\mathbf{Z}_{\mathbf{w}}(\mathbf{y})], \tag{16}$$

where H is the conjugate transpose operation and

$$\begin{aligned}\mathbf{Z}_{\mathbf{w}}(\mathbf{x}) &= e^{j\mathbf{w}^{\mathrm{T}}\mathbf{x}} = \sqrt{2}\cos(\mathbf{w}^{\mathrm{T}}\mathbf{x}+b)\\ \mathbf{Z}_{\mathbf{w}}(\mathbf{y}) &= e^{j\mathbf{w}^{\mathrm{T}}\mathbf{y}} = \sqrt{2}\cos(\mathbf{w}^{\mathrm{T}}\mathbf{y}+b),\end{aligned} \tag{17}$$

with b being drawn from the uniform distribution on $[0, 2\pi]$ [12].

Based on m random Fourier features $\mathbf{w}_1, ..., \mathbf{w}_m$ and m uniformly random numbers $b_1, ..., b_m$, the approximation of kernel function in (16) can be rewritten as the empirical average of m random components:

$$\mathbf{Z}^{\mathrm{T}}_{\mathbf{w},b}(\mathbf{x})\mathbf{Z}_{\mathbf{w},b}(\mathbf{y}) \approx \frac{1}{m}\sum_{i=1}^{m}\mathbf{Z}_{\mathbf{w}_i,b_i}(\mathbf{x})\mathbf{Z}_{\mathbf{w}_i,b_i}(\mathbf{y}), \tag{18}$$

where

$$\begin{aligned}\mathbf{Z}_{\mathbf{w},b}(\mathbf{x}) &= [\mathbf{Z}_{\mathbf{w}_i,b_i}(\mathbf{x}), ..., \mathbf{Z}_{\mathbf{w}_m,b_m}(\mathbf{x})]^{\mathrm{T}}\\ &= \left[\sqrt{\frac{2}{m}}\cos(\mathbf{w}_1^{\mathrm{T}}\mathbf{x}+b_1), ..., \sqrt{\frac{2}{m}}\cos(\mathbf{w}_m^{\mathrm{T}}\mathbf{x}+b_m)\right]^{\mathrm{T}}.\end{aligned} \tag{19}$$

The random Fourier features method for approximating the Gaussian kernel is similar to the approximation method of standard Monte Carlo. Therefore, the transformed input data $\mathbf{z}(\mathbf{x}_k)$ are

$$\mathbf{z}(\mathbf{x}_k) = \sqrt{\frac{2}{m}}\left[\cos(\mathbf{w}_1^{\mathrm{T}}\mathbf{x}_k+b_1), ..., \cos(\mathbf{w}_m^{\mathrm{T}}\mathbf{x}_k+b_m)\right]^{\mathrm{T}}, \tag{20}$$

which is called random Fourier mapping (RFM). The dimension of subspace that belongs to $\mathbf{z}(\mathbf{x}_k)$ is finite. Therefore, a linear adaptive filtering structure is given based on the transformed input $\mathbf{z}(\mathbf{x}_k)$.

The Gaussian kernel function in (12) can be represented by the inner product of the transformed input vector as

$$\kappa(\mathbf{x}_k, \mathbf{x}_l) = \mathbf{z}^{\mathrm{T}}(\mathbf{x}_k)\mathbf{z}(\mathbf{x}_l), \tag{21}$$

with the low approximation error ε provided by $m = O(n\varepsilon^{-2}\log\frac{1}{\varepsilon^{-2}})$ [14].

Combining (12) and (21), the filter output can be recomputed by $f(\mathbf{x}_k) = (\mathbf{\Omega}^r)^{\mathrm{T}}\mathbf{z}(\mathbf{x}_k)$, where $\mathbf{\Omega}^r$ is a finite-dimensional weight vector in RFFS. Thus, an infinite-dimensional implicit feature space is embedded into a relatively low-dimensional explicit feature space. The expression of filter output is similar to the linear least mean square (LMS) [2]. Therefore, a huge amount of time is saved in the RFM-based algorithms. The fixed-dimensional structure will be used to develop the following algorithm. Denote $\mathbf{z}(\mathbf{x}_k) = \mathbf{z}_k$ for simplicity hereafter.

3.2. RFFCCG Algorithm

The Cauchy loss function is presented for robust adaptive filtering as follows:

$$J_{\mathbf{\Omega}^r} = \frac{1}{N}\sum_{k=1}^{N}\ln\left(1+\frac{e_k^2}{\gamma^2}\right) = \frac{1}{N}\sum_{k=1}^{N}\ln\left(1+\frac{\left(d_k-(\mathbf{\Omega}^r)^{\mathrm{T}}\mathbf{z}_k\right)^2}{\gamma^2}\right), \tag{22}$$

where $e_k = d_k - (\mathbf{\Omega}^r)^{\mathrm{T}}\mathbf{z}_k$ is the prediction error regarding the transformed input $\mathbf{z}_k \in \mathbb{R}^m$, $d_k \in \mathbb{R}$ is the desired output, and $\mathbf{\Omega}^r \in \mathbb{R}^m$ is the weight vector in RFFS. From (5), $(\mathbf{\Omega}^r)^{\mathrm{T}}\mathbf{z}_k$ corresponds to y_k.

The gradient of the loss function with respect to $\mathbf{\Omega}^r$ is given by

$$\begin{aligned}\nabla J_{\mathbf{\Omega}^r} &= -\frac{1}{N}\sum_{k=1}^{N}\frac{2}{\gamma^2+e_k^2}\left(d_k-(\mathbf{\Omega}^r)^{\mathrm{T}}\mathbf{z}_k\right)\mathbf{z}_k \\ &= \frac{1}{N}\sum_{k=1}^{N}\frac{2}{\gamma^2+e_k^2}d_k\mathbf{z}_k-\frac{1}{N}\sum_{k=1}^{N}\frac{2}{\gamma^2+e_k^2}\mathbf{z}_k\mathbf{z}_k^{\mathrm{T}}\mathbf{\Omega}^r \\ &= \frac{1}{N}\sum_{k=1}^{N}\theta_k d_k\mathbf{z}_k-\frac{1}{N}\sum_{k=1}^{N}\theta_k\mathbf{z}_k\mathbf{z}_k^{\mathrm{T}}\mathbf{\Omega}^r,\end{aligned} \tag{23}$$

where $\theta_k=\frac{2}{\gamma^2+e_k^2}$. Hence, we have that θ_k tends to 0 as $e_k\to\infty$.

Letting $\nabla J_{\mathbf{\Omega}^r}=\mathbf{0}$ generates

$$\sum_{k=1}^{N}\theta_k\mathbf{z}_k\mathbf{z}_k^{\mathrm{T}}\mathbf{\Omega}^r=\sum_{k=1}^{N}\theta_k d_k\mathbf{z}_k, \tag{24}$$

where weighted auto-correlation matrix $\mathbf{R}^r$ and weighted cross-correlation vector $\mathbf{b}^r$ are given by

$$\begin{cases}\mathbf{R}^r=\sum_{k=1}^{N}\theta_k\mathbf{z}_k\mathbf{z}_k{}^{\mathrm{T}}\in\mathbb{R}^{m\times m}\\ \mathbf{b}^r=\sum_{k=1}^{N}\theta_k d_k\mathbf{z}_k\in\mathbb{R}^m\end{cases}. \tag{25}$$

According to (24), (25) can be rewritten as

$$\mathbf{R}^r\mathbf{\Omega}^r=\mathbf{b}^r, \tag{26}$$

where the optimal solution $\mathbf{\Omega}^r$, in practice, can be obtained by the CG method instead of estimating $(\mathbf{R}^r)^{-1}$.

For online learning in RFFS, $\mathbf{R}^r$ and $\mathbf{b}^r$ are given by using an exponentially decaying data window as follows:

$$\begin{cases}\mathbf{R}^r_{k+1}=\lambda\mathbf{R}^r_k+\theta_k\mathbf{z}_{k+1}\mathbf{z}^{\mathrm{T}}_{k+1}\\ \mathbf{b}^r_{k+1}=\lambda\mathbf{b}^r_k+\theta_k d_{k+1}\mathbf{z}_{k+1}\end{cases}, \tag{27}$$

where positive forgetting factor λ which is very close to but smaller than one (i.e., $\lambda\in(0,1)$) is used to scale down past data. Since θ_k tends to 0 as $e_k\to\infty$ for outliers, $\mathbf{R}^r_{k+1}$ and $\mathbf{b}^r_{k+1}$ have almost no change from (27). In such case, according to (26), $\mathbf{\Omega}^r$ almost has no update, which is robust against outliers.

Then, we use an important concept called the search direction vector here. A fundamental relation between the current optimal weight and the previous optimal weight in RFFS is obtained [25]. We have the recursion to update $\mathbf{\Omega}^r$ as

$$\mathbf{\Omega}^r_{k+1}=\mathbf{\Omega}^r_k+\alpha_k\mathbf{p}^r_k, \tag{28}$$

which suggests that we can estimate the optimal weight $\mathbf{\Omega}^r_{k+1}$ of a nonlinear dynamical system by combining the previous $\mathbf{\Omega}^r_k$ with the search direction vector $\mathbf{p}^r_k$ multiplied by a step-size parameter α_k. Furthermore, by the conjugate gradient theory [24,25], we can express α_k as

$$\alpha_k=\frac{(\mathbf{s}^r_k)^{\mathrm{T}}\mathbf{p}^r_k}{(\mathbf{p}^r_k)^{\mathrm{T}}\mathbf{R}^r_{k+1}\mathbf{p}^r_k}. \tag{29}$$

Now, a residual vector of normal equations $\mathbf{s}^r$ is introduced in the kernel space. From (27)–(29), we have the recursive residual vector of RFFCCG as follows:

$$\begin{aligned}\mathbf{s}_{k+1}^r &= \mathbf{b}_{k+1}^r - \mathbf{R}_{k+1}^r \mathbf{\Omega}_{k+1}^r \\ &= \lambda \mathbf{s}_k^r - \alpha_k \mathbf{R}_{k+1}^r \mathbf{p}_k^r + \mathbf{z}_{k+1} \theta_{k+1} e_{k+1},\end{aligned} \tag{30}$$

where $e_{k+1} = d_{k+1} - (\mathbf{\Omega}_k^r)^{\mathrm{T}} \mathbf{z}_{k+1}$ is the prediction error estimated by the desired and the real output in RFFS. These residual vectors are orthogonal to each other, that is, $(\mathbf{s}_k^r)^{\mathrm{T}} \mathbf{s}_l^r = 0$, for $l = 0, ..., k-1$.

In [25], the Hessian matrix must be recalculated at each iteration for the Hestenes–Stiefel method, which requires a large amount of computation, and the numerator of the Fletcher–Reeves method may be close zero, resulting in poor performance. From the results of the global convergence characteristics analysis, using the Polak–Ribière method [22] which adopts a degenerated scheme to calculate β_k, the performance of the conjugate gradient is the best. Thus, a proper coefficient parameter β_k (k = 1,2,...) is expressed to update the current search direction $\mathbf{p}_{k+1}^r$ automatically by

$$\beta_k = \frac{\left(\mathbf{s}_{k+1}^r\right)^{\mathrm{T}} (\mathbf{s}_{k+1}^r - \mathbf{s}_k^r)}{\left(\mathbf{s}_k^r\right)^{\mathrm{T}} \mathbf{s}_k^r}. \tag{31}$$

By using (31) directly, a linear formulation of the new search direction is given by

$$\mathbf{p}_{k+1}^r = \mathbf{s}_{k+1}^r + \beta_k \mathbf{p}_k^r, \tag{32}$$

where β_k is set to provide the $\mathbf{R}$-conjugacy for the new search direction $\mathbf{p}_{k+1}^r$ with regard to the previous direction vectors (i.e., $\mathbf{p}_1^r, \mathbf{p}_2^r, ..., \mathbf{p}_k^r$). The search direction is updated per iteration to ensure the convergence of the algorithm. Depending on the spanning subspace theorem [30], the residual vector at iteration k will be orthogonal to the search direction in the Krylov subspace, that is, $\mathbf{s}_k^{\mathrm{H}} \mathbf{p}_l = 0$, for $l < k$.

Finally, combining Equations (27)–(32), we summarize the online RFFCCG algorithm in Algorithm 1.

Remark 1. *The proposed RFFCCG algorithm uses an explicit mapping method to transform the original input into RFFS, generating a linear filtering structure. Based on a data-independent vector representation, RFFs can provide good tracking performance for non-stationary circumstances. The network size of RFFCCG only depends on the dimension m, which plays a significant role in the approximation accuracy. Generally, a larger dimension simultaneously results in a higher filtering accuracy with more computational and storage burdens. In practice, to balance the filtering accuracy and complexity, an appropriate dimension can be chosen by trials to obtain the desirable performance. In addition, we can find from Figure 1 that the Cauchy loss function can provide good robustness to large outliers. Hence, the proposed RFFCCG can also be applied to channel equalization and noise cancellation [2] in non-Gaussian noise environments.*

Algorithm 1: The robust random Fourier features Cauchy conjugate gradient (RFFCCG) algorithm.

Input: Sequential input–output pairs $\{\mathbf{x}_k, d_k\}_{k=1}^{N}$, kernel bandwidth $\sigma > 0$, forgetting factor $\lambda \in (0,1)$, the dimension of RFF $m > 0$, and constant $\gamma > 0$.

Draw: i.i.d. $\mathbf{w} \sim \mathcal{N}(0, \sigma^2 I_n)$, where the dimension of original data space $n > 0$.
i.i.d. $b \sim U[0, 2\pi]$, where U denotes the uniform distribution.

Initialization: $\mathbf{z}_1 = \sqrt{\frac{2}{m}}\left[\cos(\mathbf{w}_1^{\mathrm{T}}\mathbf{x}_1 + b_1), ..., \cos(\mathbf{w}_m^{\mathrm{T}}\mathbf{x}_1 + b_m)\right]^{\mathrm{T}}$,
$\mathbf{\Omega}_1^r = \mathbf{0}$, $d_1 = e_1$, $\theta_1 = \frac{2}{\gamma^2 + e_1^2}$, $\mathbf{R}_1^r = \theta_1 \mathbf{z}_1 \mathbf{z}_1^{\mathrm{T}}$, $\mathbf{b}_1^r = \theta_1 d_1 \mathbf{z}_1$, $\mathbf{p}_1^r = \mathbf{b}_1^r - \mathbf{R}_1^r \mathbf{\Omega}_1^r$, $\mathbf{p}_1^r = \mathbf{s}_1^r$.

Computation:

while $\{\mathbf{x}_k, d_k\}$ is available, do

1. $\mathbf{z}_{k+1} = \sqrt{\frac{2}{m}}\left[\cos(\mathbf{w}_1^{\mathrm{T}}\mathbf{x}_{k+1} + b_1), ..., \cos(\mathbf{w}_m^{\mathrm{T}}\mathbf{x}_{k+1} + b_m)\right]^{\mathrm{T}}$,
2. $y_{k+1} = \left\langle \mathbf{\Omega}_{k+1}^r, \mathbf{z}_{k+1} \right\rangle$,
3. $e_{k+1} = d_{k+1} - y_{k+1}$,
4. $\theta_{k+1} = \frac{2}{\gamma^2 + e_{k+1}^2}$,
5. $\mathbf{R}_{k+1}^r = \lambda \mathbf{R}_k^r + \theta_{k+1} \mathbf{z}_{k+1} \mathbf{z}_{k+1}^{\mathrm{T}}$,
6. $\alpha_{k+1} = \frac{\left(\mathbf{s}_k^r\right)^{\mathrm{T}} \mathbf{p}_k^r}{\left(\mathbf{p}_k^r\right)^{\mathrm{T}} \mathbf{R}_{k+1}^r \mathbf{p}_k^r}$,
7. $\mathbf{\Omega}_{k+1}^r = \mathbf{\Omega}_k^r + \alpha_k \mathbf{p}_k^r$,
8. $\mathbf{s}_{k+1}^r = \lambda \mathbf{s}_k^r - \alpha_k \mathbf{R}_{k+1}^r \mathbf{p}_k^r + \mathbf{z}_{k+1} \theta_{k+1} e_{k+1}$,
9. $\beta_k = \frac{\left(\mathbf{s}_{k+1}^r\right)^{\mathrm{T}} \left(\mathbf{s}_{k+1}^r - \mathbf{s}_k^r\right)}{\left(\mathbf{s}_k^r\right)^{\mathrm{T}} \mathbf{s}_k^r}$,
10. $\mathbf{p}_{k+1}^r = \mathbf{s}_{k+1}^r + \beta_k \mathbf{p}_k^r$.

end while

3.3. Complexity

In this section, we mainly discuss the complexities of RFFCCG in terms of computation and storage. The computational complexity of RFFCCG at each iteration is summarized in detail as follows. It can be seen from Algorithm 1 that at each iteration, RFFCCG requires a total of $4m^2 + (n+12)m + 1$ multiplications from Steps 1, 2, and 5–10; $3m^2 + (n+11)m + 1$ additions from Steps 1, 2, and 4–10; and 4 divisions from Steps 1, 4, 6, and 9. The computation of $\cos(.)$ is not considered here since it can be ignored compared with the cost of RFFCCG. We compare the computational complexities of RFFCCG with the kernel least mean square (KLMS) algorithm [3], kernel recursive least squares (KRLS) algorithm [5], kernel conjugate gradient (KCG) algorithm [30], and kernel recursive maximum correntropy (KRMC) algorithm [31] in Table 1, where k is the number of iterations and n and m are the dimensions of original data space and RFFS, respectively. We clearly see from Table 1 that the proposed RFFCCG algorithm has a fixed computational complexity, but the computational complexities of KLMS, KRLS, KCG, and KRMC increase with the network size. Thus, RFFCCG can significantly reduce the computational requirements.

Further, we discuss the storage complexity of RFFCCG based on the matrix number and size. From Algorithm 1, there is only an $m \times m$ symmetric matrix $\mathbf{R}$ that needs to be stored. Therefore, the storage complexity of RFFCCG is $\mathrm{O}(m)$. Table 2 lists the compared results with other algorithms based on matrix. Note that the storage complexity of RFFCCG is also fixed, and other algorithms have increasing network sizes. In summary, compared with the KAFs without sparsification in Table 1 and Table 2, RFFCCG can efficiently alleviate the computational and storage burdens of KAFs.

Table 1. Computational cost of algorithms per iteration. KCG: kernel conjugate gradient; KLMS: kernel least mean square; KRLS: kernel recursive least squares; KRMC: kernel recursive maximum correntropy.

Algorithm	Addition	Multiplication	Division
1-4 KLMS [3]	k	k	0
KRLS [5]	$4k^2+4k$	$4k^2+4k$	1
KRMC [31]	$4k^2+4k$	$4k^2+4k$	2
KCG [30]	$2k^2+8k$	$2k^2+10k$	3
RFFCCG	$3m^2+(n+11)m+1$	$4m^2+(n+12)m+1$	4

Table 2. Storage cost of algorithms per iteration.

Algorithm	Matrix		
	$m \times m$	$n \times k$	$k \times k$
KLMS [3]	0	1	0
KRLS [5]	0	1	1
KRMC [31]	0	1	1
KCG [30]	0	1	1
RFFCCG	1	0	0

4. Simulation

To demonstrate the superior performance of the proposed RFFCCG algorithm in this section, simulations were performed on the Mackey–Glass chaotic time series prediction and nonlinear system identification, respectively. Due to the modest complexity and excellent performance, representative algorithms (i.e., random Fourier features kernel least mean square (RFFKLMS) algorithm [13], quantized kernel recursive least squares (QKRLS) algorithm [32], random Fourier features maximum correntropy (RFFMC) algorithm [14], kernel recursive maximum correntropy algorithm with novelty criterion (KRMC-NC) [31], and random Fourier features conjugate gradient (RFFCG) algorithm [15]) were selected to compare the performance of RFFCCG. Among these algorithms, RFFMC and KRMC-NC are typical robust algorithms, while RFFKLMS, QKRLS, and RFFCG with no robustness are also used for the filtering performance reference. For all simulations, we ran 50 independent Monte Carlo simulations to reduce disturbances using Matlab R2016b on Windows 10, where PC is configured with 3.30 GHz of CPU and 8 GB of RAM.

To evaluate the filtering performance of algorithms, the testing mean-square error (MSE) is defined as:

$$\mathrm{MSE(dB)} = 10\log_{10}\left(\frac{1}{N}\sum_{k=1}^{N}(d_k - y_k)^2\right), \tag{33}$$

where y_k is the prediction of d_k and N is the length of testing data.

The non-Gaussian noise model considered in this section is the impulsive noise [33], which was modeled by the combination of two mutually independent noise processes. We assumed the mixture noise model in the form of $v_k = (1-a_k)A_k + a_k B_k$, where $a_k \in (0,1)$ is a binary distribution with occurrence probability $P(a_k = 1) = c$ and $P(a_k = 0) = 1-c$ $(0 \leq \mathrm{c} \leq 1)$. Without mentioning otherwise, the parameter c was configured to 0.1 and A_k is a zero-mean Gaussian distribution with $\sigma_A^2 = 0.01$. For B_k, we mainly considered the α-stable noise (heavy-tailed impulsive noise) process with characteristic function [34]:

$$\varphi(t) = \exp\{j(\varepsilon t) - \eta|t|^{\alpha}[1 + j(\beta\,\mathrm{sgn}(t)S(t,\alpha))]\}, \tag{34}$$

where

$$S(t,\alpha) = \begin{cases} \tan\frac{\alpha\pi}{2}, & \alpha \neq 1, \\ \frac{2}{\pi}\log|t|, & \alpha = 1, \end{cases} \tag{35}$$

where $\alpha \in (0,2]$ is the characteristic factor to measure the heaviness of the tail and a smaller parameter α means a larger impulse, η is the dispersion factor that controls the number of impulses, $-\infty < \varepsilon < +\infty$ is the location factor, $\beta \in [-1,1]$ is the symmetry factor, $\mathrm{sgn}(\cdot)$ is the sign function, and $j = \sqrt{-1}$. The parameter vector of the noise model is written as $V_{\alpha-stable} = [\alpha, \beta, \eta, \varepsilon]$. Here, we chose the parameter vector $V_{\alpha-stable} = [0.8, 0, 0.1, 0]$ in the following simulations.

4.1. Mackey–Glass Time Series

Since the Mackey–Glass (MG) chaotic system is a benchmark problem for nonlinear learning problems, we first considered the MG chaotic time series [2] in the following simulations, which is generated by the delayed differential equation as follows:

$$\frac{du(t)}{dt} = -bu(t) + \frac{au(t-\tau)}{1+u(t-\tau)^n}, \tag{36}$$

with $a = 0.2$, $b = 0.1$, $n = 10$, and $\tau = 30$. The time series was discretized at the sampling period of 6 s and corrupted by the noise model mentioned above. We used the previous seven points $\{u_{k-7}, u_{k-6}, ..., u_{k-1}\}^{\mathrm{T}}$ to predict the current value u_k. The prediction was trained by 2000 data points, and tested with another 200.

The parameter γ is key in the proposed RFFCCG algorithm. In the first simulation, we discuss the influence of γ on the filtering accuracy of RFFCCG to combat non-Gaussian noises. The parameter was selected within the range $\gamma = [0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 1, 2, 4, 6]$. The influence of γ is shown in Figure 2, where the steady-state MSEs are derived by averaging the last 200 iterations. For RFFCCG, the Gaussian kernel bandwidth was set as $\sigma = 1$, the forgetting factor β = 0.999, and the dimension of RFF $m = 100$. It can be seen from Figure 2 that the parameter γ had a direct influence on the filtering performance of RFFCCG. The RFFCCG algorithm could achieve the highest filtering accuracy when $\gamma = 0.3$, and thus too large or too small γ will cause performance degradation. An appropriate γ can combat impulsive noises efficiently. Therefore, we set the parameter $\gamma = 0.3$ for RFFCCG in the following simulations.

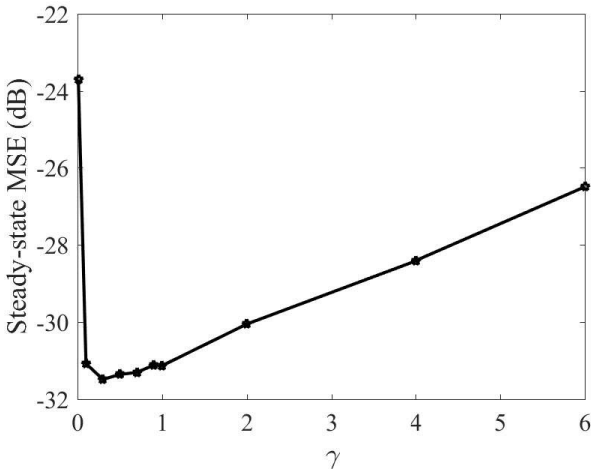

Figure 2. Steady-state MSEs of RFFCCG versus different γ in Mackey–Glass (MG) time series prediction for non-Gaussian noises.

In addition, the steady-state MSEs and the averaged time are plotted in Figure 3 regarding different dimension m. Here, the simulation environment and kernel bandwidth setting of RFFCCG were similar to those of Figure 2. The range of m was set as [1,100]. From Figure 3, we observe: (1) the average consumed time increased linearly with m; (2) the filtering accuracy of RFFCCG could be improved by increasing m to some extent, however it remained almost unchanged when $m \geq 60$. In addition, a larger dimension m resulted in higher filtering accuracy at the expense of increasing computational time. Thus, the dimension $m = 60$ was set for RFFCCG to provide a trade-off between filtering accuracy and computational time.

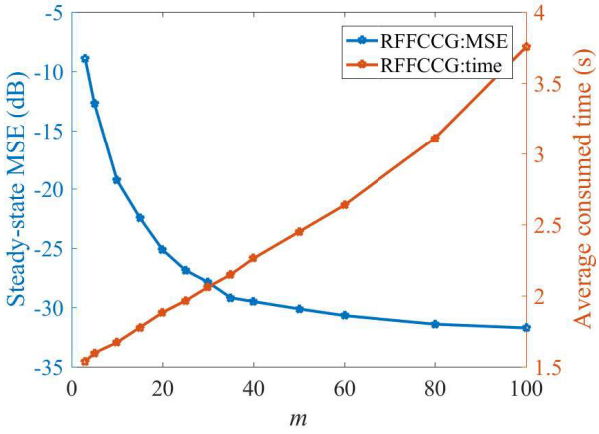

Figure 3. Comparison of steady-state MSEs and average consumed time of RFFCCG versus m in MG time series prediction for non-Gaussian noises.

In this example, we compared the filtering accuracy and robust performance of RFFCCG with other filtering algorithms. The parameters of each algorithm were set to achieve the desired filtering accuracy and to have the same convergence rate. The bandwidth of Gaussian kernels was set to 1 for all algorithms; the step size was $\eta = 0.1$ in RFFKLMS and RFFMC; the threshold $\varepsilon = 0.05$ was chosen for QKRLS; the distance threshold and the error threshold were set as $\delta_1 = 0.1$ and $\delta_2 = 0.1$, respectively, and the regularization parameter $\lambda = 0.1$ for the KRMC-NC; for RFFCG and RFFCCG, the forgetting factor was set to $\beta = 0.999$; $\gamma = 0.3$ was chosen for RFFCCG; the dimension of RFFS was $m = 60$. From Figure 4, we observe that the performance of quadratic-based algorithms (i.e., RFFKLMS, QKRLS, and RFFCG) became worse in the non-Gaussian noise environment, while RFFMC, KRMC-NC, and RFFCCG always generated stable performance and achieved desirable performance when impulse noise appeared. Especially, the filtering performance of RFFCCG was very close to that of the recursive KRMC-NC algorithm and better than that of the SGD-based RFFMC algorithm. Table 3 lists the detailed simulation results in terms of the dictionary size, steady-state MSE, and average consumed time. One also can observe that RFFCCG could produce the comparable filtering accuracy to KRMC-NC with less consumed time and storage requirements. Thus, RFFCCG is more efficient in the compared algorithms for the MG time series prediction.

Table 3. Simulation results of RFFKLMS, QKRLS, RFFKMC, KRMC-NC, RFFCG, and RFFCCG in MG time series prediction for non-Gaussian noises.

Algorithm	Size	MSE (dB)	Consumed Time (s)
RFFKLMS [13]	60	N/A	2.6305
QKRLS [32]	182	N/A	8.2586
RFFMC [14]	60	−22.3870	2.6282
KRMC-NC [31]	500	−29.6680	3.6299
RFFCG [15]	60	N/A	2.6183
RFFCCG	60	−30.5060	2.6750

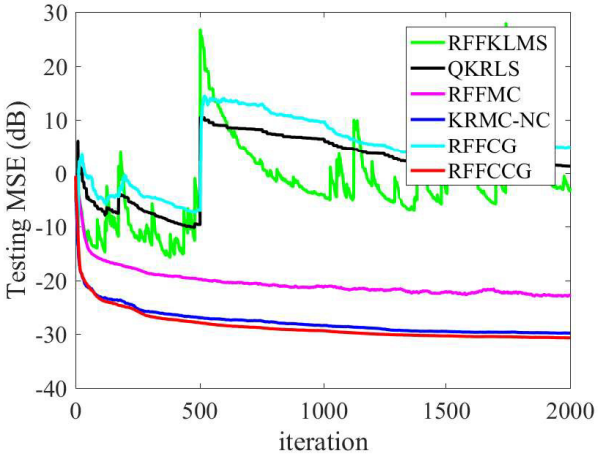

Figure 4. Learning curves of RFFCCG and different algorithms versus $m = 60$ in MG time series prediction for non-Gaussian noises. KRMC-NC: kernel recursive maximum correntropy algorithm with novelty criterion; QKLMS: quantized kernel recursive least squares; RFFCG: random Fourier features conjugate gradient; RFFKLMS: random Fourier features kernel least mean square; RFFMC: random Fourier features maximum correntropy.

4.2. Nonlinear System Identification

To further validate the superiority of RFFCCG, we chose the problem of nonlinear system identification, where the nonlinear system is of the form [35]

$$u_k = (a_1 - a_2 \exp(u_{k-1}^2))u_{k-1} - (a_3 + a_4 \exp(-u_{k-1}^2))u_{k-2} + a_5 \sin(u_{k-1}\pi), \tag{37}$$

where u_k denotes the output at discrete time k, $u_1 = 0.1$, and $u_2 = 0.1$ were configured as the initial values, and $\mathbf{a} = [a_1, a_2, a_3, a_4, a_5]^{\mathrm{T}}$ denotes the coefficient vector. The setting for prediction task is shown as follows: the previous two values (i.e., $\mathbf{u} = [u_{k-1}, u_{k-2}]$) were used as the input vector to predict the current value u_k. We considered stationary and non-stationary scenarios in the following simulations. The data were corrupted by the noise model mentioned above and the Gaussian kernel with kernel parameter $\sigma = 1$ was used for all the tested algorithms.

In the stationary case, the coefficient vector was fixed at $\mathbf{a} = [0.8, 0.5, 0.3, 0.9, 0.1]^{\mathrm{T}}$. The first 2000 data points were used for training and the additional 200 for testing. We compared the testing MSE of RFFCCG with those of RFFKLMS, QKRLS, RFFMC, KRMC-NC, and RFFCG due to their modest complexities and excellent performance under the stationary system. The parameters were chosen to obtain the best results as follows: $\eta = 0.1$ for RFFKLMS; $\varepsilon = 0.002$ for QKLMS; $\eta = 0.4$ for RFFMC; $\delta_1 = 0.01$, $\delta_2 = 0.01$, and $\lambda = 0.1$ for KRMC-NC; $\beta = 0.999$ for RFFCG; $\beta = 0.999$ and $\gamma = 0.3$ for RFFCCG. To balance the accuracy and computational time, $m = 50$ was also configured for the dimension of RFFS. The learning curves of all the algorithms are shown in Figure 5. In this case, the RFFCCG algorithm still had satisfactory prediction ability and achieved comparable performance to KRMN-NC and better performance than RFFMC, while others exhibited poor performance. This also means that RFFCCG has strong robustness against impulsive noises. Table 4 shows the dictionary size, steady-state MSEs, and average consumed time of all algorithms. As can be clearly seen from Figure 5 and Table 4, RFFCCG consumed less time and achieved a faster convergence rate and higher filtering accuracy than the compared algorithms including RFFCG.

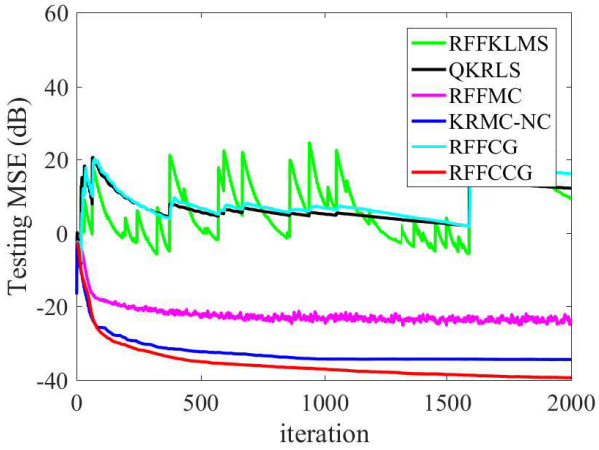

Figure 5. Learning curves of RFFCCG and different algorithms versus $m = 50$ in nonlinear system identification of a stationary environment for non-Gaussian noises.

Table 4. Simulation results of RFFKLMS, RFFMCC, RFFCG, and RFFCCG in nonlinear system identification of a stationary environment for non-Gaussian noises.

Algorithm	Size	MSE (dB)	Consumed Time (s)
RFFKLMS [13]	50	N/A	2.2773
QKRLS [32]	160	N/A	7.1883
RFFMC [14]	50	−23.6263	2.3359
KRMC-NC [31]	500	−34.3570	2.9277
RFFCG [15]	50	N/A	2.3768
RFFCCG	50	−38.9975	2.3390

The tracking performance was evaluated in a non-stationary system where two different coefficient vectors were used for data generation as follows: $\mathbf{a} = [0.8, 0.5, 0.3, 0.3, 0.1]^{\mathrm{T}}$ was selected in the first 2000 data, and $\mathbf{a} = [0.4, 0.7, 0.6, 0.6, 0.2]^{\mathrm{T}}$ was set in the following 2000 data. We compared the testing MSE of RFFCCG with those of RFFKLMS, RFFMC, RFFCG, and RFFCCG due to their modest complexities and excellent performance in the non-stationary system. To compute the convergence curve, a total of 4000 data points were used for training with a sudden change at the 2001-th data point. Regarding the test process, 400 data points were used with a sudden change at the 201-th data point. With the same criterion of parameters setting, the step sizes η of RFFKLMS and RFFMC were chosen as 0.1 and 0.3, respectively, the forgetting factor $\beta = 0.999$ was used in RFFCG and RFFCCG, and the dimension of RFF was set as $m = 50$. The performance comparison is presented in Figure 6. It can be observed that all of the RFF-based algorithms were capable of tracking the change of the system. However, RFFCCG outperformed all the compared algorithms with $\gamma = 0.3$ when abrupt change occurred. Note that the dictionary size, steady-state MSEs, and consumed time of the tested algorithms averaged by the points of $1500 \sim 2000$ and $3500 \sim 4000$ are summarized in Table 5. As observed in Figure 6 and Table 5, RFFCCG provided good tracking performance for a non-stationary system in non-Gaussian noises.

Therefore, in both stationary and non-stationary circumstances for the nonlinear system identification, the proposed RFFCCG algorithm offers excellent filtering performance in terms of filtering accuracy, convergence rate, robustness, and computational and space complexities.

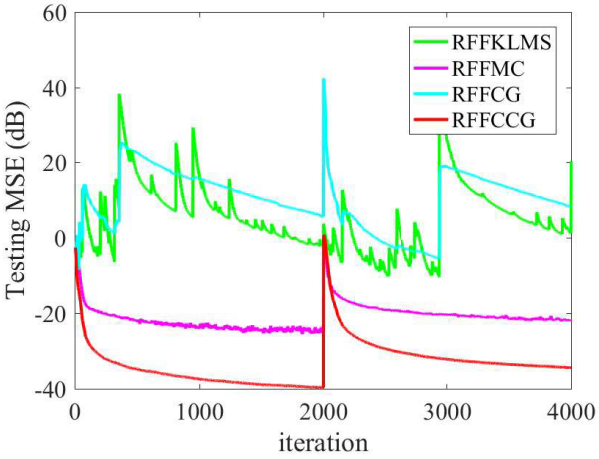

Figure 6. Learning curves of RFFCCG and different algorithms versus $m = 50$ in nonlinear system identification of a non-stationary environment for non-Gaussian noises.

Table 5. Simulation results of RFFKLMS, RFFMCC, RFFCG, and RFFCCG in nonlinear system identification of a non-stationary environment for non-Gaussian noises.

Algorithm	Size	MSE (dB)	Consumed Time (s)
RFFKLMS [13]	50	N/A	0.1048
RFFMC [14]	50	−24.4093/−21.5835	0.1039
RFFCG [15]	50	N/A	0.1794
RFFCCG	50	−39.3061/−34.0029	0.1786

5. Conclusions

In this paper, the robust random Fourier features Cauchy conjugate gradient (RFFCCG) algorithm was proposed and analyzed by integrating random Fourier mapping (RFM) into the Cauchy loss function with the conjugate gradient (CG) optimization method for nonlinear applications in a non-Gaussian noise circumstance. Random Fourier mapping (RFM) for a kernel adaptive filter (KAF) generates an effective finite-dimensional sparsification approach to obtain a more accurate and compact network. The developed RFFCCG algorithm with low computational and space complexities could significantly improve the filtering performance in comparison with other representative robust filters against non-Gaussian noise interferences. We discussed the influence of free parameters, and obtained optimal parameter values for RFFCCG. Simulation results in the presence of non-Gaussian noises validated the superiority of RFFCCG in terms of the filtering accuracy, robustness, tracking performance, and time and storage consumption.

Author Contributions: Conceptualization, X.H. and S.W.; methodology, X.H. and K.X.; software, K.X. and S.W.; validation, S.W. and X.H.; formal analysis, X.H. and K.X.; investigation, X.H. and K.X.; resources, S.W.; data curation, S.W. and X.H.; writing—original draft preparation, X.H.; writing—review and editing, S.W. and K.X.; visualization, X.H.; supervision, S.W.; project administration, S.W. and K.X.; funding acquisition, S.W. and K.X.

Funding: This work was supported by the National Natural Science Foundation of China (61671389) and Fundamental Research Funds for the Central Universities (XDJK2019B011).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kivinen, J.; Smola, A.J.; Williamson, R.C. Online learning with kernels. *IEEE Trans. Signal Process.* **2004**, *52*, 1540–1547. [CrossRef]
2. Liu, W.; Príncipe, J.C.; Haykin, S. *Kernel Adaptive Filtering: A Comprehensive Introduction;* John Wiley & Sons: Hoboken, NJ, USA, 2010.

3. Liu, W.; Pokharel, P.P.; Príncipe, J.C. The kernel least mean square algorithm. *IEEE Trans. Signal Process.* **2008**, *56*, 543–554. [CrossRef]
4. Liu, W.; Príncipe, J.C. Kernel affine projection algorithms. *EURASIP J. Adv. Signal Process.* **2008**, *2008*, 1–12.
5. Engel, Y.; Mannor, S.; Meir, R. The kernel recursive least squares algorithm. *IEEE Trans. Signal Process.* **2004**, *52*, 2275–2285. [CrossRef]
6. Liu, W.; Park, I.; Príncipe, J.C. An information theoretic approach of designing sparse kernel adaptive filters. *IEEE Trans. Neural Netw.* **2009**, *20*, 1950–1961. [CrossRef]
7. Richard, C.; Bermudez, J.C.M.; Honeine, P. Online prediction of time series data with kernels. *IEEE Trans. Signal Process.* **2009**, *57*, 1058–1067. [CrossRef]
8. Wang, S.; Zheng, Y.; Duan, S.; Wang, L.; Tan, T. Quantized kernel maximum correntropy and its mean square convergence analysis. *Dig. Signal Process.* **2017**, *63*, 164–176. [CrossRef]
9. Zhao, S.; Chen, B.; Zhu, P.; Príncipe, J.C. Fixed budget quantized kernel least mean square algorithm. *Signal Process.* **2013**, *93*, 2759–2770. [CrossRef]
10. Vaerenbergh, S.V.; Vía, J.; Santamaría, I. A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. In Proceedings of the 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings (ICASSP), Toulouse, France, 14–19 May 2006; pp. 789–792.
11. Wang, S.; Wang, W.; Dang, L.; Jiang, Y. Kernel least mean square based on the Nyström method. *Circuits Syst. Signal Process.* **2019**, *38*, 3133–3151. [CrossRef]
12. Rahimi, A.; Recht, B. Random features for large-scale kernel machines. In Proceedings of the 21th Annual Conference on Neural Information Processing Systems (ACNIPS), Vancouver, BC, Canada, 3–6 December 2007; pp. 1177–1184.
13. Singh, A.; Ahuja, N.; Moulin, P. Online learning with kernels: Overcoming the growing sum problem. In Proceedings of the 2012 IEEE International Workshop on Machine Learning for Signal Process (MLSP), Santander, Spain, 23–26 September 2012; pp. 1–6.
14. Wang, S.; Dang, L.; Chen, B.; Duan, S.; Wang, L.; Tse, C.K. Random Fourier filters under maximum correntropy criterion. *IEEE Trans. Circuits Syst. I Reg. Pap.* **2018**, *65*, 3390–3403. [CrossRef]
15. Xiong, K.; Wang, S. The online random Fourier features conjugate gradient algorithm. *IEEE Signal Process. Lett.* **2019**, *26*, 740–744. [CrossRef]
16. Wu, Q.; Li, Y.; Xue, W. A kernel recursive maximum versoria-like criterion algorithm for nonlinear channel equalization. *Symmetry* **2019**, *11*, 1067. [CrossRef]
17. Mathews, V.J.; Cho, S.H. Improved convergence analysis of stochastic gradient adaptive filters using the sign algorithm. *IEEE Trans. Acoust. Speech Signal Process.* **1987**, *35*, 450–454. [CrossRef]
18. Walach, E.; Widrow, B. The least mean fourth (LMF) adaptive algorithm and its family. *IEEE Trans. Inf. Theory* **1984**, *42*, 275–283. [CrossRef]
19. Príncipe, J.C. *Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives*; Springer: New York, NY, USA, 2010.
20. Li, Y; Wang, Y.; Sun, L. A proportionate normalized maximum correntropy criterion algorithm with correntropy induced metric constraint for identifying sparse systems. *Symmetry* **2018**, *10*, 683. [CrossRef]
21. Gallagher, C.H.; Fisher, T.J.; Shen, J. A cauchy estimator test for autocorrelation. *J. Stat. Comput. Simul.* **2015**, *85*, 1264–1276. [CrossRef]
22. Luenberger, D.G. *Linear and Nonlinear Programming*, 4th ed.; Prentice Hall: Englewood Cliffs, NJ, USA, 2016.
23. Yang, J.; Ye, F.; Rong, H.J.; Chen, B. Recursive least mean *p*-Power extreme learning machine. *Neural Netw.* **2017**, *91*, 22–33. [CrossRef]
24. Boray, G.K.; Srinath, M.D. Conjugate gradient techniques for adaptive filtering. *IEEE Trans. Circuits Syst. I Fundam. Theory Appl.* **1992**, *39*, 1–10. [CrossRef]
25. Chang, P.S.; Willson, A.N. Analysis of conjugate gradient algorithms for adaptive filtering. *IEEE Trans. Signal Process.* **2008**, *48*, 409–418. [CrossRef]
26. Hestenes, M.R.; Stiefel, E. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.* **1952**, *49*, 409–436. [CrossRef]
27. Dassios, I.; Fountoulakis, K.; Gondzio, J. A preconditioner for a primal-dual newton conjugate gradients method for compressed sensing problems. *SIAM J. Sci. Comput.* **2015**, *37*, 2783–2812. [CrossRef]

28. Heravi, A.R.; Hodtani, G.A. A new correntropy-based conjugate gradient backpropagation algorithm for improving training in neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 6252–6263. [CrossRef] [PubMed]
29. Caliciotti, A.; Fasano, G.; Roma, M. Preconditioned nonlinear conjugate gradient methods based on a modified secant equation. *Appl. Math. Comput.* **2018**, *318*, 196–214. [CrossRef]
30. Zhang, M.; Wang, X.; Chen, X.; Zhang, A. The kernel conjugate gradient algorithms. *Trans. Signal Process.* **2018**, *66*, 4377–4387. [CrossRef]
31. Wu, Z.; Shi, J.; Zhang, X.; Ma, W.; Chen, B. Kernel recursive maximum correntropy. *Signal Process.* **2015**, *117*, 11–16. [CrossRef]
32. Chen, B.; Zhao, S.; Zhu, P.; Príncipe, J.C. Quantized kernel recursive least squares algorithm. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *24*, 1484–1491. [CrossRef]
33. Chen, B.; Xing, L.; Zhao, H.; Zheng, N.; Príncipe, J.C. Generalized correntropy for robust adaptive filtering. *IEEE Trans. Signal Process.* **2016**, *64*, 3376–3387. [CrossRef]
34. Weng, B.; Barner, K.E. Nonlinear system identification in impulsive environments. *IEEE Trans. Signal Process.* **2005**, *53*, 2588–2594. [CrossRef]
35. Chen, S.; Billings, S.A.; Grant, P.M. Recursive hybrid algorithm for non-linear system identification using radial basis function networks. *Int. J. Control* **1992**, *55*, 1051–1070. [CrossRef]