A Study of Efficient Kernel Adaptive Filtering Algorithm Based on Parallel Projection, Shrinkage Operator, and Model Learning

February 2021

TAKIZAWA, Masaaki

A Thesis for the Degree of Ph.D. in Engineering

A Study of Efficient Kernel Adaptive Filtering Algorithm Based on Parallel Projection, Shrinkage Operator, and Model Learning

February 2021

Graduate School of Science and Technology
Keio University

TAKIZAWA, Masaaki

Abstract

Various tasks in signal processing and machine learning can be cast as estimations of nonlinear functions. With the rise of big data, online/adaptive learning algorithms to solve nonlinear estimation problems in an online fashion have attracted particular attention. Typical methods for online nonlinear estimation include neural network, Volterra filter, and Kalman filter, each of which has the following issues: local minima, high computational complexity, and model dependency. Although online/adaptive learning method with reproducing kernels, which is called the kernel adaptive filtering (KAF), overcome those issues, KAF suffers from the following efficiency issues: (i) convergence speed of algorithms, (ii) increase of the dictionary size, and (iii) the design of kernel parameters. This study aims to propose online nonlinear estimation algorithms that overcome those issues (i)-(iii) and verify the efficacy of the algorithm through computer simulations.

First, an adaptive technique to enhance the convergence speed of KAF algorithms is presented. The proposed technique is based on the use of the geometric structure of reproducing kernel Hilbert space (RKHS)s as well as data-reusing implemented with the parallel projection. Thanks to the proposed technique, the proposed KAF algorithm has the advantages: high estimation-accuracy, low complexity, and fast convergence/tracking as well as noise robustness.

Second, an adaptive dictionary-reconstruction scheme based on a shrinkage operator is presented to avoid performance degradations of the algorithm caused by obsolete elements in the dictionary, as well as the increases of memory usages and computational complexities caused by the dictionary-size explosion. The proposed algorithm is shown to enjoy the monotone approximation property for a certain function.

Third, an online algorithm that adapts the parameters of Gaussian functions (scales and centers) iteratively by a coordinate descent-based algorithm is presented. It is pointed out that the initial Gaussian scales largely affect the efficiency and the precision of the estimate. To alleviate the sensitivity to the initial scales, the multiscale screening method is presented which selects an adequate initial scale from multiple values efficiently. The proposed algorithm includes the adaptive learning of the coefficient vector of Gaussians as well as the Gaussian parameters, thereby learning mathematical models adaptively.

Acknowledgment

This thesis completes my studies in the undergraduate course at Niigata University from 2012 to 2013, in the master's course at Keio University from 2013 to 2015, and in the doctoral course at Keio University from 2018 to 2021. I am deeply grateful to my adviser Associate Professor Masahiro Yukawa for his guidance, advice, and comments throughout all these years. I would also express my deep gratitude to Professor Isao Yamada of Tokyo Institute of Technology for his support. Furthermore, it is my pleasure to thank all the current and formal students of the Yukawa laboratory for the friendship. My special thanks to Ms. Ishii for her careful support on my overall activities related to my research. I would like to thank Professor Masaaki Ikehara, Professor Shuichi Adachi, and Professor Yoshimitsu Aoki for serving as members of the examining committee of this thesis. My works related to this thesis were supported by KAKENHI Grant numbers 18J21595 and 18H01446.

Contents

List of Figures

List of Tables

Abbreviations

RKHS	reproducing kernel Hilbert space
KAF	kernel adaptive filtering
MKAF	multikernel adaptive filtering
Φ-PASS	parallel hyperplane projection along affine subspace algorithm
DR-Φ-PASS	dual regularization Φ-PASS algorithm
ONEGAP	online nonlinear estimation with adaptations of Gaussian parameters algorithm
LMS	least mean square algorithm
NLMS	normalized least mean square algorithm
APA	affine projection algorithm
RLS	recursive least square algorithm
APSM	adaptive projection subgradient method
APFBS	adaptive proximal forward-backward splitting
NORMA	naive online risk minimization algorithm
KLMS	kernel least mean square algorithm
KNLMS	kernel normalized least mean square algorithm
KAPA	kernel affine projection algorithm
QKLMS	quantized kernel least mean square algorithm
KAPSM	kernel adaptive projection subgradient method
KRLS	kernel recursive least square algorithm
FOBOS-KLMS	kernel least mean square algorithm with forward-backward-splitting
MKNLMS	multi-kernel least mean square algorithm
KAW	kernel adaptive width algorithm
QKLMS-AKS	QKLMS with adaptive kernel size
QKAPSM	quantized kernel APSM algorithm
RBF	radial basis function network
GP	Gaussian process
MFFN	Multilayer feed-forward network

Notations

$\mathbb{R}$	the set of all real numbers
$\mathbb{R}_{++}$	the set of all strictly positive real numbers
$\mathbb{N}$	the set of all nonnegative integers
$\mathbb{N}^*$	the set of all positive integers
$(\cdot)^\mathsf{T}$	vector or matrix transpose
$\boldsymbol{A}^{-1}$	inverse of matrix $\boldsymbol{A}$
$E(\cdot)$	expected value of a random variable
$\mathrm{tr}(\cdot)$	trace of matrix
∇	gradient
$\mathcal{U}$	input space
$\mathbf{0}$	the null vector of the input space
$\mathcal{H}$	reproducing kernel Hilbert space (RKHS)
θ	the null vector of the RKHS
$\boldsymbol{I}$	identity matrix
L	dimension of the input space
$\boldsymbol{u}$	input vector
d	output if system
$\boldsymbol{\kappa}$	kernelized input vector
$\boldsymbol{h}$	coefficients vector
e	error
$\langle \cdot, \cdot \rangle_{\mathcal{H}}$	inner product in $\mathcal{H}$
$\|\cdot\|_{\mathcal{H}}$	norm in $\mathcal{H}$
$\langle \cdot, \cdot \rangle$	the canonical inner product
$\|\cdot\|$	the Euclidean norm
$\kappa(\cdot, \cdot)$	reproducing kernel
ψ	target function
φ	estimate of ψ
n	time index

r	number of dictionary elements
$\mathcal{J}$	index set of dictionary elements
$\mathcal{M}$	dictionary subspace
s	number of selected elements
$\tilde{\mathcal{J}}$	index set of selected elements
$\tilde{\mathcal{M}}$	subspace spanned by selected elements
$\mathcal{V}$	affine subspace
p	number of projections computed in parallel
$\mathcal{I}$	index set of data used in parallel projection
$\mathcal{S}$	bounded-instantaneous-error hyperslab
ρ	error bound for $\mathcal{S}$
Π	zero instantaneous error hyperplane in $\mathcal{H}$
H	zero instantaneous error hyperplane in $\mathbb{R}^r$
δ	coherence parameter
ϵ	threshold for error criterion
μ	extrapolation coefficient
λ	step size
ω	weight of parallel projections and distance function
w	weight of the ℓ_1 norm
τ	regularization parameter
γ	Lipschitz constant
Θ	cost function of the proposed algorithm
$P_{\mathcal{K}}(\cdot)$	projection onto a nonempty closed convex set $\mathcal{K}$
$\boldsymbol{G}$	Gram matrix

Chapter 1

General Introduction

1.1 Background

Nonlinear functions appear in science and engineering, e.g., system identification, channel equalization, acoustic echo cancellation, time-series prediction, and so on. In particular, online (sequential) estimation techniques for nonlinear functions have attracted particular attention with the rise of big data due to small computational costs and memory usages of online learning algorithms. In Table 1.1, nonlinear estimation techniques are summarized. Typical methods for online nonlinear estimation include Kalman filter, Volterra filter, and multilayer feed-forward network (MFFN). Extended and unscented Kalman filters [1, 2] are dominant choices when the target system has a state-space formulation. The state-space model determines the modeling capability and the complexity of the Kalman filters as well as the convexity of problem. Parameter estimations for the state-space models have been studied in a variety of industrial applications, including the control of motor activity [3, 4], state estimation of power systems [5, 6], state-of-charge estimation for lithium-ion batteries [7], and control and estimation in vehicle systems [8, 9]. Volterra filter [10] is widely applied in acoustics applications [11] and nonlinear system identification [12]. However, the order of the Volterra series expansion is limited typically to two (or three at most) due to the increase of computational loads, which means that the Volterra filter has limited capabilities to capture the nonlinearity of the target. MFFN is widely used for nonlinear estimation [13] due to its universal approximation property [14]. Despite the critical property, the learning of MFFNs optimization problem usually faces the problem of local-minima, i.e., there is no guarantee that solutions reach the global-minima [15, 16].

Computational methods for nonlinear signal processing tasks based on *reproducing kernels* have been developed during the last decades, as witnessed by the success of support vector machine/regression and principal

Table 1.1: Comparisons of nonlinear estimation methods.

	Adaptivity	Modeling capability	Convexity	Complexity
Kalman filter	Yes	Depending on state-space model		
Volterra filter	Yes	Limited	Yes	Large
MFFN	Yes	Universal	No	Large
GP	No	Universal	Yes	Large
KAF	Yes	Universal	Yes	Moderate

component analysis [17, 18]. While traditional kernel methods consider essentially butch settings, i.e., all training data are available in advance, online/adaptive methods have gained attention during the past decade. A typical example is Gaussian process (GP) regression [19, 20] is a nonlinear regression method based on reproducing kernels. GP regression mainly considers for butch settings due to its sizeable computational complexity of the kernel-matrix-inversion operation.

Recently, *Kernel adaptive filtering (KAF)* [21, 22, 23, 24, 25, 26, 27, 28, 29, 30] has gathered significant attention due to the following three reasons [24].

1. **Convexity:** Kernel adaptive filtering can be viewed as linear adaptive filtering in a higher-dimensional feature space, which means that a wealth of knowledge on adaptive filtering, based on convex optimization, is available.

2. **Computational efficiency:** Due to the reproducing property of reproducing kernels (see Section 1.3.1), the inner product operation in the feature space is computed efficiently.

3. **High estimation capability:** Under an appropriate choice of reproducing kernels, a kernel adaptive filter enjoys high estimation capability, as well as computational efficiency and convexity. In the case of Gaussian kernel (see Section 1.3.1), filters have the universal approximation property (see Section 1.5.7).

1.2 Notations and Problem Settings

Let $\mathbb{R}$, $\mathbb{R}_{++}$, and $\mathbb{N}$ be the sets of real numbers, strictly positive real numbers, and nonnegative integers, respectively. The superscript $(\cdot)^{\mathsf{T}}$ denotes the transpose of a vector/matrix. The L-dimensional Euclidean space is denoted by $\mathbb{R}^L$, and the Euclidean inner product and norm are denoted by $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$, respectively. A real Hilbert space $\mathcal{H}$ equipped with an inner product will be denoted by $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ as well as its induced norm $\|\cdot\|_{\mathcal{H}}$.

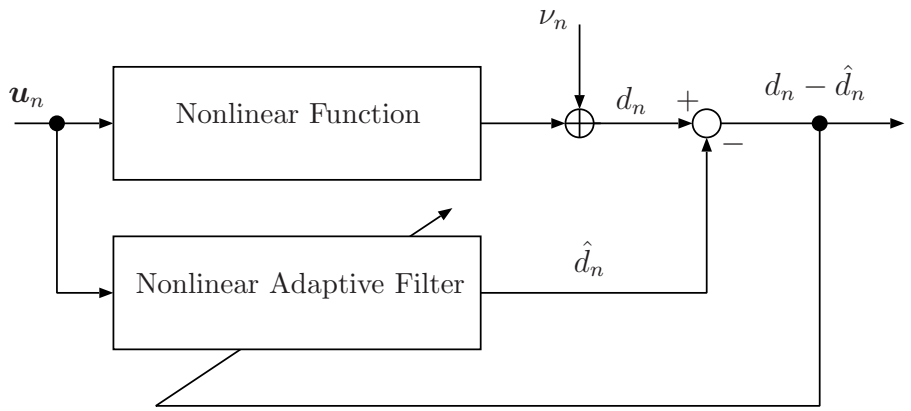

Figure 1.1: Estimation of a nonlinear system with a nonlinear adaptive filter.

The online nonlinear estimation problem considered in the present study is the following (see Figure 1.1): estimate an unknown nonlinear function $\psi_n : \mathcal{U} \to \mathbb{R}$ by means of sequentially arriving input vectors $\boldsymbol{u}_n \in \mathcal{U}$ and its output $d_n := \psi_n(\boldsymbol{u}_n) + \nu_n \in \mathbb{R}$ contaminated by additive noise $\nu_n \in \mathbb{R}$. Here, we denote by $\mathcal{U} \subset \mathbb{R}^L$ the input space in which the input vectors $\boldsymbol{u}_n$ arise, where $n \in \mathbb{N}$ is the time index. No prior knowledge is assumed available about the structure of ψ and the input signals.

1.3 Kernel Adaptive Filtering

1.3.1 Reproducing Kernels

In the field of adaptive filtering [31, 32], online learning schemes for linear function have been studied for a long time. A possible way to extend the scope of linear models to nonlinear processing is to map the input data into a high dimensional feature space using a nonlinear function. The idea is the following; transform the input data into a high dimensional feature space via a function such that the inner product operation in the feature space is computed efficiently. Let us start from the definition of *reproducing kernel Hilbert space (RKHS)*.

Definition 1.1 (Reproducing kernel Hilbert space) *Let $\mathcal{H}$ be a real Hilbert space of functions $f : \mathcal{U} \to \mathbb{R}$. The space $\mathcal{H}$ is a reproducing kernel Hilbert space when there exists a function $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ that satisfies the following properties.*

1. $\kappa(\cdot, \boldsymbol{u}) \in \mathcal{H}, \forall \boldsymbol{u} \in \mathcal{U}$.
2. $f(\boldsymbol{u}) = \langle f, \kappa(\cdot, \boldsymbol{u}) \rangle_{\mathcal{H}}, \forall f \in \mathcal{H}, \forall \boldsymbol{u} \in \mathcal{U}$.

The property 2 is called the *reproducing property* and a function $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ is called a *reproducing kernel* of $\mathcal{H}$. In particular, for any $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{U}$,

$$\kappa(\boldsymbol{u}, \boldsymbol{v}) = \kappa(\boldsymbol{v}, \boldsymbol{u}) = \langle \kappa(\cdot, \boldsymbol{u}), \kappa(\cdot, \boldsymbol{v}) \rangle_{\mathcal{H}}. \tag{1.1}$$

Some examples of the reproducing kernel are given below:

Linear kernel:

$$\kappa(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^{\mathsf{T}}\boldsymbol{v}, \ \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^L \tag{1.2}$$

Polynomial kernel:

$$\kappa(\boldsymbol{u}, \boldsymbol{v}) = \left(\boldsymbol{u}^{\mathsf{T}}\boldsymbol{v} + c\right)^p, \ p \in \mathbb{N}, \ c \geq 0, \ \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^L \tag{1.3}$$

Laplacian kernel:

$$\kappa(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\beta \sum_{l=1}^{L} |u_l - v_l|\right), \ \beta > 0, \ \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^L \tag{1.4}$$

where $u_l = [\boldsymbol{u}]_l$ and $v_l = [\boldsymbol{v}]_l$

Gaussian kernel:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\frac{-\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\xi}\right), \ \xi > 0, \ \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^L \tag{1.5}$$

1.3.2 Kernel Adaptive Filter

Let $\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j \in \mathcal{J}_n}$ be the *dictionary* with the *dictionary index aet* $\mathcal{J}_n := \{j_n^{(1)}, j_n^{(2)}, \cdots, j_n^{(r_n)}\} \subset \{0, 1, 2, \cdots, n\}$. Here, the cardinality $|\mathcal{J}_n| = r_n$ denotes the dictionary size at time n. A nonlinear model can be defined as

$$\varphi_n := \sum_{j \in \mathcal{J}_n} h_n^{(j)} \kappa(\cdot, \boldsymbol{u}_j), \ h_n^{(j)} \in \mathbb{R}, \ n \in \mathbb{N}. \tag{1.6}$$

Assume here that the dictionary is linearly independent. Figure 1.2 illustrates an example of estimation with four Gaussian kernels. Using the *reproducing property*, the output of the filter φ_n to the input $\boldsymbol{u}_n$ is given by

$$\varphi_n(\boldsymbol{u}_n) = \langle \varphi_n, \kappa(\cdot, \boldsymbol{u}_n) \rangle_{\mathcal{H}} = \sum_{j=1}^{r_n} h_n^{(j)} \kappa(\boldsymbol{u}_n, \boldsymbol{u}_j). \tag{1.7}$$

1.3.3 Two Classes of Kernel Adaptive Filtering Algorithms

Under the strategy of minimizing the instantaneous squared error $(d_n - \varphi_n(\boldsymbol{u}_n))^2$, the filter updating rule is desired to satisfy the following

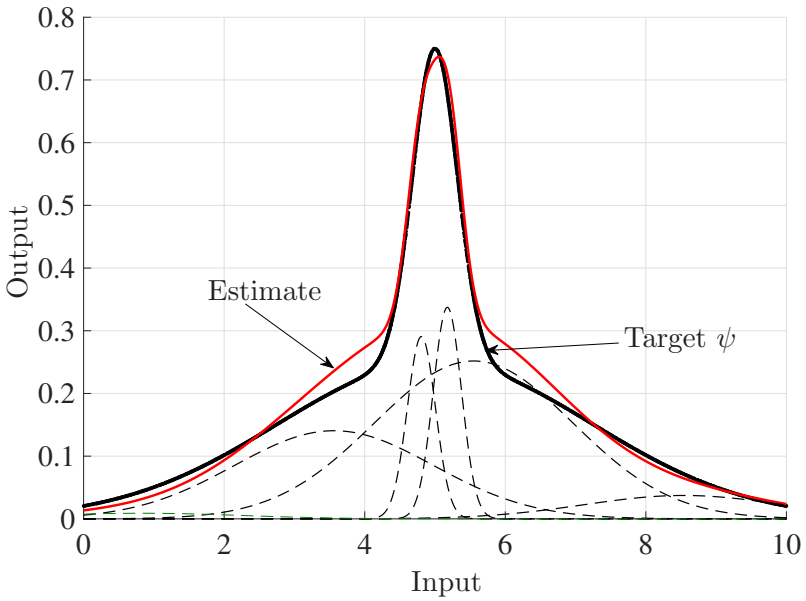

Figure 1.2: Estimation of the nonlinear function ψ with the Gaussian kernels.

conditions (the principle of minimal disturbance).

C1. Vanishing the instantaneous error ($\varphi_n(\boldsymbol{u}_n) = d_n$).

C2. Being close to the current filter φ_n.

Based on the above conditions C1 and C2, Kivinen *et al.* have derived a stochastic gradient descent method in an RKHS $\mathcal{H}$, named the Naive online regularized risk minimization algorithm (NORMA) algorithm [21]. When the new data is observed, NORMA updates an estimate in the direction of the function $\kappa(\cdot, \boldsymbol{u}_n)$ centered on the new data $\boldsymbol{u}_n$, which means that the current estimate is updated in the direction of the normal vector of the following hyperplane:

$$\Pi_n := \{\varphi \in \mathcal{H} : \varphi(\boldsymbol{u}_n) = \langle \varphi, \kappa(\cdot, \boldsymbol{u}_n)\rangle_{\mathcal{H}} = d_n\}. \tag{1.8}$$

Here, the equality $\varphi(\boldsymbol{u}_n) = \langle \varphi, \kappa(\cdot, \boldsymbol{u}_n)\rangle$ is due to the reproducing property of the RKHS $\mathcal{H}$. The set Π_n is a collection of functions (estimates) of which the output $\varphi(\boldsymbol{u}_n)$ to the input $\boldsymbol{u}_n$ equals the output d_n of the system. Each update is quite similar to the least mean square (LMS) algorithm, which is well known in the field of linear adaptive filtering. It should be remarked that φ_n and φ_{n+1} lie in the *dictionary subspace*

$$\mathcal{M}_n := \operatorname{span}\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j \in \mathcal{J}_n} \subset \mathcal{H}, \ n \in \mathbb{N}. \tag{1.9}$$

We refer to this approach based on the RKHS-inner-product expression as *functional-space approach*. In the functional-space approach, the filter φ_n is updated to reduce the distance between Π_n.

The parameter-space approach is obtained by reconsidering the conditions C1 and C2. For $f := \sum_{j=1}^{r} h^{(j)} \kappa(\cdot, \boldsymbol{x}^{(j)}) \in \mathcal{H}$, we obtain

$$\langle f, \kappa(\cdot, \boldsymbol{u}) \rangle_{\mathcal{H}} = f(\boldsymbol{u}) = \sum_{j=1}^{r} h^{(j)} \kappa(\boldsymbol{u}, \boldsymbol{x}^{(j)}) = \langle \boldsymbol{\kappa}, \boldsymbol{h} \rangle \,, \tag{1.10}$$

where

$$\boldsymbol{\kappa} := [\kappa(\boldsymbol{u}, \boldsymbol{x}^{(1)}), \kappa(\boldsymbol{u}, \boldsymbol{x}^{(2)}), \cdots, \kappa(\boldsymbol{u}, \boldsymbol{x}^{(r)})]^{\mathsf{T}} \in \mathbb{R}^r \tag{1.11}$$

and

$$\boldsymbol{h} := [h^{(1)}, h^{(2)}, \cdots, h^{(r)}]^{\mathsf{T}} \in \mathbb{R}^r. \tag{1.12}$$

In the parameter-space, the 'closeness' in the condition C2 is measured by the Euclidean norm of coefficients vectors rather than the Hilbertian norm of functions. The idea of the *parameter-space approach* is the followings: update the coefficient vector $\boldsymbol{h}$ in the direction of the normal vector of the following hyperplane:

$$H_n := \{\boldsymbol{h} \in \mathbb{R}^{r_n} : \langle \boldsymbol{\kappa}_n, \boldsymbol{h} \rangle = d_n\}, \tag{1.13}$$

where

$$\boldsymbol{\kappa}_n := [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{j_n^{(1)}})), \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{j_n^{(2)}}), \cdots, \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{j_n^{(r_n)}})]^{\mathsf{T}} \tag{1.14}$$

is the kernelized input vector.

1.3.4 Review of Kernel Adaptive Filtering Algorithms

There have been many studies on kernel adaptive filtering and many kernel adaptive filtering algorithms have been proposed [27, 28, 33, 29].

Algorithms: In Figure 1.3, existing KAF algorithms are summarized. As described in the previous subsection, the existing kernel adaptive filtering algorithms can be classified into two general categories: (i) the *parameter-space approach* and (ii) the *functional-space approach*. The parameter-space approach updates a filter by finding a *coefficients vector* which gives a small (or zero) instantaneous error, and the update can be explained as the update in parameter-space (the number of coefficients-dimensional Euclidian space). The kernel recursive least square (KRLS) algorithm [22], the KRLS Tracker algorithm [30], kernel normalized least mean square (KNLMS) algorithm, the kernel affine projection (KAP) algorithm [27], and the multikernel normalized least mean square (MKNLMS) algorithm [28] fall into this approach. In contrast, the functional-space approach updates a filter by finding a *function* which gives a small (or zero) instantaneous error. The kernelized least mean square (KLMS) algorithm [33], kernel adaptive projected subgradient

method (KAPSM) [26], and the quantized KLMS (QKLMS) algorithm [29] fall into this approach.

Novelty Criteria: Under the update based on functional-space approach, the dictionary size r_n grows linearly as time goes by, i.e., $\kappa(\cdot, \boldsymbol{u}_n) \in \mathcal{M}_n$ for any $n \in \mathbb{N}$. To keep it bounded, a simple truncation rule is used in [21], discarding the oldest datum and accepting the new one at each time. This seems to be natural in terms of adaptivity, but it possibly results in discarding useful data for estimation. For another approach to bound dictionaries, novelty criteria have therefore been proposed to construct a well-organized dictionary by accepting novel data only; e.g., the approximate linear dependency of Engel *et al.* [22], the coherence of Richard *et al.* [27], and the surprise (a subjective information-theoretic criterion) of Liu *et al.* [34].

Dictionary Refinement Techniques: An adaptive dictionary-refinement technique based on the proximity operator of a weighted (block) ℓ_1 norm for kernel adaptive filtering has initially been proposed by Yukawa in 2010 [35, 36, 28] for the parameter-space approach in the multikernel adaptive filtering (MKAF) context. Specifically, to enhance the model efficiency, a weighted (block) ℓ_1 norm regularization is applied to the cost function, which is applied in various fields [37, 38, 39, 40]. A similar algorithm (for the monokernel case) has been presented and analyzed by Gao *et al.* in 2013 [41], under the name of KNLMS with forward-backward-splitting (FOBOS-KNLMS) algorithm. A sparse algorithm classified in functional-space approach was proposed by Chen *et al.* under the name of Sparse-QKLMS [42]. Another line of researches related to the dictionary refinement techniques are the so-called budget learning which keeps the dictionary size at some pre-specified budget [43, 30].

1.4 Motivation of This Study

Although KAF has attracted significant attention due to its computational efficiency caused by the reproducing property of the positive definite kernel as well as being free from the issue of local minima, KAF suffers from the following efficiency issues: (i) convergence speed of algorithms, (ii) increase of the dictionary size, and (iii) the design of kernel parameters. Each issue is specifically described below.

1.4.1 Issue (i): Convergence Speed of Algorithm

Applying one of the novelty criteria to KAF algorithms based on the functional-space approach (e.g., NORMA and KLMS), the filter cannot move when the new data is not added into the dictionary, implying that the guarantee of each update no longer exists. This means that once one of those criteria is introduced, the observed datum that is regarded as insufficiently novel is simply discarded and makes no contributions to estimation even though

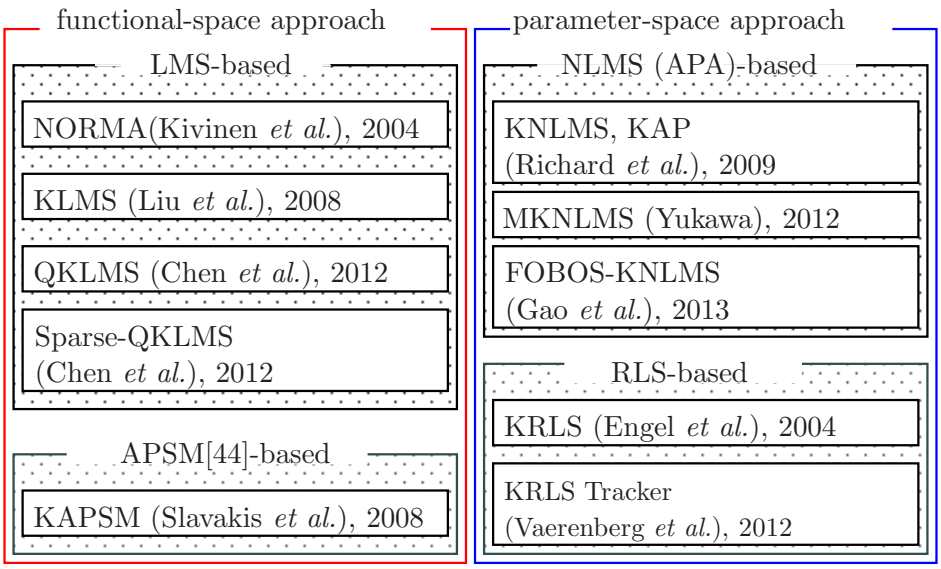

Figure 1.3: A summary of the existing KAF algorithms.

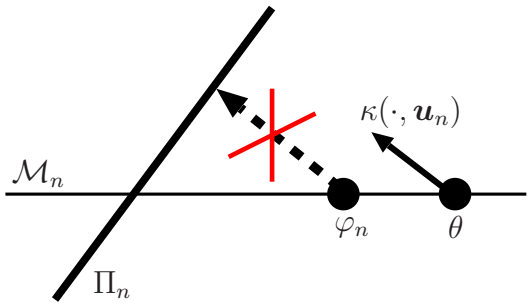

Figure 1.4: A situation when $\kappa(\cdot, \boldsymbol{u}_n) \notin \mathcal{M}_n$.

it can be informative enough to adjust the coefficients. Specifically, once the admission control is operated under one of those criteria, it is no longer guaranteed that the normal vector $\kappa(\cdot, \boldsymbol{u}_n)$ of Π_n lies in the dictionary subspace $\mathcal{M}_n$. Figure 1.4 illustrates such a situation that $\kappa(\cdot, \boldsymbol{u}_n) \notin \mathcal{M}_n$. It is seen that the estimate φ_n has to leave $\mathcal{M}_n$ to be projected onto Π_n. This implies that one cannot update the estimate straightforwardly in this case since the next estimate φ_{n+1} should lie in $\mathcal{M}_n$. This is waste-of-resources which leads to slow convergences of algorithms.

Remark 1.1 *The methods proposed in [22, 27] update the estimate even when $\kappa(\cdot, \boldsymbol{u}_n) \in \mathcal{M}_n$. Our current focus is, however, on the methods based on projections in the RKHS. The method in [22] performs a recursive least square update in terms of the coefficients. The method in [27] operates the*

orthogonal projection in a Euclidean space, not in the RKHS.

1.4.2 Issue (ii): Increase of Dictionary Size

In [41], it has been reported that some obsolete elements still exist, i.e., there are elements of which the coefficients are small even though we perform admission controls with the novelty criteria. Moreover, obsolete elements negatively affect the estimation, and therefore those elements should be pruned from the dictionary. Considering time-dependent function ψ_n, dictionary elements, which are 'not' obsolete at some point in time, may become obsolete when ψ_n changes and has zero-value at the radius of those elements. Those obsolete elements increase the dictionary size senselessly.

As described in Section 1.3.4, some dictionary refinement techniques are proposed. Those techniques, however, are mainly presented for the parameter-space approach. As a functional-space algorithm with ℓ_1 norm regularization, the Sparse QKLMS algorithm has been presented in [42], which exploits a subgradient, defined in the RKHS, of an instantaneous squared error penalized by the ℓ_1 norm. However it involves the computation of the inverse of a Gram matrix, which increases the overall complexity of the algorithm.

1.4.3 Issue (iii): Design of Kernel Parameters

One of the largest difficulties of kernel adaptive filtering lies in finding an appropriate kernel to construct an "efficient model" in the sense of reducing the estimation errors with low (or affordable at most) complexity. If improper kernels are used, the kernel adaptive filter may need a large-size dictionary, which causes slow convergence. Under the use of Gaussian kernel, the efficacy of models relies on the scales and centers of Gaussian kernels. Regarding the kernel scales, a reasonable scale parameter has been assumed available prior to adaptation in the early studies of kernel adaptive filtering. This assumption is however unrealistic, particularly when the data under consideration is nonstationary.

The MKAF algorithms [45, 28, 46, 47, 48, 49, 50] has been proposed as a convex analytic approach with multiple different scales, and the concept of online model selection and learning has been presented in [51, 52] based on the MKAF framework, selecting appropriate scales from a hundred possible scales by shrinking the coefficient vector for each scale while learning those parameters as well as reducing the estimation errors simultaneously. Although those selection schemes for the Gaussian parameters yield reasonably good results, there is still sufficient room for improvements in the sense of "efficiency" of the estimate. In the kernel adaptive filtering context, moreover, some methods have been proposed to adapt the kernel scales [53, 54, 55] and centers [56, 57] in the dictionary. The method proposed in

[54] uses a common scale parameter for all kernel functions. The method in [55, 58] updates both scales and centers individually, as in the proposed approach.

We focus on the fact that the performance of the aforementioned alternating update approach depends highly on the initial scales [59]. Specifically, the initial Gaussian scales affect the efficiency and the accuracy of the estimate significantly when the selected scale was far from the actual ones of the target function due to the "gradient vanishment" (see Section 4.3.1).

1.4.4 Contribution of This Thesis

The contributions of my doctoral study are three folds.

First, we present an efficient KAF algorithm for adaptive nonlinear estimation based on projections aiming to enhance the learning speed of the algorithm by using the information of the observed data effectively. The proposed algorithm, named *parallel hyperslab projection along affine subspaces* (Φ-PASS), is based on three ideas: projection-along-subspace, selective update, and parallel projection. The projection-along-subspace yields excellent performances with small dictionary sizes. The selective update effectively reduces the complexity without any serious degradation of performance. The parallel projection leads to fast convergence/tracking accompanied by noise robustness.

Second, we present an adaptive dictionary-reconstruction scheme based on a shrinkage operator, named *the parallel hyperplane projection along affine subspaces algorithm with dictionary refinements* (DR-Φ-PASS). We apply the popular ℓ_1 norm regularization [35, 28, 41] for the sake of adaptive refinements of model complexities. A straightforward approach is to apply the adaptive proximal forward-backward splitting (APFBS) algorithm [60] to the cost function (which is the sum of smooth and nonsmooth functions) in the functional subspace. However, the proximity operator defined in the functional subspace has no closed-form expression. We, therefore, propose a heuristic, but efficient, algorithm that employs the proximity operator defined in the parameter space. This heuristic idea drastically reduces the computational complexity of the algorithm. Although the proposed algorithm uses different inner products between the forward and backward steps, we show that it still enjoys a monotone approximation property regarding a cost function with a certain modified weighted ℓ_1 norm under some conditions.

Third, focusing on the model defined by a weighted sum of Gaussian functions, we propose an efficient adaptive method updating the Gaussian parameters (scales and centers) and the coefficients alternately to reduce the instantaneous squared errors named *online nonlinear estimation with adaptations of Gaussian parameters* (ONEGAP). The Gaussian function, as a basis, has the following advantages in the problem setting of this thesis:

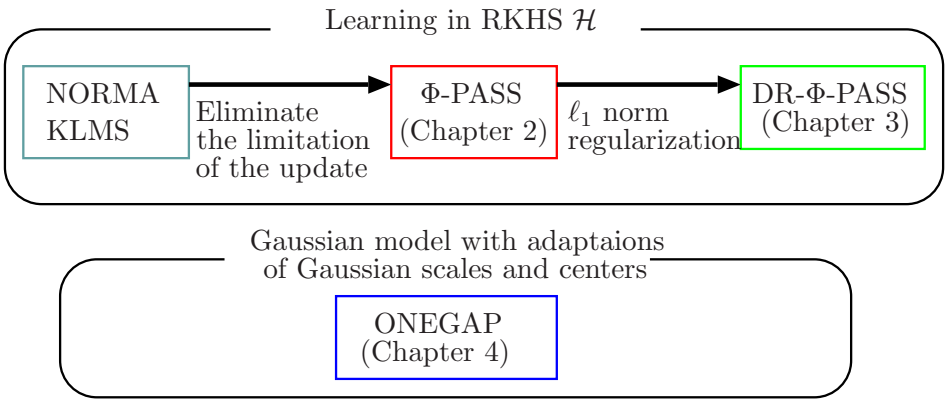

Figure 1.5: Relations of the proposed algorithms and related algorithms.

1. Universality:
   Since no prior knowledge is assumed about the structure of the target system, the model should have a strong capability to approximate functions. The Gaussian function has the universality [61] and therefore the Gaussian model can represent a wide variety of functions. For more details about the universality see Section 1.5.7.

2. Unimodality:
   Under the online setting considered in this paper, a possible strategy for updating the estimate is minimizing the instantaneous squared error with the observed input and output $(\boldsymbol{u}_n, d_n)$. If a certain periodic function is employed as a basis, for instance, each function affects a wide range of the estimate. This is against the above updating strategy since updating the coefficient of a basis changes the estimate over a wide range. The Gaussian function, on the other hand, is unimodal, and therefore the coefficient update affects only a local region.

The key difference between the proposed algorithm and the methods in [53, 54, 55, 56, 57] is a novel online dictionary growing technique, which builds a dictionary with multiple initial scales selected by a hierarchical selection strategy. The use of multiple initial values, however, may cause undesirable growths of the dictionary size (which involve high computational complexities and large memory size). To avoid this, we present an efficient dictionary growing strategy named *multiscale screening method*, which 'screens' the large- and small- scale Gaussians based on the error and novelty criteria.

Figure 1.5 shows the relations between the proposed algorithms and related KAF algorithms. We remark that ONEGAP is no longer KAF algorithm, since the RKHS, in which the filter is updated, changes through estimation due to the adaptation of the Gaussian scales.

This study: In Chapter 1, the background, the objective of this study, and the previous studies were presented. In the rest of Chapter 1, some ingredients used throughout this thesis were presented.

In Chapter 2, an adaptive technique to enhance the convergence speed of KAF algorithms, named Φ-PASS, was presented. The proposed technique was based on the use of the geometric structure of reproducing kernel Hilbert spaces as well as data-reusing implemented with the parallel projection.

In Chapter 3, first, the relation between the functional-space and Euclidean-space approaches were investigated to kernel adaptive filtering based on the isomorphism between two real Hilbert spaces: the so-called dictionary subspace of a functional space and a Euclidean space equipped with a dictionary-dependent inner product. Second, an efficient functional-space projection algorithm, named DR-Φ-PASS, that refines the dictionary adaptively was proposed. The proposed algorithm was shown to enjoy the monotone approximation property for a certain function.

In Chapter 4, ONEGAP which adapts the parameters of Gaussian functions (scales and centers) iteratively by a coordinate descent-based algorithm with the proposed multiscale screening method was presented. ONEGAP consists of two steps: (i) the dictionary growing step and (ii) the parameter updating step. In the first step, the dictionary grows under a hierarchical selection strategy. In the second step, the Gaussian parameters are updated in sequence.

In Chapter 5, the results of this study were summarized.

1.5 Preliminaries

1.5.1 Convexity, Proper, and Lower Semi-continuity

Let $\mathcal{H}$ be a real Hilbert space. A set $\mathcal{C} \in \mathcal{H}$ is call a convex set if $(1-\zeta)\boldsymbol{x} + \zeta\boldsymbol{y} \in \mathcal{C}$, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$, $\forall \zeta \in [0,1]$.

If a set $\mathcal{C}$ is closed as well as convex, it is called a closed convex set.

Definition 1.2 (convex function, proper function) *A function $f : \mathcal{H} \to (-\infty, \infty] := \mathbb{R} \cup \{\infty\}$ is called convex function if*

$$f\left((1-\zeta)\boldsymbol{x} + \zeta\boldsymbol{y}\right) \leq (1-\zeta) f\left(\boldsymbol{x}\right) + \eta f\left(\boldsymbol{y}\right), \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{H}, \forall \zeta \in (0,1)\,. \quad (1.15)$$

In particular, a convex function $f : \mathcal{H} \to (-\infty, \infty]$ is called proper if

$$\mathrm{dom}\left(f\right) := \{\boldsymbol{x} \in \mathcal{H} | f\left(\boldsymbol{x}\right) < \infty\} \neq \emptyset. \quad (1.16)$$

Definition 1.3 (lower semicontinuous function) *A function $f : \mathcal{H} \to (-\infty, \infty]$ is called lower semicontinuous if the set*

$$\mathrm{lev}_{\leq a}\left(f\right) := \{\boldsymbol{x} \in \mathcal{H} | f\left(\boldsymbol{x}\right) \leq a\} \quad (1.17)$$

is closed for every $a \in \mathbb{R}$.

The set of all proper lower semicontinuous convex functions is denoted by $\Gamma_0(\mathcal{H})$.

Definition 1.4 (strictly convex) *A function* $f \in \Gamma_0(\mathcal{H})$ *is called strictly convex if*

$$(\boldsymbol{x} \neq \boldsymbol{y}, \zeta \in (0,1)) \Rightarrow f((1-\eta)\boldsymbol{x} + \zeta\boldsymbol{y}) < (1-\zeta) f(\boldsymbol{x}) + \zeta f(\boldsymbol{y}). \quad (1.18)$$

Definition 1.5 (coercivity) *A function* $f \in \Gamma_0(\mathcal{H})$ *is called coercive if*

$$\|\boldsymbol{x}\|_{\mathcal{H}} \to \infty \Rightarrow f(\boldsymbol{x}) \to \infty. \quad (1.19)$$

1.5.2 Gradients of Convex Functions

Proposition 1.1 (Properties of gradients of convex functions [62]) *Suppose* $f : \mathbb{R}^L \to \mathbb{R}$ *is convex and in* $\mathcal{C}^1$. *Then following statements are equivalent.*

1. *Lipschitz continuity of* $\nabla f(x)$*: there exists and* $L > 0$ *such that*

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L \|\boldsymbol{x} - \boldsymbol{y}\|, \ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom} f. \quad (1.20)$$

2. $g(\boldsymbol{x}) := \frac{L}{2}\|\boldsymbol{x}\|^2 - f(\boldsymbol{x})$ *is convex.*

3. *Quadratic upper bound:*

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \nabla f(\boldsymbol{y}) \rangle + \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2. \quad (1.21)$$

4. *Co-coercivity*

$$\langle \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle \geq \frac{1}{L}\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|^2. \quad (1.22)$$

1.5.3 Subgradient

Definition 1.6 (Subgradient) *Let* $f : \mathcal{H} \to \mathbb{R}$ *be continuous and convex. Then for any* $\boldsymbol{x} \in \mathcal{H}$, *there always exists* $\tilde{x} \in \mathcal{H}$ *satisfying*

$$\|\boldsymbol{y} - \boldsymbol{x}\|_{\mathcal{H}} \tilde{\boldsymbol{x}} + f(\boldsymbol{x}) \leq f(\boldsymbol{y}), \ \forall \boldsymbol{y} \in \mathcal{H}. \quad (1.23)$$

The vector $\tilde{\boldsymbol{x}}$ *is called a subgradient of* f *at* $\boldsymbol{x}$. *The set of all such vectors is called the subdifferential of* f *at* $\boldsymbol{x}$, *and it is denoted as* $\partial f(\boldsymbol{x})$.

1.5.4 Nonexpansive and Quasi-nonexpansive Mappings

A point $\boldsymbol{x} \in \mathcal{H}$ is called a fix point of a mapping $T : \mathcal{H} \to \mathcal{H}$ if $T(\boldsymbol{x}) = \boldsymbol{x}$. The set of all fix points of T is denoted by $\mathrm{Fix}(T) := \{\boldsymbol{x} \in \mathcal{H} \mid T(\boldsymbol{x}) = \boldsymbol{x}\}$. The mapping $T : \mathcal{H} \to \mathcal{H}$ is called nonexpansive if

$$\|T(\boldsymbol{x}) - T(\boldsymbol{y})\|_{\mathcal{H}} \leq \|\boldsymbol{x} - \boldsymbol{y}\|_{\mathcal{H}} \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{H}. \tag{1.24}$$

A mapping $T : \mathcal{H} \to \mathcal{H}$ is called β-Lipschitz continuous with a Lipschitz constant β over $\mathcal{H}$ if there exists $\beta > 0$ satisfying

$$\|T(\boldsymbol{x}) - T(\boldsymbol{y})\|_{\mathcal{H}} \leq \beta \|\boldsymbol{x} - \boldsymbol{y}\|_{\mathcal{H}}, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{H}. \tag{1.25}$$

In particular, a mapping $T : \mathcal{H} \to \mathcal{H}$ is called strictly contractive if there exists $\kappa \in (0, 1)$ such that $\|T(\boldsymbol{x}) - T(\boldsymbol{y})\|_{\mathcal{H}} \leq \|\boldsymbol{x} - \boldsymbol{y}\|_{\mathcal{H}}, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{H}$.

Suppose that the mapping $T : \mathcal{H} \to \mathcal{H}$ has at least one fix point.

Definition 1.7 (quasi-nonexpansive mapping) *The mapping $T : \mathcal{H} \to \mathcal{H}$ is called quasi-nonexpansive if T satisfies*

$$\|T(\boldsymbol{x}) - \boldsymbol{z}\|_{\mathcal{H}} \leq \|\boldsymbol{x} - \boldsymbol{z}\|_{\mathcal{H}}, \forall \boldsymbol{x} \in \mathcal{H}, \forall \boldsymbol{z} \in \mathrm{Fix}(T). \tag{1.26}$$

1.5.5 Proximity Operator

Definition 1.8 (proximity operator) *The proximity operator of index $\tau \in (0, \infty)$ of $f \in \Gamma_0(\mathcal{H})$ is defined by*

$$\mathrm{prox}_{\tau f} : \mathcal{H} \to \mathcal{H} : \boldsymbol{x} \mapsto \underset{\boldsymbol{y} \in \mathcal{H}}{\mathrm{argmin}} \left\{ f(\boldsymbol{y}) + \frac{1}{2\tau} \|\boldsymbol{x} - \boldsymbol{y}\|_{\mathcal{H}}^2 \right\}. \tag{1.27}$$

The existence and the uniqueness of the minimizer of (1.27) are guaranteed respectively by the coercivity and the strict convexity of $f(\cdot) + \frac{1}{2\tau} \|\boldsymbol{x} - \cdot\|_{\mathcal{H}}^2$. Equivalently, for every $\boldsymbol{x} \in \mathcal{H}$, $\mathrm{prox}_{\tau f}(\boldsymbol{x})$ is characterized as a unique point satisfying

$$\left\{\mathrm{prox}_{\tau f}(\boldsymbol{x})\right\} = \{\boldsymbol{z} \in \mathcal{H} | \boldsymbol{z} + \tau \partial f(\boldsymbol{z}) \ni \boldsymbol{x}\}, \tag{1.28}$$

i.e.,

$$\mathrm{prox}_{\tau f}(\boldsymbol{x}) = (I + \tau \partial f)^{-1}(\boldsymbol{x}), \tag{1.29}$$

which is again equivalent to

$$\left\langle \boldsymbol{y} - \mathrm{prox}_{\tau f}(\boldsymbol{x}), \frac{\boldsymbol{x} - \mathrm{prox}_{\tau f}(\boldsymbol{x})}{\tau} \right\rangle_{\mathcal{H}} + f\left(\mathrm{prox}_{\tau f}(\boldsymbol{x})\right) \leq f(\boldsymbol{y}), \forall \boldsymbol{y} \in \mathcal{H}. \tag{1.30}$$

The proximity operator is firmly nonexpansive, i.e., $\mathrm{rprox}_{\tau f} := 2\mathrm{prox}_{\tau f} - I : \mathcal{H} \to \mathcal{H}$, is nonexpansive:

$$\left\| \left(2\mathrm{prox}_{\tau f} - I\right) \boldsymbol{x} - \left(2\mathrm{prox}_{\tau f} - I\right) \boldsymbol{y} \right\|_{\mathcal{H}} \geq \|\boldsymbol{x} - \boldsymbol{y}\|_{\mathcal{H}}. \tag{1.31}$$

If $\operatorname{argmin}_{\boldsymbol{x}\in\mathcal{H}} f(\boldsymbol{x}) \neq \emptyset$, the set of all minimizers of f is equal to that of the Moreau envelope and also expressed as the fixed point set of $\operatorname{prox}_{\tau f} : \mathcal{H} \to \mathcal{H}$; i.e.,

$$\operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{H}} f(\boldsymbol{x}) = \operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{H}}{}^{\tau} f(\boldsymbol{x}) = \operatorname{Fix}\left(\operatorname{prox}_{\tau f}\right). \tag{1.32}$$

1.5.6 Projections onto Subspace and Linear Variety

Definition 1.9 (metric projection) *Given a nonempty closed convex set $\mathcal{C} \subset \mathcal{H}$ and any point $\boldsymbol{x} \in \mathcal{H}$, there exists a unique point $P_{\mathcal{C}}(\boldsymbol{x}) \in \mathcal{C}$ satisfying*

$$d_{\mathcal{C}} := \min_{\boldsymbol{z}\in\mathcal{C}} \|\boldsymbol{x} - \boldsymbol{z}\|_{\mathcal{H}} = \|x - P_{\mathcal{C}}(\boldsymbol{x})\|_{\mathcal{H}}. \tag{1.33}$$

The mapping $\mathcal{H} \ni \boldsymbol{x} \mapsto P_{\mathcal{C}}(\boldsymbol{x}) \in \mathcal{C}$ is called the metric projection onto $\mathcal{C}$.

Given nonzero and linearly independent vectors $\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n$ and and arbitrary vector $\boldsymbol{x}$, the metric projection $P_{\mathcal{M}}(\boldsymbol{x})$ of $\boldsymbol{x}$ onto the subspace $\mathcal{M} := \operatorname{span}\{\boldsymbol{y}_j\},\ j = 1, \cdots, n$ is given by

$$P_{\mathcal{M}}(\boldsymbol{x}) = \sum_{j=1}^{n} \alpha_j \boldsymbol{y}_j. \tag{1.34}$$

Here, $\alpha_1, \cdots, \alpha_n$ is obtained by solving the following normal equation

$$\boldsymbol{G}\boldsymbol{\alpha} = [\langle \boldsymbol{x}, \boldsymbol{y}_1 \rangle, \cdots, \langle \boldsymbol{x}, \boldsymbol{y}_n \rangle]^{\mathsf{T}}, \tag{1.35}$$

where

$$\boldsymbol{G} := \begin{bmatrix} \langle \boldsymbol{y}_1, \boldsymbol{y}_1 \rangle_{\mathcal{H}} & \langle \boldsymbol{y}_1, \boldsymbol{y}_2 \rangle_{\mathcal{H}} & \cdots & \langle \boldsymbol{y}_1, \boldsymbol{y}_r \rangle_{\mathcal{H}} \\ \langle \boldsymbol{y}_2, \boldsymbol{y}_1 \rangle_{\mathcal{H}} & \langle \boldsymbol{y}_2, \boldsymbol{y}_2 \rangle_{\mathcal{H}} & \cdots & \langle \boldsymbol{y}_2, \boldsymbol{y}_r \rangle_{\mathcal{H}} \\ \vdots & \vdots & \ddots & \vdots \\ \langle \boldsymbol{y}_r, \boldsymbol{y}_1 \rangle_{\mathcal{H}} & \langle \boldsymbol{y}_r, \boldsymbol{y}_2 \rangle_{\mathcal{H}} & \cdots & \langle \boldsymbol{y}_r, \boldsymbol{y}_r \rangle_{\mathcal{H}} \end{bmatrix} \tag{1.36}$$

is called Gram matrix. Given a closed subspace $\mathcal{M}$ in a Hilbert space $\mathcal{H}$, define a linear variety as $\mathcal{V} := \mathcal{M} + \boldsymbol{v}$ for some $\boldsymbol{v} \in \mathcal{H}$. Then, the projection $P_{\mathcal{V}}(\boldsymbol{x})$ of $\boldsymbol{x}$ onto $\mathcal{V}$ is has the following expressions.

1. $P_{\mathcal{V}}(\boldsymbol{x}) = P_{\mathcal{M}}(\boldsymbol{x} - \boldsymbol{v}) + \boldsymbol{v}$.
2. $P_{\mathcal{V}}(\boldsymbol{x}) = P_{\mathcal{M}}(\boldsymbol{x}) + P_{\mathcal{V}}(\theta)$.

We next show that the projection depends on metric design. Given any positive definite matrix $\left(\mathbb{R}^{L\times L} \ni\right) \boldsymbol{G} \succ 0$, we define an inner product and its induced norm, respectively, as $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{G}} := \boldsymbol{x}^{\mathsf{T}}\boldsymbol{G}\boldsymbol{y}, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^L$, and $\|\boldsymbol{x}\|_{\boldsymbol{G}} := \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\boldsymbol{G}}}, \forall \boldsymbol{x} \in \mathbb{R}^L$. Hereafter, we regard $\boldsymbol{G}$ as a *metric* which determines the metric-distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ by $\|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{G}}$.

Definition 1.10

(a) **$\boldsymbol{G}$-orthogonal:** *Vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are said to be $\boldsymbol{G}$-orthogonal, if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{G}} = 0$.*

(b) **$\boldsymbol{G}$-projection:** *Given a closed convex set $\mathcal{C} \subset \mathbb{R}$, the metric projection of $\boldsymbol{x}$ onto $\mathcal{C}$ in terms of the metric $\boldsymbol{G}$ is defined as*

$$P_{\mathcal{C}}^{\boldsymbol{G}}(\boldsymbol{x}) := \underset{\boldsymbol{y} \in \mathcal{C}}{\operatorname{argmin}} \|\boldsymbol{y} - \boldsymbol{x}\|_{\boldsymbol{G}}, \tag{1.37}$$

which is referred to as the $\boldsymbol{G}$-projection of $\boldsymbol{x}$ onto $\mathcal{C}$.

Example 1.1 *Define the hyperplane*

$$\Pi := \left\{ \boldsymbol{h} \in \mathbb{R}^L : \boldsymbol{h}^{\mathsf{T}} \boldsymbol{u} = d = \left\langle \boldsymbol{h}, \boldsymbol{G}^{-1} \boldsymbol{u} \right\rangle_{\boldsymbol{G}} \right\}, n \in \mathbb{N}. \tag{1.38}$$

In the Hilbert space equipped with the inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{A}}$, the normal vector of Π is $\boldsymbol{G}^{-1}\boldsymbol{u}$, and the $\boldsymbol{G}$-projection of $\boldsymbol{x}$ onto the hyperplane Π can be computed as

$$P_{\Pi}^{\boldsymbol{G}}(\boldsymbol{x}) = \boldsymbol{x} + \frac{d - \boldsymbol{x}^{\mathsf{T}} \boldsymbol{u}}{\boldsymbol{u}^{\mathsf{T}} \boldsymbol{G}^{-1} \boldsymbol{u}} \boldsymbol{G}^{-1} \boldsymbol{u}. \tag{1.39}$$

1.5.7 Universal approximation property of the Gaussian kernel

Definition 1.11 (Universality [61]) *A continuous kernel $\kappa(\cdot, \cdot)$ on a compact metric space (X, d) is called universal if the space of all functions induced by κ is dense in the space $C(X)$ of all continuous functions $f : X \to \mathbb{R}$, i.e., for every function $f \in C(X)$ and every $\epsilon > 0$ there exists a function g induced by κ with*

$$\|f - g\|_{\infty} \leq \epsilon, \tag{1.40}$$

where $\|\cdot\|_{\infty} := \sup_{x \in X} |f(x)|$ is the supremum norm.

Example 1 in [61] shows that the Gaussian kernel (1.5) is universal.

1.5.8 Normal Distribution

A normal distribution of a variate $x \in \mathbb{R}$ is a statistic distribution with probability density function

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2} \right) \tag{1.41}$$

and is denoted by $\mathcal{N}(\mu, \sigma^2)$, where $\mu \in \mathbb{R}$ is the mean and $\sigma > 0$ is the variance.

Chapter 2

A Fast KAF Algorithm Based on Parallel Projection

2.1 Introduction

In this chapter, we propose an efficient kernel adaptive filtering algorithm that is based on the following three key ideas: (i) *projection-along-subspace*, (ii) *selective updating*, and (iii) *data reusing by means of parallel projection*. Consider the set of vectors on the dictionary subspace $\mathcal{M}_n$ (see (1.9)) that nullify the instantaneous error. It is clearly the intersection of $\mathcal{M}_n$ and the hyperplane Π_n (see (1.8)) and it would thus be natural to project the current estimate φ_n onto $\mathcal{M}_n \cap \Pi_n$. It can be viewed that φ_n is projected onto Π_n along $\mathcal{M}_n$ (see Section 2.2.1). This is the idea of *projection-along-subspace*, enabling effective utilization of resources. The projection can be computed simply using the projection of $\kappa(\cdot, \boldsymbol{u}_n)$ onto $\mathcal{M}_n$, of which the computation involves the inversion of an $r_n \times r_n$ Gram matrix. This could make a significant impact to the overall complexity of the algorithm when r_n is large. Our idea to alleviate this computational issue is the *selective updating*: pick up a few elements that are maximally coherent to $\kappa(\cdot, \boldsymbol{u}_n)$ and then update their coefficients with the other coefficients fixed. Geometrically, the selection of the elements determines an affine subspace passing through the current estimate φ_n, and the φ_n is projected onto Π_n along the affine subspace. The third idea is *data reusing using parallel projection*. Some of the past data, as well as the current datum, are exploited for updating the estimate at each time, accommodated in hyperplanes, or more generally 'hyperslabs'. The current estimate φ_n is projected onto multiple hyperslabs in parallel, and then the projections are combined convexly. The errors for multiple data are suppressed simultaneously, and this contributes to enhancing the convergence/tracking speeds. As reported in [63] already, the parallel projection strategy is more robust to noise compared to the affine projection strategy. These three key ideas are unified into the *Φ-PASS) algorithm*.

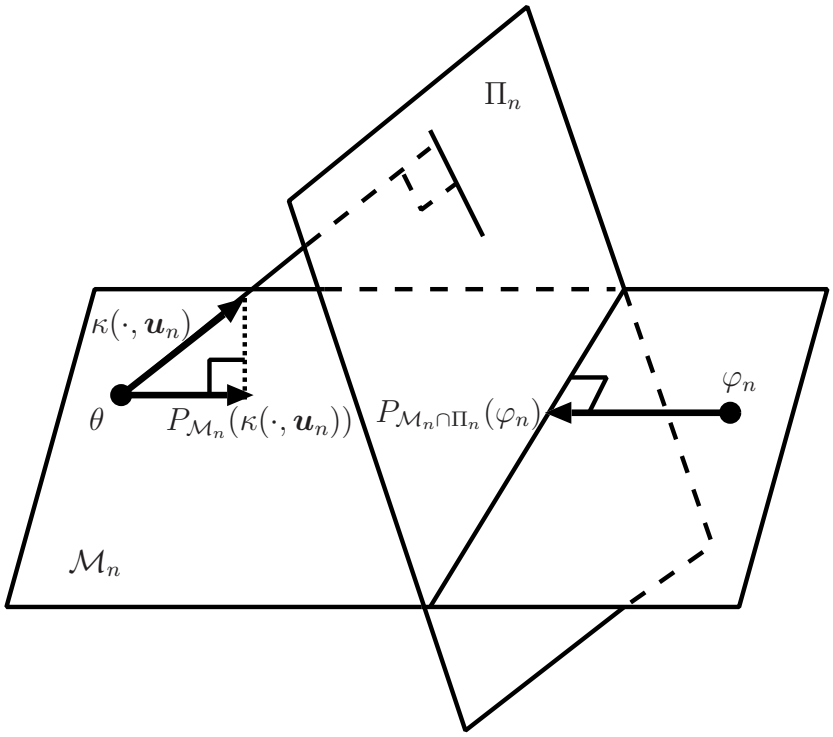

Figure 2.1: Projection-along-subspace when $\kappa(\cdot, \boldsymbol{u}_n) \notin \mathcal{M}_n$.

2.2 The Φ-PASS Algorithm

2.2.1 Key Ideas of the Proposed Algorithm

We present three key ideas of the proposed algorithm: (A) *projection-along-subspace*, (B) *selective update*, and (C) *parallel projection*. These ideas lead respectively to (A) high estimation-accuracy with a small dictionary size, (B) low complexity, and (C) fast convergence/tracking as well as noise robustness. Those properties are of particular importance in online scenarios.

Projection-along-subspace

The basic idea is the following: (i) nullify the instantaneous error for the current measurements, (ii) minimize the traveling distance from the current estimate φ_n to minimally disturb the previous learning, and (iii) stay in the dictionary subspace $\mathcal{M}_n$. This clearly suggests the use of the projection $P_{\mathcal{M}_n \cap \Pi_n}(\varphi_n)$ of the current point φ_n onto the intersection $\mathcal{M}_n \cap \Pi_n$, where Π_n is defined in (1.8). Figure 2.1 presents a geometric interpretation of $P_{\mathcal{M}_n \cap \Pi_n}(\varphi_n)$. One can first project the normal vector $\kappa(\cdot, \boldsymbol{u}_n)$ of Π_n onto the subspace $\mathcal{M}_n$. The projection $P_{\mathcal{M}_n \cap \Pi_n}(\varphi_n)$ can then be given in the form of $\varphi_n + \beta_n P_{\mathcal{M}_n}(\kappa(\cdot, \boldsymbol{u}_n))$ for some $\beta_n \in \mathbb{R}$ (see Section 2.2.2). It can be

seen in the figure that the current estimate φ_n is projected onto Π_n along the subspace $\mathcal{M}_n$.

Although the use of the projection-along-subspace yields excellent performance with a reasonable dictionary size, it involves the inversion of the $r_n \times r_n$ Gram matrix

$$\boldsymbol{G}_n = \begin{bmatrix} \kappa\left(\boldsymbol{u}_{j_1^{(n)}}, \boldsymbol{u}_{j_1^{(n)}}\right) & \cdots & \kappa\left(\boldsymbol{u}_{j_1^{(n)}}, \boldsymbol{u}_{j_{r_n}^{(n)}}\right) \\ \vdots & \ddots & \vdots \\ \kappa\left(\boldsymbol{u}_{j_{r_n}^{(n)}}, \boldsymbol{u}_{j_1^{(n)}}\right) & \cdots & \kappa\left(\boldsymbol{u}_{j_{r_n}^{(n)}}, \boldsymbol{u}_{j_{r_n}^{(n)}}\right) \end{bmatrix}. \tag{2.1}$$

which may need to be avoided in real time implementations. We therefore introduce the idea of the *selective update* to reduce the computational loads for the inversion in the following subsection.

Selective Update

The idea is the following: select a few elements that are maximally coherent to $\kappa(\cdot, \boldsymbol{u}_n)$ and then update the coefficients of the selected elements only. Let $\{\kappa(\cdot, \boldsymbol{u}_j))\}_{j \in \tilde{\mathcal{J}}_n}$ be the selected elements where

$$\tilde{\mathcal{J}}_n := \{\tilde{j}_1^{(n)}, \tilde{j}_2^{(n)}, \cdots, \tilde{j}_{s_n}^{(n)}\} \subset \mathcal{J}_n. \tag{2.2}$$

Here, $s_n (\leq r_n)$ is the number of selected elements at time n. A specific example of $\tilde{\mathcal{J}}_n$ will be given in Section 2.2.2. To update the coefficients of the selected elements with the other coefficients fixed, the direction vector from φ_n should lie in the subspace

$$\tilde{\mathcal{M}}_n := \mathrm{span}\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j \in \tilde{\mathcal{J}}_n} \subset \mathcal{M}_n. \tag{2.3}$$

This implies that the next estimate φ_{n+1} should lie in the linear variety (affine subspace)

$$\mathcal{V}_n := \tilde{\mathcal{M}}_n + \varphi_n := \{f + \varphi_n : f \in \tilde{\mathcal{M}}_n\}.$$

To embody the above arguments, we modify the item (iii) of the basic idea in Section 2.2.1 into the following: stay in the affine subspace $\mathcal{V}_n$. Under this modification, the current estimate φ_n is projected onto the intersection $\mathcal{V}_n \cap \Pi_n$.

Figure 2.2 presents a geometric interpretation of the selective update by means of $P_{\mathcal{V}_n \cap \Pi_n}(\varphi_n)$. It can be seen that the current estimate φ_n is projected onto Π_n along the affine subspace $\mathcal{V}_n$. In analogy with the case in the previous subsection, $P_{\mathcal{V}_n \cap \Pi_n}(\varphi_n)$ can be given in the form of $\varphi_n + \beta_n P_{\tilde{\mathcal{M}}_n}(\kappa(\cdot, \boldsymbol{u}_n))$ for some $\beta_n \in \mathbb{R}$.

The selective update reduces the inversion of the $r_n \times r_n$ Gram matrix

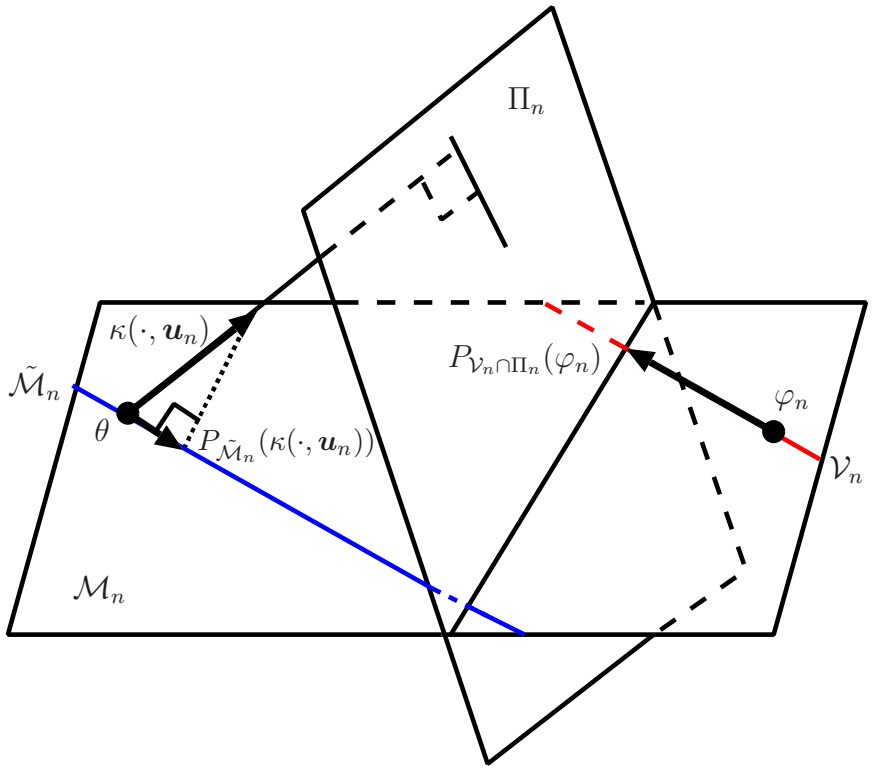

Figure 2.2: Selective update by means of the projection along an affine subspace $\mathcal{V}_n$ when $\kappa(\cdot, \boldsymbol{u}_n) \notin \mathcal{M}_n$.

to that of an $s_n \times s_n$ Gram matrix. We will show in Section 2.3 that the use of $s_n = 1$ gives reasonable performances and it brings a drastic reduction of computational loads.

Parallel Projection

We finally explain the *parallel projection*. The idea is the following: use the p most recent measurements at each time instant (data reusing), and suppress the errors for those p measurements simultaneously by operating p projections in parallel. The p measurements $\{(\boldsymbol{u}_\iota, d_\iota)\}_{\iota \in \mathcal{I}_n}$, $\mathcal{I}_n := \{n, n-1, \cdots, n-p+1\}$, are accommodated into the hyperplanes Π_ι, $\iota \in \mathcal{I}_n$. The current estimate φ_n is projected onto the hyperplanes in parallel, and then the projections are combined convexly. The resulting point (an average point of the projections) gives the direction in which the estimate travels. Figure 2.3 presents a geometric interpretation of the parallel projection combined with the idea of the projection-along-subspace.

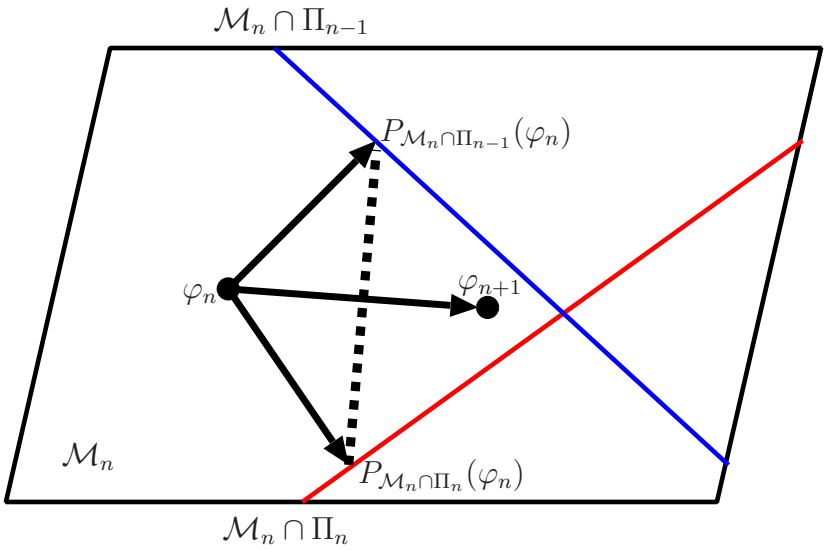

Figure 2.3: Parallel projection for $p = 2$ combined with the idea of projection-along-subspace.

2.2.2 The Proposed Algorithm

We first present the proposed algorithm which is based on the key ideas presented in Section 2.2.1. We then discuss the selective updating strategy, computational issues, and the relation to prior works. We finally present a convergence analysis of the proposed algorithm for the mathematically tractable case of non-selective update. For each of the p measurements $(\boldsymbol{u}_\iota, d_\iota)$, $\iota \in \mathcal{I}_n$, let $\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j \in \tilde{\mathcal{J}}_{n,\iota}}$ be the set of selected elements with the set of their associated indices $\tilde{\mathcal{J}}_{n,\iota} (\subset \mathcal{J}_n)$. Then, its corresponding affine subspace is given by

$$\mathcal{V}_{n,\iota} := \text{span}\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j \in \tilde{\mathcal{J}}_{n,\iota}} + \varphi_n \subset \mathcal{M}_n. \tag{2.4}$$

Instead of a zero-instantaneous-error hyperplane, we define its relaxed set, a bounded-instantaneous-error hyperslab, as follows:

$$\mathcal{S}_\iota := \{\varphi \in \mathcal{H} : (\varphi(\boldsymbol{u}_\iota) - d_\iota)^2 \leq \rho\}, \ \iota \in \mathcal{I}_n, \tag{2.5}$$

where $\rho \geq 0$ is the error bound. The hyperslabs $\mathcal{S}_\iota^{(n)}$ are referred to as *stochastic property sets* [63] as each of the sets contains the desired solution with high probability under some stochastic assumptions. The set-theoretic adaptive filtering approach [63, 44, 64, 65], stemming from the set-theoretic estimation framework [66], seeks to push an estimate towards the intersection of those stochastic property sets. It has been observed in [63, 44, 64, 67, 68, 65] that the use of parallel projection brings efficient adaptive algorithms. Note here that $\mathcal{S}_\iota = \Pi_\iota$ in the special case of $\rho = 0$.

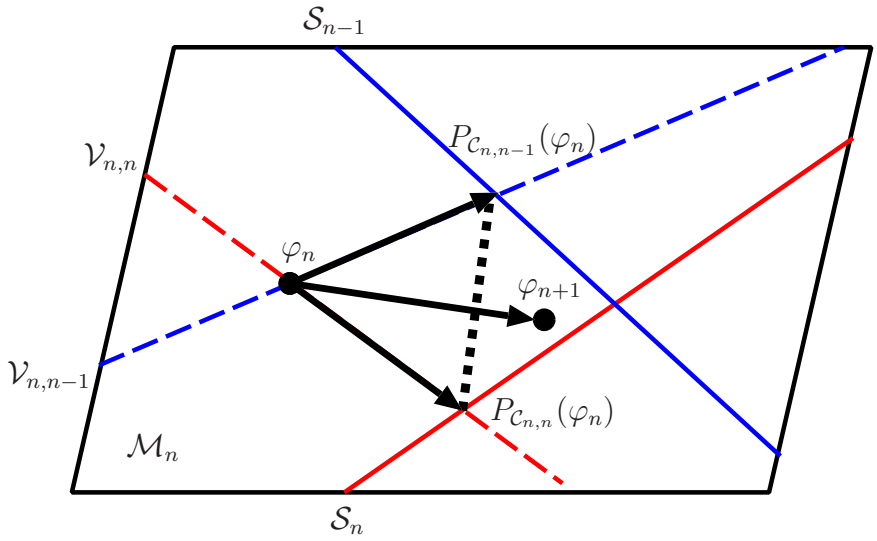

Figure 2.4: The Φ-PASS algorithm for $\rho = 0$ and $p = 2$.

The proposed algorithm projects the current estimate φ_n onto the following closed convex sets in parallel:

$$\mathcal{C}_{n,\iota} := \mathcal{V}_{n,\iota} \cap \mathcal{S}_\iota \subset \mathcal{M}_n,\ n \in \mathbb{N},\ \iota \in \mathcal{I}_n. \tag{2.6}$$

The proposed algorithm is given as follows; the derivation is given in Section 2.2.4.

Algorithm 2.1 (The Φ-PASS algorithm) *For the initial estimate* $\varphi_0 := \theta$*, generate the sequence* $(\varphi_n)_{n\in\mathbb{N}}$ *of nonlinear estimates by*

$$\varphi_{n+1} := \varphi_n + \lambda_n \left(\sum_{\iota \in \mathcal{I}_n} \omega_{n,\iota} P_{\mathcal{C}_{n,\iota}}(\varphi_n) - \varphi_n \right),\ n \in \mathbb{N}, \tag{2.7}$$

where $\omega_{n,\iota} \geq 0, \forall \iota \in \mathcal{I}_n$*, satisfying* $\sum_{\iota\in\mathcal{I}_n} \omega_{n,\iota} = 1$*, is the weight assigned to each set* $\mathcal{C}_{n,\iota}$ *and* $\lambda_n \in [0, 2\mu_n]$ *is the step size with the extrapolation coefficient*

$$\mu_n := \frac{\sum_{\iota\in\mathcal{I}_n} \omega_\iota^{(n)} \left\| P_{\mathcal{C}_{n,\iota}}(\varphi_n) - \varphi_n \right\|^2}{\left\| \sum_{\iota\in\mathcal{I}_n} \omega_\iota^{(n)} P_{\mathcal{C}_{n,\iota}}(\varphi_n) - \varphi_n \right\|^2} \geq 1. \tag{2.8}$$

In the trivial case that the denominator is zero, $\mu_n := 1$.

A geometric interpretation of the proposed algorithm is given in Figure 2.4. Geometrically, the current estimate φ_n is projected onto the hyperslabs $\mathcal{S}_n$ and $\mathcal{S}_{n-1}$ in parallel along the affine subspaces $\mathcal{V}_{n,n}$ and $\mathcal{V}_{n,n-1}$, respectively.

Let us now present the definition of *coherence* [27]:

$$c\left(\boldsymbol{u},\boldsymbol{v}\right) := \frac{\left|\kappa\left(\boldsymbol{u},\boldsymbol{v}\right)\right|}{\sqrt{\kappa\left(\boldsymbol{u},\boldsymbol{u}\right)}\sqrt{\kappa\left(\boldsymbol{v},\boldsymbol{v}\right)}},\ \boldsymbol{u},\ \boldsymbol{v} \in \mathcal{U}. \tag{2.9}$$

We present examples of dictionary construction and selective update below.

Example 2.1 (Dictionary construction with coherence [27]) *With $\mathcal{J}_{-1} := \emptyset$, the dictionary index set $\mathcal{J}_n$ is defined recursively as*

$$\mathcal{J}_n := \begin{cases} \mathcal{J}_{n-1} \cup \{n\}, & \text{if } \max_{j\in\mathcal{J}_n} c(\boldsymbol{u}_n,\boldsymbol{u}_j) \le \delta, \\ \mathcal{J}_{n-1}, & \text{otherwise,} \end{cases}$$

where $\delta \in (0,1)$.

Example 2.2 (Selective updating strategy with coherence) *For each $\iota \in \mathcal{I}_n$, construct $\tilde{\mathcal{J}}_{n,\iota}$ such that $c(\boldsymbol{u}_\iota,\boldsymbol{u}_j) \ge c(\boldsymbol{u}_\iota,\boldsymbol{u}_k)$ for all $j \in \tilde{\mathcal{J}}_{n,\iota}$ and $k \in \mathcal{J}_n \setminus \tilde{\mathcal{J}}_{n,\iota}$; i.e., select such elements that are maximally coherent to $\kappa(\cdot,\boldsymbol{u}_\iota)$.*

We finally show how to compute the projection in (2.7). Let $\mathcal{M}$ be a subspace of $\mathcal{H}$ such that $\kappa(\cdot,\boldsymbol{u}_j) \in \mathcal{M}$, $j \in \mathcal{J} := \{1,2,\cdots,s\}$, where $\boldsymbol{u}_j \in \mathcal{U}$. Given any $\varphi \in \mathcal{M}$, define

$$\mathcal{V} := \operatorname{span}\{\kappa(\cdot,\boldsymbol{u}_j)\}_{j\in\mathcal{J}} + \varphi = \tilde{\mathcal{M}} + \varphi \subset \mathcal{M}, \tag{2.10}$$

where $\tilde{\mathcal{M}} := \operatorname{span}\{\kappa(\cdot,\boldsymbol{u}^{(j)})\}_{j\in\mathcal{J}}$. Given $\boldsymbol{u} \in \mathcal{U}$ and $d \in \mathbb{R}$, define a hyperslab $\mathcal{S} := \{f \in \mathcal{H} : \left(f\left(\boldsymbol{u}\right)-d\right)^2 \le \rho\}$. We assume that $\kappa(\cdot,\boldsymbol{u}) \not\perp \tilde{\mathcal{M}}$; this assumption will be justified later on. It can then be shown that $C := \mathcal{V} \cap \mathcal{S}$ is nonempty. The projection onto the nonempty closed convex set $C(\subset \mathcal{M})$ is given by

$$P_{\mathcal{C}}(\varphi) = \varphi + bP_{\tilde{\mathcal{M}}}(\kappa(\cdot,\boldsymbol{u})) \tag{2.11}$$

where

$$b = \varsigma\frac{\max\{|d-\varphi\left(\boldsymbol{u}\right)|-\sqrt{\rho},0\}}{\sum_{j\in\mathcal{J}}\alpha_j\kappa\left(\boldsymbol{u},\boldsymbol{u}_j\right)}, \tag{2.12}$$

$$P_{\tilde{\mathcal{M}}}(\kappa(\cdot,\boldsymbol{u})) = \sum_{j\in\mathcal{J}}\alpha_j\kappa(\cdot,\boldsymbol{u}_j). \tag{2.13}$$

Here, $\varsigma := \operatorname{sign}(d-\varphi(\boldsymbol{u}))$ with the signum function $\operatorname{sign}(\cdot)$, and the coefficient vector $\boldsymbol{\alpha} := [\alpha^{(1)},\alpha^{(2)},\cdots,\alpha^{(s)}]^{\mathsf{T}} \in \mathbb{R}^s$ is obtained by solving the following

Table 2.1: Summary of the Φ-PASS algorithm.

The Φ-PASS Algorithm
Requirement : $\lambda_n \in [0, 2\mu_n]$, $\omega_{n,\iota} \geq 0$ with $\sum_{\iota \in \mathcal{I}_n} \omega_{n,\iota} = 1$
Estimate output : $\varphi_n(\boldsymbol{u}_n) := \sum_{j \in \mathcal{J}_n} h_n^{(j)} \kappa(\boldsymbol{u}_n, \boldsymbol{u}_j)$
Estimate update :
1. Construct $\mathcal{J}_n$ based on a novelty criterion (e.g., the coherence)
2. $h_{n,n} := 0$ if $n \in \mathcal{J}_n$
3. Construct $\tilde{\mathcal{J}}_{n,\iota} (\subset \mathcal{J}_n)$ (see Example 2.2)
4. Compute φ_{n+1} based on (2.7) and (2.11)–(2.16).

normal equation:

$$\boldsymbol{G}\boldsymbol{\alpha} = \boldsymbol{y} \tag{2.14}$$

for

$$\boldsymbol{G} := \begin{bmatrix} \kappa(\boldsymbol{u}_1, \boldsymbol{u}_1) & \cdots & \kappa(\boldsymbol{u}_1, \boldsymbol{u}_s) \\ \vdots & \ddots & \vdots \\ \kappa(\boldsymbol{u}_s, \boldsymbol{u}_1) & \cdots & \kappa(\boldsymbol{u}_s, \boldsymbol{u}_s) \end{bmatrix} \tag{2.15}$$

$$\boldsymbol{y} := [\kappa(\boldsymbol{u}, \boldsymbol{u}_1), \kappa(\boldsymbol{u}, \boldsymbol{u}_2), \cdots \kappa(\boldsymbol{u}, \boldsymbol{u}_s)]^{\mathsf{T}}. \tag{2.16}$$

In the case of $\varphi \in C (\Leftrightarrow \beta = 0)$, it is clear by definition that $P_{\mathcal{C}}(\varphi) = \varphi$ so that (2.11) holds. Assume now that $\varphi \notin C (\Leftrightarrow \beta \neq 0)$. In this case, one can verify that the $P_{\mathcal{C}}(\varphi)$ in (2.11) lies on the boundary (closer to φ) of $\mathcal{C}$ in the affine subspace $\mathcal{V}$. Hence, to prove (2.11), it is sufficient to show that $P_{\tilde{\mathcal{M}}}(\kappa(\cdot, \boldsymbol{u})) \perp \tilde{\mathcal{M}} \cap \tilde{\Pi}$, where $\tilde{\Pi} := \{f \in \mathcal{H} : \langle f, \kappa(\cdot, \boldsymbol{u}) \rangle = 0\}$ is a translation of the boundary hyperplanes of $\mathcal{S}$ [69]. This can be readily proved by noting the orthogonal decomposition $\kappa(\cdot, \boldsymbol{u}) = P_{\tilde{\mathcal{M}}}(\kappa(\cdot, \boldsymbol{u})) + P_{\tilde{\mathcal{M}}^{\perp}}(\kappa(\cdot, \boldsymbol{u}))$ together with $\kappa(\cdot, \boldsymbol{u}) \perp \tilde{\Pi}$ and $P_{\tilde{\mathcal{M}}^{\perp}}(\kappa(\cdot, \boldsymbol{u})) \perp \tilde{\mathcal{M}}$. Here, $\tilde{\mathcal{M}}^{\perp}$ is the orthogonal complement of $\tilde{\mathcal{M}}$.

The denominator in (2.12) can be written as

$$\sum_{j \in \mathcal{J}} \alpha^{(1)} \kappa(\boldsymbol{u}, \boldsymbol{u}_1) = \left\langle P_{\tilde{\mathcal{M}}}(\kappa(\cdot, \boldsymbol{u})), \kappa(\cdot, \boldsymbol{u}) \right\rangle = \left\| P_{\tilde{\mathcal{M}}}(\kappa(\cdot, \boldsymbol{u})) \right\|^2. \tag{2.17}$$

The use of the coherence for the dictionary construction excludes the trivial case of $\kappa(\cdot, \boldsymbol{u}) \perp \mathcal{M}$ since the new measurement is perfectly novel in this case. Under the use of the coherence also for the selective update (see Example 2.2), it can readily be verified that $\kappa(\cdot, \boldsymbol{u}) \not\perp \tilde{\mathcal{M}}$, which implies that $P_{\tilde{\mathcal{M}}}(\kappa(\cdot, \boldsymbol{u})) \neq \theta$. Hence, the denominator in (2.12) is ensured to be nonzero under the use of the coherence criterion. The coherence is thus a reasonable criterion for the proposed algorithm. The Φ-PASS algorithm is summarized in Table 2.1.

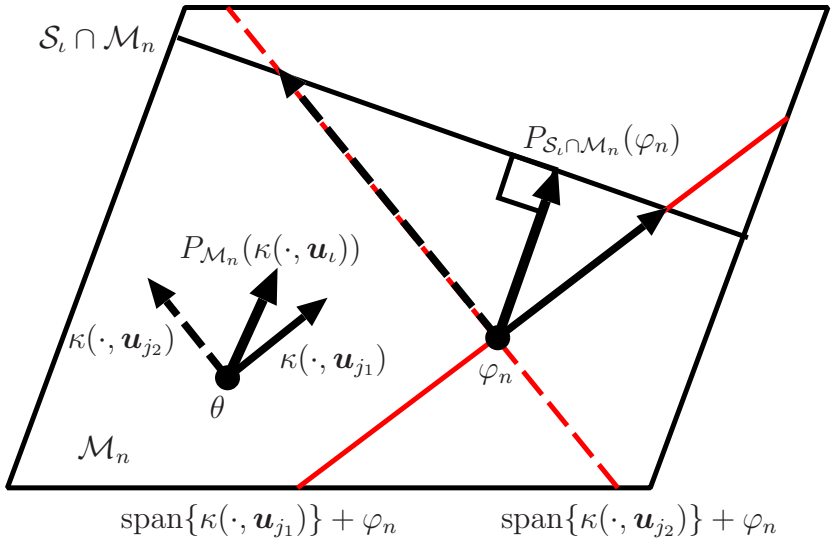

Figure 2.5: The selective updating strategy in Example 2.2.

2.2.3 Discussions

On the selective updating strategy in Example 2.2: Given $\iota \in \mathcal{I}_n$, it holds that $\langle \kappa(\cdot, \boldsymbol{u}_\iota), \kappa(\cdot, \boldsymbol{u}_j) \rangle = \langle P_{\mathcal{M}_n}(\kappa(\cdot, \boldsymbol{u}_\iota)), \kappa(\cdot, \boldsymbol{u}_j) \rangle$ for any $j \in \mathcal{J}_n$, since $\kappa(\cdot, \boldsymbol{u}_j) \in \mathcal{M}_n$. One can thus verify that

$$c(\boldsymbol{u}_\iota, \boldsymbol{u}_j) = \left| \left\langle \frac{P_{\mathcal{M}_n}(\kappa(\cdot, \boldsymbol{u}_\iota))}{\|\kappa(\cdot, \boldsymbol{u}_\iota)\|}, \frac{\kappa(\cdot, \boldsymbol{u}_j)}{\|\kappa(\cdot, \boldsymbol{u}_j)\|} \right\rangle \right|. \tag{2.18}$$

Since

$$\operatorname*{argmax}_{j \in \mathcal{J}_n} c(\boldsymbol{u}_\iota, \boldsymbol{u}_j) = \operatorname*{argmax}_{j \in \mathcal{J}_n} \left| \left\langle \frac{P_{\mathcal{M}_n}(\kappa(\cdot, \boldsymbol{u}_\iota))}{\|P_{\mathcal{M}_n}(\kappa(\cdot, \boldsymbol{u}_\iota))\|}, \frac{\kappa(\cdot, \boldsymbol{u}_j)}{\|\kappa(\cdot, \boldsymbol{u}_j)\|} \right\rangle \right|,$$

one can see that such elements $\kappa(\cdot, \boldsymbol{u}_j)$ that have the least angles with the line $\mathrm{span}\{P_{\mathcal{M}_n}(\kappa(\cdot, \boldsymbol{u}_\iota))\}$ are selected. Note here that

$$P_{S_\iota \cap \mathcal{M}_n}(\varphi_n) - \varphi_n \in \mathrm{span}\{P_{\mathcal{M}_n}(\kappa(\cdot, \boldsymbol{u}_\iota))\}. \tag{2.19}$$

A geometric interpretation is given in Figure 2.5. One can see that $\kappa(\cdot, \boldsymbol{u}_{j_1})$ has a smaller angle with $\mathrm{span}\{P_{\mathcal{M}_n}(\kappa(\cdot, \boldsymbol{u}_\iota))\}$ than $\kappa(\cdot, \boldsymbol{u}_{j_2})$. We remark here that the projection along $\mathrm{span}\{\kappa(\cdot, \boldsymbol{u}_{j_1})\}$ gives a better approximation of $P_{S_\iota \cap \mathcal{M}_n}(\varphi_n)$ than the other one, the projection along $\mathrm{span}\{\kappa(\cdot, \boldsymbol{u}_{j_2})\}$. The Pythagorean theorem guarantees in general that the smaller the angle, the better the approximation. This justifies the strategy presented in Example 2.2.

Table 2.2: Computational complexities of the proposed and existing algorithms.

QKLMS	$(L+2)r_n$
Φ-PASS ($p=1$)	$(L+2)r_n + s_n^2 + 4s_n + \upsilon_{\mathrm{inv}}(s_n)$
KAP	$(L+p^2+2p+1)r_n + 2p^2 + p + \upsilon_{\mathrm{inv}}(p)$
KRLS Tracker	$2r_n^2 + (L+9)r_n + \upsilon_{\mathrm{inv}}(r_n)$
Φ-PASS	$(L+2p+2)r_n + s_n^2(2p+1) + s_n(6p+1) + p\upsilon_{\mathrm{inv}}(s_n)$
Φ-PASS-Full	$(2p+1)r_n^2 + (L+7p+2)r_n + \upsilon_{\mathrm{inv}}(r_n)$

Computational complexity: We discuss the complexity of the proposed algorithm under the selective updating strategy in Example 2.2. The computational complexity of kernel adaptive filtering algorithms at each time instant $n \in \mathbb{N}$ is generally given in terms of the dictionary size r_n as well as the dimension L of the input space $\mathcal{U}$. As the complexity depends on the kernel employed, it is supposed that a Gaussian kernel is employed (see Section 2.3). The computational complexity of the proposed algorithm depends also on the cardinality $\left|\tilde{\mathcal{J}}_{n,\iota}\right|$ which is supposed to be a constant s_n for all $\iota \in \mathcal{I}_n$ at each time instant $n \in \mathbb{N}$. The case of the non-selective update (i.e., $\tilde{\mathcal{J}}_{n,\iota} = \mathcal{J}_n$, $\forall \iota \in \mathcal{I}_n$) is referred to specially as the fully-updating Φ-PASS algorithm, abbreviated as Φ-PASS-Full. In the Φ-PASS-Full algorithm, the Gram matrix (cf. $\boldsymbol{G}$ in (2.15)) involved in the computation of $P_{\mathcal{C}_{n,\iota}}(\varphi_n)$ is kept constant as long as the dictionary remains unchanged. When the dictionary is updated, the Gram matrix evolves and its inverse of the previous $(r_n-1)\times(r_n-1)$ Gram matrix can be updated into the inverse of the new $r_n \times r_n$ Gram matrix by using the formula for the inverse of a partitioned matrix together with the matrix inversion lemma [70].[1]

Table 2.2 summarizes the overall per-iteration (asymptotic) complexity (the number of real multiplications) of the proposed algorithm and the existing algorithms: QKLMS [29], KAP [27], and KRLS Tracker [30]. The precise complexity (rather than the asymptotic one) of Φ-PASS for $p=1$ is given by $(L+s_n+1)r_n + s_n^2 + 4s_n$, excluding the complexity $O(s_n^3)$ for the inversion. The complexity contains all the computations required per iteration such as that for the dictionary construction and the selection of the coefficients updated. A comparison is counted as a multiplication. Figure 2.6 illustrates the complexity as a function of the dictionary size r_n for $L=4$. The curves are plotted based on the precise complexities with $O(N^t)$ counted as N^t for any positive integers N and t. For instance, the curve of Φ-PASS for $p=1$ is plotted with $(L+s_n+1)r_n + s_n^3 + s_n^2 + 4s_n$ for $s_n = 1$. The affine

[1]The size of the matrix is incremented by one and its $(r_n-1)\times(r_n-1)$ submatrix is exactly the same as the Gram matrix at the previous iteration. We, therefore, do not count the computation for the inverse of the submatrix.

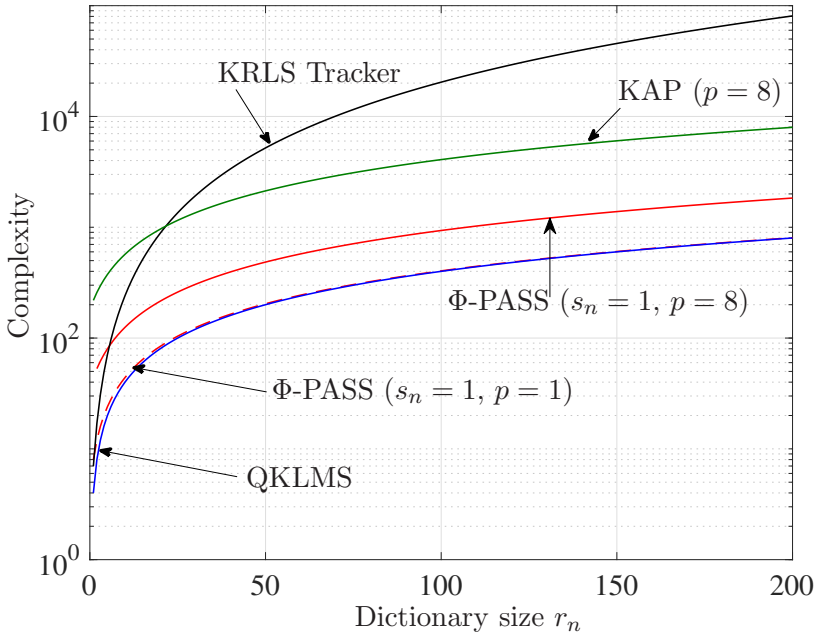

Figure 2.6: Comparisons of the proposed and conventional algorithms in computational complexity for $L = 4$.

order (the data-reusing factor) of KAP is set to $p = 8$. For $r_n = 200$, for instance, Φ-PASS ($p = 8$) approximately requires 3.9 times less complexity than KAP and 136 times less complexity than KRLS Tracker, respectively. Although Φ-PASS ($p = 8$) requires higher complexities than QKLMS, it has significant advantages in performance, as shown in Section 2.3.

Relation to other prior works: Let $\rho = 0$ and $p = 1$, employ a Gaussian kernel, and adopt the updating strategy presented in Example 2.2 for $|\tilde{\mathcal{J}}_{n,\iota}| = 1,\ \forall \iota \in \mathcal{I}_n,\ n \in \mathbb{N}$. In this case, $\tilde{\mathcal{J}}_{n,n} := \operatorname{argmax}_{j\in\mathcal{J}_n} c(\boldsymbol{u}_n, \boldsymbol{u}_j) = \operatorname{argmin}_{j\in\mathcal{J}_n} \|\boldsymbol{u}_n - \boldsymbol{u}_j\|$, meaning that $\boldsymbol{u}_n$ is quantized to its nearest point in $\{\boldsymbol{u}_j\}_{j\in\mathcal{J}_n}$. The Φ-PASS algorithm is thus reduced in this case to the QKLMS algorithm [29].

Let $\rho = 0$, $p = 1$, and $\lambda_n := \frac{\sum_{j\in\mathcal{J}_n} \alpha^{(j)} \kappa(\boldsymbol{u}_n, \boldsymbol{u}_j)}{\kappa(\boldsymbol{u}_n, \boldsymbol{u}_n)}$ for the Φ-PASS-Full algorithm with a Gaussian kernel employed. In this case, the proposed algorithm is reduced to the sparse sequential algorithm [71]. One can verify that $\lambda_n \leq 1 (< 2\mu_n)$ by substituting $\boldsymbol{u} = \boldsymbol{u}_n$ into (2.17) and by noting that $\left\|P_{\tilde{\mathcal{M}}}(\kappa(\cdot, \boldsymbol{u}))\right\|^2 \leq \|\kappa(\cdot, \boldsymbol{u})\|^2 = \kappa(\boldsymbol{u}, \boldsymbol{u})$.

To sum up the above arguments, the proposed algorithm for $p = 1$ includes QKLMS and the sparse sequential algorithm as its two extreme cases in the following sense: QKLMS updates only one coefficient at each iteration whereas the sparse sequential algorithm updates all coefficients. *The*

proposed algorithm bridges the gap between the two important algorithms in a systematic way. In the sparse sequential algorithm, the estimate φ_n is projected onto Π_n and, if the projection is sufficiently close to the dictionary subspace $\mathcal{M}_n$, it is again projected back onto $\mathcal{M}_n$. The sparse sequential algorithm therefore requires the inversion of an $r_n \times r_n$ Gram matrix to compute the projection onto $\mathcal{M}_n$. Likewise, the proposed algorithm *with non-selective update* requires the inversion of an $r_n \times r_n$ Gram matrix. At the price of computing the inversion, the convergence behavior is improved significantly by updating all the coefficients at every iteration (see Section 2.2.1). A unique, and also attracting, feature of the proposed algorithm is *the selective updating* which yields a drastic reduction of the complexity by avoiding the computation of the full $r_n \times r_n$ Gram matrix. While the proposed algorithm can select and update an arbitrary number of coefficients, the QKLMS algorithm can select and update *only one coefficient* at each iteration. It has been reported that updating only one coefficient could cause considerable performance degradations, and updating slightly more (such as $s_n = 5$) coefficients often yields nearly identical results to Φ-PASS-full with a marginal increase of complexity [46].

In the machine learning community, online classification methods with kernels have been proposed [72, 73, 74]. One of the central issues of online kernel-based classification is an unbounded number of support vectors, which corresponds to the dictionary size in the present context. The common approach among the above methods to handling this issue is fixing the number of support vectors at the predefined *budget*. The idea of *budget learning* has been applied indeed to kernel adaptive filtering [30, 43]. In Section 2.3, the proposed algorithm is compared with the method in [30] which is one of the state-of-the-art algorithms based on the budget learning.

2.2.4 Monotone Approximation and Convergence Analysis

The proposed algorithm is based on the framework of set-theoretic adaptive filtering [63, 44, 64, 65], which stems from the set-theoretic estimation concept [66]. In this framework, a desired solution (a best approximation of the ψ in $\mathcal{M}_n$) is characterized as a common point of the so-called *stochastic property sets* at each time instant. In the present context, the convex sets $\mathcal{C}_{n,\iota}$ correspond to the stochastic property sets, containing the desired solution with high probability under some stochastic assumptions (see [63] for details). The idea of set-theoretic adaptive filtering is pushing an estimate towards the intersection of the stochastic property sets.

Define a sequence of convex functions $(\Theta_n)_{n\in\mathbb{N}}$ as follows. If $\varphi_n \in \bigcap_{\iota\in\mathcal{I}_n} \mathcal{C}_{n,\iota}$, then let $\Theta_n(\varphi) := 0$ for all $\varphi \in \mathcal{H}$. Otherwise, we have $\nu_n := \sum_{\iota\in\mathcal{I}_n} \omega_{n,\iota} d(\varphi_n, \mathcal{C}_{n,\iota}) \neq 0$, where $d(\varphi, \mathcal{C}_{n,\iota}) := \min_{f\in\mathcal{C}_{n,\iota}} \|\varphi - f\|$ $= \left\|\varphi - P_{\mathcal{C}_{n,\iota}}(\varphi)\right\|$ is a metric distance function, and let

$$\Theta_n(\varphi) := \sum_{\iota\in\mathcal{I}_n} \frac{\omega_{n,\iota} d(\varphi_n, \mathcal{C}_{n,\iota})}{\nu_n} d(\varphi, \mathcal{C}_{n,\iota}),\ \varphi \in \mathcal{H}. \tag{2.20}$$

The additional weight $d(\varphi_n, \mathcal{C}_{n,\iota})$ emphasizes those sets which are more distant from the current estimate φ_n than the other sets. A minimizer of Θ_n is such a point that is maximally consistent with the information accommodated by each set $\mathcal{C}_{n,\iota}$. The set of minimizers of Θ_n is therefore expected to contain the desired solution. The set of minimizers is simply the intersection of $\mathcal{C}_{n,\iota}$s if the intersection is nonempty. One can thus make an estimate closer to the desired solution by suppressing the 'time-varying' objective function Θ_n at each iteration. (See [65] for comprehensive descriptions of the set-theoretic adaptive filtering.) We use APSM [44] presented below to minimize the sequence $(\Theta_n)_{n\in\mathbb{N}}$ of 'time-varying' objective functions asymptotically.

Scheme 2.1 (APSM [44] with no constraint)

$$\varphi_{n+1} := \begin{cases} \varphi_n - \lambda_n \dfrac{\Theta_n(\varphi_n)}{\|\Theta_n'(\varphi_n)\|^2} \Theta_n'(\varphi_n), & \text{if } \Theta_n'(\varphi_n) \neq \theta, \\ \varphi_n, & \text{otherwise,} \end{cases} \tag{2.21}$$

where $\varphi_0 \in \mathcal{H}$, $\lambda_n \in [0, 2]$, *and* $\Theta_n'(\varphi_n)$ *is a subgradient of* Θ_n *at* φ_n *[44].*

Substituting $\Theta_n'(\varphi_n) := \frac{1}{\nu_n} \sum_{\iota\in\mathcal{I}_n} \omega_{n,\iota}(\varphi_n - P_{\mathcal{C}_{n,\iota}}(\varphi_n))$ and (2.20) for $\varphi = \varphi_n$ with $d(\varphi_n, \mathcal{C}_{n,\iota}) = \left\|\varphi_n - P_{\mathcal{C}_{n,\iota}}(\varphi_n)\right\|$ into (2.21) reproduces the Φ-PASS algorithm in (2.7) and (2.8). In the following, we present a deterministic analysis of the proposed algorithm. We first show that φ_n approaches every minimizer of Θ_n monotonically at each iteration.

Theorem 2.1 (Monotone Approximation) *Assume that* $\Omega_n := \operatorname{arginf}_{\varphi\in\mathcal{H}} \Theta_n(\varphi) \neq \emptyset$ *and that* $\varphi_n \notin \Omega_n$. *Then, it holds that* $\|\varphi_{n+1} - \varphi^*\| < \|\varphi_n - \varphi^*\|$, $\forall \varphi^* \in \Omega_n$, *for any* $\lambda_n \in \left(0, 2\left(1 - \frac{\Theta_n^*}{\Theta_n(\varphi_n)}\right)\right)$, *where* $\Theta_n^* := \inf_{\varphi\in\mathcal{H}} \Theta_n(\varphi)$.

Proof: The claim can be verified directly by [44, Theorem 2(a)]. □

The monotone approximation property shown in Theorem 2.1 suggests that φ_n would approach the desired solution monotonically at each iteration since the desired solution is expected to be a minimizer of Θ_n. This property

is quite important in practice to ensure the stability of the proposed algorithm. It is also a fundamental question of theoretical interests whether the sequence $(\varphi_n)_{n\in\mathbb{N}}$ generated by the proposed algorithm is convergent; if it is convergent, questions are whether the limit point as well as the sequence $(\varphi_n)_{n\in\mathbb{N}}$ has any optimality, and how the limit point can be characterized. To allow further analyses to answer this question, we make the following set of assumptions.

Assumption 2.1 (Required for Theorem 2.2)

1. *Step-size condition:* $(\lambda_n)_{n\in\mathbb{N}} \subset [\mu_n\epsilon_1, \mu_n(2-\epsilon_2)] \subset (0, 2\mu_n)$, $\exists \epsilon_1, \epsilon_2 > 0$.

2. *Dictionary constancy: there exists some* $N_0 \in \mathbb{N}$ *such that* $\mathcal{M}_n = \mathcal{M}_{N_0}$ *for all* $n \geq N_0$.

3. *Data consistency: there exists some* $N_1 \geq N_0$ *such that* $\mathcal{C}_{N_1} := \bigcap_{\iota\in\tilde{\mathcal{I}}_n, n\geq N_1} \mathcal{C}_{n,\iota}$ *has a relative interior with respect to the subspace* $\mathcal{M}_{N_0}$*, where* $\tilde{\mathcal{I}}_n := \{\iota \in \mathcal{I}_n : \varphi_n \notin \mathcal{C}_{n,\iota}\}$.[2]

The dictionary constancy (Assumption 2.1.2) is a realistic assumption under the use of the coherence criterion (see Example 2.1) because it has been proven in [27] that the dictionary size remains finite as the time index n goes to infinity provided that the input space $\mathcal{U}$ is compact. The data consistency (Assumption 2.1.3) assumes, roughly speaking, the existence of a small 'mass' of points in $\mathcal{M}_{N_0}$ each of which makes bounded instantaneous errors for all data observed after the time instant N_1. It is therefore assumed implicitly that the noise contained in d_n is bounded (see [65] for instance). Under the use of a Gaussian kernel, one can naturally assume that $\psi \in \mathcal{H}$ (allowing an infinitesimal discrepancy) due to its universality [75]. If the dictionary is well constructed, one may also assume that there are good estimates of ψ in $\mathcal{M}_{N_0}$; small errors could be regarded as noise. Another implicit assumption behind the data consistency is a disuse of the selective updating strategy. This is because, if $\tilde{\mathcal{J}}_{n,\iota} \subsetneq \mathcal{J}_n$, $\mathcal{V}_{n,\iota} = \text{span}\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j\in\tilde{\mathcal{J}}_{n,\iota}} + \varphi_n \subsetneq \mathcal{M}_{N_0} = \text{span}\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j\in\mathcal{J}_n}$ and thus $\mathcal{V}_{n,\iota}$ has no relative-interior with respect to $\mathcal{M}_{N_0}$, meaning that there is no chance for its subset $\mathcal{C}_{n,\iota}(\subset \mathcal{V}_{n,\iota})$ to have a relative interior in this case. To sum up, the data-consistency assumption comes with five implications: (i) an appropriate design of the error bound ρ appearing in (2.5) (cf. [63]), (ii) bounded noise, (iii) appropriate kernels such as Gaussian, (iv) well-constructed dictionaries, and (v) the disuse of the selective updating strategy. In fact, the appropriate design of ρ involves multiple factors such as environmental noise, modeling errors, and imperfection of the dictionary. We stress however that

[2] A point $\tilde{\varphi} \in \mathcal{C}_{N_1}$ is called a relative interior of $\mathcal{C}_{N_1}$ with respect to $\mathcal{M}_{N_0}$ if there exists some $\varepsilon > 0$ such that $\{\varphi \in \mathcal{M}_{N_0} : \|\varphi - \tilde{\varphi}\| < \varepsilon\} \subset \mathcal{C}_{N_1}$.

the above implications never restrict the applicability of the proposed algorithm. It will actually be shown in Section 2.3 that the proposed algorithm performs well with $\rho := 0$ and also with the selective updating strategy used, and that the use of the selective update brings substantial reductions of computational costs without causing any serious degradation of performance. We also make the following assumption, which is optional and can be eliminated, for simplifying the proof of our convergence analysis.

Assumption 2.2 (Optional for Theorem 2.2) *The dictionary is constructed in an incremental manner, i.e., $\mathcal{M}_0 \subseteq \mathcal{M}_1 \subseteq \mathcal{M}_2 \subseteq \mathcal{M}_3 \cdots$, so that $\varphi_n \in \mathcal{M}_{N_0}$ for all $n \in \mathbb{N}$ under Assumption 2.1.2.*

Assumption 2.2 is automatically guaranteed under the use of the coherence criterion (see Example 2.1). We now present the convergence analysis below.

Theorem 2.2 (Convergence Analysis) *Under Assumptions 2.1 and 2.2, the following statements hold.*

(a) Convergence and asymptotic optimality: The sequence $(\varphi_n)_{n\in\mathbb{N}}$ generated by Algorithm 2.1 converges to some $\hat{\varphi} \in \mathcal{H}$ and $\lim_{n\to\infty} \Theta_n(\varphi_n) = \lim_{n\to\infty} \Theta_n(\hat{\varphi}) = 0$.

(b) Characterization of the limit point: Suppose that $\inf_{n\geq N_1, \iota\in\mathcal{I}_n} \omega_{n,\iota} > 0$. Then, the limit point is characterized by $\hat{\varphi} \in \overline{\liminf_{n\to\infty} \bigcap_{\iota\in\tilde{\mathcal{I}}_n} \mathcal{C}_{n,\iota}}$, where $\liminf_{n\to\infty} \bigcap_{\iota\in\tilde{\mathcal{I}}_n} \mathcal{C}_{n,\iota} := \bigcup_{n=0}^{\infty} \bigcap_{k\geq n} \left(\bigcap_{\iota\in\tilde{\mathcal{I}}_k} \mathcal{C}_{k,\iota} \right)$ and the overline denotes the closure.

Proof: To apply [44, Theorem 2(b)–(d)] directly for verifying the claims, we need to assume that the set $\mathcal{C}_{N_1}$ has an interior. This is however unrealistic because $\mathcal{M}_{N_0}(\supset \mathcal{C}_{N_1})$ is strictly smaller than the whole space $\mathcal{H}$ in most cases and thus has no interior. Strictly speaking, to verify only Theorem 2.2(a), the assumption can be relaxed into the following: $\mathcal{C}_{N_1}$ has a relative interior with respect to some hyperplane. This is still unrealistic for the same reason unfortunately.

Under Assumptions 2.1.2 and 2.2, we regard $\mathcal{M}_{N_0}$ as a (whole) Hilbert space since $\varphi_n \in \mathcal{M}_{N_0}$ for all $n \in \mathbb{N}$; the domain of Θ_n is restricted to $\mathcal{M}_{N_0}$ accordingly. Then, the existence of an interior of $\mathcal{C}_{N_1}$ is ensured by Assumption 2.1.3. The other conditions in [44, Theorem 2(b)–(d)] can be verified by following the way in [44, Proposition 3(a), (b)]. Thus, [44, Theorem 2(b)–(d)] can be applied to verify the claims. □

The reader may refer to [76, 77, 78] for more information about the proof technique shown above; the discussion is presented there under general linear constraints. Theorem 2.2 guarantees that the point sequence $(\varphi_n)_{n\in\mathbb{N}}$ converges to a point $\hat{\varphi}$ in $\overline{\liminf_{n\to\infty} \bigcap_{\iota\in\tilde{\mathcal{I}}_n} \mathcal{C}_{n,\iota}}$ and that the limit point

$\hat{\varphi}$ as well as the point sequence $(\varphi_n)_{n\in\mathbb{N}}$ is asymptotically optimal (i.e., it minimizes the sequence $(\Theta_n)_{n\in\mathbb{N}}$ of objective functions asymptotically).

2.3 Experiments of the Φ-PASS Algorithm

We show the practical advantages of the Φ-PASS algorithm in its applications to online prediction of the following time-series data.

Data A : Stationary data generated by

$$\begin{aligned} d_n &:= \psi_{\mathrm{A}}(d_{n-1}, d_{n-2}) \\ &:= (0.2 - 0.7\exp(-d_{n-1}^2))d_{n-1} - 0.8(1 + \exp(-d_{n-1}^2)d_{n-2}) \\ &\quad + 0.2\sin(d_{n-1}\pi) \end{aligned} \tag{2.22}$$

for $d_{-2} := d_{-1} := 0.1$. We show that the Φ-PASS algorithm enjoys high estimation-accuracy with a small dictionary size, fast convergence, low complexity, and noise robustness in Section 2.3.2.

Data B : Nonstationary data generated by

$$\begin{aligned} d_n &:= (0.8 - 0.5\exp(d_{n-1}^2))d_{n-1} - (0.3 + 0.9\exp(-d_{n-1}^2))d_{n-2} \\ &\quad + 0.1\sin(d_{n-1}\pi) \end{aligned} \tag{2.23}$$

for $0 \le n \le 2500$ ($d_{-2} := d_{n-1} := 0.1$) and $d_n := \psi_{\mathrm{A}}(d_{n-1}, d_{n-2})$ for $n > 2500$. We show the tracking capability to the nonstationarity in Section 2.3.3.

Data C : Real data of the mean daily temperature measured at Fisher River near Dallas with a one-day sampling rate over a period of approximately four years, Jan. 01, 1988 to Dec. 31, 1991. The data can be downloaded from the website of Data Market (http://datamarket.com/). We show the advantages over the state-of-the-art methods in Section 2.3.4.

The dictionary of the proposed algorithm is constructed with the coherence criterion (see Example 2.1). We focus on two extreme cases: the non-selective update (Φ-PASS-Full) and the selective update (Φ-PASS) with the strategy described in Example 2.2 for $\left|\tilde{\mathcal{J}}_{n,\iota}\right| = 1$.

2.3.1 Parameter design

The parameters s and p control the tradeoff between the computational complexity and the performance of the algorithm, and users can design those parameters for each application. The larger the parameters δ, the larger the maximal dictionary size. Although the use of the large dictionary tends to

yield fast convergence and low MSEs, this may cause also an explosion of the computational complexity.

The stepsize λ_n is an important parameter that determines the performance of the algorithm. Although strict adjustments are necessary to obtain the best performance of the algorithm, as a rule of thumb, setting $\lambda_n \in [0.01\mu_n, 0.1\mu_n]$ gives a reasonable performance. The detailed settings of those parameters are presented in each of the experiments.

We employ a Gaussian kernel and uniform weights; i.e.,

$$\omega_{n,\iota} = (\min\{p, n+1\})^{-1}, \ \iota \in \mathcal{I}_n. \tag{2.24}$$

We set the error bound to $\rho := 0$ for showing that the proposed algorithm works well without its delicate tuning.

2.3.2 Experiment A — Stationary Data

Experiment A1: Comparisons with NORMA and KLMS

We predict each datum d_n with the vector $\boldsymbol{u}_n := [\hat{d}_{n-1}, \hat{d}_{n-2}, \cdots, \hat{d}_{n-L}]^{\mathsf{T}} \in \mathcal{U} \subset \mathbb{R}^L$ of past data. Here, $\hat{d}_n := d_n + v_n$, $n \in \mathbb{N}$, where $v_n \sim \mathcal{N}(0, 1.0\times10^{-2})$ is the additive white Gaussian noise. The nonlinear filter φ_n is updated with $(\boldsymbol{u}_n, \hat{d}_n)$ at each time $n \in \mathbb{N}$. We mention that the datum $d_n := \psi_{\mathrm{A}}(d_{n-1}, d_{n-2})$ depends indirectly on all the past data since the data d_{n-1} and d_{n-2} are functions of $(d_{n-2},\ d_{n-3})$ and $(d_{n-3},\ d_{n-4})$, respectively, and so on. In this experiment, we let $L = 4$ as it yields reasonable performance. We test 100 independent runs with the additive noise generated randomly, and the mean squared error (MSE) is computed by averaging the instantaneous squared errors $(d_n - \varphi_n(\boldsymbol{u}_n))^2$ over the 100 runs. The parameter of the Gaussian kernel is set to $\xi = \sqrt{8.0}$. The proposed algorithm is compared with NORMA [21] and KLMS. To demonstrate the efficiency coming from the idea of projection-along-subspace, we test KLMS without any admission control for the dictionary. Table 2.3 summarizes the set of parameters employed in this experiment and the average computational complexities (the number of real multiplications). For NORMA, $\epsilon > 0$, $\lambda \in (0, 1/\epsilon)$, and $\tau \in \mathbb{N}^*$ are the regularization parameter, the step size, and the upper bound of the dictionary size, respectively. For KLMS, $\lambda > 0$ is the step size. The dictionary size of NORMA is chosen so that it is nearly identical to that of Φ-PASS at the end of the adaptation. We mention that Φ-PASS ($p = 1$) with $|\tilde{\mathcal{J}}_{n,\iota}| = 1$ coincides with QKLMS [29]. We remark that the stepsize significantly affect to the KAF algorithms and therefore should be carefully tuned. The step size for each algorithm is chosen so that the steady-state MSEs become close to each other. The average complexities are calculated with the dictionary size averaged over both iterations and runs.

Figures 2.7 and 2.8 depict the MSE learning curves and the evolutions of the dictionary size, respectively. We remark that the learning curve of

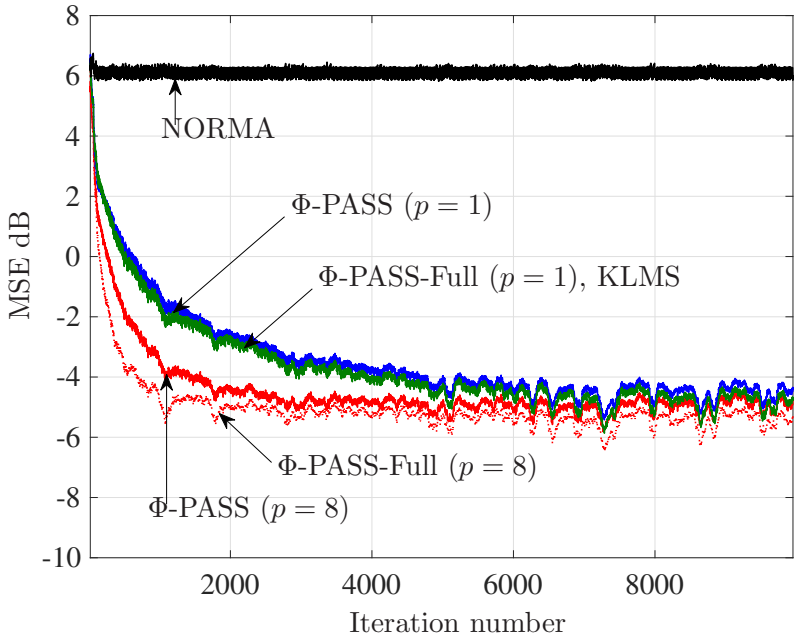

Figure 2.7: Results of Experiment A1: MSEs of the Φ-PASS algorithms, KLMS, and NORMA.

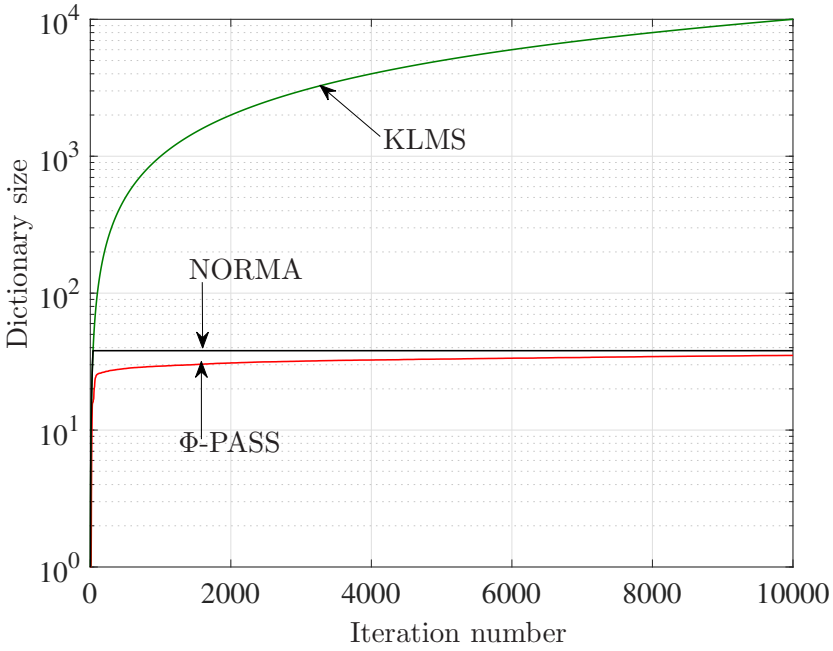

Figure 2.8: Results of Experiment 1: dictionary sizes of the Φ-PASS algorithms, KLMS, and NORMA.

Table 2.3: Parameter settings and complexities for Experiment A1.

	parameter		complexity	dictionary size (mean)
Φ-PASS ($p = 1$)	$\lambda_n = 0.4$	$\delta = 0.975$	199	32.4
Φ-PASS-Full ($p = 1$)	$\lambda_n = 0.6$		2195	
Φ-PASS ($p = 8$)	$\lambda_n = 0.03\mu_n$		780	
Φ-PASS-Full ($p = 8$)	$\lambda_n = 0.1\mu_n$		20523	
NORMA	$\lambda = 0.3,\ \epsilon = 10^{-5},\ \tau = 38$		228	38.0
KLMS	$\lambda = 0.1$		25003	5000.5

the dictionary sizes are continuous since the curves are average over the multiple runs. The figure together with Table 2.3 gives the following three observations. Observation 1: Φ-PASS ($p = 8$) attains a significantly better performance than NORMA, KLMS, and Φ-PASS ($p = 1$) with fairly low complexity. Observation 2: Φ-PASS ($p = 8$) attains a substantial reduction of complexity compared to Φ-PASS-Full ($p = 8$) while the performance degradation is marginal. This shows the efficacy of the use of past data together with the selective update. Observation 3: Φ-PASS ($p = 1$) achieves approximately the same performance as KLMS and yields a drastic reduction in complexity due to the use of the small size of dictionary. This efficiency comes from the idea of projection-along-subspace. We finally remark that the observed poor performance of NORMA is due to the small dictionary size which implies that the dictionary is reconstructed with unacceptably-high frequency.

Experiment A2: Comparisons with KAP

We use the same stationary data (Data A) under the same problem settings as in Section 2.3.2 with different noise levels:

Case 1 : Large variance $v_n \sim \mathcal{N}(0, 1.0 \times 10^{-1})$.

Case 2 : Small variance $v_n \sim \mathcal{N}(0, 1.0 \times 10^{-4})$.

To show the noise robustness due to the use of the parallel projection rather than the affine projection, the proposed algorithm is compared with KAP [27]. Table 2.4 summarizes the parameters and the average computational complexities. For KAP, $\lambda > 0$ and $\epsilon > 0$ are the step size and the regulariza-

Table 2.4: Parameter settings and complexities for Experiment A2.

		parameter		complexity	dictionary size (mean)
Case 1	Φ-PASS ($p = 8$)	$\lambda_n = 0.01\mu_n$	$\delta = 0.975$	2140	93.9
	KAP	$\lambda = 0.05$ $p = 8,\ \epsilon = 10^{-5}$		8632	
Case 2	Φ-PASS ($p = 8$)	$\lambda_n = 0.03\mu_n$		587	23.4
	KAP	$\lambda = 0.06$ $p = 8,\ \epsilon = 10^{-5}$		2636	

tion parameter, respectively. The step size and the regularization parameter are chosen so that each algorithm achieves the best performance.

Figures 2.9 and 2.10 depict the results. It can be seen that the proposed algorithm significantly outperforms KAP in the noisy situation (Case 1) although the difference in the quiet situation (Case 2) is minor. This shows the noise robustness of the proposed algorithm. We mention that the large noise variance increases the variance of the measurements $\hat{d}_n$, thus enlarging the dictionary size under the same coherence threshold.

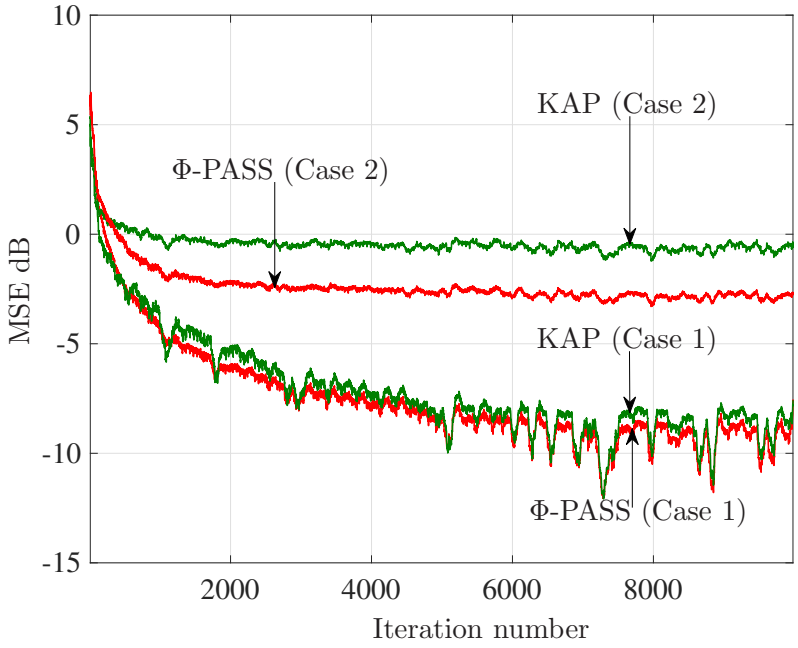

Figure 2.9: Results of Experiment A2: MSEs of Φ-PASS and KAP. Case 1: large noise variance. Case 2: small noise variance.

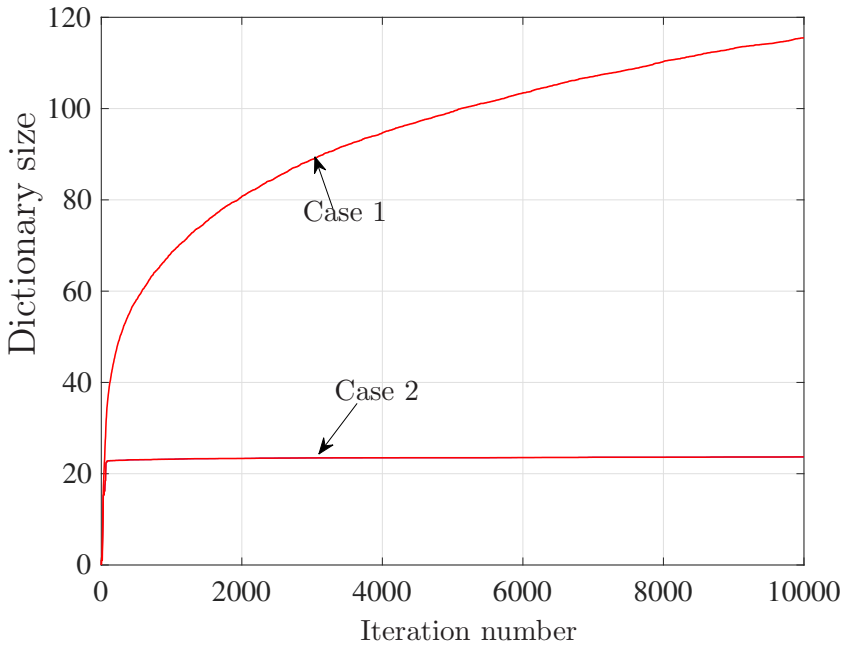

Figure 2.10: Results of Experiment A2: dictionary sizes of Φ-PASS and KAP. Case 1: large noise variance. Case 2: small noise variance.

Table 2.5: Parameter settings and complexities for Experiment B.

	parameter		complexity	dictionary size (mean)
Φ-PASS ($p = 8$)	$\lambda_n = 0.05\mu_n$	$\delta = 0.97$	610	24.3
Φ-PASS-Full ($p = 8$)	$\lambda_n = 0.05\mu_n$		3761	
QKLMS	$\lambda = 0.4$		145	
KRLS	$\zeta = 1.5 \times 10^{-3}$		3008	26.7

2.3.3 Experiment B — Nonstationary Data

The simulation settings are basically the same as in Section 2.3.2. We predict each d_n for the nonstationary data (Data B) from noisy measurements $\hat{d}_n$ with $v_n \sim \mathcal{N}(0, 1.0 \times 10^{-2})$ for $L = 4$ and the Gaussian scale $\xi = \sqrt{8.0}$. The proposed algorithm is compared with the existing algorithms: QKLMS [29] and KRLS [22]. (KRLS is built upon the stationarity assumption but we show its performance for reference.) We adopt the coherence criterion for QKLMS.[3] Table 2.5 summarizes the parameters and the average complexities. For QKLMS, $\lambda > 0$ is the step size. For KRLS, $\zeta > 0$ is the threshold of the approximate linear dependency criterion. The step size is chosen so that the steady-state MSEs of the proposed algorithm and QKLMS are nearly identical to each other in the first half of adaptation. The parameters δ and ζ are chosen so that the dictionary sizes of all algorithms are nearly identical in the second half.

Figures 2.11 and 2.12 depict the results. It can be seen that the proposed algorithm exhibits a better tracking performance than the existing ones. Also, the proposed algorithm with the selective update is quite efficient; its complexity is much lower than that of KRLS.

[3]Bounding the coherence $c(\boldsymbol{u}, \boldsymbol{v})$ from above is essentially equivalent to bounding the distance $\|\boldsymbol{u} - \boldsymbol{v}\|$ from below, where $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{U}$.

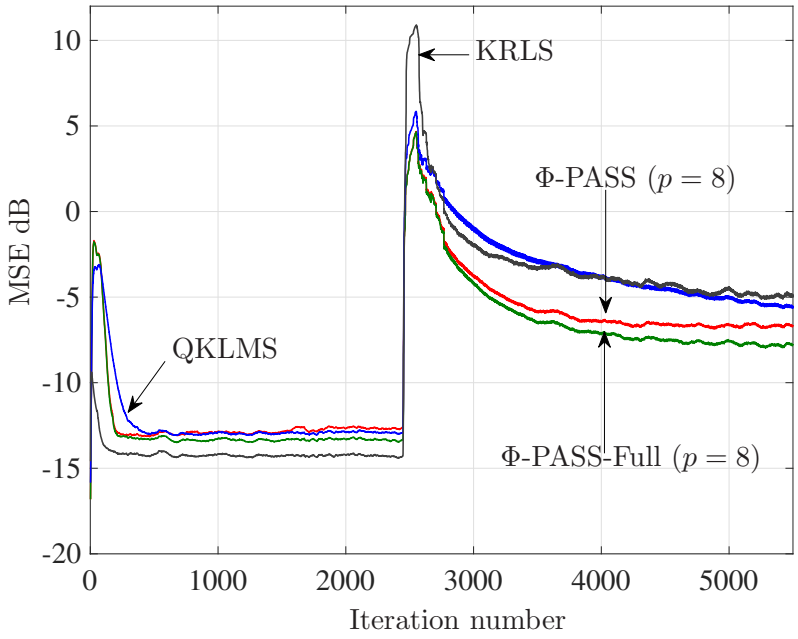

Figure 2.11: Results of Experiment B: MSEs of Φ-PASSs, KRLS, and QKLMS.

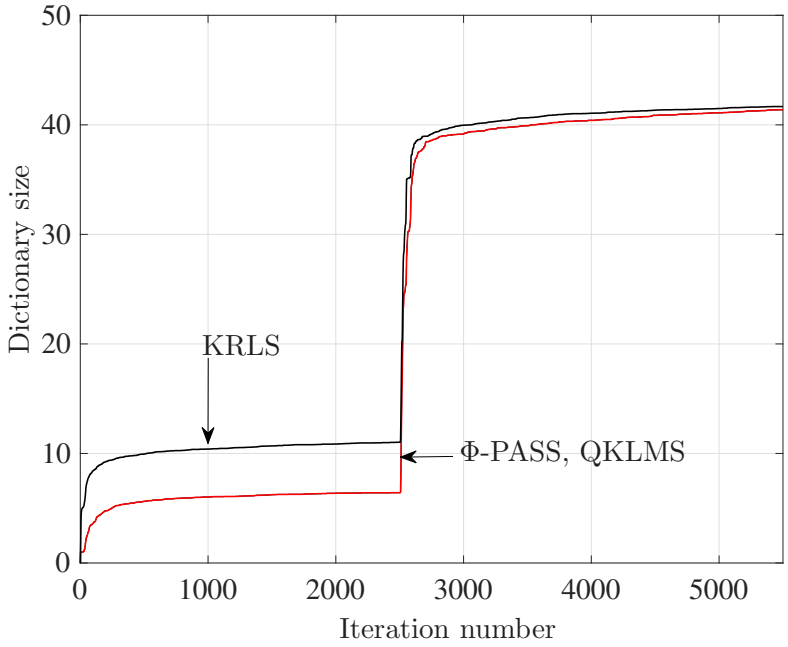

Figure 2.12: Results of Experiment B: dictionary sizes of Φ-PASSs, KRLS, and QKLMS.

Table 2.6: Parameter settings and complexities for Experiment C.

	parameter		complexity	squared error dB	dictionary size (mean)
Φ-PASS ($p = 40$)	$\lambda_n = 0.2\mu_n$	$\delta = 0.94$	3225	-20.61	33.3
QKLMS	$\lambda = 0.1$		200	-12.40	
KAP	$\lambda = 0.02$ $p = 40,\ \epsilon = 10^{-3}$		12333	-17.58	
KRLS Tracker	$M = 41$ $\xi = 1,\ \epsilon = 3 \times 10^{-4}$		4927	-19.77	38.77

2.3.4 Experiment C — Mean Daily Temperature

We finally use the real data, Data C plotted in Figure 2.13, to show the advantages of the proposed algorithm over the state-of-the-art algorithms: QKLMS [29], and KAP [27], KRLS Tracker [30]. As a pre-processing, the data are divided by its maximum value for normalization. The first 380 samples are used as training data for online learning, and the rest is used as test data to evaluate the generalization abilities; this follows the way in [33]. The temperature d_n is predicted with $\boldsymbol{u}_n := [d_{n-1}, d_{n-2}, \cdots, d_{n-L}]^{\mathsf{T}} \in \mathcal{U} \subset \mathbb{R}^L$ for $L = 4$ and the Gaussian scale $\xi = 1.0$. Table 2.6 summarizes the parameters, the computational complexities based on the dictionary sizes *averaged over the training phase*, and the MSEs *averaged over the testing phase*. For KRLS Tracker, $M \in \mathbb{N}^*$, $\xi \in (0, 1]$, and $\epsilon > 0$ are the budget, the forgetting factor, and the regularization parameter, respectively. The step size, the regularization parameter, and the forgetting factor are chosen so that each algorithm achieves the best performance. The budget M for KRLS Tracker is chosen so that its dictionary size *in the testing phase (not in the training phase)* is identical to those of the other algorithms to make the MSE comparisons in the testing phase more meaningful.

Figures 2.14 and 2.15 depict the results. It is seen that the proposed algorithm considerably outperforms KAP and QKLMS. We remark here that the approximately 3 dB gain from KAP comes with another benefit of lower complexity. The proposed algorithm exhibits better generalization ability and lower complexity than KRLS Tracker, although its convergence speed in the training phase is slightly slower.

2.3.5 Advantages Shown Through the Experiments

To sum up the simulation results, the major advantages of the proposed algorithm are summarized as follows.

1. The proposed algorithm achieved excellent performances with small

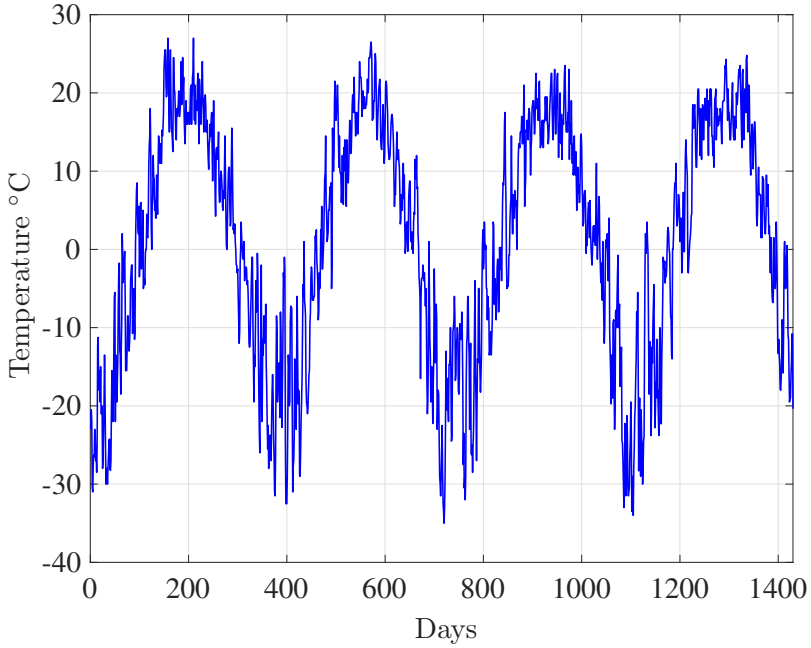

Figure 2.13: Time-series data used in Experiment C.

dictionary sizes in all the experiments. In particular, despite the small dictionary size, it attains, by the use of parallel projection, even better performances than KLMS. This comes from the key idea of projection-along-subspace presented in Section 2.2.1.

2. The proposed algorithm achieved reasonably good performances with fairly small complexities in all the experiments. This comes from the idea of selective update presented in Section 2.2.1.

3. The proposed algorithm achieved fast convergence and tracking, even for nonstationary data, by the use of parallel projection presented in Section 2.2.1. The use of parallel projection also brought the robustness to noise.

4. For the real data, the proposed algorithm significantly outperformed QKLMS and KAP, while its generalization ability is slightly better than KRLS Tracker and it saves approximately 35 % of the complexity.

2.4 Conclusion

We proposed the Φ-PASS algorithm for adaptive nonlinear estimation tasks. The key ideas were projection-along-subspace, selective update, and parallel projection. The projection-along-subspace systematically eliminates the

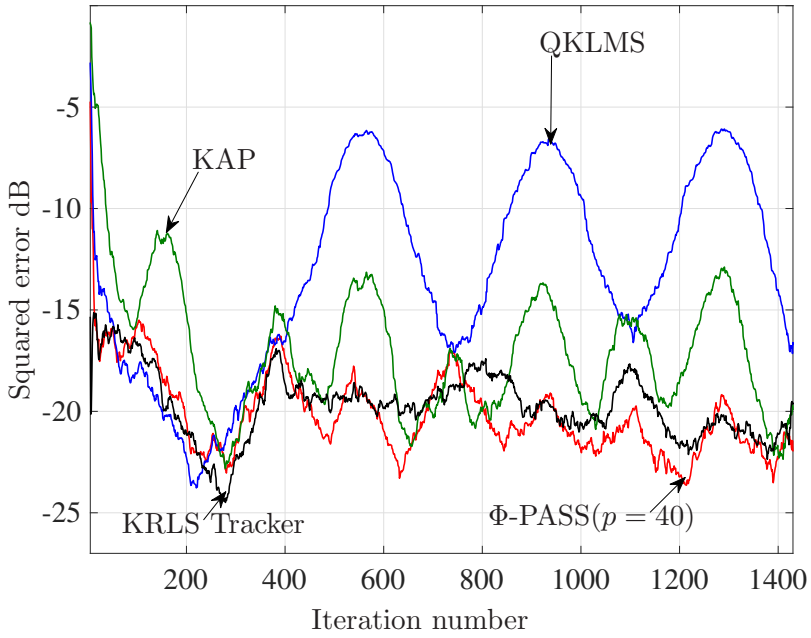

Figure 2.14: Results of Experiment C: MSEs of Φ-PASS, KRLS Tracker, KAP, QKLMS, RAN, and Kalman filter.

waste-of-resources issue, yielding excellent performances with small dictionary sizes. The selective update effectively reduces the complexity without any serious degradation of performance, as justified by a geometric interpretation. The parallel projection leads to fast convergence/tracking accompanied by noise robustness. The convergence analysis in the particular case of non-selective update was presented by using the framework of APSM. Numerical examples demonstrated the benefits from the three ideas as well as the advantages of the proposed algorithm over the state-of-the-art algorithms. Interestingly, the proposed algorithm bridges a pair of important algorithms: QKLMS and the sparse sequential algorithm.

We remark that some related researches have been studied in [46, 79]. In [46], a multikernel extension of Φ-PASS (with single projection) is proposed. The method in [46] performs projections in the Cartesian-product of the multiple RKHSs.

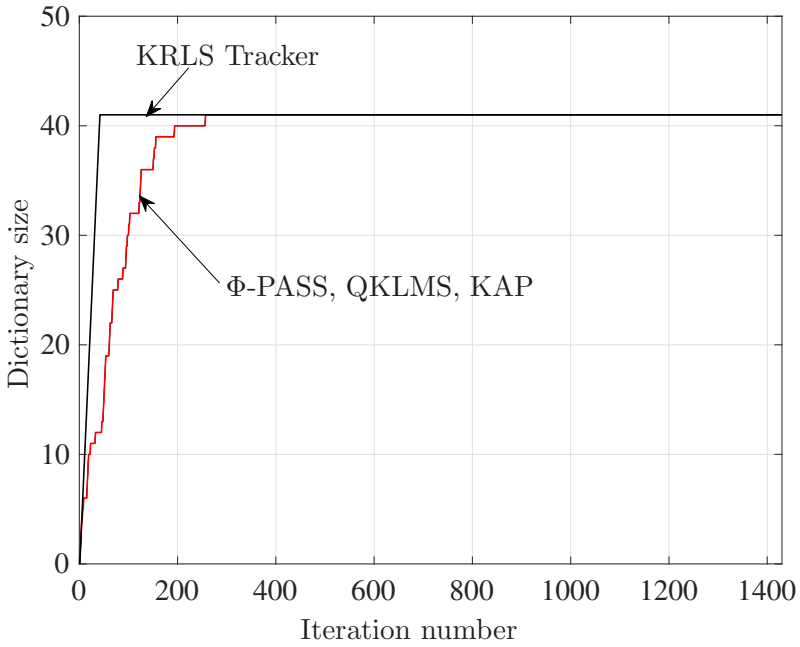

Figure 2.15: Results of Experiment C: dictionary sizes of Φ-PASS, KRLS Tracker, KAP, QKLMS, RAN, and Kalman filter.

Chapter 3

An Efficient KAF Algorithm with Dictionary Refinements

3.1 Introduction

First, we provide a basis to clarify the relationship between the two classes of KAF algorithms: the functional and parameter-space approaches. We show that the dictionary subspace and a parameter space are isomorphic under the inner product defined with the Gram matrix $\boldsymbol{G}$. This means that the learning in the dictionary subspace can be regarded as the learning in the parameter space with the particular $\boldsymbol{G}$-inner product. Based on the isomorphism between the dictionary subspace and the parameter space, we define the *restricted gradient*, which is the gradient of the cost functional under the restriction to the dictionary subspace. The restricted gradient, together with the isomorphism, provides a way to view the behaviors of the two approaches in a common space, either in the dictionary subspace or in the parameter space. With the restricted gradient, we derive a stochastic restricted-gradient descent method in the dictionary subspace, named the *Constrained KLMS (CKLMS) algorithm* [80]. To clarify the relationship of two approaches, we compare CKLMS and KNLMS from the error surface view.

Second, we derive a promising functional-space algorithm that suppresses the weighted squared-distance functions penalized by the ℓ_1 norm. A straightforward idea of the proposed algorithm is to apply APFBS algorithm to the cost function (which is the sum of smooth and nonsmooth functions) under the $\boldsymbol{G}$-inner product, unfortunately, the proximity operator defined with the $\boldsymbol{G}$-inner product has no closed form expression. Moreover, the computation of the proximity operator defined with the $\boldsymbol{G}$-inner product includes the inverse of the Gram matrix $\boldsymbol{G}$. We therefore propose a heuristic, but efficient, algorithm that employs the proximity operator defined with the standard inner product which has the closed form expression. Although the

proposed algorithm uses different inner products between the forward and backward steps of APFBS, we show that it still enjoys a monotone approximation property regarding a cost function with a certain modified weighted ℓ_1 norm under some conditions.

3.2 Fundamental Results on Kernel Adaptive Filtering

3.2.1 Preliminaries

To facilitate understanding of the underlying relation between two classes of kernel adaptive filtering algorithms, we assume in this section is generated with a fixed set of functions $\{\kappa(\cdot, \boldsymbol{x}^{(j)})\}_{j=1}^{r}$ for some given $\boldsymbol{x}^{(j)} \in \mathcal{U}$. Assume that the dictionary is linearly independent so that the Gram matrix

$$\boldsymbol{G} := \begin{bmatrix} \kappa\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}\right) & \cdots & \kappa\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(r)}\right) \\ \vdots & \ddots & \vdots \\ \kappa\left(\boldsymbol{x}^{(r)}, \boldsymbol{x}^{(1)}\right) & \cdots & \kappa\left(\boldsymbol{x}^{(r)}, \boldsymbol{x}^{(r)}\right) \end{bmatrix} \in \mathbb{R}^{r \times r} \tag{3.1}$$

is positive definite [69]. We can thus define the $\boldsymbol{G}$-*inner product* (for more details, see Section 1.5.6). The dictionary elements span the dictionary subspace

$$\mathcal{M} := \operatorname{span}\{\kappa(\cdot, \boldsymbol{x}^{(j)})\}_{j=1}^{r} \subset \mathcal{H}. \tag{3.2}$$

The output of a filter $\varphi := \sum_{j=1}^{r} h^{(j)} \kappa(\cdot, \boldsymbol{x}^{(j)}),\ h^{(j)} \in \mathbb{R}$, to the input $\boldsymbol{u}_n$ is given by

$$\varphi(\boldsymbol{u}_n) = \langle \varphi, \kappa(\cdot, \boldsymbol{u}_n) \rangle_{\mathcal{H}} = \sum_{j=1}^{r} h^{(j)} \kappa(\boldsymbol{u}_n, \boldsymbol{x}^{(j)}) = \langle \boldsymbol{h}, \boldsymbol{\kappa}_n \rangle, \tag{3.3}$$

where $\boldsymbol{\kappa}_n := [\kappa(\boldsymbol{u}_n, \boldsymbol{x}^{(1)}), \kappa(\boldsymbol{u}_n, \boldsymbol{x}^{(2)}), \cdots, \kappa(\boldsymbol{u}_n, \boldsymbol{x}^{(r)})]^{\mathsf{T}}$ is the kernelized input vector and $\boldsymbol{h} := [h^{(1)}, h^{(2)}, \cdots, h^{(r)}]^{\mathsf{T}}$ is the coefficient vector. Due to the two inner-product expressions in (3.3), the output $\varphi(\boldsymbol{u}_n)$ can be regarded as a linear functional $\langle \varphi, \kappa(\cdot, \boldsymbol{u}_n) \rangle_{\mathcal{H}}$ of φ, and also as a linear function $\langle \boldsymbol{h}, \boldsymbol{\kappa}_n \rangle$ of $\boldsymbol{h}$. The MSE $E(d_n - \varphi(\boldsymbol{u}_n))^2$ can thus be regarded as a functional of φ, and also as a function of $\boldsymbol{h}$:

$$J_{\varphi}(\varphi) := E(\langle \varphi, \kappa(\cdot, \boldsymbol{u}_n) \rangle_{\mathcal{H}}^2) - 2E(d_n \langle \varphi, \kappa(\cdot, \boldsymbol{u}_n) \rangle_{\mathcal{H}}) + E(d_n^2), \tag{3.4}$$

$$J_{\boldsymbol{h}}(\boldsymbol{h}) := \boldsymbol{h}^{\mathsf{T}} \boldsymbol{R} \boldsymbol{h} - 2\boldsymbol{p}^{\mathsf{T}} \boldsymbol{h} + E(d_n^2), \tag{3.5}$$

where $\boldsymbol{R} := E(\boldsymbol{\kappa}_n \boldsymbol{\kappa}_n^{\mathsf{T}})$ is the autocorrelation matrix of the kernelized input vector $\boldsymbol{\kappa}_n$ and $\boldsymbol{p} := E(d_n \boldsymbol{\kappa}_n)$ is the cross-correlation vector between d_n and $\boldsymbol{\kappa}_n$. By abuse of notation, we simply denote $J_{\boldsymbol{h}}(\boldsymbol{h})$ and $J_{\varphi}(\varphi)$ by $J(\boldsymbol{h})$ and $J(\varphi)$, respectively. As shown in the following section, the Euclidean-

space and functional-space approaches seek to minimize $J(\boldsymbol{h})$ and $J(\varphi)$, respectively, by, e.g., a stochastic gradient method.

3.2.2 Restricted Gradient and Isomorphism

We define (i) the gradient of the functional J under the restriction to the dictionary subspace $\mathcal{M}$ and (ii) the metric projection onto a closed convex subset of a real Hilbert space.[1]

Definition 3.1 (Restricted gradient) *Given any $\varphi := \sum_{i=1}^{r} h^{(i)}\kappa(\cdot, \boldsymbol{x}^{(i)})$, the gradient $\nabla_{|\mathcal{M}} J(\varphi)$ restricted to $\mathcal{M}$ is defined as a unique vector $g \in \mathcal{M}$ such that*

$$\langle g, \varphi_\epsilon \rangle_{\mathcal{H}} = \langle \nabla J(\boldsymbol{h}), \boldsymbol{\epsilon} \rangle \,, \ \forall \boldsymbol{\epsilon} := [\epsilon^{(1)}, \epsilon^{(2)}, \cdots, \epsilon^{(r)}]^{\mathsf{T}} \in \mathbb{R}^r, \tag{3.6}$$

where $\varphi_\epsilon := \sum_{j=1}^{r} \epsilon^{(j)}\kappa(\cdot, \boldsymbol{x}^{(j)}) \in \mathcal{M}$ and $\nabla J(\boldsymbol{h}) := 2(\boldsymbol{R}\boldsymbol{h} - \boldsymbol{p}) \in \mathbb{R}^r$ is the ordinary gradient (the Fréchet differential in $\mathbb{R}^r$) of $J(\boldsymbol{h})$ at $\boldsymbol{h} := [h^{(1)}, h^{(2)}, \cdots, h^{(r)}]^{\mathsf{T}}$.

The basic fact shown below plays a key role in the present study.

Lemma 3.1 *The pair of real Hilbert spaces $(\mathcal{M}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ and $(\mathbb{R}^r, \langle \cdot, \cdot \rangle_{\boldsymbol{G}})$ are isomorphic under the correspondence*

$$\mathcal{M} \ni \varphi := \sum_{j=1}^{r} h^{(j)}\kappa(\cdot, \boldsymbol{x}^{(j)}) \longleftrightarrow [h^{(1)}, h^{(2)}, \cdots, h^{(r)}]^{\mathsf{T}} =: \boldsymbol{h} \in \mathbb{R}^r. \tag{3.7}$$

Proof: The linear independence of the dictionary implies that the correspondence is clearly a bijective mapping. Given any vector $\hat{\boldsymbol{h}} := [\hat{h}^{(1)}, \cdots, \hat{h}^{(r)}]^{\mathsf{T}} \in \mathbb{R}^r$, the inner product between φ and $\hat{\varphi} := \sum_{j=1}^{r} \hat{h}^{(j)}\kappa(\cdot, \boldsymbol{x}^{(j)})$ is $\langle \varphi, \hat{\varphi} \rangle_{\mathcal{H}} = \sum_{i=1}^{r}\sum_{j=1}^{r} h^{(i)}\hat{h}^{(j)}\kappa(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \boldsymbol{h}^{\mathsf{T}}\boldsymbol{G}\hat{\boldsymbol{h}} = \left\langle \boldsymbol{h}, \hat{\boldsymbol{h}} \right\rangle_{\boldsymbol{G}}$, verifying that the mapping is inner product preserving. □

Lemma 3.1 states that learning in the functional subspace $\mathcal{M}$ can be expressed equivalently as learning in the Euclidean space $\mathbb{R}^r$ with the inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{G}}$. By the isomorphism in Lemma 3.1, the restricted gradient $\nabla_{|\mathcal{M}} J(\varphi)$ can be expressed as below (see Figure 3.1).

Proposition 3.1

1. *The restricted gradient $\nabla_{|\mathcal{M}} J(\varphi)$ is the Fréchet differential, defined in $\mathcal{M}$, of J at φ.*

[1] A set $\mathcal{K}$ is said to be convex if $\beta x + (1-\beta)y \in \mathcal{K}, \ \forall x, y \in \mathcal{K}, \ \forall \beta \in (0, 1)$. If a set is closed and convex, we say that it is closed convex.

2. *Through the correspondence in* (3.7)*, the restricted gradient* $\nabla_{|\mathcal{M}} J(\varphi)$ *can be expressed as*

$$\nabla_{|\mathcal{M}} J(\varphi) = P_{\mathcal{M}}(\nabla J(\varphi)) \longleftrightarrow \nabla_{\boldsymbol{G}} J(\boldsymbol{h}) := \boldsymbol{G}^{-1} \nabla J(\boldsymbol{h}), \tag{3.8}$$

where $\nabla J(\varphi)$ *is the Fréchet differential, defined in* $\mathcal{H}$*, of* J *at* φ*.*

Proof of Proposition 3.1.1. Due to the correspondence $\varphi_\epsilon \longleftrightarrow \boldsymbol{\epsilon}$, the isomorphism presented in Lemma 3.1 indicates that $\|\boldsymbol{\epsilon}\|_{\boldsymbol{G}}^2 = \|\varphi_\epsilon\|_{\mathcal{H}}^2$. Since $\lambda_{\min} \|\boldsymbol{\epsilon}\|^2 \leq \|\boldsymbol{\epsilon}\|_{\boldsymbol{G}}^2$, where $\lambda_{\min} > 0$ is the minimal eigenvalue of $\boldsymbol{G}$, one can verify that

$$\begin{aligned} 0 &\leq \frac{\left| J(\varphi + \varphi_\epsilon) - J(\varphi) - \left\langle \nabla_{|\mathcal{M}} J(\varphi), \varphi_\epsilon \right\rangle_{\mathcal{H}} \right|}{\|\varphi_\epsilon\|_{\mathcal{H}}} \\ &\leq \lambda_{\min}^{-1/2} \frac{|J(\boldsymbol{h} + \boldsymbol{\epsilon}) - J(\boldsymbol{h}) - \langle \nabla J(\boldsymbol{h}), \boldsymbol{\epsilon} \rangle|}{\|\boldsymbol{\epsilon}\|} \to 0 \text{ as } \|\boldsymbol{\epsilon}\| \to 0. \end{aligned} \tag{3.9}$$

Proof of Proposition 3.1.2. By the definition of the restricted gradient $\nabla_{|\mathcal{M}} J(\varphi)$, we have

$$\left\langle \nabla_{|\mathcal{M}} J(\varphi), \varphi_\epsilon \right\rangle_{\mathcal{H}} = \langle \nabla J(\boldsymbol{h}), \boldsymbol{\epsilon} \rangle = \left\langle \boldsymbol{G}^{-1} \nabla J(\boldsymbol{h}), \boldsymbol{\epsilon} \right\rangle_{\boldsymbol{G}}, \ \forall \boldsymbol{\epsilon} \in \mathbb{R}^r. \tag{3.10}$$

A simple inspection of (3.10) under Lemma 3.1 verifies $\nabla_{|\mathcal{M}} J(\varphi) \longleftrightarrow \nabla_{\boldsymbol{G}} J(\boldsymbol{h})$. On the other hand, it can be readily verified that $P_{\mathcal{M}}(\nabla J(\varphi)) \longleftrightarrow \boldsymbol{G}^{-1} \boldsymbol{b}_\varphi$, where $\boldsymbol{b}_\varphi := [\left\langle \nabla J(\varphi), \kappa(\cdot, \boldsymbol{x}^{(1)}) \right\rangle_{\mathcal{H}}, \cdots, \left\langle \nabla J(\varphi), \kappa(\cdot, \boldsymbol{x}^{(r)}) \right\rangle_{\mathcal{H}}]^{\mathsf{T}} \in \mathbb{R}^r$. The proof will hence be completed by showing that $\boldsymbol{b}_\varphi = \nabla J(\boldsymbol{h})$. Since

$$\begin{aligned} \langle \boldsymbol{b}_\varphi, \boldsymbol{\epsilon} \rangle_{\mathbb{R}^r} &= \sum_{j=1}^{r} \epsilon^{(j)} \left\langle \nabla J(\varphi), \kappa(\cdot, \boldsymbol{x}^{(j)}) \right\rangle_{\mathcal{H}} \\ &= \left\langle \nabla J(\varphi), \sum_{j=1}^{r} \epsilon^{(j)} \kappa(\cdot, \boldsymbol{x}^{(j)}) \right\rangle_{\mathcal{H}} = \langle \nabla J(\varphi), \varphi_\epsilon \rangle_{\mathcal{H}} \end{aligned} \tag{3.11}$$

and $\|\varphi_\epsilon\|_{\mathcal{H}}^2 = \|\boldsymbol{\epsilon}\|_{\boldsymbol{G}}^2 \leq \lambda_{\max} \|\boldsymbol{\epsilon}\|^2$, where $\lambda_{\max} > 0$ is the maximal eigenvalue of $\boldsymbol{G}$, it follows that

$$\begin{aligned} 0 &\leq \frac{|J(\boldsymbol{h} + \boldsymbol{\epsilon}) - J(\boldsymbol{h}) - \langle \boldsymbol{b}_\varphi, \boldsymbol{\epsilon} \rangle|}{\|\boldsymbol{\epsilon}\|} \\ &\leq \lambda_{\max}^{1/2} \frac{|J(\varphi + \varphi_\epsilon) - J(\varphi) - \langle \nabla J(\varphi), \varphi_\epsilon \rangle_{\mathcal{H}}|}{\|\varphi_\epsilon\|_{\mathcal{H}}} \to 0 \text{ as } \|\varphi_\epsilon\|_{\mathcal{H}} \to 0. \end{aligned} \tag{3.12}$$

This implies that $\boldsymbol{b}_\varphi$ is the Fréchet differential of J at $\boldsymbol{h}$ in $\mathbb{R}^r$. □

As the restricted gradient $\nabla_{|\mathcal{M}} J(\varphi)$ is the Fréchet differential in $\mathcal{M}$ (Proposition 3.1.1), it gives the steepest ascent direction, within the dictionary subspace $\mathcal{M}$, of the tangent plane of the functional $J(\varphi)$ at the

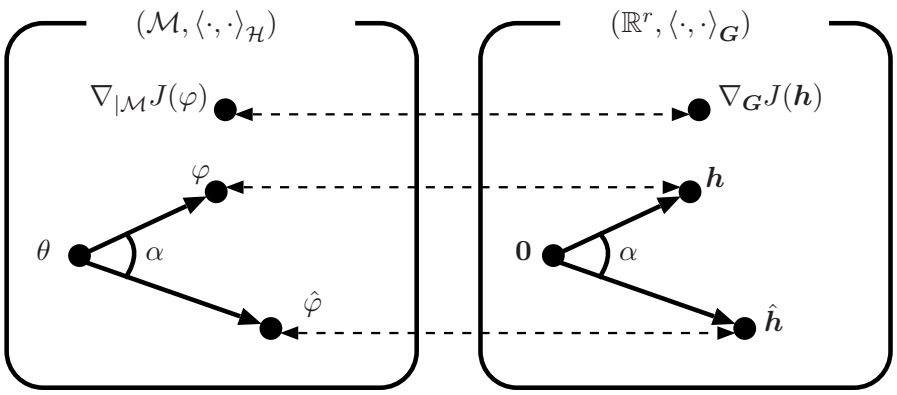

Figure 3.1: The isomorphism between $\mathcal{M}$ and $\mathbb{R}^r$. $\alpha \in [0, \pi]$ satisfies $\cos\alpha = \dfrac{\langle \varphi, \hat{\varphi}\rangle_{\mathcal{H}}}{\|\varphi\|_{\mathcal{H}} \|\hat{\varphi}\|_{\mathcal{H}}} = \dfrac{\left\langle \boldsymbol{h}, \hat{\boldsymbol{h}} \right\rangle_{\boldsymbol{G}}}{\|\boldsymbol{h}\|_{\boldsymbol{G}} \left\|\hat{\boldsymbol{h}}\right\|_{\boldsymbol{G}}}$.

point φ; indeed we have

$$\begin{aligned} \underset{\Delta\varphi\in\mathcal{M},\ \|\Delta\varphi\|_{\mathcal{H}}=1}{\operatorname{argmax}} \langle \nabla J(\varphi), \Delta\varphi\rangle_{\mathcal{H}} &= \underset{\Delta\varphi\in\mathcal{M},\ \|\Delta\varphi\|_{\mathcal{H}}=1}{\operatorname{argmax}} \langle P_{\mathcal{M}}(\nabla J(\varphi)), \Delta\varphi\rangle_{\mathcal{H}} \\ &= \frac{P_{\mathcal{M}}(\nabla J(\varphi))}{\|P_{\mathcal{M}}(\nabla J(\varphi))\|_{\mathcal{H}}} = \frac{\nabla_{|\mathcal{M}} J(\varphi)}{\left\|\nabla_{|\mathcal{M}} J(\varphi)\right\|_{\mathcal{H}}}. \end{aligned} \tag{3.13}$$

Here, the first equality is verified by the orthogonal decomposition $\nabla J(\varphi) = P_{\mathcal{M}}(\nabla J(\varphi)) + P_{\mathcal{M}^\perp}(\nabla J(\varphi))$ and by $\langle P_{\mathcal{M}^\perp}(\nabla J(\varphi)), \Delta\varphi\rangle_{\mathcal{H}} = 0$ for any $\Delta\varphi \in \mathcal{M}$, where $\mathcal{M}^\perp := \{\boldsymbol{\pi} \in \mathcal{H} : \langle \boldsymbol{m}, \boldsymbol{\pi}\rangle_{\mathcal{H}} = 0,\ \forall \boldsymbol{m} \in \mathcal{M}\}$ is the orthogonal complement of $\mathcal{M}$.

3.2.3 The CKLMS Algorithm

We present a very basic stochastic algorithm based on the restricted gradient. Replacing $\boldsymbol{R}$ and $\boldsymbol{p}$ by their instantaneous approximation $\boldsymbol{\kappa}_n \boldsymbol{\kappa}_n^{\mathsf{T}}$ and $d_n \boldsymbol{\kappa}_n$, respectively, an instantaneous approximation of the gradient $\nabla J(\boldsymbol{h})$ is defined as [81, 32, 31]

$$\hat{\nabla} J(\boldsymbol{h}) := -2\left(d_n - \langle \boldsymbol{h}, \boldsymbol{\kappa}_n\rangle\right)\boldsymbol{\kappa}_n,\ n \in \mathbb{N}. \tag{3.14}$$

Based on (3.8), an instantaneous approximation of the restricted gradient $\nabla_{\boldsymbol{G}} J(\boldsymbol{h})$, expressed in the Euclidean space $\mathbb{R}^r$, can be defined as

$$\hat{\nabla}_{\boldsymbol{G}} J(\boldsymbol{h}) := \boldsymbol{G}^{-1} \hat{\nabla} J(\boldsymbol{h}) = -2\left(d_n - \langle \boldsymbol{h}, \boldsymbol{\kappa}_n\rangle\right)\boldsymbol{G}^{-1}\boldsymbol{\kappa}_n,\ n \in \mathbb{N}. \tag{3.15}$$

By (3.15) and the correspondence [46, Lemma 1]

$$P_{\mathcal{M}}\left(\kappa(\cdot,\boldsymbol{u}_n)\right) \longleftrightarrow \boldsymbol{G}^{-1}\boldsymbol{\kappa}_n, \tag{3.16}$$

the instantaneous approximation $\hat{\nabla}_{\boldsymbol{G}} J(\boldsymbol{h})$ can be expressed in the functional space $\mathcal{M}$ as (see also (3.3))

$$\hat{\nabla}_{|\mathcal{M}} J(\varphi) = -2\left(d_n - \langle \varphi, \kappa(\cdot,\boldsymbol{u}_n)\rangle_{\mathcal{H}}\right) P_{\mathcal{M}}(\kappa(\cdot,\boldsymbol{u}_n)) \left(\longleftrightarrow \hat{\nabla}_{\boldsymbol{G}} J(\boldsymbol{h})\right). \tag{3.17}$$

Let $\varphi_0 := \theta \longleftrightarrow \boldsymbol{h}_0 := \boldsymbol{0}$ be the initial filter. The CKLMS is then given by

$$\varphi_{n+1} := \varphi_n - \frac{\lambda}{2}\hat{\nabla}_{|\mathcal{M}} J(\varphi_n) = \varphi_n + \lambda e_n P_{\mathcal{M}}(\kappa(\cdot,\boldsymbol{u}_n)),\ n \in \mathbb{N}, \tag{3.18}$$

where $\lambda > 0$ is the step size and $e_n := d_n - \varphi_n(\boldsymbol{u}_n)$. To express CKLMS in $\mathbb{R}^r$, we parameterize a kernel adaptive filter as follows:

$$\varphi_n(\cdot) = \sum_{j=1}^{r} h_n^{(j)} \kappa(\cdot, \boldsymbol{x}^{(j)}) \in \mathcal{H},\ n \in \mathbb{N}, \tag{3.19}$$

where $h_n^{(j)} \in \mathbb{R}$.

We can then verify the following proposition.

Proposition 3.2 *The CKLMS algorithm in* (3.18) *can be regarded as the LMS algorithm for the pair* $(\tilde{\boldsymbol{\kappa}}_n, d_n)_{n\in\mathbb{N}}$ *of the modified input vector* $\tilde{\boldsymbol{\kappa}}_n := \boldsymbol{G}^{-\frac{1}{2}}\boldsymbol{\kappa}_n$ *and the output.*[2] *In other words, it is a stochastic gradient descent method for the modified MSE cost function*

$$(J(\boldsymbol{h}) =)\ \tilde{J}(\tilde{\boldsymbol{h}}) = \tilde{\boldsymbol{h}}^{\mathsf{T}}\tilde{\boldsymbol{R}}\tilde{\boldsymbol{h}} - 2\tilde{\boldsymbol{p}}^{\mathsf{T}}\tilde{\boldsymbol{h}} + E(d_n^2) \tag{3.20}$$

of $\tilde{\boldsymbol{h}} := \boldsymbol{G}^{\frac{1}{2}}\boldsymbol{h}$, *where* $\tilde{\boldsymbol{R}} := E\left(\tilde{\boldsymbol{\kappa}}_n\tilde{\boldsymbol{\kappa}}_n^{\mathsf{T}}\right) = \boldsymbol{G}^{-\frac{1}{2}}\boldsymbol{R}\boldsymbol{G}^{-\frac{1}{2}}$ *and* $\tilde{\boldsymbol{p}} := E\left(d_n\tilde{\boldsymbol{\kappa}}_n\right) = \boldsymbol{G}^{-\frac{1}{2}}\boldsymbol{p}$.

Proof: Under the correspondence in Lemma 3.1, (3.18) is equivalent to

$$\boldsymbol{h}_{n+1} := \boldsymbol{h}_n - \frac{\lambda}{2}\hat{\nabla}_{\boldsymbol{G}} J(\boldsymbol{h}_n) = \boldsymbol{h}_n + \lambda e_n \boldsymbol{G}^{-1}\boldsymbol{\kappa}_n,\ n \in \mathbb{N}, \tag{3.21}$$

where $\boldsymbol{h}_n := [h_n^{(1)}, h_n^{(2)}, \cdots, h_n^{(r)}]^{\mathsf{T}} \in \mathbb{R}^r$. Left-multiplying both-sides of (3.21) by $\boldsymbol{G}^{\frac{1}{2}}$ yields

$$\tilde{\boldsymbol{h}}_{n+1} = \tilde{\boldsymbol{h}}_n + \lambda e_n \tilde{\boldsymbol{\kappa}}_n,\ n \in \mathbb{N}, \tag{3.22}$$

where $\tilde{\boldsymbol{h}}_n := \boldsymbol{G}^{\frac{1}{2}}\boldsymbol{h}_n$. Since $\varphi_n(\boldsymbol{u}_n) = \langle \boldsymbol{h}_n, \boldsymbol{\kappa}_n\rangle = \left\langle \tilde{\boldsymbol{h}}_n, \tilde{\boldsymbol{\kappa}}_n \right\rangle$ (see (3.3)), the

[2] For any positive semi-definite matrix $\boldsymbol{Q}$, there exists a unique square root $\boldsymbol{Q}^{\frac{1}{2}}$ satisfying $\boldsymbol{Q} = \boldsymbol{Q}^{\frac{1}{2}}\boldsymbol{Q}^{\frac{1}{2}}$.

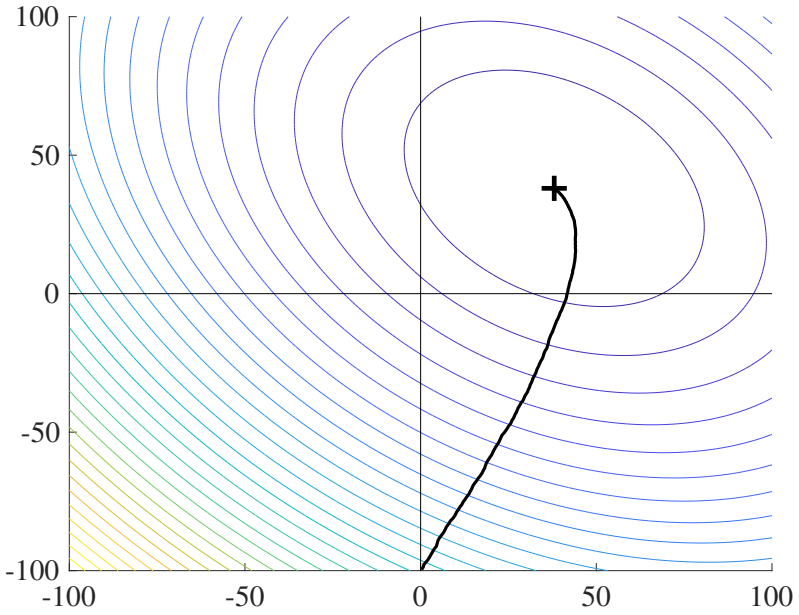

Figure 3.2: Equal error contour of $\tilde{J}(\tilde{\boldsymbol{h}})$ in Case (i).

instantaneous error e_n can be rewritten as

$$e_n = d_n - \left\langle \tilde{\boldsymbol{h}}_n, \tilde{\boldsymbol{\kappa}}_n \right\rangle. \tag{3.23}$$

From (3.22) and (3.23), it can be seen that (3.18) can be regarded as the LMS algorithm for $(\tilde{\boldsymbol{\kappa}}_n, d_n)_{n\in\mathbb{N}}$. Also, (3.22) can be rewritten as

$$\tilde{\boldsymbol{h}}_{n+1} = \tilde{\boldsymbol{h}}_n - \frac{\lambda}{2}\hat{\nabla}\tilde{J}(\tilde{\boldsymbol{h}}_n),\ n \in \mathbb{N}, \tag{3.24}$$

where

$$\hat{\nabla}\tilde{J}(\tilde{\boldsymbol{h}}_n) := -2e_n\tilde{\boldsymbol{\kappa}}_n. \tag{3.25}$$

□

Proposition 3.2 tells us that the error surface for the CKLMS algorithm is governed by the modified autocorrelation matrix $\tilde{\boldsymbol{R}} = \boldsymbol{G}^{-\frac{1}{2}}\boldsymbol{R}\boldsymbol{G}^{-\frac{1}{2}}$. On the other hand, the error surface of (the unnormalized version of) the KNLMS algorithm [27] is governed by $\boldsymbol{R}$, since it performs the LMS update for the ordinary MSE cost function $J(\boldsymbol{h})$. We illustrate these arguments with some particular examples in the following subsection.

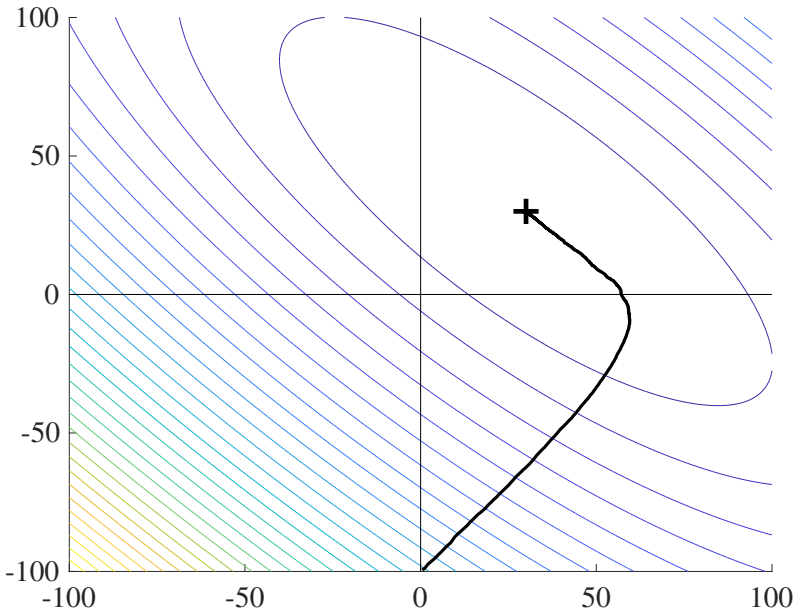

Figure 3.3: Equal error contour of $J(\boldsymbol{h})$ in Case (i).

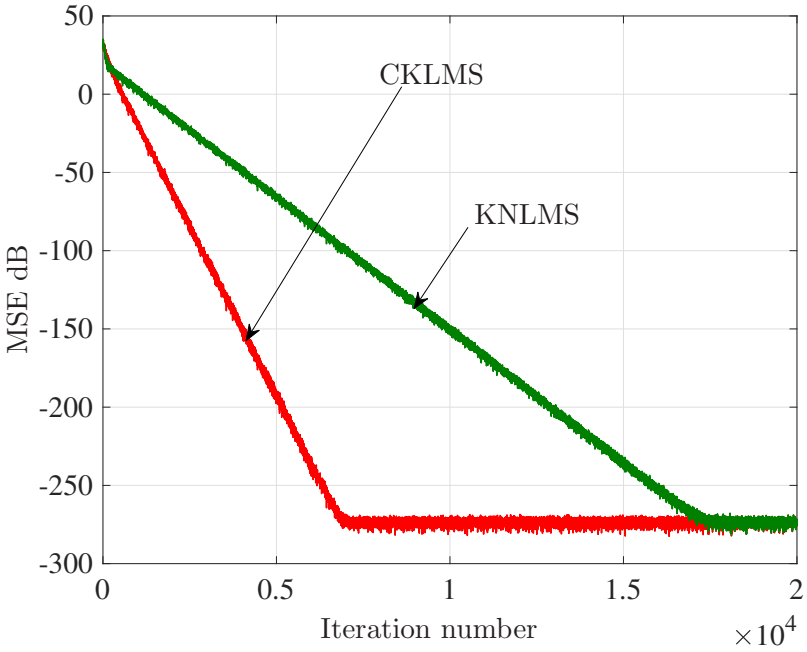

Figure 3.4: MSE learning curves for $r = 2$ in Case (i).

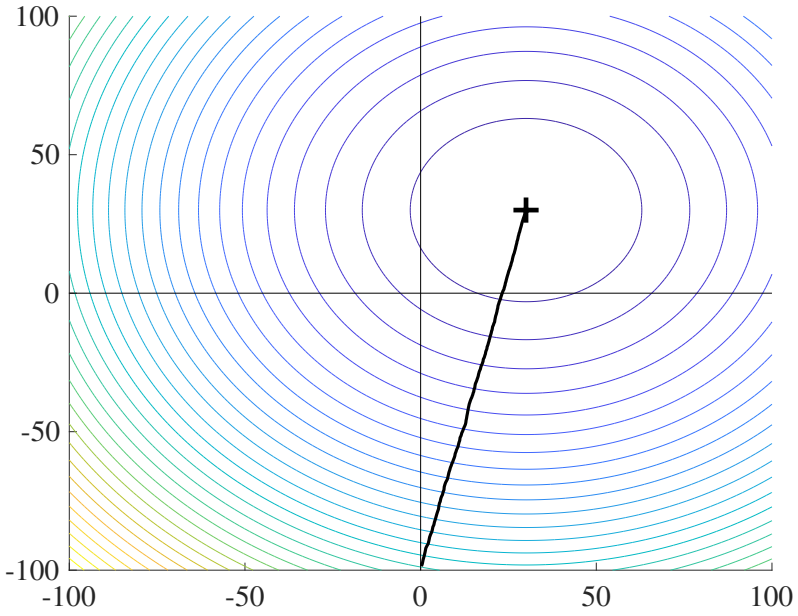

Figure 3.5: Equal error contour of $\tilde{J}(\tilde{\boldsymbol{h}})$ in Case (ii).

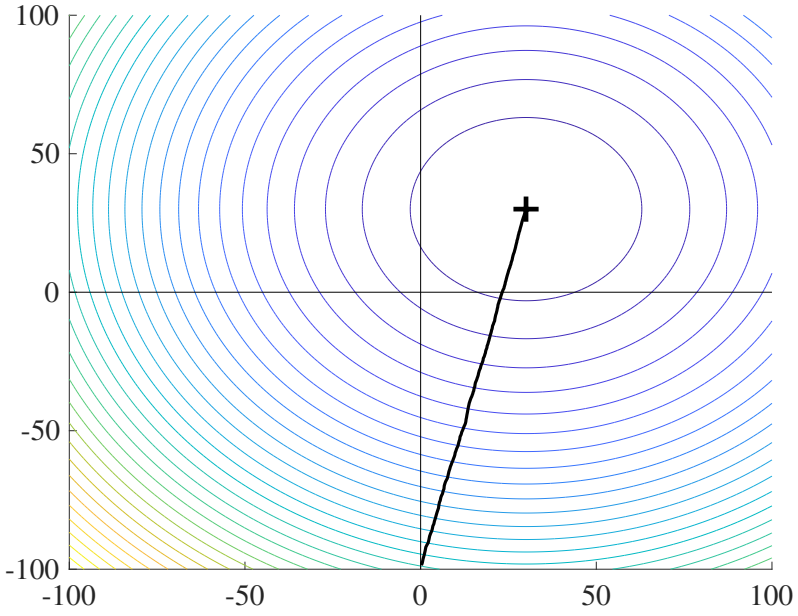

Figure 3.6: Equal error contour of $J(\boldsymbol{h})$ in Case (ii).

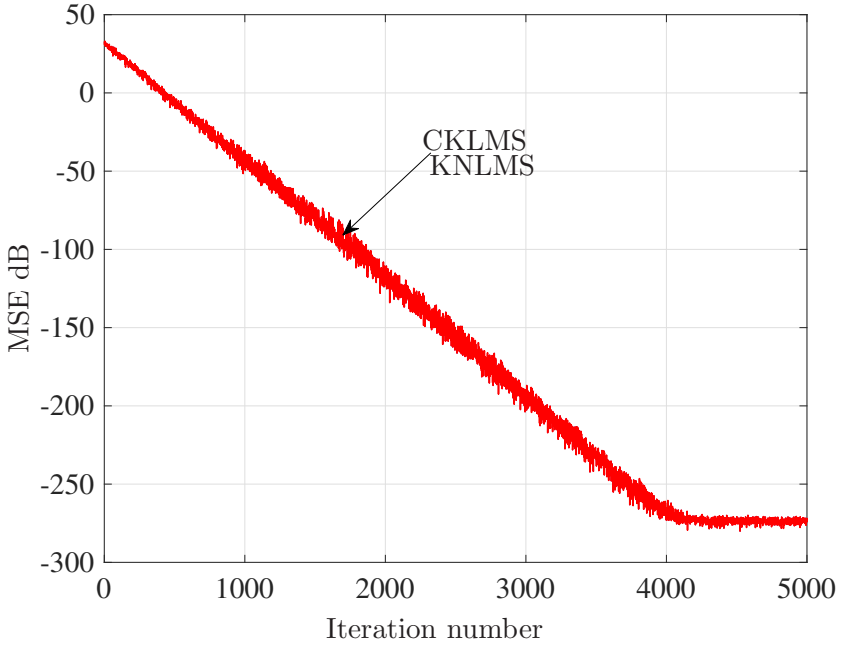

Figure 3.7: MSE learning curves for $r = 2$ in Case (ii).

3.2.4 Error Surface Analysis

We consider an estimation problem of the superposition of two Gaussian functions: $\psi(u) = 30\exp\left(-\frac{(u-x^{(1)})^2}{2\times 0.1^2}\right) + 30\exp\left(-\frac{(u-x^{(2)})^2}{2\times 0.1^2}\right)$, for $x^{(1)} := 0.4$ and $x^{(2)} := 0.5$. The input $u_n \in \mathbb{R},\ n \in \mathbb{N}$, is generated randomly from a uniform distribution within the range of $[-1, 1]$, and the output is generated by $d_n := \psi(u_n),\ n \in \mathbb{N}$. We let $\lambda = 0.1$ and $\sigma = 0.1$, and assume that the dictionary is $\{\kappa(\cdot, x^{(1)}), \kappa(\cdot, x^{(2)})\}$ whose linear span contains the ψ.

Figure 3.2 depicts the equal error contours of $\tilde{J}(\tilde{\boldsymbol{h}})$ with the trajectory of the vectors generated by CKLMS. Likewise, Figure 3.3 depicts those of $J(\boldsymbol{h})$ with the trajectories for KNLMS. The position of each point on the trajectories was computed by taking an average over 100 independent trials. Figure 3.4 plots the MSE curves, showing how much impact the difference in the contour shapes between $\tilde{J}(\tilde{\boldsymbol{h}})$ and $J(\boldsymbol{h})$ gives on the MSE performance. The MSEs are computed by averaging the instantaneous squared errors over the 100 trials. One can see that the equal error contours of $\tilde{J}(\tilde{\boldsymbol{h}})$ is better conditioned and CKLMS performs better while the convergence of KNLMS is deteriorated due to the 'squeezed' shape of contours as depicted in Figure 3.3. The eigenvalue spreads of $\boldsymbol{R}$ and $\tilde{\boldsymbol{R}}$ are $\mathrm{cond}_2(\boldsymbol{R}) := \|\boldsymbol{R}\|_2 \left\|\boldsymbol{R}^{-1}\right\|_2 = 8.13$ $\mathrm{cond}_2(\tilde{\boldsymbol{R}}) = 1.99$, respectively, where $\|\cdot\|_2$ denotes the spectral norm. We have empirically found that the convergence speed of KNLMS tends to be

Table 3.1: Eigenvalue spreads of $\boldsymbol{R}$ and $\tilde{\boldsymbol{R}}$.

	$\mathrm{cond}_2(\boldsymbol{R})$	$\mathrm{cond}_2(\tilde{\boldsymbol{R}})$
Case (i)	8.13	1.99
Case (ii)	1.07	1.07

slow (see Figure 3.4) particularly when the dictionary elements are closely located to each other. It is beyond the scope of the present study to verify this empirical finding.

Case (ii): In this case, $\boldsymbol{R} \approx \tilde{\boldsymbol{R}}$ because $\boldsymbol{G} \approx \boldsymbol{I}$, and hence the equal error contours in Figures 3.5 and 3.6 look similar and the convergence behaviors of the two algorithms are nearly identical (see Figure 3.7).

3.3 The DR-Φ-PASS Algorithm

3.3.1 Key Ingredients

The proposed algorithm shares the two key ingredients with the Φ-PASS algorithm proposed in Chapter 2: (i) *Parallel projection*: the p most recent data are exploited at each iteration for attaining fast convergence. (ii) *Selective update*: only a few coefficients associated with those dictionary elements which are coherent to $\kappa(\cdot, \boldsymbol{u}_n)$ are updated for complexity reduction. Moreover, as well as the ingredients (i) and (ii), the proposed algorithm has the following third key ingredient: (iii) *Dictionary refinement*: a dictionary is refined dynamically by discarding obsolete elements from the dictionary by means of iterative shrinkage. We discuss those key ingredients in detail below.

(i) **Parallel $\boldsymbol{G}_n$-projection.** We use the p most recent input-output pairs $(\boldsymbol{u}_n, d_n), (\boldsymbol{u}_{n-1}, d_{n-1}), \cdots, (\boldsymbol{u}_{n-p+1}, d_{n-p+1})$ to polish the coefficients; i.e, the filter φ_n is updated in such a way that the estimation errors $e_n, e_{n-1}, \cdots, e_{n-p+1}$ are suppressed. The set of indices that indicate the set of data exploited at time $n \in \mathbb{N}$ is denoted by $\mathcal{I}_n := \{n, n-1, \cdots, n-p+1\}$; for mathematical rigor, we let $\mathcal{I}_n := \{n, n-1, 0\}$ for $n < p-1$. The proposed algorithm is based on the projections, with the $\boldsymbol{G}_n$ inner product, onto multiple hyperplanes

$$H_{n,\iota} := \left\{ \boldsymbol{h} \in \mathbb{R}^{r_n} : \left\langle \boldsymbol{h}, \boldsymbol{G}_n^{-1} \boldsymbol{\kappa}_{n,\iota} \right\rangle_{\boldsymbol{G}_n} = d_\iota \right\}, \ \iota \in \mathcal{I}_n, \tag{3.26}$$

where $\boldsymbol{\kappa}_{n,\iota} := [\kappa(\boldsymbol{u}_\iota, \boldsymbol{u}_{j_1^{(n)}}) \ \kappa(\boldsymbol{u}_\iota, \boldsymbol{u}_{j_2^{(n)}}), \cdots, \kappa(\boldsymbol{u}_\iota, \boldsymbol{u}_{j_{r_n}^{(n)}})]^\mathsf{T}$. For more details, see Section 2.2.1

(ii) **Selective update.** We approximate the projection by selectively updating the coefficients te reduce the computational complexity of the algorithm. To update the selected coefficients with the other coefficients fixed,

the update-direction vector (in the functional space $\mathcal{H}$) should lie in the functional subspaces

$$\tilde{\mathcal{M}}_{n,\iota} := \mathrm{span}\{\kappa(\cdot, \boldsymbol{u}^{(j)})\}_{j\in\tilde{\mathcal{J}}_{n,\iota}} \subset \mathcal{M}_n \subset \mathcal{H},\ \iota \in \mathcal{I}_n. \tag{3.27}$$

Under the correspondence in Lemma 3.1, we define Euclidean affine subspaces $\mathcal{V}_{n,\iota} \subset \mathbb{R}^{r_n},\ \iota \in \mathcal{I}_n$, via the correspondence

$$\mathcal{M}_n \supset \tilde{\mathcal{M}}_{n,\iota} + \varphi_n := \{\boldsymbol{m} + \varphi_n \mid \boldsymbol{m} \in \tilde{\mathcal{M}}_{n,\iota}\} \longleftrightarrow \mathcal{V}_{n,\iota} \subset \mathbb{R}^{r_n}. \tag{3.28}$$

Note here that $\boldsymbol{h}_n \in \mathcal{V}_{n,\iota}$ since $\varphi_n \in \tilde{\mathcal{M}}_{n,\iota} + \varphi_n$. In the Euclidean space $\mathbb{R}^{r_n}$, the selective update can be accomplished by projecting $\boldsymbol{h}_n$ onto each hyperplane $H_{n,\iota}$ along the affine subspace $\mathcal{V}_{n,\iota}$. This is equivalent in fact to projecting $\boldsymbol{h}_n$ onto

$$H^{\mathcal{V}}_{n,\iota} := H_{n,\iota} \cap \mathcal{V}_{n,\iota},\ \iota \in \mathcal{I}_n. \tag{3.29}$$

(iii) **Dictionary refinement.** Our growing strategy of the dictionary is based on the coherence which has initially been proposed in [27]. It does not, however, take into account the statistics of inputs nor the characteristics of the nonlinear system ψ to be estimated. This implies that the dictionary constructed only with the coherence criterion may contain obsolete elements. To discard such elements adaptively, we penalize our cost function by the weighted ℓ_1 norm to sparsify the coefficient vector as in [28, 41, 35, 36, 42]. It has been shown in [41] that obsolete dictionary-elements would give negative impacts on the performance of adaptive algorithms. The use of the weighted ℓ_1-norm penalty is expected to prevent such negative impacts.

3.3.2 Cost Function and a Straightforward Idea

Based on the arguments presented in Section 3.3.1, we define a sequence of convex functions $(\Theta_n)_{n\in\mathbb{N}}$ as follows:

$$\Theta_n(\boldsymbol{h}) := \Phi_n(\boldsymbol{h}) + \tau\Omega_n(\boldsymbol{h}),\ \boldsymbol{h} \in \mathbb{R}^{r_n}, \tag{3.30}$$

where $\tau > 0$ is the regularization parameter. Here,

$$\Phi_n(\boldsymbol{h}) :=\ \frac{1}{2}\sum_{\iota\in\mathcal{I}_n} \omega_{n,\iota} d^2_{\boldsymbol{G}_n}(\boldsymbol{h}, H^{\mathcal{V}}_{n,\iota})\ \text{(smooth)} \tag{3.31}$$

is a weighted squared-distance function for some given weights $\omega_{n,\iota} > 0$ such that $\sum_{\iota\in\mathcal{I}_n} \omega_{n,\iota} = 1,\ \iota \in \mathcal{I}_n$, where

$$d_{\boldsymbol{G}_n}(\boldsymbol{h}, H^{\mathcal{V}}_{n,\iota}) := \min_{\boldsymbol{g}\in H^{\mathcal{V}}_{n,\iota}} \|\boldsymbol{h} - \boldsymbol{g}\|_{\boldsymbol{G}_n} \tag{3.32}$$

is the $\boldsymbol{G}_n$-metric distance from $\boldsymbol{h}$ to the closed convex sets $H_{n,\iota}^{\mathcal{V}}$. The second term

$$\Omega_n(\boldsymbol{h}) := \|\boldsymbol{w}_n \circ \boldsymbol{h}\|_1 \text{ (nonsmooth)} \tag{3.33}$$

is a weighted ℓ_1 norm for dictionary refinements, where $\boldsymbol{w}_n := [w_n^{(j_1^{(n)})}, w_n^{(j_2^{(n)})}, \cdots, w_n^{(j_{r_n}^{(n)})}]^{\mathsf{T}}$, $w_n^{(j)} > 0$, $\forall j \in \mathcal{J}_n$, is the weights, $\|\boldsymbol{x}\|_1 := \sum_{j=1}^{r_n} \left|x^{(j)}\right|$ is the ℓ_1 norm of $\boldsymbol{x} := [x^{(1)}, x^{(2)}, \cdots, x^{(r_n)}] \in \mathbb{R}^{r_n}$, and $\boldsymbol{w}_n \circ \boldsymbol{h}$ denotes the Hadamard (componentwise) product between $\boldsymbol{w}_n$ and $\boldsymbol{h}$.

The adaptive proximal forward-backward splitting (APFBS) algorithm [60] is an efficient method to minimize a sequence of convex functions each of which is the sum of smooth and nonsmooth convex functions (see (3.30), (3.31), and (3.33)). A straightforward idea would be to apply APFBS to the function sequence $(\Theta_n)_{n\in\mathbb{N}}$ with the inner product $\langle\cdot,\cdot\rangle_{\boldsymbol{G}_n}$. This approach, however, is impractical unfortunately, since the proximity operator of Ω_n in the Hilbert space $(\mathbb{R}^{r_n}, \langle\cdot,\cdot\rangle_{\boldsymbol{G}_n})$ has no closed-form expression. We therefore present a modified algorithm in the following subsection.

3.3.3 The Proposed Algorithm

Inspired by APFBS, the proposed algorithm includes the following two steps in addition to two more steps for dictionary updates.

Gradient (forward) step. The proposed algorithm exploits the $\boldsymbol{G}_n$-gradient [60]

$$\nabla_{\boldsymbol{G}_n} \Phi_n(\boldsymbol{h}) = \boldsymbol{h} - \sum_{\iota\in\mathcal{I}_n} \omega_{n,\iota} P_{H_{n,\iota}^{\mathcal{V}}}^{\boldsymbol{G}_n}(\boldsymbol{h}),\ \boldsymbol{h} \in \mathbb{R}^{r_n}. \tag{3.34}$$

The use of the $\boldsymbol{G}_n$ metric leads to fast convergence as illustrated in Section 3.2.4. It has indeed been reported in [67, 68] that the use of adequately-designed time-variable metric could yield substantial accelerations of convergence speed. We stress here that the $\boldsymbol{G}_n$ metric is naturally induced by the metric in the RKHS $\mathcal{H}$.

Proximal (backward) step. We adopt the canonical inner product $\langle\cdot,\cdot\rangle$ for the proximal step of APFBS rather than the $\boldsymbol{G}_n$ inner product $\langle\cdot,\cdot\rangle_{\boldsymbol{G}_n}$, i.e., the proposed algorithm exploits the proximity operator [82]

$$\begin{aligned}
\operatorname{prox}_{\lambda_n\tau\Omega_n}(\boldsymbol{h}) &:= \operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{R}^{r_n}} \left(\Omega_n(\boldsymbol{x}) + \frac{1}{2\lambda_n\tau}\|\boldsymbol{h}-\boldsymbol{x}\|^2\right) \\
&= \sum_{j=1}^{r_n} \max\left\{0, 1 - \frac{\lambda_n\tau w_n^{(j)}}{|h^{(j)}|}\right\} h^{(j)} \boldsymbol{e}_n^{(j)}, \\
\boldsymbol{h} &:= [h^{(1)}, h^{(2)}, \cdots, h^{(r_n)}]^{\mathsf{T}} \in \mathbb{R}^{r_n}.
\end{aligned} \tag{3.35}$$

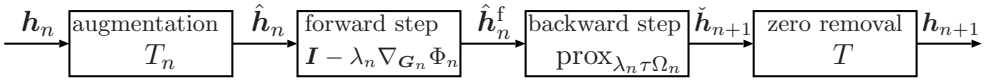

Figure 3.8: A block diagram of the proposed algorithm.

Here, $\boldsymbol{e}_n^{(j)}$ denotes the length-r_n unit vector with one at the jth position and zeros elsewhere.

We present below the DR-Φ-PASS algorithm, which is based on APFBS applied to $(\Theta_n)_{n\in\mathbb{N}}$ in (3.30).

Algorithm 3.1 (DR-Φ-PASS) *Let* $\hat{\boldsymbol{h}}_0 := 0 \in \mathbb{R}$ *and generate the sequence* $(\boldsymbol{h}_n)_{n\in\mathbb{N}}$ *by*

$$\begin{aligned} \hat{\boldsymbol{h}}_n &:= T_n(\boldsymbol{h}_n) \in \mathbb{R}^{r_n}, n \in \mathbb{N}, \\ \boldsymbol{h}_{n+1} &:= T\left\{\mathrm{prox}_{\lambda_n \tau \Omega_n}\left[\hat{\boldsymbol{h}}_n - \lambda_n \nabla_{\boldsymbol{G}_n} \Phi_n(\hat{\boldsymbol{h}}_n)\right]\right\}, \ n \in \mathbb{N}. \end{aligned} \tag{3.36}$$

Here, $\lambda_n \in [0,2]$ *is the step size.* T_n *is the operator that augments a vector by adding zero as a new entry at the bottom (i.e., adds* $\kappa(\cdot, \boldsymbol{u}_n)$ *into the dictionary) only when* $\kappa(\cdot, \boldsymbol{u}_n)$ *is sufficiently novel under some novelty criterion such as the coherence (see Example 3.1 below), and* T *is the operator that shrinks the length of a vector by removing zero components (i.e., discards obsolete elements from the dictionary).*

We define $\hat{\boldsymbol{h}}_n^{\mathrm{f}} := \hat{\boldsymbol{h}}_n - \lambda_n \nabla_{\boldsymbol{G}_n} \Phi_n(\hat{\boldsymbol{h}}_n)$ and $\check{\boldsymbol{h}}_{n+1} := \mathrm{prox}_{\lambda_n \tau \Omega_n}\left(\hat{\boldsymbol{h}}_n^{\mathrm{f}}\right)$ for $n \in \mathbb{N}$ and show a block diagram of the DR-Φ-PASS algorithm in Figure 3.8.

Example 3.1 (Novelty criterion based on coherence [27]) *Given a dictionary* $\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j\in\mathcal{J}}$, $\kappa(\cdot, \boldsymbol{u}_n)$ *is regarded novel if* $\max_{j\in\mathcal{J}} c(\boldsymbol{u}_n, \boldsymbol{u}_j) \le \delta$ *for some threshold* $\delta \in (0,1)$.

From the definition of $\mathcal{V}_{n,\iota}$, it is clear that the $\boldsymbol{G}_n$-projection $P_{H_{n,\iota}^{\mathcal{V}}}^{\boldsymbol{G}_n}(\hat{\boldsymbol{h}}_n)$ only changes the selected coefficients $h^{(j)}$, $j \in \tilde{\mathcal{J}}_{n,\iota}$, while keeping the unselected coefficients $h^{(j)}$, $j \in \tilde{\mathcal{J}}_{n,\iota}^{\mathrm{c}}$, unchanged. Therefore, the jth component of the projection can be written as

$$[P_{H_{n,\iota}^{\mathcal{V}}}^{\boldsymbol{G}_n}(\hat{\boldsymbol{h}}_n)]_j = \begin{cases} h_n^{(j)}, & j \in \tilde{\mathcal{J}}_{n,\iota}^{\mathrm{c}}, \\ p_{n,\iota}^{(j)}, & j \in \tilde{\mathcal{J}}_{n,\iota}, \end{cases} \tag{3.37}$$

where [69]

$$\begin{aligned} \boldsymbol{p}_{n,\iota} &:= [p_{n,\iota}^{(\tilde{j}_1^{(n,\iota)})}, p_{n,\iota}^{(\tilde{j}_2^{(n,\iota)})}, \cdots, p_{n,\iota}^{(\tilde{j}_{s_n}^{(n,\iota)})}]^{\mathsf{T}} := P_{\tilde{H}_{n,\iota}}^{\tilde{\boldsymbol{G}}_{n,\iota}}(\tilde{\boldsymbol{h}}_{n,\iota}) \\ &= \tilde{\boldsymbol{h}}_{n,\iota} + \frac{d_\iota - \boldsymbol{h}_n^{\mathsf{T}} \boldsymbol{\kappa}_{n,\iota}}{\tilde{\boldsymbol{\kappa}}_{n,\iota}^{\mathsf{T}} \tilde{\boldsymbol{G}}_{n,\iota}^{-1} \tilde{\boldsymbol{\kappa}}_{n,\iota}} \tilde{\boldsymbol{G}}_{n,\iota}^{-1} \tilde{\boldsymbol{\kappa}}_{n,\iota} \in \mathbb{R}^{s_{n,\iota}}. \end{aligned} \tag{3.38}$$

Table 3.2: Summary of the proposed algorithm.

The DR-Φ-PASS algorithm
Requirement : step size $\lambda_n \in [0, 2]$
Initialization : $\mathcal{J}_{-1}^{\neq 0} := \emptyset$, $\mathcal{J}_0 := \emptyset$
Filter output : $\varphi_n(\boldsymbol{u}_n) := \sum_{j\in\mathcal{J}_n} h_n^{(j)} \kappa(\boldsymbol{u}_n, \boldsymbol{u}_j)$
Filter update :
1. Add a new element into the dictionary based on some novelty criterion such as the coherence (see Example 3.1). If $\kappa(\cdot, \boldsymbol{u}_n)$ is novel, $\mathcal{J}_n := \mathcal{J}_{n-1}^{\neq 0} \cup \{n\}$. Otherwise, $\mathcal{J}_n := \mathcal{J}_{n-1}^{\neq 0}$.
2. If $n \in \mathcal{J}_n$, let $\hat{\boldsymbol{h}}_n := [\boldsymbol{h}_n^{\mathsf{T}}\ 0]^{\mathsf{T}}$. Otherwise let $\hat{\boldsymbol{h}}_n := \boldsymbol{h}_n$.
3. Construct $\tilde{\mathcal{J}}_{n,\iota} (\subset \mathcal{J}_n)$.
4. Compute $P_{H_{n,\iota}^{\mathcal{V}}}^{\boldsymbol{G}_n}(\hat{\boldsymbol{h}}_n)$ based on (3.37) and (3.38).
5. Compute $\hat{\boldsymbol{h}}_n^{\mathrm{f}} = \hat{\boldsymbol{h}}_n + \lambda_n \left(\sum_{\iota\in\mathcal{I}_n} \omega_\iota^{(n)} P_{H_{n,\iota}^{\mathcal{V}}}^{\boldsymbol{G}_n}(\hat{\boldsymbol{h}}_n) - \hat{\boldsymbol{h}}_n \right)$.
6. Compute $\check{\boldsymbol{h}}_{n+1} = \mathrm{prox}_{\lambda_n \tau \Omega_n}(\hat{\boldsymbol{h}}_n^{\mathrm{f}})$ based on (3.35).
7. Discard obsolete dictionary elements: $\boldsymbol{h}_{n+1} = T(\check{\boldsymbol{h}}_{n+1})$ and let $\mathcal{J}_n^{\neq 0} := \{j \in \mathcal{J}_n : \check{h}_{n+1}^{(j)} \neq 0\}$.

Here, $\tilde{\boldsymbol{h}}_{n,\iota} := [\tilde{h}^{(\tilde{j}_1^{(n,\iota)})}, \tilde{h}^{(\tilde{j}_2^{(n,\iota)})}, \cdots, \tilde{h}^{(\tilde{j}_{s_{n,\iota}}^{(n,\iota)})}]^{\mathsf{T}}$,
$\tilde{\boldsymbol{\kappa}}_{n,\iota} := [\kappa(\boldsymbol{u}_\iota, \boldsymbol{u}_{\tilde{j}_1^{(n,\iota)}})\ \kappa(\boldsymbol{u}_\iota, \boldsymbol{u}_{\tilde{j}_2^{(n,\iota)}}), \cdots, \kappa(\boldsymbol{u}_\iota, \boldsymbol{u}_{\tilde{j}_{s_{n,\iota}}^{(n,\iota)}})]^{\mathsf{T}}$,

$$\tilde{\boldsymbol{G}}_{n,\iota} := \begin{bmatrix} \kappa\left(\boldsymbol{u}_{\tilde{j}_1^{(n,\iota)}}, \boldsymbol{u}_{\tilde{j}_1^{(n,\iota)}}\right) & \cdots & \kappa\left(\boldsymbol{u}_{\tilde{j}_1^{(n,\iota)}}, \boldsymbol{u}_{\tilde{j}_{s_{n,\iota}}^{(n,\iota)}}\right) \\ \vdots & \ddots & \vdots \\ \kappa\left(\boldsymbol{u}_{\tilde{j}_{s_{n,\iota}}^{(n,\iota)}}, \boldsymbol{u}_{\tilde{j}_1^{(n,\iota)}}\right) & \cdots & \kappa\left(\boldsymbol{u}_{\tilde{j}_{s_{n,\iota}}^{(n,\iota)}}, \boldsymbol{u}_{\tilde{j}_{s_{n,\iota}}^{(n,\iota)}}\right) \end{bmatrix} \in \mathbb{R}^{s_{n,\iota}\times s_{n,\iota}}, \tag{3.39}$$

and

$$\tilde{H}_{n,\iota} := \left\{ \tilde{\boldsymbol{h}} \in \mathbb{R}^{s_{n,\iota}} : \left\langle \tilde{\boldsymbol{h}}, \tilde{\boldsymbol{G}}_{n,\iota}^{-1} \tilde{\boldsymbol{\kappa}}_{n,\iota} \right\rangle_{\tilde{\boldsymbol{G}}_{n,\iota}} = d_\iota \right\},\ \iota \in \mathcal{I}_n. \tag{3.40}$$

The DR-Φ-PASS algorithm is summarized in Table 3.2.

3.3.4 Discussions

Monotone approximation property

We present a monotone approximation property of DR-Φ-PASS below.

Theorem 3.1 (Monotone approximation) *Assume that*
(i) $\mathrm{sgn}(\boldsymbol{G}_n \boldsymbol{W}_n \mathrm{sgn}(\hat{\boldsymbol{h}}_n^{\mathrm{f}})) = \mathrm{sgn}(\hat{\boldsymbol{h}}_n^{\mathrm{f}})$, *where* $\mathrm{sgn}(\cdot)$ *denotes the signum function defined as* $\mathrm{sgn}(0) := 0$, $\mathrm{sgn}(x) := 1$, *if* $x > 0$, *and* $\mathrm{sgn}(x) := -1$, *if* $x < 0$, *and*
(ii) $\hat{\boldsymbol{h}}_n^{\mathrm{f}} \in \mathcal{D}_n := \{\boldsymbol{h} \in \mathbb{R}^{r_n} : |h^{(i)}| > \lambda_n \tau w_n^{(j_i^{(n)})}, i = 1, 2, \cdots, r_n\}$.
Define $\tilde{\Omega}_n(\boldsymbol{h}) = \|\tilde{\boldsymbol{w}}_n \circ \boldsymbol{h}\|_1$, $\boldsymbol{h} \in \mathbb{R}^{r_n}$, *for the modified weights* $\tilde{\boldsymbol{w}}_n := \boldsymbol{G}_n \boldsymbol{W}_n \mathrm{sgn}(\hat{\boldsymbol{h}}_n^{\mathrm{f}})$, *where* $\boldsymbol{W}_n := \mathrm{diag}(\boldsymbol{w}_n)$. *Let*

$$\tilde{\Theta}_n(\boldsymbol{h}) := \Phi_n(\boldsymbol{h}) + \tau \tilde{\Omega}_n(\boldsymbol{h}), \ \boldsymbol{h} \in \mathbb{R}^{r_n}. \tag{3.41}$$

Then, the monotone approximation property holds between $\hat{\boldsymbol{h}}_n \in \mathbb{R}^{r_n}$ *and* $\check{\boldsymbol{h}}_{n+1} \in \mathbb{R}^{r_n}$*:*

$$\left\|\check{\boldsymbol{h}}_{n+1} - \boldsymbol{h}^*\right\|_{\boldsymbol{G}_n} < \left\|\hat{\boldsymbol{h}}_n - \boldsymbol{h}^*\right\|_{\boldsymbol{G}_n} \tag{3.42}$$

for any $\boldsymbol{h}^* \in \Xi_\iota^{(n)} := \mathrm{argmin}_{\boldsymbol{h} \in \mathbb{R}^{r_n}} \tilde{\Theta}_n(\boldsymbol{h}) \neq \emptyset$, *if* $\boldsymbol{h}_n \notin \Xi_\iota^{(n)}$.

Proof: We first define the subdifferential of a continuous convex function.

Definition 3.2 (Subdifferential) *Given a positive definite matrix* $\boldsymbol{A} \in \mathbb{R}^{r_n \times r_n}$, *let* $(\mathbb{R}^{r_n}, \langle \cdot, \cdot \rangle_{\boldsymbol{A}})$ *be a real Hilbert space. Then, the subdifferential of a continuous convex function* $f : \mathbb{R}^{r_n} \to \mathbb{R}$ *at* $\boldsymbol{x} \in \mathbb{R}^{r_n}$ *is defined as*

$$\partial_{\boldsymbol{A}} f(\boldsymbol{x}) := \{\tilde{\boldsymbol{x}} \in \mathbb{R}^{r_n} : \langle \boldsymbol{y} - \boldsymbol{x}, \tilde{\boldsymbol{x}} \rangle_{\boldsymbol{A}} + f(\boldsymbol{x}) \leq f(\boldsymbol{y}), \ \forall \boldsymbol{y} \in \mathbb{R}^{r_n}\} \neq \emptyset. \tag{3.43}$$

Proof of Theorem 3.1: By (3.35) and the assumption (ii), the jth element of $\check{\boldsymbol{h}}_{n+1} := \mathrm{prox}_{\lambda_n \tau \Omega_n}(\hat{\boldsymbol{h}}_n^{\mathrm{f}})$ is given by

$$\check{h}_{n+1}^{(j)} = \max\left\{0, 1 - \frac{\lambda_n \tau w_n^{(j)}}{|\hat{h}_n^{(j),\mathrm{f}}|}\right\} \hat{h}_n^{(j),\mathrm{f}} = \left(1 - \frac{\lambda_n \tau w_j^{(n)}}{|\hat{h}_n^{(j),\mathrm{f}}|}\right) \hat{h}_n^{(j),\mathrm{f}} \tag{3.44}$$

By (3.44) and the assumption (i), we obtain

$$\mathrm{sgn}(\check{\boldsymbol{h}}_{n+1}) = \mathrm{sgn}(\hat{\boldsymbol{h}}_n^{\mathrm{f}}) = \mathrm{sgn}(\boldsymbol{G}_n \boldsymbol{W}_n \mathrm{sgn}(\hat{\boldsymbol{h}}_n^{\mathrm{f}})) = \mathrm{sgn}(\tilde{\boldsymbol{w}}_n), \tag{3.45}$$

from which the subdifferential of $\tilde{\Omega}_n$ at $\check{\boldsymbol{h}}_{n+1}$ with the canonical inner product is given by

$$\partial_{\boldsymbol{I}} \tilde{\Omega}_n(\check{\boldsymbol{h}}_{n+1}) = \{|\tilde{\boldsymbol{w}}_n| \circ \mathrm{sgn}(\check{\boldsymbol{h}}_{n+1})\} = \{|\tilde{\boldsymbol{w}}_n| \circ \mathrm{sgn}(\tilde{\boldsymbol{w}}_n)\} = \{\tilde{\boldsymbol{w}}_n\}, \tag{3.46}$$

where $|\cdot|$ denotes componentwise absolute value. By (3.45) and (3.46), one can verify that

$$\begin{aligned} \partial_{\boldsymbol{I}} \Omega_n(\check{\boldsymbol{h}}_{n+1}) &= \{\boldsymbol{W}_n \mathrm{sgn}(\check{\boldsymbol{h}}_{n+1})\} = \{\boldsymbol{W}_n \mathrm{sgn}(\hat{\boldsymbol{h}}_n^{\mathrm{f}})\} \\ &= \{\boldsymbol{G}_n^{-1} \tilde{\boldsymbol{w}}_n\} = \boldsymbol{G}_n^{-1} \partial_{\boldsymbol{I}} \tilde{\Omega}_n(\check{\boldsymbol{h}}_{n+1}) = \partial_{\boldsymbol{G}_n} \tilde{\Omega}_n(\check{\boldsymbol{h}}_{n+1}). \end{aligned} \tag{3.47}$$

Here, the last equality can be verified by observing that

$$\begin{aligned}
&\tilde{\boldsymbol{x}} \in \partial_{\boldsymbol{G}_n} \tilde{\Omega}_n(\check{\boldsymbol{h}}_{n+1}) \\
&\Leftrightarrow \left\langle \boldsymbol{y} - \check{\boldsymbol{h}}_{n+1}, \tilde{\boldsymbol{x}} \right\rangle_{\boldsymbol{G}_n} + \tilde{\Omega}_n(\check{\boldsymbol{h}}_{n+1}) \leq \tilde{\Omega}_n(\boldsymbol{y}),\ \forall \boldsymbol{y} \in \mathbb{R}^{r_n} \\
&\Leftrightarrow \left\langle \boldsymbol{y} - \check{\boldsymbol{h}}_{n+1}, \boldsymbol{G}_n \tilde{\boldsymbol{x}} \right\rangle_{\boldsymbol{I}} + \tilde{\Omega}_n(\check{\boldsymbol{h}}_{n+1}) \leq \tilde{\Omega}_n(\boldsymbol{y}),\ \forall \boldsymbol{y} \in \mathbb{R}^{r_n} \\
&\Leftrightarrow \boldsymbol{G}_n \tilde{\boldsymbol{x}} \in \partial_{\boldsymbol{I}} \tilde{\Omega}_n(\check{\boldsymbol{h}}_{n+1}).
\end{aligned} \tag{3.48}$$

Since the proximity operator can be characterized as the resolvent of the subdifferential [82], i.e.,

$$\mathrm{prox}_{\lambda_n \tau \Omega_n} = \left(I + \lambda_n \tau \partial_{\boldsymbol{I}} \Omega_n\right)^{-1}, \tag{3.49}$$

one can verify with (3.47) that

$$\begin{aligned}
&\check{\boldsymbol{h}}_{n+1} = \mathrm{prox}_{\lambda_n \tau \Omega_n}(\hat{\boldsymbol{h}}_n^{\mathrm{f}}) = \left(I + \lambda_n \tau \partial_{\boldsymbol{I}} \Omega_n\right)^{-1} (\hat{\boldsymbol{h}}_n^{\mathrm{f}}) \\
&\Leftrightarrow \hat{\boldsymbol{h}}_n^{\mathrm{f}} \in \left(I + \lambda_n \tau \partial_{\boldsymbol{I}} \Omega_n\right) (\check{\boldsymbol{h}}_{n+1}) = \left(I + \lambda_n \tau \partial_{\boldsymbol{G}_n} \tilde{\Omega}_n\right) (\check{\boldsymbol{h}}_{n+1}) \\
&\Leftrightarrow \check{\boldsymbol{h}}_{n+1} = \left(I + \lambda_n \tau \partial_{\boldsymbol{G}_n} \tilde{\Omega}_n\right)^{-1} (\hat{\boldsymbol{h}}_n^{\mathrm{f}}) \\
&= \mathrm{prox}^{\boldsymbol{G}_n}_{\lambda_n \tau \tilde{\Omega}_n}(\hat{\boldsymbol{h}}_n^{\mathrm{f}}) = \mathrm{prox}^{\boldsymbol{G}_n}_{\lambda_n \tau \tilde{\Omega}_n}(\hat{\boldsymbol{h}}_n - \lambda_n \nabla_{\boldsymbol{G}_n} \Phi_n(\hat{\boldsymbol{h}}_n)).
\end{aligned} \tag{3.50}$$

This is APFBS with the $\boldsymbol{G}_n$ inner product applied to $\tilde{\Theta}_n(\boldsymbol{h}) = \Phi_n(\boldsymbol{h}) + \tau \tilde{\Omega}_n(\boldsymbol{h})$, and hence (4.25) holds (cf. [60]). □

The proposed algorithm uses different inner products in each of the forward and backward steps. Nevertheless, Theorem 4.1 clarifies its underlying principle. The theorem states that the proposed algorithm pushes the estimate towards the set of minimizers of $\tilde{\Theta}_n$ with the $\boldsymbol{G}_n$ inner product, whereas the algorithm has been derived from the cost function Θ_n. This is simply because the proximity operator, with the canonical inner product, of the weighted ℓ_1 norm can be regarded basically as the proximity operator, with the $\boldsymbol{G}_n$ inner product, of the ℓ_1 norm with the modified weights $\tilde{\boldsymbol{w}}_n$. The efficacy of the proposed algorithm will be shown experimentally in Section 3.4.

Proposition 3.3 (A sufficient condition for the assumption (i)) *Assume that* $[\boldsymbol{G}_n]_{ii} = \kappa(\boldsymbol{u}^{(j_i^{(n)})}, \boldsymbol{u}^{(j_i^{(n)})}) > 0$ *for all* $i = 1, 2, \cdots, r_n$, *where* $[\boldsymbol{G}_n]_{ij}$ *is the* (i, j) *component of* $\boldsymbol{G}_n$ *for any* $i, j = 1, 2, \cdots, r_n$. *Assume also that* $\boldsymbol{W}_n = \boldsymbol{I}$ *and that* $\boldsymbol{G}_n$ *is strictly diagonally dominant, i.e.,*

$$[\boldsymbol{G}_n]_{ii} > \sum_{i \neq j} [\boldsymbol{G}_n]_{ij},\ \forall i = 1, \cdots, r_n. \tag{3.51}$$

Then, the assumption (i) of Theorem 4.1 holds.

Proof: Let $\boldsymbol{a} := [a_1, a_2, \cdots, a_{r_n}]^{\mathsf{T}} \in \{1, -1\}^{r_n}$. Then, the ith element of

$\boldsymbol{G}_n\boldsymbol{a}$ is given by

$$[\boldsymbol{G}_n\boldsymbol{a}]_i = [\boldsymbol{G}_n]_{ii}a_i + \sum_{i\neq j}[\boldsymbol{G}_n]_{ij}a_j. \tag{3.52}$$

Since $a_i \in \{1, -1\}$, we have

$$\begin{aligned}\mathrm{sgn}([\boldsymbol{G}_n\boldsymbol{a}]_i) &= \mathrm{sgn}\left([\boldsymbol{G}_n]_{ii}a_i + \sum_{i\neq j}[\boldsymbol{G}_n]_{ij}a_j\right)\\ &= \mathrm{sgn}\left[\left([\boldsymbol{G}_n]_{ii} + \sum_{i\neq j}[\boldsymbol{G}_n]_{ij}a_j/a_i\right)a_i\right]\\ &= \mathrm{sgn}(a_i) = a_i,\ \forall i = 1, \cdots, r_n. \end{aligned}\tag{3.53}$$

Here, the third equality is verified by

$$[\boldsymbol{G}_n]_{ii} + \sum_{i\neq j}[\boldsymbol{G}_n]_{ij}a_j/a_i \geq [\boldsymbol{G}_n]_{ii} - \sum_{i\neq j}|[\boldsymbol{G}_n]_{ij}| > 0, \tag{3.54}$$

where the strict inequality is due to (3.51). By (3.53), the assumption (i) is directly verified.

□

The assumption (ii) is violated if $\hat{\boldsymbol{h}}_n$ contains some nearly zero components. In such a case, however, those minor components are discarded by the proximity operator and T, and hence violations of the assumption (ii) would give no major impacts on the overall performance. Indeed, the proposed algorithm is robust against violations of the assumptions (i) and (ii), as will be shown in Section 3.4.

Relation to the Φ-PASS algorithm

Roughly speaking, the DR-Φ-PASS algorithm is the Φ-PASS algorithm with the additional operation of dictionary refinements. To be precise, removing the proximity operator and T from (3.36), DR-Φ-PASS is reduced to the original Φ-PASS algorithm with a narrower step-size range. The narrower range is a consequence of the use of the differentiable function Φ_n in (3.30) within the framework of APFBS; the original Φ-PASS algorithm employs a nondifferentiable function (the sum of 'non-squared' distance functions) within the framework of the adaptive projected subgradient method [44]. This would cause no practical disadvantage though because the step size is set typically to a small value in the presence of ambient noise.

Table 3.3: Computational complexity of the proposed and conventional algorithms.

DR-Φ-PASS	$[L+p(s_n+1)+3]r_n + s_n^2 + ps_n + p\upsilon_{\mathrm{inv}}(s_n)$
Sparse QKLMS [42]	$r_n^2 + (L+3)r_n$
FOBOS-KLMS [41]	$(L+6)r_n$

3.3.5 Computational Complexity

The complexity of DR-Φ-PASS depends on the cardinality $\left|\tilde{\mathcal{J}}_{n,\iota}\right|$ which is supposed to be a small constant $s_n \ll r_n$ for all $\iota \in \mathcal{I}_n$ at each time instant $n \in \mathbb{N}$; $s_n \leq 5$ typically. The computational complexity (the number of real multiplications) of the DR-Φ-PASS and existing algorithms are presented under the use of a Gaussian kernel in Table 3.3. Here, $\upsilon_{\mathrm{inv}}(s_n)$ represents the complexity to compute the inverse of an $s_n \times s_n$ Gram matrix. We mention that Sparse QKLMS requires another $O(r_n^2)$ computation for $\boldsymbol{G}_n^{-1}$, in addition to the complexity presented in Table 3.3, when the dictionary is updated; here we postulate the use of the matrix inversion lemma [70]. DR-Φ-PASS makes no use of $\boldsymbol{G}_n^{-1}$ and hence enjoys lower complexity than Sparse QKLMS. This is due to (i) the use of the canonical inner product for the proximity operator and (ii) the selective updating strategy. DR-Φ-PASS exhibits faster convergence, at the price of higher complexity, than FOBOS-KLMS due to the use of the $\boldsymbol{G}_n$-projection (cf. Section 3.2.4). In the following section, we show that DR-Φ-PASS outperforms the existing algorithms in MSE with reasonably low complexities.

3.4 Experiments of the DR-Φ-PASS Algorithm

We show the efficacy of the proposed algorithm in applications to online estimation of nonlinear functions and online prediction of time series data.

3.4.1 Parameter Settings

The parameters λ and τ are affect significantly to the performance of DR-Φ-PASS. The parameter τ is affect significantly to the dictionary size of DR-Φ-PASS, and therefore strict adjustments are necessary to obtain the best performance of the algorithm. However, as a rule of thumb, setting $\tau \in [10^{-5}, 10^{-3}]$ gives a reasonable performance. For setting λ, see Section 2.3.1. The detailed settings of those parameters are presented in each of the experiments.

For the proposed algorithm, the uniform weights $\omega_\iota^{(n)} = (\min\{p, n+1\})^{-1}$, $\iota \in \mathcal{I}_n$, are used for the squared-distance function, and the weights $w_n^{(j)} :=$

Table 3.4: Parameter settings for Experiment A-1.

	parameter
DR-Φ-PASS	$\lambda_n = 0.3,\ p = 1,\ s_{n,\iota} = 1,\ \tau = 0.01$
Φ-PASS	$\lambda_n = 0.3,\ p = 1,\ s_{n,\iota} = 1$

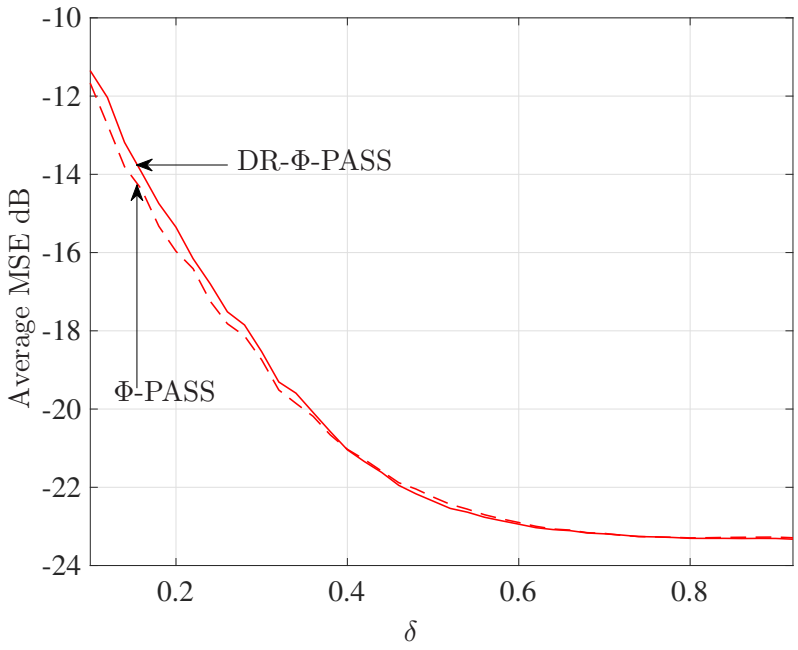

Figure 3.9: Results of Experiment A-1: average MSEs of DR-Φ-PASS and Φ-PASS.

$\frac{|\hat{h}_n^{(j),\mathrm{f}}|^{-1}}{\sum_{\iota\in\mathcal{J}_n}|\hat{h}_n^{(\iota),\mathrm{f}}|^{-1}}$, $j \in \mathcal{J}_n$, are used for the weighted ℓ_1 norm, where $\hat{h}_n^{(\iota),\mathrm{f}}$ is a component of $\hat{\boldsymbol{h}}_n^{\mathrm{f}}$ associated with $\kappa(\cdot, \boldsymbol{u}_\iota)$. We employ a Gaussian kernel with the coherence criterion (see Example 3.1).

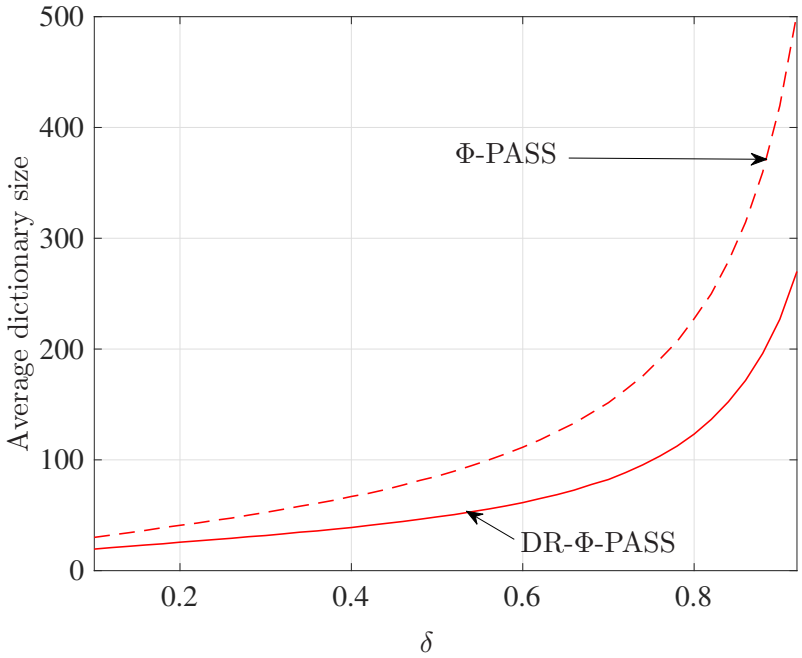

Figure 3.10: Results of Experiment A-1: average dictionary sizes of DR-Φ-PASS and Φ-PASS.

3.4.2 Experiment A — Function Estimation

A-1) A basic performance of the proposed algorithm

We consider the following nonlinear function:

$$\psi(\boldsymbol{u}) = \exp\left(-\frac{\left\|\boldsymbol{u}-\boldsymbol{c}^{(1)}\right\|^2}{2\times 0.6^2}\right) + 1.5\exp\left(-\frac{\left\|\boldsymbol{u}-\boldsymbol{c}^{(2)}\right\|^2}{2\times 0.6^2}\right) - 0.5\exp\left(-\frac{\left\|\boldsymbol{u}-\boldsymbol{c}^{(3)}\right\|^2}{2\times 0.6^2}\right) - 1.25\exp\left(-\frac{\left\|\boldsymbol{u}-\boldsymbol{c}^{(4)}\right\|^2}{2\times 0.6^2}\right), \tag{3.55}$$

where $\boldsymbol{c}^{(1)} := [-0.2, 0.7]^\mathsf{T}$, $\boldsymbol{c}^{(2)} := [0.2, -0.3]^\mathsf{T}$, $\boldsymbol{c}^{(3)} := [0.6, 0.2]^\mathsf{T}$, and $\boldsymbol{c}^{(4)} := [-0.3, -0.8]^\mathsf{T}$. The observed signal is generated as $d_n := \psi(\boldsymbol{u}_n) + v_n$, $n \in \mathbb{N}$, where $\boldsymbol{u}_n$ is the input data randomly generated from a uniform distribution within the region $[-1, 1]^2$ and $v_n \sim \mathcal{N}(0, 1.0 \times 10^{-2})$ is the additive white Gaussian noise. The kernel parameter is set to $\xi = \sqrt{0.3}$. To verify that the proposed algorithm refines the dictionary efficiently, we compare the performance of the proposed algorithm with the Φ-PASS algorithm (see Remark 3.3.4). The set of parameters employed in this experiment is summarized in Table 3.4.

Figures 3.9, 3.9, and 3.11 depict the MSE, the dictionary size averaged

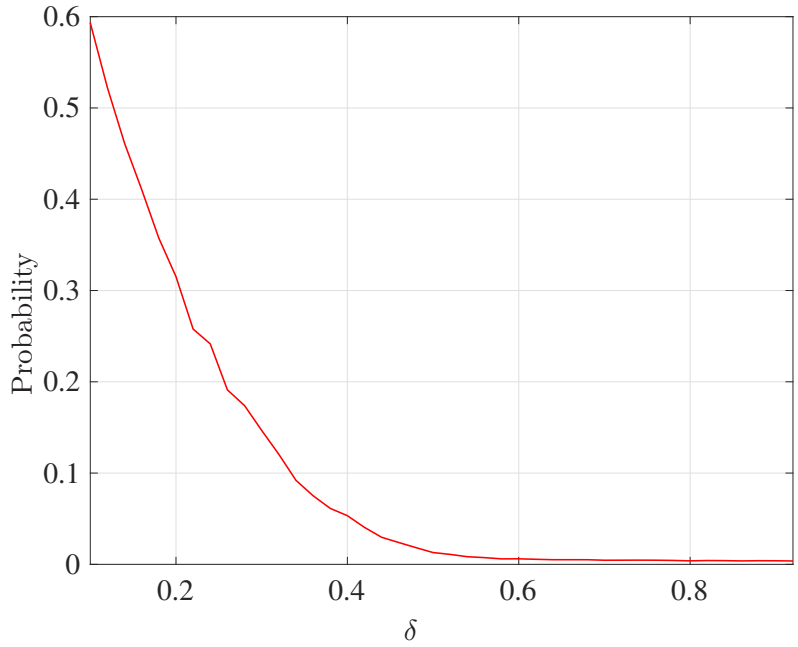

Figure 3.11: Results of Experiment A-1: probabilities of DR-Φ-PASS and Φ-PASS.

over 5000 iterations and also over 100 runs for each coherence parameter δ, and the probability that the assumption (i) of Theorem 3.1 holds true. It can be seen that DR-Φ-PASS reduces the dictionary size by half approximately compared to that of Φ-PASS while the performance degradation is negligible. It can also be seen that, even though the probability is approximately zero for $\delta \geq 0.7$, DR-Φ-PASS performs well. This indicates the robustness of DR-Φ-PASS against violations of the assumption (i) of Theorem 3.1.

A-2) A comparison with the existing algorithms using ℓ_1 regularization

We show the advantages of the proposed algorithm over the existing algorithms that refine the dictionary using ℓ_1 regularization: the FOBOS-KLMS algorithm [41] and the Sparse QKLMS algorithm [42]. We consider a fluid-flow control problem [13]:

$$\begin{cases} y_n := 0.1044u_n + 0.0883u_{n-1} + 1.4138y_{n-1} - 0.6065y_{n-2} \\ d_n := 0.3163y_n/\sqrt{0.1 + 0.9y_n^2} + \nu_n^{(1)}, \end{cases} \tag{3.56}$$

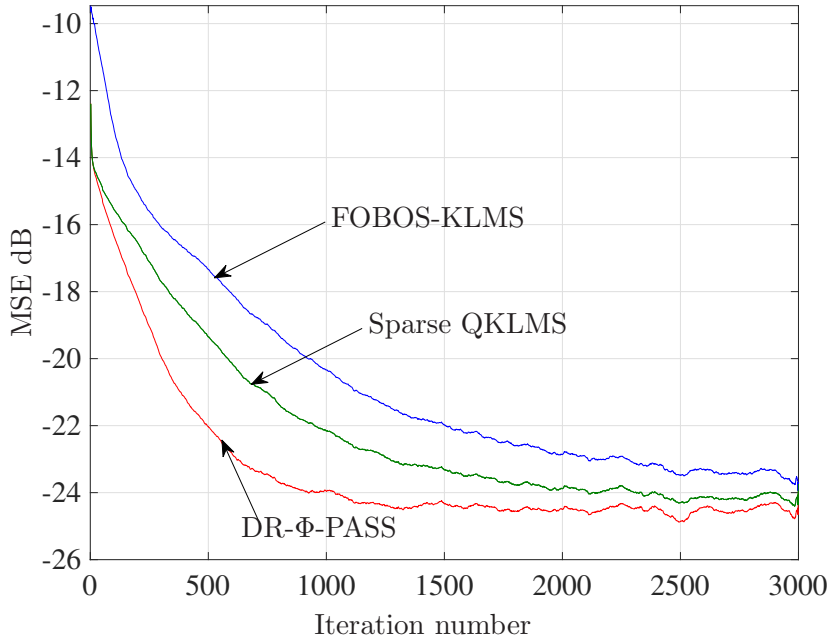

Figure 3.12: Results of Experiment A-2: MSEs of DR-Φ-PASS, FOBOS-KLMS, and Sparse-QKLMS.

Table 3.5: Parameter settings for Experiment A-2.

	parameter		dictionary size (mean)	complexity
DR-Φ-PASS	$\lambda_n = 0.02,\ p = 4$ $s_{n,\iota} = 1,\ \tau = 1.0 \times 10^{-2}$	$\delta = 0.9$	14.75	284
Sparse QKLMS	$\eta = 0.015,\ \lambda = 4.0 \times 10^{-3}$ $\gamma = 1.0 \times 10^{-4}$		23.50	670
FOBOS-KLMS	$\mu = 0.01,\ \lambda = 2.5 \times 10^{-4}$ $\epsilon_\alpha = 1.0 \times 10^{-6}$		15.83	126

where the input u_n is generated by

$$u_0 := 0 \text{ and } u_n := 0.5u_{n-1} + \sqrt{1 - 0.5^2}\nu_n^{(2)},\ n \in \mathbb{N}^*. \tag{3.57}$$

Here, $\nu_n^{(2)} \sim \mathcal{N}(0, 1.0)$ and $\nu_n^{(1)} \sim \mathcal{N}(0, 2.5 \times 10^{-3})$ are the additive white Gaussian noise. The output d_n is predicted with input $\boldsymbol{u}_n = [u_n, u_{n-1}]^\mathsf{T}$ ($L = 2$). We test 100 independent runs by generating the noise randomly and MSE is computed by averaging the instantaneous squared error over the 100 runs. Table 3.5 summarizes the parameters. For FOBOS-KLMS, the step size and the regularization parameter of the ℓ_1 norm are denoted by η and λ, respectively. For Sparse QKLMS, the step size, the regularization

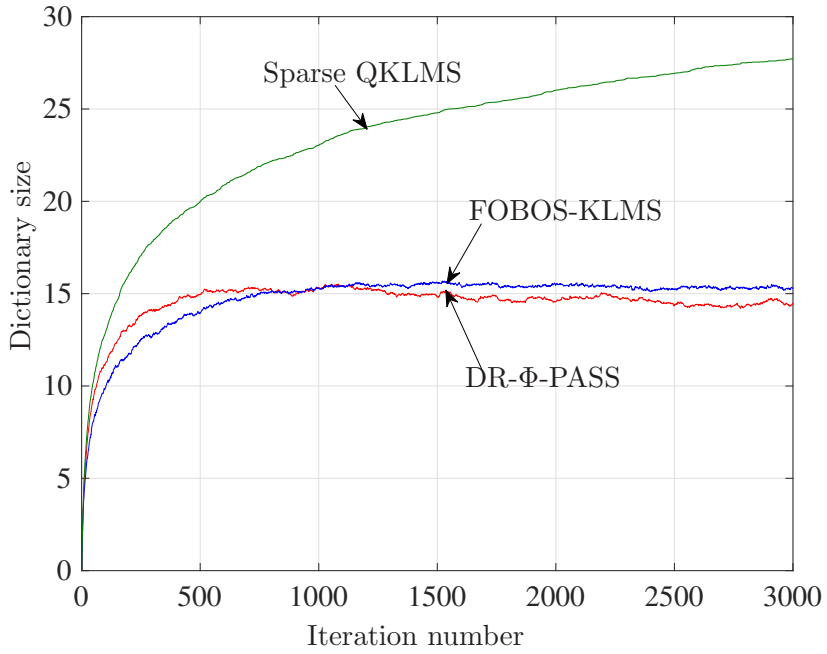

Figure 3.13: Results of Experiment A-2: dictionary sizes of DR-Φ-PASS, FOBOS-KLMS, and Sparse-QKLMS.

parameter for the ℓ_1 norm, and the regularization parameter for the Gram matrix $\boldsymbol{G}_n$ are denoted by η, λ, and γ, respectively. For the proposed algorithm and FOBOS-KLMS, the regularization parameters for the ℓ_1 norm are chosen so that the dictionary sizes of both algorithms are nearly identical to each other. Sparse QKLMS operates iterative shrinkage to obtain a sparse coefficient-vector, but keeps all the dictionary elements even though some coefficients are nearly/exactly zero. For comparison, the regularization parameters λ of Sparse QKLMS is chosen so that the number of coefficients larger than the threshold 0.01 is nearly identical to the dictionary sizes of the other algorithms. For all algorithms, the step sizes are chosen so that the steady-state MSEs are nearly identical to each other. The other parameters are chosen so that each algorithm achieves the best performance. For FOBOS-KLMS, the weights of the ℓ_1 norm are set to $w_{j,n} = (|h_{j,n}| + \epsilon_\alpha)^{-1}$ for some small constant $\epsilon_\alpha > 0$.

Figures 3.12 and 3.13 depict the MSE learning curves and the growth of dictionary size. Table 3.5 summarizes the average dictionary sizes and the average computational complexities. One can see that DR-Φ-PASS attains faster convergence than Sparse QKLMS due to the use of the parallel projection. Moreover, thanks to (i) the use of the canonical inner product for the backward step of APFBS and (ii) the selective updating strategy, DR-Φ-PASS requires lower complexities than Sparse QKLMS. One may notice

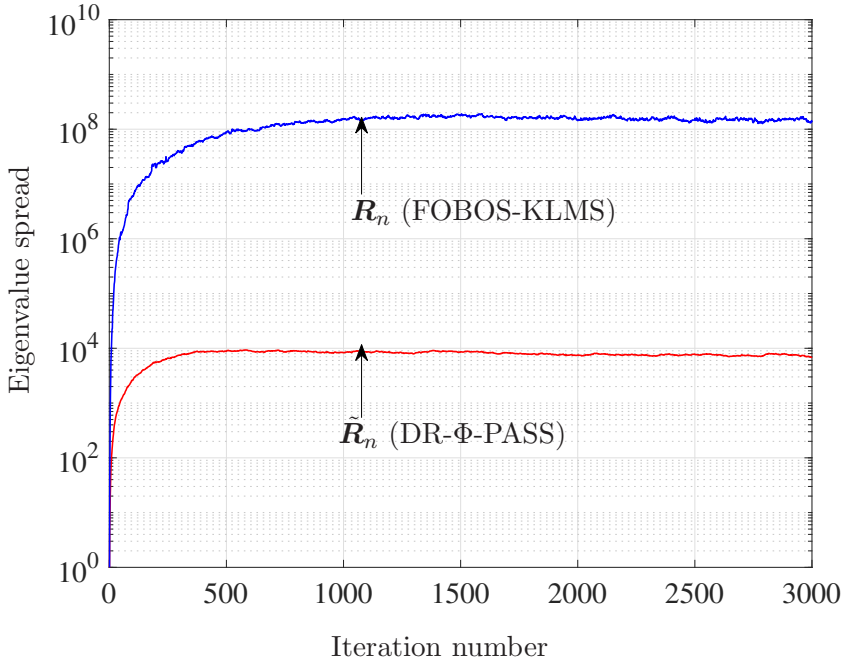

Figure 3.14: Eigenvalue spreads of $\boldsymbol{R}_n$ (of FOBOS-KLMS) and $\tilde{\boldsymbol{R}}_n$ (of DR-Φ-PASS).

Table 3.6: Parameter settings and complexities for Experiment B.

	parameter	dictionary size (mean)	complexity
DR-Φ-PASS	$\lambda_n = 0.3,\ p = 8,$ $s_{n,\iota} = 1,\ \delta = 1,\ \tau = 0.025$	16.54	851
KRLS Tracker	$M = 17,\ \lambda = 0.999,\ \sigma_n^2 = 0.1$	17	1055

that FOBOS-KLMS converges slowly compared with the other algorithms. This is because the error surface for FOBOS-KLMS is squeezed due to the large eigenvalue spread of the autocorrelation matrix $\boldsymbol{R}_n$ (cf. Section 3.2.4). (Note that $\boldsymbol{R}_n$ depends on the dictionary and hence on time n in general.) To verify this, we plot the eigenvalue spreads, averaged over 300 runs, of $\boldsymbol{R}_n$ and $\tilde{\boldsymbol{R}}_n := \boldsymbol{G}_n^{-1/2}\boldsymbol{R}_n\boldsymbol{G}_n^{-1/2}$ in Figure 3.14. One can see that the eigenvalue spreads of $\tilde{\boldsymbol{R}}_n$ are smaller than those of $\boldsymbol{R}_n$, which implies that the error surface for DR-Φ-PASS is better shaped than that for FOBOS-KLMS. The matrix $\boldsymbol{R}_n$ was computed according to the closed-form expression presented in [83].

3.4.3 Experiment B — Nonstationary Data Prediction

We consider the nonstationary data given as follows:

$$\begin{aligned} d_n &:= (0.8 - 0.5\exp(d_{n-1}^2))d_{n-1} - (0.3 + 0.9\exp(-d_{n-1}^2))d_{n-2} \\ &+ 0.1\sin(d_{n-1}\pi) \text{ for } 0 \le n \le 10000, \\ d_n &:= (0.2 - 0.7\exp(-d_{n-1}^2))d_{n-1} \\ &- 0.8(1 + \exp(-d_{n-1}^2)d_{n-2}) + 0.2\sin(d_{n-1}\pi) \text{ for } n > 10000, \end{aligned} \tag{3.58}$$

with $d_{-2} := d_{-1} := 0.1$. In this experiment, we evaluate the tracking capability to the nonstationarity. Each datum d_n is predicted with the input vector $\boldsymbol{u}_n := [\hat{d}_{n-1}, \hat{d}_{n-2}, \cdots, \hat{d}_{n-L}]^\mathsf{T} \in \mathcal{U} \subset \mathbb{R}^L$ ($L = 4$). Here, $\hat{d}_n := d_n + v_n$, $n \in \mathbb{N}$, where $v_n \sim \mathcal{N}(0, 1.0 \times 10^{-2})$ is the additive white Gaussian noise. The kernel parameter is set to $\xi = 1$. DR-Φ-PASS is compared with one of the state-of-the-art algorithms based on budget learning: KRLS Tracker [30]. We test 100 independent runs by generating the noise randomly and MSEs are computed by averaging the instantaneous squared errors over the 100 runs. Table 3.6 summarizes the set of parameters, the average dictionary sizes, and the average complexity of each algorithm. The budget size, the forgetting factor, and the regularization parameter for KRLS Tracker are denoted by M, λ, and σ_n^2, respectively. The parameters are chosen so that each algorithm achieves the best performance and the average dictionary sizes are close to each other.

Figures 3.15 and 3.16 depict the MSE learning curves and the growth of dictionary size. It can be seen that DR-Φ-PASS well controls the dictionary size, leading to fast convergence and tracking with low complexity.

3.4.4 Experiment C — Real Data

We compare the performance of the proposed algorithm with the conventional algorithms in an application to online prediction of the chaotic laser time series from the Santa Fe time series competition [84]. Each datum d_n is predicted with $\boldsymbol{u}_n := [d_{n-1}, d_{n-2}, \cdots, d_{n-L}]^\mathsf{T} \in \mathcal{U} \subset \mathbb{R}^L$ for $L = 8$ and $\sigma = 0.25$. The compared algorithm includes the linear Kalman filter and the resource allocating network algorithm [85]. The state-space model for the Kalman filter is given as

$$\begin{aligned} \boldsymbol{h}_{n+1} &= \boldsymbol{h}_n + \mathbf{1}\nu_n^{(1)} \\ \hat{d}_n &= \boldsymbol{u}_n^\mathsf{T}\boldsymbol{h}_n + \nu_n^{(2)}, \end{aligned} \tag{3.59}$$

where $\nu_n^{(1)}$ and $\nu_n^{(2)}$ are noises generated by Gaussian distributions and $\mathbf{1} = [1, 1, \cdots, 1]^\mathsf{T} \in \mathbb{R}^L$. Table 3.7 summarizes the parameters, and Table 3.8 summarizes the average dictionary sizes, the average complexities, and the average squared errors. Parameters are chosen so that each algo-

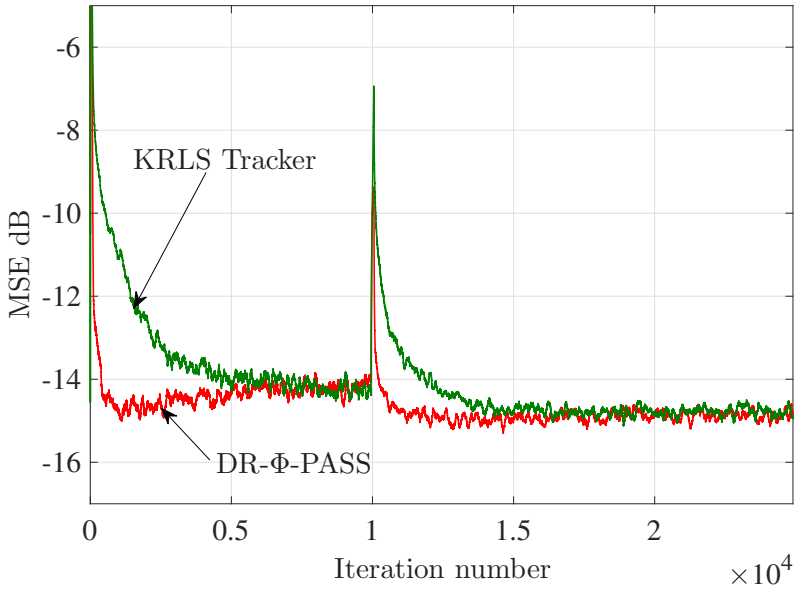

Figure 3.15: Results of Experiment B: MSEs of DR-Φ-PASS and KRLS Tracker.

rithm achieves the best performance. The kernel parameter is set to $\xi = 1$. The dictionary size M for KRLS Tracker is chosen so that its average dictionary size is comparable to that of the proposed algorithm. Each algorithm operates a single run over the data.

Figures 3.17 and 3.18 depict the results. It can be seen that DR-Φ-PASS outperforms the other algorithms excluding the initial phase. It can also be seen from Table 3.8 that the average MSE and the complexity of DR-Φ-PASS are smaller than those of the other algorithms; DR-Φ-PASS attains approximately 2 dB and 2.5 dB lower MSE compared with KRLS Tracker and Sparse QKLMS, respectively. It should be mentioned that KRLS Tracker uses a large size of dictionary to attain fast initial convergence.

3.4.5 Wrapping Up

The important outcomes obtained through the experiments are listed below.

1. The proposed algorithm enjoyed low complexities as well as efficient dictionary refinements in all the experiments. The dictionary refinements are due to the use of the ℓ_1 regularization, keeping the dictionary suitable for the statistics of inputs. The low complexity is a direct consequence of small dictionary-size but also comes from (i) the use of the canonical inner product for the backward step and (ii) the selective

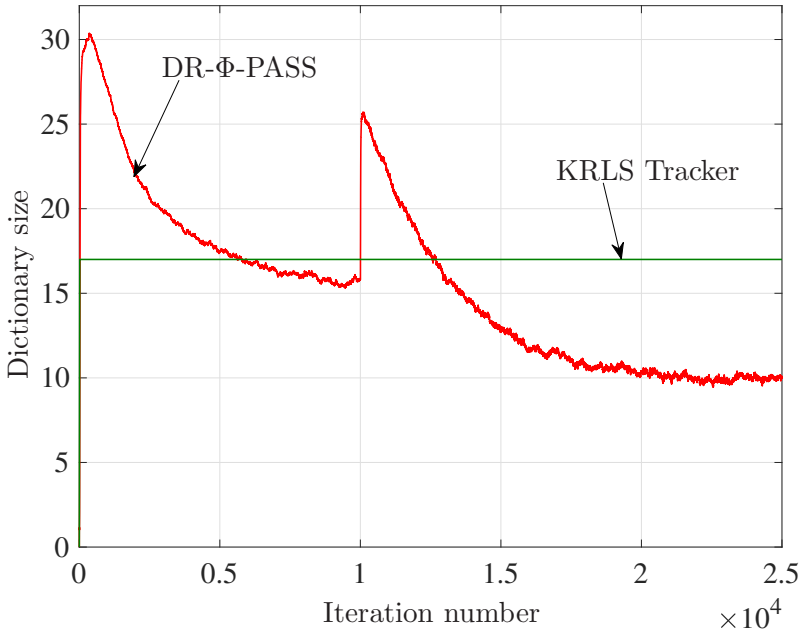

Figure 3.16: Results of Experiment B: dictionary sizes of DR-Φ-PASS and KRLS Tracker.

updating strategy. It should be emphasized that the dictionary refinements were achieved without any severe performance degradation.

2. The proposed algorithm exhibited fast convergence and tracking. In particular, its superior performance to FOBOS-KLMS is due to the use of the $\boldsymbol{G}_n$ inner product; this was verified by analyzing the eigenvalue spreads of the autocorrelation matrices $\boldsymbol{R}_n$ and $\tilde{\boldsymbol{R}}_n$. The fast tracking for the nonstationary data is a consequence of the efficient dictionary refinements.

3. For the real data, the proposed algorithm significantly outperformed the state-of-the-art algorithms: Sparse QKLMS, FOBOS-KLMS, and KRLS Tracker.

4. The proposed algorithm worked robustly against the violation of the assumptions (i) and (ii) of Theorem 3.1.

3.5 Conclusion

In the first part, we studied the relation between the functional-space and Euclidean-space approaches under the isomorphism between the dictionary

Table 3.7: Parameter settings and results for Experiment C.

	parameter
DR-Φ-PASS	$\lambda_n = 0.3,\ p = 4,\ s_{n,\iota} = 1,\ \tau = 0.01,\ \delta = 0.6$
KRLS Tracker	$M = 11,\ \lambda = 0.995,\ \sigma_n^2 = 0.08$
FOBOS-KLMS	$\eta = 0.3,\ \lambda = 3 \times 10^{-6},\ \epsilon_\alpha = 1 \times 10^{-3}, \delta = 0.6$
Sparse QKLMS	$\eta = 0.3,\ \lambda = 1 \times 10^{-3},\ \gamma = 1 \times 10^{-4}, \delta = 0.6$
Kalman filter	$\nu_n^{(1)}, \nu_n^{(2)} \sim \mathcal{N}(0, 10^{-2})$
RAN	$\eta = 0.3,\ \epsilon = 1 \times 10^{-3},\ \delta_{\min} = 1 \times 10^{-2}, \delta_{\max} = \tau = 100$

Table 3.8: Results of Experiment C.

	dictionary size (mean)	complexity	squared error dB
DR-Φ-PASS	24.33	1050	−42.12
KRLS Tracker	24.95	2243	−39.95
FOBOS-KLMS	25.92	363	−38.67
Sparse QKLMS	61.72	4488	−38.38
Kalman	–	224	−22.37
RAN	29.14	641	−28.54

subspace and the Euclidean space equipped with the particular inner product defined with the kernel matrix. The CKLMS algorithm was presented as a stochastic restricted-gradient method for the MSE cost functional and it was shown that the shape of its error surface is governed (the kernelized input vector left-multiplied by the square-root inverse of the kernel matrix).

We remark that a theoretical result about the autocorrelation matrix of the functional-space approach is given in [86]. In [86], it is shown that the eigenvalue spread of the autocorrelation matrix of CKLMS is approximately a square root of that for KNLMS. In [79], an online estimation algorithm associated with a result in [86] is proposed. The algorithm in [79] is based on L^2-space projections to employ the best metric that induces a perfect decorrelation property for online nonlinear estimation.

In the second part, we proposed the DR-Φ-PASS algorithm having three ingredients: (i) parallel $\boldsymbol{G}_n$-projection, (ii) selective update, and (iii) dictionary refinement. The proposed algorithm is basically the APFBS method, but it performs the gradient step with the dictionary-dependent inner product and the proximal step with the canonical inner product. It was shown that the sequence of coefficient vectors generated by the proposed algorithm monotonically approaches, in the $\boldsymbol{G}_n$-norm sense, the set of minimizers of the weighted squared-distance cost function penalized by an ℓ_1 norm with the modified weights. The numerical examples showed that the proposed algorithm achieved effective dictionary refinements as well as fast convergence/tracking under possible data-nonstationarity with low computational complexity.

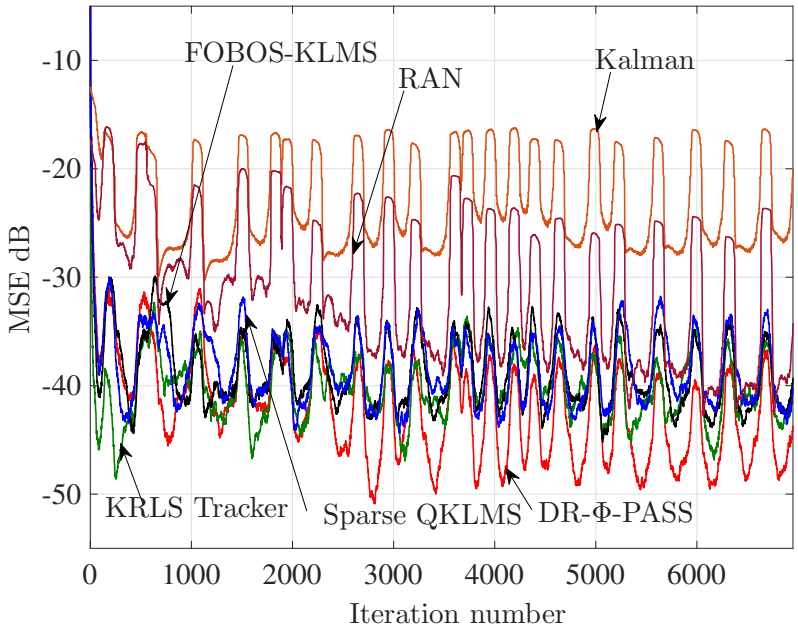

Figure 3.17: Results of Experiment C: MSEs of DR-Φ-PASS, Sparse-QKLMS, FOBOS-KLMS, KRLS Tracker, Kalman filter, and RAN.

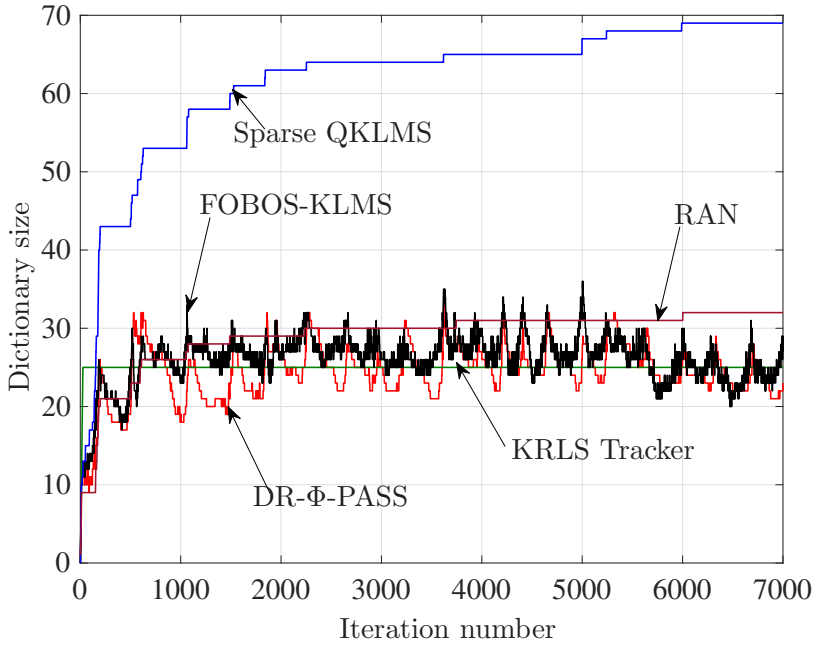

Figure 3.18: Results of Experiment C: dictionary sizes of DR-Φ-PASS, Sparse-QKLMS, FOBOS-KLMS, KRLS Tracker, Kalman filter, and RAN.

Chapter 4

An Efficient KAF Algorithm with Adaptations of Kernel Scales and Centers

4.1 Introduction

The primal goal of this chapter is to devise an online algorithm that finds an efficient model (and the coefficients simultaneously) being able to express the nonlinear function accurately with reasonably short expansion length. The efficient model would yield a number of practical benefits such as avoidance of over-fitting, reduction of computational complexity, saving of memory storage, improvements of convergence behavior, as well as disclosure of the latent dimension (interpretability of the resultant estimate).

We propose an efficient adaptive method updating the parameters (scales and centers) of the Gaussian Kernel and the coefficients alternately to reduce the instantaneous squared errors. To enhance the model efficiency, the instantaneous cost is penalized by the weighted ℓ_1 norm of the coefficient vector, which leads to dictionary pruning. In the kernel adaptive filtering context, some related works have been proposed to adapt the kernel scales [53, 54, 55] and centers [56, 57] in the dictionary. The method in [53] updates the scales only when each kernel enters the dictionary and keeps those scales unchanged after that. Its performance is therefore rather limited. The method proposed in [54] uses a common scale parameter for all kernel functions. The method in [55, 58] updates both scales and centers individually, as in the way of the proposed approach. The key difference between the proposed algorithm and those methods is a novel online dictionary growing technique, under which the dictionary grows with multiple initial scales selected by a hierarchical selection strategy. The proposed dictionary growing technique is motivated by the fact that the performance of the aforementioned alternating update approach depends highly on the initial scales [59].

Specifically, the initial Gaussian scales affect the efficiency and the precision of the estimate significantly when the selected scale was far from the actual scales of the target function due to the vanishment of gradients.

The major properties of the proposed algorithm are summarized below. Multiple initial values for the Gaussian scales are employed to alleviate the sensitivity to the initial conditions. It is expected here that at least some of the initial scales are relevant to the estimation task. The use of multiple initial values, however, may cause undesirable growths of the dictionary size (which involve high computational complexities and large memory size). To avoid this, we present an efficient dictionary growing strategy named *multi-scale screening method*, which 'screens' the large- and small- scale Gaussians based on the error and novelty criteria. The computational complexity is reasonably low thanks to a certain selection strategy for dictionary growing and scale/center updating.

4.2 Nonlinear Model and Cost Function

We define the Gaussian function as

$$g(\boldsymbol{u};\xi,\boldsymbol{c}) := \exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{c}\|^2}{2\xi}\right) \tag{4.1}$$

with the scale (variance) parameter $\xi > 0$ and the center (mean) vector $\boldsymbol{c} \in \mathbb{R}^L$. Our time-varying model is then given as

$$\varphi_n(\boldsymbol{u}) := \sum_{j=1}^{r_n} h_n^{(j)} g(\boldsymbol{u};\xi_n^{(j)},\boldsymbol{c}_n^{(j)}),\ \boldsymbol{u} \in \mathcal{U}, \tag{4.2}$$

with the height $h_n^{(j)} \in \mathbb{R}$, scale $\xi_n^{(j)} > 0$, and center $\boldsymbol{c}_n^{(j)} \in \mathbb{R}^L$ of the jth Gaussian. In this study, the scale $\xi_n^{(j)}$ and the center $\boldsymbol{c}_n^{(j)}$ of each Gaussian (atom) in the dictionary $\{g(\cdot;\xi_n^{(j)},\boldsymbol{c}_n^{(j)})\}_{j=1}^{r_n}$, $n \in \mathbb{N}$, are regarded as *variables*. Namely, those parameters are updated iteratively so that our estimate φ_n becomes an efficient approximation of the target nonlinear function ψ.

At time instant n, the variables (heights, scales, and centers of r_n Gaussians) can be expressed respectively as $\boldsymbol{h} := [h^{(1)}, h^{(2)}, \cdots, h^{(r_n)}]^\mathsf{T} \in \mathbb{R}^{r_n}$, $\boldsymbol{\xi} := [\xi^{(1)}, \xi^{(2)}, \cdots, \xi^{(r_n)}]^\mathsf{T} \in \mathbb{R}_{++}^{r_n}$, and $\boldsymbol{C} := [\boldsymbol{c}^{(1)}\ \boldsymbol{c}^{(2)}\ \cdots\ \boldsymbol{c}^{(r_n)}] \in \mathbb{R}^{L\times r_n}$. The instantaneous cost function is then given by

$$\Theta_n(\boldsymbol{h},\boldsymbol{\xi},\boldsymbol{C}) := J_n(\boldsymbol{h},\boldsymbol{\xi},\boldsymbol{C}) + \tau\Omega_n(\boldsymbol{h},\boldsymbol{\xi},\boldsymbol{C}), \tag{4.3}$$

where $\tau > 0$ is the regularization parameter, and

$$J_n(\boldsymbol{h},\boldsymbol{\xi},\boldsymbol{C}) := \frac{1}{2}(d_n - \varphi_n(\boldsymbol{u}_n))^2 \tag{4.4}$$

$$\Omega_n\left(\boldsymbol{h},\boldsymbol{\xi},\boldsymbol{C}\right) := \sum_{j=1}^{r_n} w_n^{(j)} \left|h^{(j)}\right|, \tag{4.5}$$

for some positive weights $w_n^{(j)} > 0$. The weighted ℓ_1 norm serves to discard those redundant/obsolete Gaussians that make no contribution to the estimation, yielding parsimonious estimates without causing serious performance degradations [28, 87, 35, 36, 42, 88]. Indeed, it has shown in [87] that obsolete Gaussians remaining in the dictionary may give negative impacts on the performance, and the weighted-ℓ_1 regularization mitigates such negative impacts. See [41, 28, 88, 42] for more details about the dictionary refinement techniques.

Since the squared error function (4.4) is nonconvex for the widths and centers, it is not practical to give an algorithm and its analysis based on convex optimization as in Chapter 2 and 3. In this chapter, we thus propose a heuristic, but efficient, algorithm based on the coordinate descent algorithm [89] as well as presenting the stepsizes that guarantee the monotonic decreasing of the cost function (4.3). For more details, see Section 4.3.2.

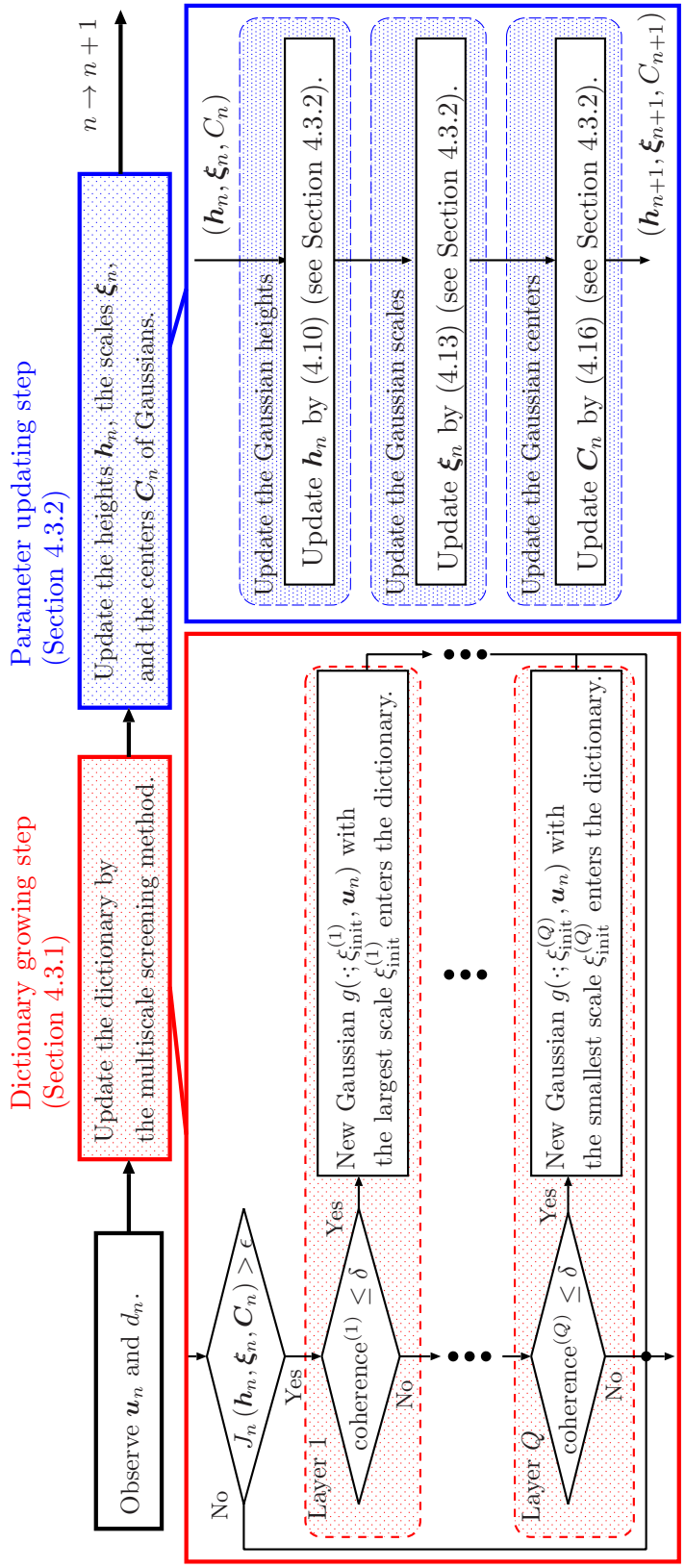

Figure 4.1: A diagram of ONEGAP. At each time instant n, ONEGAP updates the estimate in two steps.

4.3 Proposed Algorithm

The proposed algorithm, named ONEGAP, consists of two steps: (i) the dictionary growing step and (ii) the parameter updating step, where the latter includes the dictionary pruning process. The flowchart is given in Figure 4.1. In the first step, the dictionary is initialized to an empty set ($r_0 := 0$), and it grows under a hierarchical selection strategy using the sequentially coming data. In the second step, the variables $\boldsymbol{h}_n$, $\boldsymbol{\xi}_n$, and $\boldsymbol{C}_n$ are updated in sequence. Each step will be detailed below.

4.3.1 Dictionary Growing Strategy under Multiscale Screening

We pay our attention to the following fact: $J_n(\boldsymbol{h}, \boldsymbol{\xi}, \boldsymbol{C})$ is nonconvex as a function of each scale parameter $\xi^{(j)}$, and it has shallow slopes at those points which are far from the optimal point (see [59]). This means that the gradient vanishes if the initial scale is undesirably large or smaller compared to the optimal one. In such a case, the learning speeds of the poorly-initialized $\xi^{(j)}$ become unacceptably slow, and this may cause a serious deterioration of the whole estimation process. In our preliminary experiments, the use of $\xi_{\text{init}}^{(1)}$ which is hundred times larger/smaller than an adequate scale caused slow convergence. We thus employ multiple initial values for the Gaussian scales so that at least some of the initial scales are suitable for the data. To avoid undesirable growths of the dictionary size due to the use of multiple initial scales, each input vector is tested from coarse to fine 'screens' corresponding to large- to small- scale Gaussians. This efficient dictionary growing strategy is named *multiscale screening method*. The multiscale screening method consists of the following two sub-steps: (i) the error test and (ii) the novelty test.

The error test is rather simple. When the estimation error is sufficiently small, the current estimate is good enough already for the current input $\boldsymbol{u}_n$ and therefore there is no need to add the new Gaussian (centered at the current input $\boldsymbol{u}_n$) into the dictionary, as there remains little space for improvements in estimation accuracy and such a redundant Gaussian function may even give negative impacts on the performance as mentioned in the previous section. The error condition is thus given as follows:

$$J_n(\boldsymbol{h}_n, \boldsymbol{\xi}_n, \boldsymbol{C}_n) > \epsilon, \tag{4.6}$$

where $\epsilon \geq 0$ is the threshold. Here, $\boldsymbol{h}_n := [h_n^{(1)}, h_n^{(2)}, \cdots, h_n^{(r_n)}]^\mathsf{T}$, $\boldsymbol{\xi}_n := [\xi_n^{(1)}, \xi_n^{(2)}, \cdots, \xi_n^{(r_n)}]^\mathsf{T}$, and $\boldsymbol{C}_n := [\boldsymbol{c}_n^{(1)}\ \boldsymbol{c}_n^{(2)}\ \cdots\ \boldsymbol{c}_n^{(r_n)}]$. If the error condition is satisfied, the novelty test is conducted to select a Gaussian function with an adequate scale parameter; otherwise, the dictionary does not grow at this time instant.

The novelty test is performed hierarchically based on the multiscale screening to select an adequate Gaussian scale. The multiscale screening aims to enhance the model efficiency. The global structures (the low frequency components) of the nonlinear function ψ can be captured efficiently by relatively large scale Gaussian functions, while the local structures (the fine parts) of ψ can be captured efficiently by Gaussian functions of appropriately small scales. The central philosophy of the multiscale screening is the following: (i) extract the global structures at the initial phase of estimation and (ii) extract the local structures (the estimation residual after removing the global structures) gradually once the dictionary for the global ones is well developed (for more discussions about the global-to-local order of the multiscale screening, see Section 4.4.2).

We now explain how to choose the initial scale at each iteration. A wide range of scales $\xi_{\text{init}}^{(1)} > \xi_{\text{init}}^{(2)} > \cdots > \xi_{\text{init}}^{(Q)} > 0$ are usually adopted. At time instant $n := 0$, the dictionary is empty, and the largest scale $\xi_{\text{init}}^{(1)}$ to extract the global structure is selected automatically without any novelty test, which means that the function $g(\cdot; \xi_{\text{init}}^{(1)}, \boldsymbol{u}_0)$ enters the dictionary. From the second iteration, the novelty test is conducted. At time instant $n \geq 2$, the similarity between $g(\cdot; \xi_{\text{init}}^{(1)}, \boldsymbol{u}_n)$ and (a selected subset of) the current dictionary is evaluated. (Indeed, the similarity is evaluated only with a subset of the dictionary selected under some criterion as explained later on for reducing the computational costs of the novelty test.) If the similarity is sufficiently low, $g(\cdot; \xi_{\text{init}}^{(1)}, \boldsymbol{u}_n)$ is regarded novel and it enters the dictionary. If, and only if, the similarity is high, it is regarded redundant and the second Gaussian $g(\cdot; \xi_{\text{init}}^{(2)}, \boldsymbol{u}_n)$ is tested in the same way. If the similarity is sufficiently low, $g(\cdot; \xi_{\text{init}}^{(2)}, \boldsymbol{u}_n)$ enters the dictionary, and, if (and only if) the similarity is high, it is regarded redundant and the third one is tested. This continues until some Gaussian is regarded novel; if all the Gaussian functions are regarded redundant, the dictionary does not grow at that time instant. Suppose that some $g(\cdot; \xi_{\text{init}}^{(q)}, \boldsymbol{u}_n)$ enters the dictionary. Then, the sizes of the variable vectors and matrix are augmented: the augmented vectors and matrices are given by $\hat{\boldsymbol{h}}_n := [\boldsymbol{h}_n^{\mathsf{T}}\ 0]^{\mathsf{T}} \in \mathbb{R}^{r_n+1}$, $\hat{\boldsymbol{\xi}}_n := [\boldsymbol{\xi}_n^{\mathsf{T}}\ \xi_{\text{init}}^{(q)}]^{\mathsf{T}} \in \mathbb{R}^{r_n+1}$, and $\hat{\boldsymbol{C}}_n := [\boldsymbol{C}_n\ \boldsymbol{u}_n] \in \mathbb{R}^{L\times(r_n+1)}$, respectively. Suppose in contrast that no Gaussian enters the dictionary. In this case, we let $\hat{\boldsymbol{h}}_n := \boldsymbol{h}_n$, $\hat{\boldsymbol{\xi}}_n := \boldsymbol{\xi}_n$, and $\hat{\boldsymbol{C}}_n := \boldsymbol{C}_n$.

Now, we present the selection strategy and the novelty criterion (the similarity measure). For computational efficiency, our strategy is the following: select a subset $\{g(\cdot; \xi_n^{(j)}, \boldsymbol{c}_n^{(j)})\}_{j\in\mathcal{J}_n} \subset \{g(\cdot; \xi_n^{(j)}, \boldsymbol{c}_n^{(j)})\}_{j=1}^{r_n}$ of the Gaussian functions in the dictionary that return the largest values at the current input $\boldsymbol{u}_n$ (see Figure 4.2). Here, $\mathcal{J}_n := \{j_1, \cdots, j_{s_n^{(NC)}}\}$ with its cardinality $|\mathcal{J}_n| = s_n^{(NC)}$ denotes the index set of the selected Gaussians. More specifi-

cally, we let $\{j_1, j_2, \cdots, j_{r_n}\} = \{1, 2, \cdots, r_n\}$ such that

$$g(\boldsymbol{u}_n; \xi_n^{(j_i)}, \boldsymbol{c}_n^{(j_i)}) \geq g(\boldsymbol{u}_n; \xi_n^{(j_k)}, \boldsymbol{c}_n^{(j_k)}),\ 1 \leq i < k \leq r_n. \tag{4.7}$$

This selection strategy is computationally efficient and is expected to include such a dictionary atom that maximizes our novelty criterion of L^2 coherence (see [27] for the detail of the coherence criterion)

$$\begin{aligned} c(\xi^{(u)}, \boldsymbol{u}, \xi^{(v)}, \boldsymbol{v}) &:= \left| \frac{\left\langle g(\cdot; \xi^{(u)}, \boldsymbol{u}), g(\cdot; \xi^{(v)}, \boldsymbol{v}) \right\rangle_{L^2}}{\left\| g(\cdot; \xi^{(u)}, \boldsymbol{u}) \right\|_{L^2} \left\| g(\cdot; \xi^{(v)}, \boldsymbol{v}) \right\|_{L^2}} \right| \\ &= \left(\frac{4\xi^{(u)}\xi^{(v)}}{(\xi^{(u)} + \xi^{(v)})^2} \right)^{L/4} \exp\left(-\frac{\|\boldsymbol{u} - \boldsymbol{v}\|^2}{2(\xi^{(u)} + \xi^{(v)})} \right) \in (0, 1], \end{aligned} \tag{4.8}$$

where $\langle \cdot, \cdot \rangle_{L^2}$ and $\| \cdot \|_{L^2}$ are the inner product and norm of the L^2 space (i.e., the space of square integrable functions). See Appendix A for the derivation of (4.8). Under the selection strategy (4.7) and the L^2-coherence criterion (4.8), the novelty test for the Gaussian $g(\cdot; \xi_{\text{init}}^{(q)}, \boldsymbol{u}_n)$ is given as follows:

$$\max_{j \in \mathcal{J}_n} \left| c(\xi_n^{(j)}, \boldsymbol{c}_n^{(j)}, \xi_{\text{init}}^{(q)}, \boldsymbol{u}_n) \right| \leq \delta, \tag{4.9}$$

where $\delta \in [0, 1]$ is the prespecified threshold. If the condition in (4.9) is satisfied, the similarity between $g(\cdot; \xi_{\text{init}}^{(q)}, \boldsymbol{u}_n)$ and the existing dictionary atoms is sufficiently low and therefore $g(\cdot; \xi_{\text{init}}^{(q)}, \boldsymbol{u}_n)$ is regarded to be novel. The complexity issue will be discussed in Section 4.4.4.

The error and novelty tests share its underlying philosophy with Platt's criteria which checks the estimation error and the Euclidean distance between the current input vector and its closest center of Gaussian in the dictionary.

4.3.2 Updates of Heights, Scales, and Centers of Gaussian

The heights $\boldsymbol{h}_n$, scales $\boldsymbol{\xi}_n$, and centers $\boldsymbol{C}_n$ of the Gaussians are updated in a sequence. To reduce the computational costs, the selection strategy (4.7) presented in Section 4.3.1 is applied to the updates of the scales and centers. We denote by $s_n^{(\xi)}$ and $s_n^{(c)}$ the sizes of the selected subsets for $\boldsymbol{\xi}_n$ and $\boldsymbol{C}_n$, respectively.

Update of the heights with dictionary pruning

We employ the APFBS algorithm [60] for the cost function in (4.3) which is a superposition of the smooth fidelity function J_n and the nonsmooth regularizer $\tau\Omega_n$. Although F_n and Ω_n are defined for the variable vectors and matrix compatible with the dictionary of size r_n, we keep the same

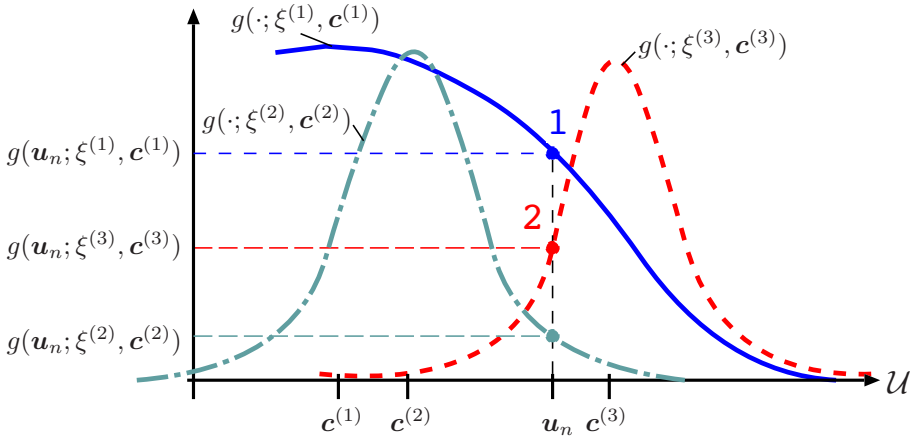

Figure 4.2: The selection strategy for $r_n = 3$ (three Gaussians) and $s_n^{(NC)} = 2$. The numbers 1 and 2 denote the priority. In this illustration, $g(\cdot;\xi^{(1)},\boldsymbol{c}^{(1)})$ and $g(\cdot;\xi^{(3)},\boldsymbol{c}^{(3)})$ are selected. The unselected Gaussian $g(\cdot;\xi^{(2)},\boldsymbol{c}^{(2)})$ is not tested for the sake of computational efficiency.

notation to denote the same functions for the resized variables in accordance with the dictionary growing and pruning.

Step 1 (Height update including dictionary pruning): Let $\lambda_h > 0$ be the stepsize parameter.

0. Switch $\boldsymbol{h}_n$, $\boldsymbol{\xi}_n$, and $\boldsymbol{C}_n$ to the possibly augmented counterparts $\hat{\boldsymbol{h}}_n \in \mathbb{R}^{\hat{r}_n}$, $\hat{\boldsymbol{\xi}}_n \in \mathbb{R}^{\hat{r}_n}_{++}$, and $\hat{\boldsymbol{C}}_n \in \mathbb{R}^{L\times\hat{r}_n}$, where $\hat{r}_n$ is either $r_n + 1$ or r_n depending on whether the dictionary grows or not (see Section 4.3.1).

1. Update the Gaussian coefficients by

$$\boldsymbol{h}_{n+1} := T\left[\mathrm{prox}_{\lambda_h\tau\Omega_n}\left(\hat{\boldsymbol{h}}_n - \lambda_h\frac{\partial J_n\left(\hat{\boldsymbol{h}}_n,\hat{\boldsymbol{\xi}}_n,\hat{\boldsymbol{C}}_n\right)}{\partial\hat{\boldsymbol{h}}}\right)\right] \in \mathbb{R}^{r_{n+1}}, \tag{4.10}$$

where

- $\mathrm{prox}_{\lambda_h\tau\Omega_n}(\boldsymbol{h})$ is the proximity operator of which the jth output can be computed as

$$[\mathrm{prox}_{\lambda_h\tau\Omega_n}(\boldsymbol{h})]_j = \max\left\{\left|h^{(j)}\right| - \lambda_h\tau w^{(j)}, 0\right\}\mathrm{sign}(h^{(j)}). \tag{4.11}$$

The dictionary is resized accordingly by discarding those Gaussian functions associated with the zero components.

The partial differential in (4.10) is given by

$$\frac{\partial J_n\left(\hat{\boldsymbol{h}}_n, \hat{\boldsymbol{\xi}}_n, \hat{\boldsymbol{C}}_n\right)}{\partial \hat{\boldsymbol{h}}} = -e_n\left(\hat{\boldsymbol{h}}_n, \hat{\boldsymbol{\xi}}_n, \hat{\boldsymbol{C}}_n\right)\boldsymbol{g}_n, \tag{4.12}$$

where $\boldsymbol{g}_n := [g(\boldsymbol{u}_n; \hat{\xi}_n^{(1)}, \hat{\boldsymbol{c}}_n^{(1)}), \cdots, g(\boldsymbol{u}_n; \hat{\xi}_n^{(\hat{r}_n)}, \hat{\boldsymbol{c}}_n^{(\hat{r}_n)})]^{\mathsf{T}} \in \mathbb{R}^{\hat{r}_n}$ and $e_n\left(\hat{\boldsymbol{h}}, \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{C}}\right) := d_n - \sum_{j=1}^{\hat{r}_n} \hat{h}^{(j)} g(\boldsymbol{u}_n; \hat{\xi}^{(j)}, \hat{\boldsymbol{c}}^{(j)})$, where $\hat{h}^{(j)}$, $\hat{\xi}^{(j)}$, and $\hat{\boldsymbol{c}}^{(j)}$ are the jth components and column of $\hat{\boldsymbol{h}}$, $\hat{\boldsymbol{\xi}}$, and $\hat{\boldsymbol{C}}$, respectively. The same notation e_n will be used to denote the same instantaneous error function for the resized variables compatible with the size-r_{n+1} dictionary.

Update of the scales

To update the scale parameters over the set $\mathbb{R}_{++}$, we derive the multiplicative gradient update for the cost function in (4.3).

Step 2 (Scale update): Let $\lambda_\xi^{(j)} > 0$ be the stepsize parameter.

0. Switch $\hat{\boldsymbol{\xi}}_n$ and $\hat{\boldsymbol{C}}_n$ again to its resized counterparts $\check{\boldsymbol{\xi}}_n \in \mathbb{R}_{++}^{r_{n+1}}$ and $\check{\boldsymbol{C}}_n \in \mathbb{R}^{L \times r_{n+1}}$, respectively, compatible with the downsized dictionary after pruning.

1. Select the index set $\{j_1, j_2, \cdots, j_{s_n^{(\xi)}}\}$ by (4.7) from the renewed dictionary $\{g(\cdot; \check{\xi}_n^{(j)}, \check{\boldsymbol{c}}_n^{(j)})\}_{j=1}^{r_{n+1}}$.

2. Update the scales of the selected Gaussians in a pseudo-code style as follows:[1]
   for $i = 1 : s_n^{(\xi)}$

$$\check{\xi}_n^{(j_i)} \leftarrow \check{\xi}_n^{(j_i)} \exp\left(-\lambda_\xi^{(j_i)} \check{\xi}_n^{(j_i)} \frac{\partial J_n\left(\boldsymbol{h}_{n+1}, \check{\boldsymbol{\xi}}_n, \check{\boldsymbol{C}}_n\right)}{\partial \check{\xi}^{(j_i)}}\right) \tag{4.13}$$

   end

   Note that the updated scale $\check{\xi}_n^{(j_i)}$ will be used to evaluate the partial differential in (4.13) for updating its subsequent scales. The same applies to Step 3 (center update) in Section 4.3.2.

3. $\boldsymbol{\xi}_{n+1} \leftarrow \check{\boldsymbol{\xi}}_n$.

[1] The notation $a \leftarrow b$ in the pseudo code means "substitute b to a".

The partial differential in (4.13) is given by

$$\frac{\partial J_n\left(\boldsymbol{h}_{n+1}, \check{\boldsymbol{\xi}}_n, \check{\boldsymbol{C}}_n\right)}{\partial \check{\xi}^{(j)}} = -\frac{e_n\left(\boldsymbol{h}_{n+1}, \check{\boldsymbol{\xi}}_n, \check{\boldsymbol{C}}_n\right) h_{n+1}^{(j)} \left\|\boldsymbol{u}_n - \check{\boldsymbol{c}}_n^{(j)}\right\|^2 g(\boldsymbol{u}_n; \check{\xi}_n^{(j)}, \check{\boldsymbol{c}}_n^{(j)})}{2(\check{\xi}_n^{(j)})^2}. \tag{4.14}$$

The multiplicative update (4.13) together with (4.14) is derived as follows. To ensure the strict positivity of $\check{\xi}^{(j)}$, we change the variable $\check{\xi}^{(j)}$ into $\check{\eta}^{(j)} := \log \check{\xi}^{(j)} \in \mathbb{R}$ which can take any real number. Then, the gradient update for the corresponding parameter $\check{\eta}_n^{(j)} := \log \check{\xi}_n^{(j)}$ is given, in a pseudo-code style, as

$$\begin{aligned}\check{\eta}_n^{(j)} \leftarrow & \check{\eta}_n^{(j)} - \lambda_\xi^{(j)} \frac{\partial J_n\left(\boldsymbol{h}_{n+1}, \check{\boldsymbol{\xi}}_n, \check{\boldsymbol{C}}_n\right)}{\partial \check{\eta}^{(j)}} \\ &= \check{\eta}_n^{(j)} - \lambda_\xi^{(j)} \check{\xi}_n^{(j)} \frac{\partial J_n\left(\boldsymbol{h}_{n+1}, \check{\boldsymbol{\xi}}_n, \check{\boldsymbol{C}}_n\right)}{\partial \check{\xi}^{(j)}},\end{aligned} \tag{4.15}$$

where the equality is due to $\partial J_n/\partial \check{\eta}^{(j)} = (\partial J_n/\partial \check{\xi}^{(j)}) \times (\partial \check{\xi}^{(j)}/\partial \check{\eta}(j)) = \check{\xi}^{(j)} \partial J_n/\partial \check{\xi}^{(j)}$. Operating the inverse map $\exp(\cdot)$ of the logarithmic function to the both sides of (4.15) yields (4.13).

Update of $\check{\boldsymbol{C}}_n$

For the Gaussian centers $\check{\boldsymbol{c}}_n^{(j)}$, we employ the standard gradient descent update.

Step 3 (Center update): Let $\lambda_c^{(j)} > 0$ be the stepsize parameter.

1. Select the index $\{j_1, j_2, \cdots, j_{s_n^{(c)}}\}$ by (4.7) from the renewed dictionary $\{g(\cdot; \xi_{n+1}^{(j)}, \check{\boldsymbol{c}}_n^{(j)})\}_{j=1}^{r_{n+1}}$.
2. Update the centers of the selected Gaussians as follows:
   for $i = 1 : s_n^{(c)}$

$$\check{\boldsymbol{c}}_n^{(j_i)} \leftarrow \check{\boldsymbol{c}}_n^{(j_i)} - \lambda_c^{(j_i)} \frac{\partial J_n\left(\boldsymbol{h}_{n+1}, \boldsymbol{\xi}_{n+1}, \check{\boldsymbol{C}}_n\right)}{\partial \check{\boldsymbol{c}}^{(j_i)}} \tag{4.16}$$

   end
3. $\boldsymbol{C}_{n+1} \leftarrow \check{\boldsymbol{C}}_n,\ j = 1, \cdots, r_{n+1}$.

The partial differential in (4.16) is given by

$$\frac{\partial J_n\left(\boldsymbol{h}_{n+1}, \boldsymbol{\xi}_{n+1}, \check{\boldsymbol{C}}_n\right)}{\partial \check{\boldsymbol{c}}^{(j)}} = -\frac{e_n\left(\boldsymbol{h}_{n+1}, \boldsymbol{\xi}_{n+1}, \check{\boldsymbol{C}}_n\right) h_{n+1}^{(j)} g(\boldsymbol{u}_n; \xi_{n+1}^{(j)}, \check{\boldsymbol{c}}_n^{(j)})(\boldsymbol{u}_n - \check{\boldsymbol{c}}_n^{(j)})}{\xi_{n+1}^{(j)}}. \tag{4.17}$$

Remark 4.1 *The order of the updates (heights, widths, and centers) affects the performance of the algorithm and the estimates obtained. If the update of heights results in a sufficiently small error, the adaptations of widths and centers stagnate and no efficient estimation is obtained. The order of the proposed method however yields efficient estimates, since even if the update of heights reduces the error, the soft thresholding operator reproduces a small error and creates a room for the adaptations of the widths and centers. The numerical experiments in Section 4.5 show the efficacy of the proposed algorithm.*

Remark 4.2 *The selective update strategy of ONEGAP is related to the set-membership approach [90, 91], which updates the estimate only when the current error is sufficiently large for reducing the computational complexity of the parameter update. Some kernel adaptive filtering algorithms based on the set-membership approach have been proposed [92, 93, 94]. Although the proposed selection strategy shares the same motivation, the criterion for selecting the Gaussians to be updated differs significantly from that of the set-membership approach: the proposed selection strategy checks the values of Gaussians at the current input $\boldsymbol{u}_n$ (see Figure 2). In the proposed selection strategy, moreover, the number of the Gaussians to be updated can be designed by the user.*

4.4 Discussions

4.4.1 Monotone Decreasing Property of Cost Function

ONEGAP alternates the (proximal) gradient updates for the Gaussian coefficients $\boldsymbol{h}_n$, scales $\boldsymbol{\xi}_n$, and centers $\boldsymbol{C}_n$. The standard analysis of the proximal gradient algorithm can be applied with (local) Lipschitz continuity of the function. Here, given a metric space $(X, \mathrm{d}(\cdot,\cdot))$, a mapping $T : X \to X$ is said to be locally Lipschitz continuous on a subset $C \subset X$ if, for any pair $(x, y) \in C \times C$, $\mathrm{d}(Tx, Ty) \leq \gamma \mathrm{d}(x, y)$ for some constant $\gamma \geq 0$ [82]. If in particular $C = X$, T is Lipschitz continuous.

For simplicity, we introduce the following shorthand notation to express J_n as a function of a specific entry $\xi^{(j)}$ of $\boldsymbol{\xi}$:

$$J_n^{(\xi^{(j)})}(\xi^{(j)}) := J_n(\boldsymbol{h}_{n+1}, \boldsymbol{\xi}, \check{\boldsymbol{C}}_n)\Big|_{\xi^{(i)} = \check{\xi}_n^{(i)},\ i \neq j}. \tag{4.18}$$

Note here that all the variables excluding $\xi^{(j)}$ are fixed to the up-to-date values. Likewise, define

$$J_n^{(c^{(j)})}(\boldsymbol{c}^{(j)}) := J_n(\boldsymbol{h}_{n+1}, \boldsymbol{\xi}_{n+1}, \boldsymbol{C})\Big|_{\boldsymbol{c}^{(i)} = \check{\boldsymbol{c}}_n^{(i)},\ i \neq j}. \tag{4.19}$$

As J_n is quadratic in $\boldsymbol{h}$, $\frac{\partial J_n}{\partial \boldsymbol{h}}$ is clearly Lipschitz continuous with constant $\gamma_n^{(h)} = \|\boldsymbol{g}_n\|^2$. As the multiplicative update of $\check{\xi}_n^{(j_i)}$ in (4.13) is derived from the (additive) gradient update of $\check{\eta}_n^{(j_i)} (:= \log \check{\xi}_n^{(j_i)})$, we consider the local Lipschitz continuity of $\frac{\partial J_n^{(\xi^{(j)})}}{\partial \eta^{(j)}}$, $\eta^{(j)} := \log \xi^{(j)}$.[2] The (local) Lipschitz continuity of $\frac{\partial J_n^{(\xi^{(j)})}}{\partial \eta^{(j)}}$ and $\frac{\partial J_n^{(\boldsymbol{c}^{(j)})}}{\partial \boldsymbol{c}^{(j)}}$ is given below.

Lemma 4.1

1. *The partial derivative* $\frac{\partial J_n^{(\xi^{(j)})}}{\partial \eta^{(j)}}$ *is locally Lipschitz on* $[t, +\infty)$, $t \in \mathbb{R}$, *with constant*

$$\gamma_{j,n}^{(\eta)}(t) := \frac{\left|h_{n+1}^{(j)}\right| \left(\left|\hat{d}_n^{(j)}\right| + \left|h_{n+1}^{(j)}\right|\right)}{2} \left\|\boldsymbol{u}_n - \check{\boldsymbol{c}}_n^{(j)}\right\|^2 \mathrm{e}^{-t}, \tag{4.20}$$

where $\hat{d}_n^{(j)} := d_n - \sum_{i \neq j} h_{n+1}^{(i)} \exp\left(-\frac{\left\|\boldsymbol{u}_n - \check{\boldsymbol{c}}_n^{(i)}\right\|^2}{2\check{\xi}_n^{(i)}}\right)$.

2. *The partial derivative* $\frac{\partial J_n^{(\boldsymbol{c}^{(j)})}}{\partial \boldsymbol{c}^{(j)}}$ *is Lipschitz continuous with constant*

$$\gamma_{j,n}^{(c)} = \delta_{j,n}^* \frac{\left|h_{n+1}^{(j)}\right|}{\xi_{n+1}^{(j)}} \left(\left|\check{d}_n^{(j)}\right| + \delta_{j,n}^* \left|h_{n+1}^{(j)}\right|\right) \leq \frac{\left|h_{n+1}^{(j)}\right|}{\xi_{n+1}^{(j)}} \left(\left|\check{d}_n^{(j)}\right| + \left|h_{n+1}^{(j)}\right|\right), \tag{4.21}$$

where

$$\delta_{j,n}^* := \max_{i=1,2,\cdots,L} \delta_{j,n}^{(i)} \in (0, 1] \tag{4.22}$$

with $\delta_{j,n}^{(i)} := \exp\left(-\frac{\sum_{k \neq i}(u_n^{(k)} - \check{c}_n^{(k)})^2}{2\xi_{n+1}^{(j)}}\right) \in (0, 1]$ *and* $\check{d}_n^{(j)} := d_n - \sum_{i \neq j} h_{n+1}^{(i)} \exp\left(-\frac{\left\|\boldsymbol{u}_n - \check{\boldsymbol{c}}_n^{(i)}\right\|^2}{2\xi_{n+1}^{(i)}}\right)$.

proof: See Appendices B and C.

Inspecting (4.20), we can see that the local Lipschitz constant $\gamma_{j,n}^{(\eta)}(t)$ decreases monotonically in t, meaning that the stepsize range allowed for

[2]The multiplicative update can also be updated with the mirror descent update with negative entropy [95], as mentioned in [96].

updating the scale parameters becomes narrower as t decreases. This implies that the stepsize bound must be smaller than $2/\gamma_{j,n}^{(\eta)}(t)$ when $\check{\eta}_n^{(j)}$ is updated to a smaller value (i.e., when its corresponding gradient is positive valued). A question of theoretical interest here is the following: under the standard stepsize range $(0, 2/\gamma_{j,n}^{(\eta)}(t))$, what is the maximal possible $t > 0$ for which the cost function is locally Lipschitz with constant $\gamma_{j,n}^{(\eta)}(t)$ at the current estimate $\check{\eta}_n^{(j)}$ both before and after the update, so that the monotone decreasing property is ensured. Such a t is clearly the updated estimate (which is smaller than before the update) when the gradient is positive. To ensure the local Lipschitz continuity for all possible stepsizes within the range $(0, 2/\gamma_{j,n}^{(\eta)}(t))$, we consider the following equation:

$$t = \check{\eta}_n^{(j)} - \frac{2}{\gamma_{j,n}^{(\eta)}(t)} \frac{\partial J_n^{(\xi^{(j)})}(\check{\xi}_n^{(j)})}{\partial \eta^{(j)}}. \tag{4.23}$$

We can now present the monotone decreasing property.

Theorem 4.1 *Let* $\lambda_h \in \left(0, \frac{2}{\gamma_n^{(h)}}\right)$, $\lambda_\xi^{(j)} \in \left(0, \frac{2}{\gamma_{j,n}^{(\eta)}(t_n^{(j)})}\right)$, *and* $\lambda_c^{(j)} \in \left(0, \frac{2}{\gamma_{j,n}^{(c)}}\right)$ *for*[3]

$$t_n^{(j)} = \begin{cases} \check{\eta}_n^{(j)}, & \dfrac{\partial J_n(\hat{\boldsymbol{h}}_{n+1}, \check{\boldsymbol{\xi}}_n, \check{\boldsymbol{C}}_n)}{\partial \eta^{(j)}} \leq 0 \\ \tilde{t}_n^{(j)}, & \dfrac{\partial J_n(\hat{\boldsymbol{h}}_{n+1}, \check{\boldsymbol{\xi}}_n, \check{\boldsymbol{C}}_n)}{\partial \eta^{(j)}} > 0, \end{cases} \tag{4.24}$$

where $\tilde{t}_n^{(j)}$ *is a unique solution of* (4.23)*. Then, after each iteration, it holds that*

$$J_n\left(\boldsymbol{h}_n, \boldsymbol{\xi}_n, \boldsymbol{C}_n\right) - J_n\left(\boldsymbol{h}_{n+1}, \boldsymbol{\xi}_{n+1}, \boldsymbol{C}_{n+1}\right) \geq 0. \tag{4.25}$$

proof: The claims for λ_h *and* $\lambda_c^{(j)}$ *are verified directly by applying the monotone decreasing properties of the (proximal) gradient descent [62, 97] in light of Lemma 4.1. In the rest, we verify the stepsize range of* $\lambda_\xi^{(j)}$*. If* $\frac{\partial J_n^{(\xi^{(j)})}(\check{\xi}_n^{(j)})}{\partial \eta^{(j)}} \leq 0$*, the estimate* $\check{\eta}_n^{(j)}$ *increases after the gradient update, and therefore the (local) Lipschitz constant* $\gamma_{j,n}^{(\eta)}(\check{\eta}_n^{(j)})$ *is valid over* $[\check{\eta}_n^{(j)}, \infty)$ *in which the updated* $\check{\eta}_n^{(j)}$ *lies. If, on the other hand,* $\frac{\partial J_n^{(\xi^{(j)})}(\check{\xi}_n^{(j)})}{\partial \eta^{(j)}} > 0$*,* $\check{\eta}_n^{(j)}$ *decreases after the update in (4.13), and therefore the maximal* t *ensuring the local Lipschitz continuity is characterized by* (4.23)*. In this case,*

[3] In (4.24), $\check{\eta}_n^{(j)}$ is the parameter before update, as it cannot be used to update the $\check{\eta}_n^{(j)}$ itself otherwise.

(4.23) *has a unique solution since* $f(t) := t - \left(\breve{\eta}_n^{(j)} - \frac{2}{\gamma_{j,n}^{(\eta)}(t)} \frac{\partial J_n^{(\xi^{(j)})}(\tilde{\xi}_n^{(j)})}{\partial \eta^{(j)}} \right)$ *is continuous and monotonically increasing with* $\lim_{t \to +\infty} f(t) = +\infty$ *and* $\lim_{t \to -\infty} f(t) = -\infty$.

Equation (4.23) has no closed form solution, and an iterative method needs to be used to find the $\tilde{t}_n^{(j)}$. This is unfavorable in online estimation. We therefore present efficient designs of the stepsize parameters based on Theorem 4.1 without solving (4.23) explicitly in Section 4.4.3.

4.4.2 On Global-to-Local Order of Multiscale Screening

We discuss the global-to-local (large-to-small scale) order of the proposed multiscale screening method. To find an economic way of expressing the unknown function ψ with our Gaussian model given in (4.2), the appropriate centers and scales of Gaussian need to be known. This is certainly unrealistic in online scenarios in which the amount of available data is rather limited especially at the early phase of estimation. The local structures need to be expressed with delicate adjustments of center points (as well as scale), and thus small-scale Gaussians are more sensitive to the mismatch of the center position than large-scale ones. This is one of the reasons for the global-to-local order.

Another reason comes from the characteristics of the data-fidelity function $J_n(\boldsymbol{h}, \boldsymbol{\xi}, \boldsymbol{C})$ in (4.4). As pointed out in Section 4.3.1, the learning speeds of ONEGAP become slow when the initial scales are far from the ones of the target due to the gradient vanishment. The sole use of an undesirably-large initial scale tends to yield an underfitting estimate, since the corresponding Gaussian does not fit the nonlinear function ψ. In contrast, the sole use of an undesirably-small initial scale tends to yield an overfitting estimate and it also causes an explosion of the dictionary size, since the learning algorithm seeks to express every detail of ψ with a peaky Gaussian individually. From the current perspective of the authors, this is caused mainly by the gradient vanishment issue mentioned above, but nevertheless we cannot deny the possibility of falling into some local minima. The goal of the present study is to build an adaptive algorithm which generates an efficient approximation of ψ, and the use of small initial scale, especially at the early learning-phrase, is therefore not recommended from this efficiency aspect. The proposed global-to-local strategy works quite well in practice.

4.4.3 Parameter Design

The parameters δ, ϵ, and Q control the tradeoff between the computational complexity and the performance of the algorithm, and users can design those parameters for each application. As a rule of thumb, the larger the parameters δ, ϵ, and Q, the larger the maximal dictionary size. Although the

use of the large dictionary tends to yield fast convergence and low MSEs, this may cause also an explosion of the computational complexity. Empirically, setting $Q = 3$ gives a reasonable performances without an explosion of the computational complexity. See Section 4.4.4 for more details about the computational complexity. Empirically, setting the selection parameters $s_n^{(NC)}, s_n^{(\xi)}, s_n^{(c)}$ from 3 to 7 gives a reasonably low computational complexity (see Section 4.5). The parameter β in the weight of the ℓ_1 norm (see (4.4)) is the regularization parameter to avoid division by zero, and it is thus set to some small value such as 10^{-4}.

The stepsize parameters λ_h, $\lambda_\xi^{(j)}$, and $\lambda_c^{(j)}$ and the regularization parameter λ affect the accuracy of the final estimate as well as the convergence speed, and thus it needs to be carefully designed. In particular, the stepsizes $\lambda_\xi^{(j)}$ and $\mu_c^{(j)}$ as well as λ govern the dictionary size and thus the efficiency of the final estimate. We present efficient designs of $\lambda_\xi^{(j)}$ and $\lambda_c^{(j)}$ in the following subsections.

Design of $\lambda_c^{(j)}$

To ensure the monotone decreasing property in Theorem 4.1, an appropriate stepsize depends on the Lipschitz constant $\gamma_{j,n}^{(c)}$. Unfortunately, $\gamma_{j,n}^{(c)}$ in (4.21) is defined with $\xi_{n+1}^{(j)}$, $\check{d}_n$, and $h_{n+1}^{(j)}$, which are unavailable for designing $\lambda_c^{(j)}$ prior to adaptation as no prior knowledge is assumed available about the structure of the target system (see Section 4.2). Fortunately, the initial scale $\xi_{\text{init}}^{(q)}$ could be used as an alternative of $\xi_{n+1}^{(j)}$, since the current $\xi_{n+1}^{(j)}$ is expected to be closer to $\xi_{\text{init}}^{(q)}$ than (at least most of) the others $\xi_{\text{init}}^{(\tilde{q})}, \tilde{q} = 1, \cdots, q-1, q+1, \cdots, Q$. Replacing $\xi_{n+1}^{(j)}$ by $\xi_{\text{init}}^{(q)}$, $\gamma_{j,n}^{(c)}$ is inversely proportional to $\xi_{\text{init}}^{(q)}$, and an appropriate stepsize is thus proportional to $\xi_{\text{init}}^{(q)}$. Based on the above discussion, below is a design scheme for the stepsize $\lambda_c^{(j)}$.

Example 4.1 (Design scheme for $\lambda_c^{(j)}$) *Set $\lambda_c > 0$. For the Gaussians initialized by $\xi_{\text{init}}^{(1)}$, set the stepsize to $\lambda_c^{(j)} = \lambda_c$. For the Gaussians initialized by each $\xi_{\text{init}}^{(q)}$, $q = 2, \cdots, Q$, set the stepsize to $\lambda_c^{(j)} = \frac{\xi_{\text{init}}^{(q)}}{\xi_{\text{init}}^{(1)}} \lambda_c$.*

Design of $\lambda_\xi^{(j)}$

In contrast to $\lambda_c^{(j)}$, one can employ the same value for all $\lambda_\xi^{(j)}$ due to the following reason. Since $\gamma_{j,n}^{(\xi)}(t)$ is monotonically increasing in $\left\| \boldsymbol{u}_n - \check{\boldsymbol{c}}_n^{(j)} \right\|^2$ with $\lim_{\left\| \boldsymbol{u}_n - \check{\boldsymbol{c}}_n^{(j)} \right\|^2 \to \infty} \gamma_{j,n}^{(\xi)}(t) = \infty$, we need to set an upper bound for $\left\| \boldsymbol{u}_n - \check{\boldsymbol{c}}_n^{(j)} \right\|^2$ to design $\lambda_\xi^{(j)}$ based on $\gamma_{j,n}^{(\xi)}(t)$. Meanwhile, a large $\left\| \boldsymbol{u}_n - \check{\boldsymbol{c}}_n^{(j)} \right\|^2$

implies a small output of the Gaussian $\exp\left(-\frac{\left\|\boldsymbol{u}_n-\check{\boldsymbol{c}}_n^{(j)}\right\|^2}{2\check{\xi}_n^{(j)}}\right)$. This means that the Gaussian gives a negligible impact on the estimation in the vicinity of $\boldsymbol{u}_n$ when $\left\|\boldsymbol{u}_n-\check{\boldsymbol{c}}_n^{(j)}\right\|^2$ is sufficiently large. Due to the above discussions, we now make an upper bound as $\left\|\boldsymbol{u}_n-\check{\boldsymbol{c}}_n^{(j)}\right\|^2 \leq -2\check{\xi}_n^{(j)}\log a$ for some small constant $0 < a \ll 1$ so that $\exp\left(-\frac{\left\|\boldsymbol{u}_n-\check{\boldsymbol{c}}_n^{(j)}\right\|^2}{2\check{\xi}_n^{(j)}}\right) \geq a$. In the same way as the design scheme for $\lambda_c^{(j)}$, we replace $\check{\xi}_n^{(j)}$ in (4.20) by the initial scale $\xi_{\text{init}}^{(q)}$ as in the design of $\lambda_c^{(j)}$ with $t := \log \xi_{\text{init}}^{(q)}$ (see also (4.24)). Substituting $\left\|\boldsymbol{u}_n-\check{\boldsymbol{c}}_n^{(j)}\right\|^2 = -2\xi_{\text{init}}^{(q)}\log a$ to (4.20), we obtain $\gamma_{j,n}^{(\xi)}(\eta_{\text{init}}^{(q)})|_{\left\|\boldsymbol{u}_n-\check{\boldsymbol{c}}_n^{(i)}\right\|^2=-2\xi_{\text{init}}^{(q)}\log a} = -\log a\left|h_{n+1}^{(j)}\right|\left(\left|\hat{d}_n^{(j)}\right|+\left|h_{n+1}^{(j)}\right|\right)$ which no longer depends on $\xi_{\text{init}}^{(q)}$. It is thus reasonable to use the same stepsizes for all js.

Design of Initial Gaussian Scales

If the components of the target function were known, one could select the initial Gaussian scales $\xi_{\text{init}}^{(q)}$ that are appropriate for those components. In many applications, however, the target components are unknown prior to estimation. In such a case, one may want to use a set of Gaussians with a variety of scales which are regular in a certain sense. As the set of inflection points of each Gaussian forms a hypersphere of radius $\sqrt{\xi^{(j)}}$, the idea is to place the hyperspheres in a regular fashion. We present a selection example of the initial Gaussian scales.

Example 4.2 (Initial Gaussian scales) *The user sets the largest and smallest scales $\xi_{\text{init}}^{(1)}$ and $\xi_{\text{init}}^{(Q)}$ to some appropriate values. The other scales are then set to $\xi_{\text{init}}^{(q)} := \sqrt{\xi_{\text{init}}^{(1)}} - (q-1)\frac{\sqrt{\xi_{\text{init}}^{(1)}}-\sqrt{\xi_{\text{init}}^{(Q)}}}{Q-1}$, $q = 2, \cdots, Q-1$.*

The one-dimensional case is illustrated in Figure 4.3. Here, the scales $\xi_{\text{init}}^{(2)}$ and $\xi_{\text{init}}^{(3)}$ are determined so that the inflection points $\sqrt{\xi_{\text{init}}^{(q)}}$ of all Gaussians $g(\cdot;\xi_{\text{init}}^{(q)},0)$, $q = 1,2,3,4$, are equally spaced in the interval $\left[\sqrt{\xi_{\text{init}}^{(4)}}, \sqrt{\xi_{\text{init}}^{(1)}}\right]$. The parameter design scheme presented in Example 4.2 works well in practice as shown in Section 4.5.

4.4.4 Computational Complexity

The computational complexity of ONEGAP at each time instant n is generally given in terms of the dictionary size r_n as well as the input dimension

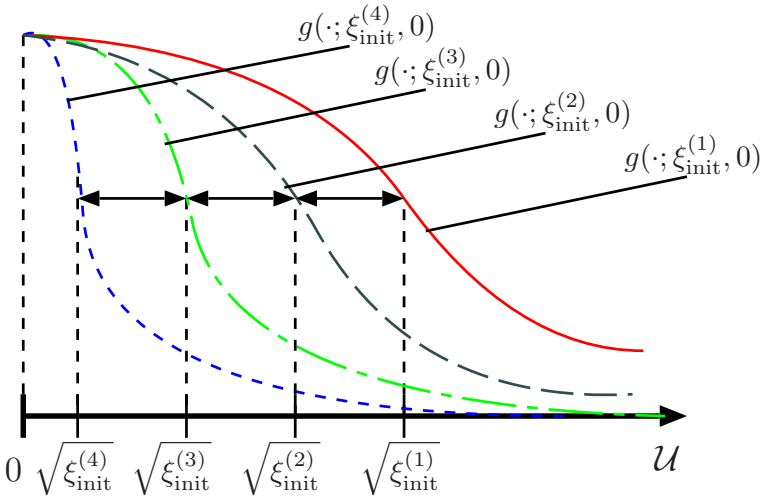

Figure 4.3: An example of initial Gaussian scales for $Q = 4$.

L. The computational complexity of ONEGAP depends also on $s^{(\text{NC})}$, $s^{(\xi)}$, $s^{(c)}$, and Q which are supposed to be constant during the adaptation.

Table 4.1 summarizes the overall complexities (the number of real multiplications) per iteration of ONEGAP and the related algorithms. Here, the KNLMS algorithm [27], which is a kernelized version of the normalized least mean square [31] algorithm, is a benchmark algorithm in the kernel adaptive filtering. The quantized kernel least mean square algorithm with adaptive kernel size (QKLMS-AKS) [53] and the kernel algorithm with adaptive width (KAW) [98] are kernel adaptive filtering algorithms which adapt the Gaussian scales. The complexity contains all the multiplications required at each time n including those for dictionary growing and parameter updating. For ONEGAP , the case of the non-selective update (i.e., $s^{(\text{NC})} = s^{(\xi)} = s^{(c)} = r_n$) is also considered to show the effectiveness of the selective update.

Figure 4.4.4 illustrates the complexities as a function of the dictionary size r_n for $L = 6$ and $Q = 3$. The figure shows that the complexity of ONEGAP ($s^{(\text{NC})} = 3$, $s^{(\xi)} = s^{(c)} = 5$) is lower than those of ONEGAP (non-selective update) and RAN. This is due to the selection strategy for dictionary growing and parameter updating. Although ONEGAP ($s^{(\text{NC})} = 3$, $s^{(\xi)} = s^{(c)} = 5$) requires the slightly higher complexity than QKLMS-AKS, it enjoys significant gains in MSE, as shown in the next section.

Table 4.1: Computational complexities of the proposed and related algorithms.

ONEGAP	$(L+5)r_n + 8s^{(\xi)} + (2L+4)s^{(c)} + Q(L/4+4)s^{(\mathrm{NC})}$
QKLMS-AKS	$r_n(L+2)$
KAW	$r_n(L+4) + (r_n-1)^2$
RAN	$r_n(2L+6)$
KNLMS	$r_n(L+3)$
NLMS	$2L$

4.5 Simulation Results

We show the efficacy of ONEGAP for system identification problems with two sets of synthetic data and time-series prediction problems with two benchmark data. For ONEGAP , the dictionaries are constructed by the multiscale screening method presented in Section 4.3.1. For the weighted ℓ_1 norm, $w_n^{(j)} = \frac{1}{\left|h_n^{(j)}\right|+\beta}$ [99] with $\beta = 10^{-4}$ is employed. The numbers of checked/updated Gaussians are set to $s_n^{(\mathrm{NC})} = s_n^{(\xi)} = s_n^{(c)} = 5$ in Experiment 1 and 2, and $s_n^{(\mathrm{NC})} = s_n^{(\xi)} = s_n^{(c)} = 4$ in Experiment 3, for all $n \in \mathbb{N}$. We remark that those parameters require any strict tuning to obtain reasonable performances for ONEGAP. The stepsizes and the regularization parameter require, on the other hand, strict tuning to control the performance or the dictionary sizes of ONEGAP. The detailed tunings of those parameters are presented in each of experiments.

Experiment 1: Effectiveness of the Adaptation of Gaussian Scales and Centers

We consider the following nonlinear function $\psi(u) = \exp\left(-\frac{\|\boldsymbol{u}-0.75\mathbf{1}\|^2}{2\xi_*^{(1)}}\right) - 3\exp\left(-\frac{\|\boldsymbol{u}-1.5\mathbf{1}\|^2}{2\xi_*^{(2)}}\right) + 2\exp\left(-\frac{\|\boldsymbol{u}-2.25\mathbf{1}\|^2}{2\xi_*^{(3)}}\right)$, $\boldsymbol{u} \in \mathbb{R}^5$, which is the sum of three Gaussian functions with $\xi_*^{(1)} = 1$, $\xi_*^{(2)} = 5$, and $\xi_*^{(3)} = 0.25$, where $\mathbf{1} = [1, \cdots, 1]^\mathsf{T} \in \mathbb{R}^5$. The observed signal is generated as $d_n := \psi(\boldsymbol{u}_n) + v_n$, $n \in \mathbb{N}$, where $\boldsymbol{u}_n$ is the input data of which each element is randomly generated from a uniform distribution over $[0,3]^5$ and $v_n \sim \mathcal{N}(0, 1.0 \times 10^{-2})$ is the additive white Gaussian noise.

To show that ONEGAP adapts the Gaussian scale and center efficiently, the performance of ONEGAP is compared with the performance of ONEGAP without the adaptation of the Gaussian scales ξ and centers $\boldsymbol{c}$ ($\lambda_\xi^{(j)} = \lambda_c^{(j)} = 0$). For ONEGAP , the initial Gaussian scales are selected according

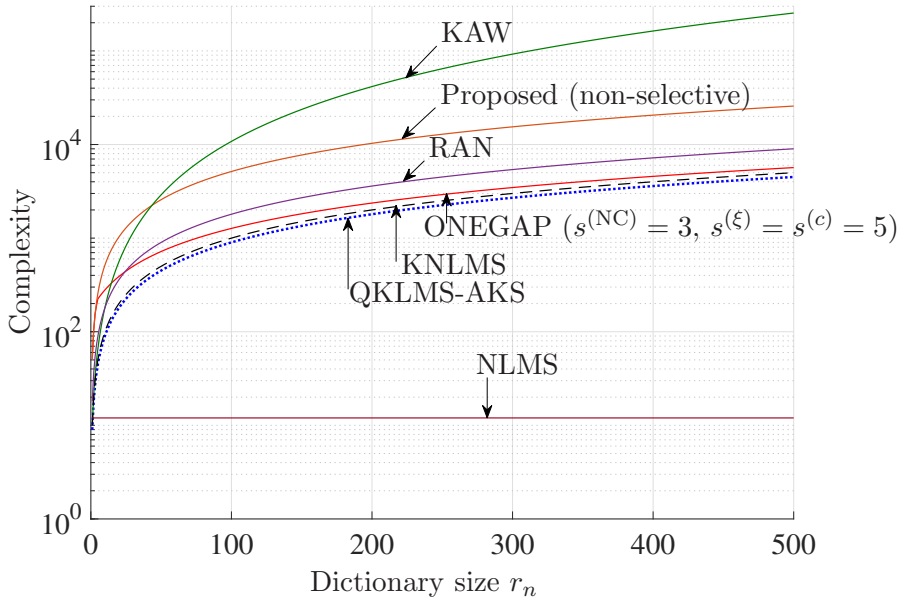

Figure 4.4: Computational complexities of the proposed and related algorithms.

to Example 4.2 with $\xi_{\mathrm{init}}^{(1)} = 10^{0.5}$ and $\xi_{\mathrm{init}}^{(3)} = 10^{-0.5}$; the stepsizes $\lambda_c^{(j)}$ for the Gaussian scales are according to Example 4.1 with $\lambda_c = 0.1$; and the other stepsizes are set to $\lambda_h = 0.1$ and $\lambda_\xi^{(j)} = 0.1$. The regularization parameter $\tau = 10^{-3}$ and the parameters of the multiscale screening method are chosen so that the dictionary size is close to the number of Gaussians contained in the target and the MSE becomes as low as possible at the steady state. To show the effectiveness of the design scheme for $\lambda_c^{(j)}$, we test the performance of ONEGAP with the constant stepsizes $\lambda_c^{(j)} = 0.05,\ j = 1, \cdots, r_{n+1}$. For ONEGAP ($\lambda_\xi^{(j)} = \lambda_c^{(j)} = 0$), $\xi_{\mathrm{init}}^{(1)} = 2.5$, $\xi_{\mathrm{init}}^{(2)} = 0.75$, and $\xi_{\mathrm{init}}^{(3)} = 0.1$ are chosen so that the algorithm achieves the best performance. We empirically found that the use of a large number of Gaussians with small scales yields small errors when adequate Gaussian parameters are unknown. Even if the scales of the target function is known, the estimation errors may become large when the centers are located at inadequate positions. For the algorithm ($\lambda_\xi^{(j)} = \lambda_c^{(j)} = 0$), the best parameters are chosen such that the maximal dictionary size is as close as possible to that of ONEGAP .

Figures 4.5, 4.6, and 4.7 depict the normalized minimum differences of the Gaussian scales between the target function and the estimate, i.e., $\min_{j=1,\cdots,r_n} \frac{\left|\xi_*^{(i)} - \xi_n^{(j)}\right|}{\xi_*^{(i)}}$, $i = 1, 2, 3$, the MSE, and the dictionary size, respectively. All results are averaged over 200 runs.

From Figure 4.5, one can see that the errors of the Gaussian scales are

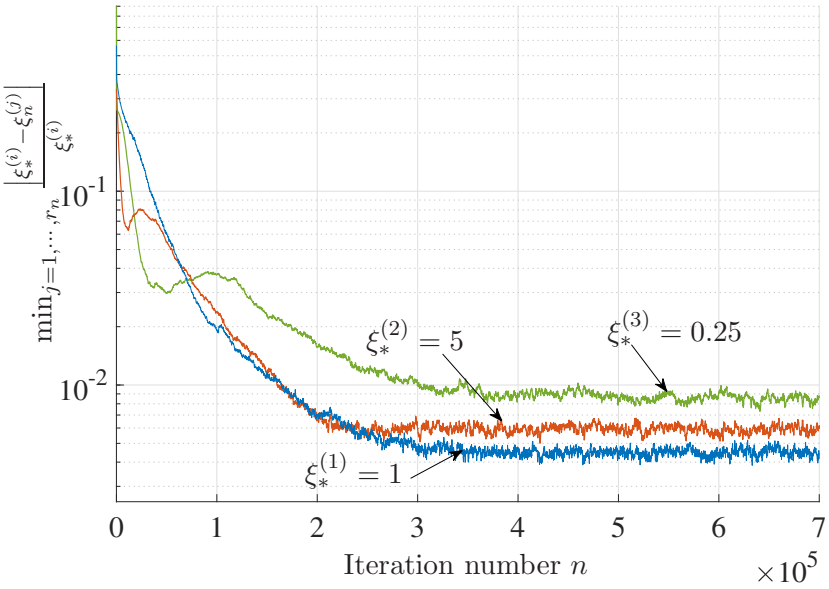

Figure 4.5: Results of Experiment 1: the minimum difference of the Gaussian scales between the target function and the atoms.

reasonably small. Furthermore, Figures 4.6 and 4.7 show that the dictionary size of ONEGAP at the steady state is nearly identical to the number of Gaussians of the target function, and also ONEGAP is superior to the algorithm ($\lambda_\xi^{(j)} = \lambda_c^{(j)} = 0$) in the sense of MSE. ONEGAP achieves the considerably small steady-state MSE (-41 dB) which is 20 dB lower than the algorithm ($\lambda_\xi^{(j)} = \lambda_c^{(j)} = 0$), thanks to the adaptations of the Gaussian scales and centers. The algorithm (const. $\lambda_c^{(j)}$) requires many iterations to reach the steady-state MSE. The proposed design scheme for $\lambda_c^{(j)}$ enables to use adequate stepsizes for each of Gaussians and consequently ONEGAP achieves the fast convergence. These results show that ONEGAP quickly yields the 'efficient' estimates by adapting the Gaussian scales and centers with adequate stepsizes $\lambda_c^{(j)}$, although ONEGAP has no guarantee to yield perfectly efficient estimates.

Experiment 2: Effectiveness of the Multiscale Screening

To show the effectiveness of the multiscale screening method presented in Section 4.3.1, estimation performances of ONEGAPs are studied for such functions that consist of extremely large and small Gaussians. Specifically,

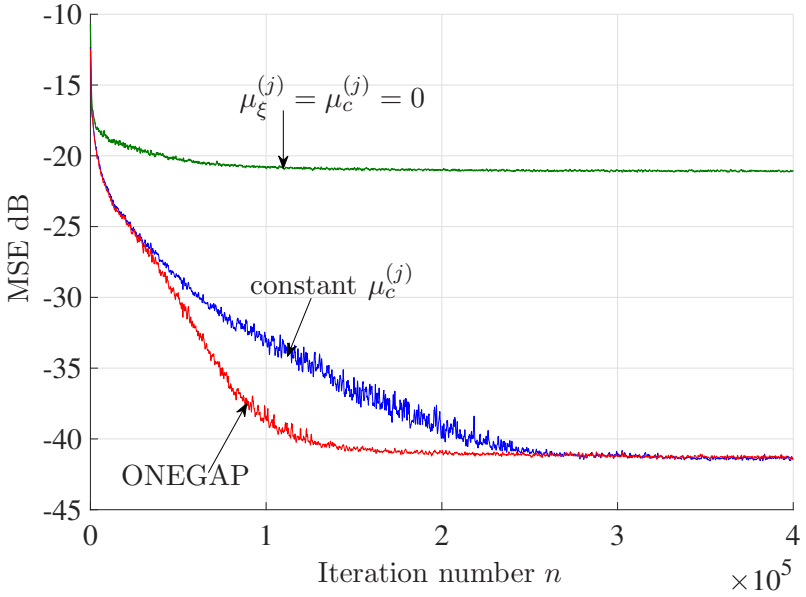

Figure 4.6: Results of Experiment 1: MSEs.

we consider the nonlinear function

$$\psi(\boldsymbol{u}) = \sum_i^5 h_*^{(i)} \exp\left(-\frac{\left\|\boldsymbol{u} - \boldsymbol{c}^{(i)}\right\|^2}{2\xi_*^{(i)}}\right),\ \boldsymbol{u} \in \mathbb{R}^3. \tag{4.26}$$

Here $\boldsymbol{c}^{(i)} \in \mathbb{R}^3$ is generated from a uniform distribution over $[0,1]^3$, $h^{(1)} = 1$, $h^{(i)} = 5,\ i = 2, \cdots, 5$, and $\xi_*^{(i)}$ is generated as $\xi_*^{(i)} = \left|\tilde{\xi_*^{(i)}}\right|$ with

$$\tilde{\xi_*^{(i)}} \sim \begin{cases} \mathcal{N}(100, 10.0),\ i = 1 \\ \mathcal{N}(1 \times 10^{-2}, 5.0 \times 10^{-3}),\ i = 2, \cdots, 5. \end{cases} \tag{4.27}$$

The observed signal is generated as $d_n := \psi(\boldsymbol{u}_n) + v_n,\ n \in \mathbb{N}$, where $\boldsymbol{u}_n$ is the input data of which each element is randomly generated from a uniform distribution over $[0,1]^3$ and $v_n \sim \mathcal{N}(0, 1.0 \times 10^{-2})$ is the additive white Gaussian noise. ONEGAP is tested with (i) the multiple initial scales ($Q = 3$) and (ii) the single initial scale ($Q = 1$). The results are averaged over 500 independent trials.

For ONEGAP, the initial Gaussian scales are selected according to Example 4.2 with $\xi_{\text{init}}^{(1)} = 10^2$ and $\xi_{\text{init}}^{(3)} = 10^{-2}$; the stepsizes $\lambda_c^{(j)}$ for the Gaussian scales are according to Example 4.1 with $\lambda_c = 0.1$; and the other stepsizes are set to $\lambda_h = 0.1$ and $\lambda_\xi^{(j)} = 0.1$. The other stepsizes are set to $\lambda_h = 0.1$ and $\lambda_\xi^{(j)} = 0.1,\ j = 1, \cdots, r_{n+1}$. The regularization parameter and

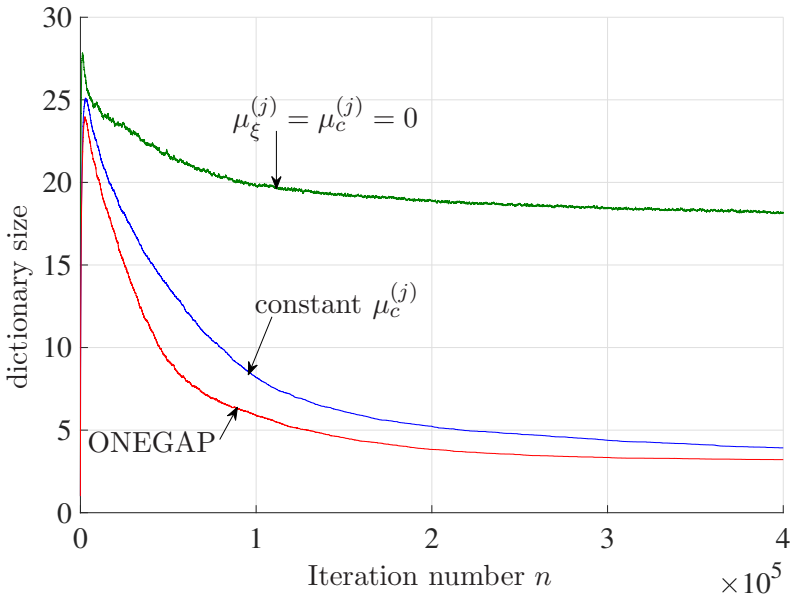

Figure 4.7: Results of Experiment 1: dictionary sizes.

the parameters of the multiscale screening method are chosen in the same way as in Experiment 1.

For ONEGAP ($Q = 1$), the two settings for the initial scales are tested: the large initial-scale $\xi_{\text{init}} = \xi_{\text{init}}^{(1)} = 10^2$ and the small initial-scale $\xi_{\text{init}} = \xi_{\text{init}}^{(3)} = 10^{-2}$. For the small initial-scale, we consider two cases: (i) small dictionary for the same maximal dictionary size, and (ii) large dictionary for the same steady-state MSE, as ONEGAP . For the large initial-scale $\xi_{\text{init}} = 10^2$, the parameters are selected so that the lowest MSE is attained.

Figures 4.8 and 4.9 illustrate the results in terms of the MSEs and the dictionary sizes, respectively. In Figure 4.9, one can see that the algorithms with $Q = 1$ yield notably high MSE for $\xi_{\text{init}} = 10^2$ and $\xi_{\text{init}} = 10^{-2}$ (small dic.). This is because, with the extremely large initial Gaussian scale $\xi_{\text{init}} = 10^2$, the adaptation of the Gaussian scales tends to stop at large scales, failing to capture fine fluctuations caused by small scale Gaussians. In contrast, in the extremely small initial Gaussian scale $\xi_{\text{init}} = 10^{-2}$, the adaptation tends to stop at small scales, failing to capture the global structure of the target. Although the MSEs of $\xi_{\text{init}} = 10^{-2}$ (large dic.) is reasonably small due to the use of the large number of small Gaussians in the initial phase of estimation, the maximal dictionary size is unacceptably large and the redundant Gaussians remain for a while as seen in Figure 4.9. These results are due to the gradient vanishment caused by the nonconvexity of the cost function as pointed out in Sections 4.3.1 and 4.4.2. ONEGAP attains the efficient estimates, preventing from the sharp rise of the dictionary size. This

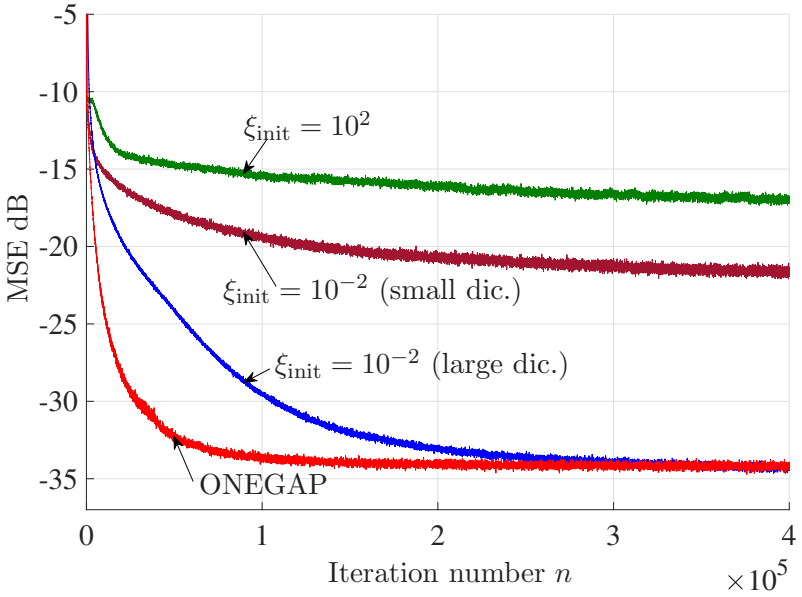

Figure 4.8: Results of Experiment 2: MSEs.

shows the effectiveness the global-to-local order of the multiscale screening method.

Experiment 3: Application to Prediction of Real and Synthetic Time-Series Data

We demonstrate the performance of ONEGAP in application to online predictions of (a) the laser data [84] from SantaFe dataset and (b) the sequence generated by Mackey-Glass equation [100].[4] Each datum d_n is predicted with $\boldsymbol{u}_n := [d_{n-1}, d_{n-2}, \cdots, d_{n-L}]^\mathsf{T} \in \mathcal{U} \subset \mathbb{R}^L$ for $L = 6$.

ONEGAP is compared with the following algorithms: (i) NLMS, (ii) RAN (a benchmark algorithm in RBF network field), (iii) the state-of-the-art algorithm [101] for online time-series estimation which is based on LSTM neural network architecture, (iv) the state-of-the-art nonlinear estimation algorithms that adapt the Gaussian scales with single initial values: QKLMS-AKS [53] and KAW [98], the linear Kalman filter with the state-space model (3.59). In contrast to ONEGAP which discards redundant atoms from the dictionary by the ℓ_1 norm regularization, QKLMS-AKS has no structure to discard the atoms, i.e., the dictionary size of QKLMS-AKS increases monotonically. Unlike ONEGAP and QKLMS-AKS, KAW fixes the dictionary size at some predefined values. To be more precise, the dictionary grows at

[4]The sequence is generated by $\frac{dx_n}{dn} = -bx_n + \frac{ax_{n-t}}{1+x_{n-t}^{10}}$. with $b = 0.1$, $a = 0.2$ and time delay $t = 30$

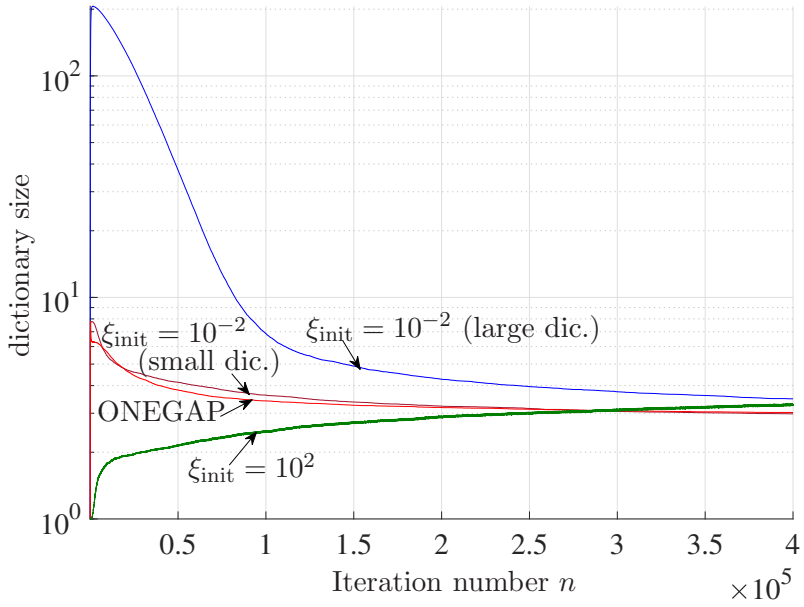

Figure 4.9: Results of Experiment 2: dictionary sizes.

every iteration until the dictionary size reaches the predefined value. If the dictionary size exceeds the predefined value, one atom is discarded from the dictionary. For ONEGAP , the initial Gaussian scales are determined by Example 4.2 with $\xi_{\text{init}}^{(1)} = 10$ and $Q = 3$. Figures 4.10 and 4.11 depict the learning curves for (a) Santa-Fe dataset and (b) Mackey-Glass data, respectively. Here, (a) 500- and (b) 200- point moving averages of the results are taken, respectively. Table 4.2 summarizes the squared errors, computational complexities, and maximal dictionary sizes averaged over all iterations. The complexities of ONEGAP , RAN, QKLMS-AKS, and KAW are shown in Table 4.1 with the dictionary sizes averaged over iterations. The complexities of NLMS and LSTM are counted as $2L$ and $N(m^2 + mp)$, respectively, where N, m, p are the numbers of particles, output nodes, and input nodes of the network, respectively. The dictionary size of QKLMS-AKS is selected so that the complexity of QKLMS-AKS is approximately the same as that of ONEGAP. The dictionary sizes of KAW and RAN is selected so that the maximal dictionary sizes of KAW and RAN are approximately the same as ONEGAP . The particle number N of LSTM is selected so as to attain the same complexity as KAW, which requires the largest complexity in these algorithms. Note that the dictionary sizes of all algorithms change dynamically.

Figures 4.10 and 4.11, and Table 4.2 show that the MSEs of ONEGAP are smaller than those of the others for both data. Furthermore, it can be seen from Table 4.2 that ONEGAP requires a lower complexity than

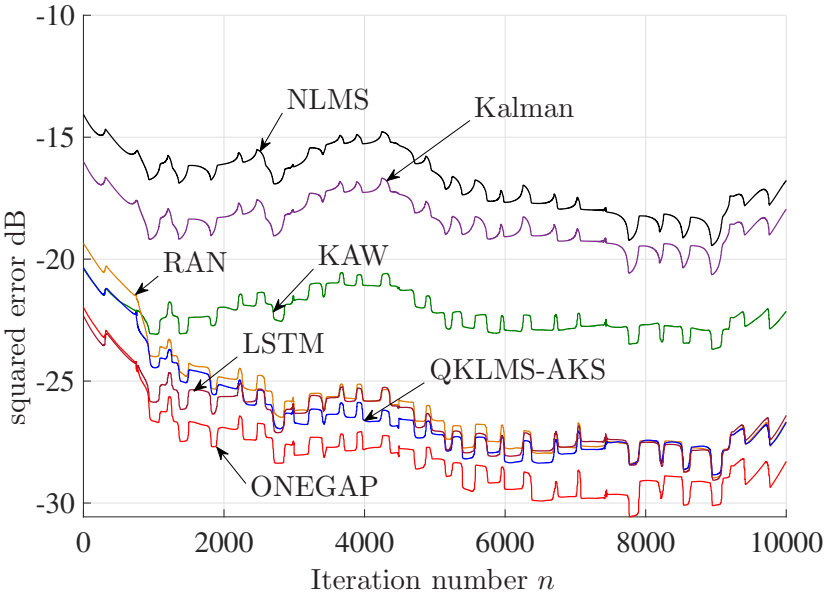

Figure 4.10: Learning curves of Experiment 3: (a) laser data from SantaFe data set.

KAW, RAN, and LSTM. The performance of NLMS and the Klamn filter are limited since its model is linear and is thus inadequate for the nonlinear time-series prediction. Again, the performance of KAW is inferior to the other nonlinear algorithms due to the use of the same Gaussian scales for all atoms. Note that KAW and LSTM may achieve lower MSEs, but with high complexity, if a larger-sized dictionary and many particles are used, respectively.

4.6 Conclusion

We proposed a learning algorithm, named ONEGAP, which adapts the model parameters, as well as the coefficients, of a weighted sum of the Gaussians. ONEGAP consisted of two steps: the dictionary growing step and the parameters updating step. In the dictionary growing step, a novel multiple initialization scheme was presented as a remedy for the gradient vanishing problem without serious increases of the dictionary size. In the parameter updating step, the Gaussian parameters were updated as well as the coefficients by the proximal gradient based algorithm. Due to the use of the ℓ_1 norm regularization, the model efficiency was enhanced. Thanks to the selection strategy for dictionary growing and scale/center updating, the complexity of ONEGAP was reasonably low. Computer simulations for the toy examples showed that ONEGAP successfully attains efficient estimates.

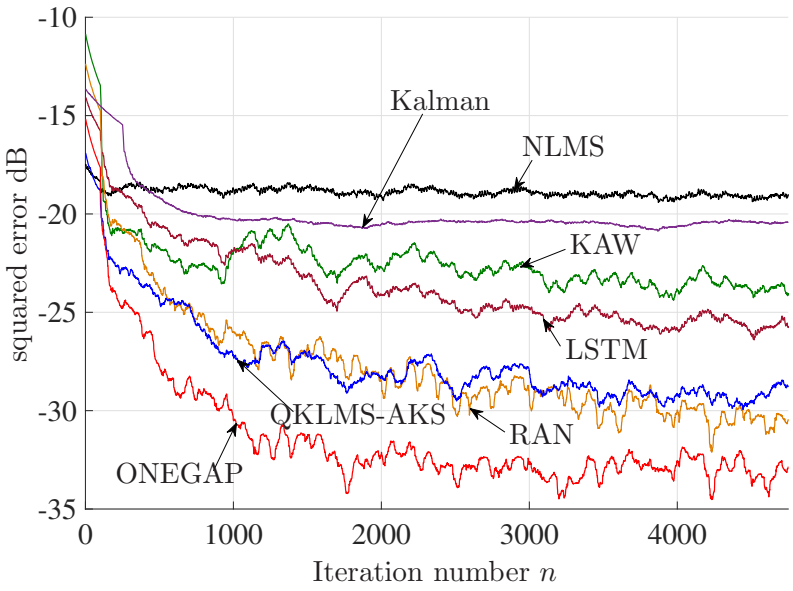

Figure 4.11: Learning curves of Experiment 3: (b) Mackey-Glass equation.

In application to the time-series data predictions, ONEGAP achieved approximately 4.7 dB lower MSE than the state-of-the-art online prediction algorithm.

Table 4.2: Results of Experiment 3.

		squared error dB	complexity	dictionary size (max)
ONEGAP	(a) Mackey-Glass	**−28.4**	339	18
	(b) SantaFe	**−27.37**	355	38
QKLMS -AKS [53]	(a) Mackey-Glass	−25.8	382	52
	(b) SantaFe	−26.5	365	49
KAW [98]	(a) Mackey-Glass	−22.1	2901	50
	(b) SantaFe	−21.5	2485	46
RAN [85]	(a) Mackey-Glass	−25.1	579	54
	(b) SantaFe	−25.0	697	51
LSTM [101]	(a) Mackey-Glass	−26.2	3040	-
	(b) SantaFe	−22.7	2560	-
NLMS [31]	(a) Mackey-Glass	−16.2	12	-
	(b) SantaFe	−17.9	12	-
Kalman	(a) Mackey-Glass	−18.2	96	-
	(b) SantaFe	−20.9	96	-

Chapter 5

General Conclusion

This thesis was devoted to algorithms for online nonlinear estimation. The proposed algorithms addressed the following three issues of conventional KAF algorithms: (i) convergence speed of algorithms, (ii) increase of the dictionary size, and (iii) the design of kernel parameters.

First, we proposed an efficient kernel adaptive filtering algorithm, called the Φ-PASS algorithm. The key ideas were projection-along-subspace, selective update, and parallel projection. The projection-along-subspace systematically eliminated the limitation of updates, yielding excellent performances with small dictionary sizes. The selective update effectively reduced the complexity without any serious degradation of performance, as justified by a geometric interpretation. The parallel projection yielded fast convergence/tracking accompanied by noise robustness. Numerical examples demonstrated the benefits from the three ideas as well as the advantages of the proposed algorithm over the state-of-the-art algorithms.

Second, we proposed the DR-Φ-PASS algorithm. DR-Φ-PASS was basically the APFBS method, but it performed the gradient step with the dictionary-dependent inner product and the proximal step with the canonical inner product. It was shown that the sequence of coefficient vectors generated by the proposed algorithm monotonically approaches, in the $\boldsymbol{G}_n$-norm sense, the set of minimizers of the weighted squared-distance cost function penalized by the ℓ_1 norm with the modified weights. The numerical examples showed that the proposed algorithm achieved effective dictionary refinements as well as fast convergence/tracking under possible data-nonstationarity with low computational complexity.

Third, we proposed an online nonlinear estimation algorithm, named ONEGAP, which adapted the Gaussian parameters (scales and centers) as well as the coefficients to obtain efficient estimates. ONEGAP consisted of two steps: the dictionary growing step and the parameters updating step. In the dictionary growing step, a novel multiple initialization scheme was presented as a remedy for the gradient vanishing problem without increasing the

dictionary size undesirably. In the parameter updating step, the Gaussian parameters and the coefficients were updated by the proximal gradient-based algorithm. Due to the use of the ℓ_1 norm regularization, the efficiency of the model enhanced. The Proposed algorithm enjoyed the reasonable computational complexity thanks to the selection strategy for dictionary growing and scale/center updating.

Future Work: The final goal of this study is constructing online nonlinear estimation algorithms that have the following desirable properties: (a) guarantee of convergence, (b) obtainability of an efficient model, and (c) parameter-free nature (including hyper-parameters).

A more in-depth analysis is required to learn the behavior and the performance of the proposed algorithms. Of particular importance is the convergence analysis and the parameter design of ONEGAP, of which no convergence result is presented in this thesis. A certain notion of convergence of ONEGAP would be guaranteed based on the knowledge of the nonconvex optimization in future work.

Our future work moreover concerns (i) applications of the proposed algorithms to online nonlinear estimation tasks, (ii) applications of the proposed multiscale screening method to other nonlinear estimation methods with Gaussian functions, and (iii) construction of a fast algorithm to track time-varying systems with the adaptations of model parameters. The details are given below:

- Many problems in signal processing and machine learning can be cast as online estimations of unknown nonlinear functions. The main focus of this thesis is on the algorithm itself. Although the efficacy of the proposed algorithms is demonstrated in the applications for time-series data predictions, more tests and experiments are required to show the effectiveness of the proposed algorithms in a range of fields.

- Designing Gaussian parameters has been recognized as a critical issue to be resolved [102] not only in the field of KAF but also RBF network, Gaussian process, and so on. I would like to validate the performance of the proposed multiscale screening method and show its efficacy, applying it to other online nonlinear estimation algorithms.

- Estimating time-varying systems is an important issue in many fields. It is expected that ONEGAP yields reasonable performances by adapting the Gaussian parameters to track the change of systems. In Chapter 4, time-invariant systems are assumed to focus on obtaining efficient estimates of the target, and no performance verification is presented for time-varying systems. Further experiments are required to construct a fast algorithm that tracks time-varying systems.

I believe that the outcomes provided by this study will contribute to developments for a wide range of applications in a variety of fields.

Appendix A

Sketch of the Derivation of (4.8)

The inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ between the two normalized Gaussians under the uniform distribution with infinite interval is given as [79]

$$\left\langle \tilde{g}(\cdot; \xi^{(u)}, \boldsymbol{u}), \tilde{g}(\cdot; \xi^{(v)}, \boldsymbol{v}) \right\rangle_{\mathcal{H}} = \frac{1}{\left(2\pi(\xi^{(u)} + \xi^{(v)})\right)^{L/2}} \exp\left(-\frac{\|\boldsymbol{u} - \boldsymbol{v}\|^2}{2(\xi^{(u)} + \xi^{(v)})} \right), \tag{A.1}$$

where $\tilde{g}(\cdot; \xi, \boldsymbol{c}) := \frac{1}{(2\pi\xi)^{L/2}} \exp\left(-\frac{\|\cdot - \boldsymbol{c}\|^2}{2\xi} \right)$, $\boldsymbol{c} \in \mathbb{R}^L$, and $\xi > 0$. Meanwhile, the inner product can be written as

$$\left\langle \tilde{g}(\cdot; \xi^{(u)}, \boldsymbol{u}), \tilde{g}(\cdot; \xi^{(v)}, \boldsymbol{v}) \right\rangle_{\mathcal{H}} = \frac{1}{\left(2\pi\sqrt{\xi^{(u)}\xi^{(v)}}\right)^{L}} \left\langle g(\cdot; \xi^{(u)}, \boldsymbol{u}), g(\cdot; \xi^{(v)}, \boldsymbol{v}) \right\rangle_{\mathcal{H}}. \tag{A.2}$$

We can also easily verify the followings:

$$\left\langle g(\cdot; \xi^{(u)}, \boldsymbol{u}), g(\cdot; \xi^{(v)}, \boldsymbol{v}) \right\rangle_{\mathcal{H}} = \left(2\pi \frac{\xi^{(u)}\xi^{(v)}}{\xi^{(u)} + \xi^{(v)}} \right)^{L/2} \exp\left(-\frac{\|\boldsymbol{u} - \boldsymbol{v}\|^2}{2(\xi^{(u)} + \xi^{(v)})} \right) \tag{A.3}$$

and

$$\left\| g(\cdot; \xi^{(u)}, \boldsymbol{u}) \right\|_{\mathcal{H}} = \sqrt{\left\langle g(\cdot; \xi^{(u)}, \boldsymbol{u}), g(\cdot; \xi^{(u)}, \boldsymbol{u}) \right\rangle_{\mathcal{H}}} = \left(\pi \xi^{(u)} \right)^{L/4}. \tag{A.4}$$

By using the above inner product and the norm, the coherence (4.8) is obtained.

Appendix B

Proof of (4.20)

For brevity, we let $\rho_n^{(j)} := \left\| \boldsymbol{u}_n - \check{\boldsymbol{c}}_n^{(j)} \right\|^2 \geq 0$. The function $J_n^{(\xi^{(j)})}$ of $\xi^{(j)}$ can then be written as

$$J_n^{(\xi^{(j)})}\left(\xi^{(j)}\right) = \frac{1}{2}\left(\hat{d}_n^{(j)} - h_{n+1}^{(j)} \exp\left(-\frac{\rho_n^{(j)}}{2\xi^{(j)}}\right)\right)^2. \tag{B.1}$$

Using the chain rule, the partial derivative of $J_n^{(\xi^{(j)})}$ with respect to $\eta^{(j)} = \log \xi^{(j)}$ is then given by

$$\frac{\partial J_n^{(\xi^{(j)})}\left(\xi^{(j)}\right)}{\partial \eta^{(j)}} = -\frac{h_{n+1}^{(j)} \rho_n^{(j)}}{2\xi^{(j)}} \exp\left(-\frac{\rho_n^{(j)}}{2\xi^{(j)}}\right)\left(\hat{d}_n^{(j)} - h_{n+1}^{(j)} \exp\left(-\frac{\rho_n^{(j)}}{2\xi^{(j)}}\right)\right). \tag{B.2}$$

For $\xi^{(j)}, \tilde{\xi}^{(j)} > 0$, one can verify by the triangle inequality that

$$\begin{aligned}
&\left|\frac{\partial J_n^{(\xi^{(j)})}(\xi^{(j)})}{\partial \eta^{(j)}} - \frac{\partial J_n^{(\xi^{(j)})}(\tilde{\xi}^{(j)})}{\partial \eta^{(j)}}\right| \\
&\leq \left|h_{n+1}^{(j)}\right| \left[\left|\hat{d}_n^{(j)}\right| \left|\frac{\rho_n^{(j)}}{2\xi^{(j)}} \exp\left(-\frac{\rho_n^{(j)}}{2\xi^{(j)}}\right) - \frac{\rho_n^{(j)}}{2\tilde{\xi}^{(j)}} \exp\left(-\frac{\rho_n^{(j)}}{2\tilde{\xi}^{(j)}}\right)\right| \right. \\
&\quad \left. + \left|h_{n+1}^{(j)}\right| \left|\frac{\rho_n^{(j)}}{2\xi^{(j)}} \exp\left(-\frac{\rho_n^{(j)}}{\xi^{(j)}}\right) - \frac{\rho_n^{(j)}}{2\tilde{\xi}^{(j)}} \exp\left(-\frac{\rho_n^{(j)}}{\tilde{\xi}^{(j)}}\right)\right|\right].
\end{aligned} \tag{B.3}$$

Letting $x := \frac{\rho_n^{(j)}}{2\xi^{(j)}}$ and $\tilde{x} := \frac{\rho_n^{(j)}}{2\tilde{\xi}^{(j)}} > 0$ in (B.3) yields

$$\begin{aligned}&\left|\frac{\partial J_n^{(\xi^{(j)})}(\xi^{(j)})}{\partial \eta^{(j)}} - \frac{\partial J_n^{(\xi^{(j)})}(\tilde{\xi}^{(j)})}{\partial \eta^{(j)}}\right| \\ &\leq \left|h_{n+1}^{(j)}\right| \left(\left|\hat{d}_n^{(j)}\right| \left|x\mathrm{e}^{-x} - \tilde{x}\mathrm{e}^{-\tilde{x}}\right| + \left|h_{n+1}^{(j)}\right| \left|x\mathrm{e}^{-2x} - \tilde{x}\mathrm{e}^{-2\tilde{x}}\right|\right).\end{aligned} \tag{B.4}$$

Considering the maximal magnitude of the gradient, the following inequalities are readily verified:

$$\left|x\mathrm{e}^{-ax} - \tilde{x}\mathrm{e}^{-a\tilde{x}}\right| \leq |x - \tilde{x}|, \ a > 0. \tag{B.5}$$

Combining (B.4) and (B.5), we obtain

$$\left|\frac{\partial J_n^{(\xi^{(j)})}(\xi^{(j)})}{\partial \eta^{(j)}} - \frac{\partial J_n^{(\xi^{(j)})}(\tilde{\xi}^{(j)})}{\partial \eta^{(j)}}\right| \leq \left|h_{n+1}^{(j)}\right| \left(\left|\hat{d}_n^{(j)}\right| + \left|h_{n+1}^{(j)}\right|\right) |x - \tilde{x}|. \tag{B.6}$$

On the other hand, by using $\xi^{(j)} = \mathrm{e}^{\eta^{(j)}}$, $|x - \tilde{x}|$ is rewritten as

$$|x - \tilde{x}| = \left|\frac{\rho_n^{(j)}}{2\xi^{(j)}} - \frac{\rho_n^{(j)}}{2\tilde{\xi}^{(j)}}\right| = \left|\frac{\rho_n^{(j)}}{2}\right| \left|\mathrm{e}^{-\eta^{(j)}} - \mathrm{e}^{-\tilde{\eta}^{(j)}}\right|. \tag{B.7}$$

Here, due to the convexity of $\mathrm{e}^{-\eta}, \eta \in \mathbb{R}$, one can verify that $\left|\mathrm{e}^{-\eta^{(j)}} - \mathrm{e}^{-\tilde{\eta}^{(j)}}\right| \leq \mathrm{e}^{-t}\left|\eta^{(j)} - \tilde{\eta}^{(j)}\right|, \ \forall \eta^{(j)}, \tilde{\eta}^{(j)} \geq t \in \mathbb{R}$, from which (B.7) leads to

$$|x - \tilde{x}| \leq \left|\frac{\rho_n^{(j)}}{2}\right| \mathrm{e}^{-t}\left|\eta^{(j)} - \tilde{\eta}^{(j)}\right|, \ \forall \eta^{(j)}, \tilde{\eta}^{(j)} \geq t. \tag{B.8}$$

Combining (B.6) and (B.8) yields

$$\begin{aligned}&\left|\frac{\partial J_n^{(\xi^{(j)})}(\xi^{(j)})}{\partial \eta^{(j)}} - \frac{\partial J_n^{(\xi^{(j)})}(\tilde{\xi}^{(j)})}{\partial \eta^{(j)}}\right| \\ &\leq \frac{\left|h_{n+1}^{(j)}\rho_n^{(j)}\right|}{2} \left(\left|\hat{d}_n^{(j)}\right| + \left|h_{n+1}^{(j)}\right|\right) \mathrm{e}^{-t}\left|\eta^{(j)} - \tilde{\eta}^{(j)}\right|, \ \forall \eta^{(j)}, \tilde{\eta}^{(j)} \geq t.\end{aligned}$$

Appendix C

Proof of (4.21)

We first prove the following lemma.

Lemma C.1 *For* $f(c) := c\exp\left(\frac{-c^2}{\xi}\right),\ c \in \mathbb{R},\ \xi > 0$, *the following inequality holds:*

$$|f(c) - f(\tilde{c})| \leq |c - \tilde{c}|\,. \tag{C.1}$$

The first and second derivatives of f *are given by* $f'(c) = \exp\left(\frac{-c^2}{\xi}\right) - \frac{2c^2}{\xi}\exp\left(\frac{-c^2}{\xi}\right)$ *and* $f''(c) = \frac{4c}{\xi^2}\left(-\frac{3}{2}\xi + c^2\right)\exp\left(\frac{-c^2}{\xi}\right)$, *respectively. By solving* $f''(c) = 0$, *we obtain the inflection points* $c = 0$ *and* $c = \pm\sqrt{\frac{3}{2}\xi}$ *of* f, *and at those points* f' *has the following values:* $f'(0) = -1$ *and* $f'\left(\pm\sqrt{\frac{3}{2}\xi}\right) = -2\mathrm{e}^{-\frac{3}{2}}$, *respectively. Since* $|f'(0)| > \left|f'\left(\pm\sqrt{\frac{3}{2}\xi}\right)\right|$ *and* $\lim_{c\to\pm\infty} f'(c) = 0$, *we obtain (C.1).*

Proof of (4.21): For brevity, we drop the time index n. We shall then prove the following inequality:

$$\left\|\frac{\partial J^{(\boldsymbol{c}^{(j)})}(\boldsymbol{c})}{\partial \boldsymbol{c}^{(j)}} - \frac{\partial J^{(\boldsymbol{c}^{(j)})}(\tilde{\boldsymbol{c}})}{\partial \boldsymbol{c}^{(j)}}\right\| \leq \delta^* \frac{\left|h^{(j)}\right|}{\xi^{(j)}}\left(\left|\check{d}^{(j)}\right| + \delta^*\left|h^{(j)}\right|\right)\|\boldsymbol{c} - \tilde{\boldsymbol{c}}\|\,, \quad \boldsymbol{c}, \tilde{\boldsymbol{c}} \in \mathbb{R}^L. \tag{C.2}$$

(Note that the inequality in (4.21) can readily be verified by $\delta^* \leq 1$.) The function $J^{(\boldsymbol{c}^{(j)})}$ of $\boldsymbol{c}^{(j)}$ can be written as

$$J^{(\boldsymbol{c}^{(j)})}\left(\boldsymbol{c}^{(j)}\right) = \frac{1}{2}\left(\check{d}^{(j)} - h^{(j)}\exp\left(-\frac{\left\|\boldsymbol{u} - \boldsymbol{c}^{(j)}\right\|^2}{2\xi^{(j)}}\right)\right)^2, \tag{C.3}$$

and the ith component its partial derivative is given as

$$\begin{aligned}&\left[\frac{\partial J^{(\boldsymbol{c}^{(j)})}(\boldsymbol{c})}{\partial \boldsymbol{c}^{(j)}}\right]_i \\ &= -\frac{h^{(j)}}{\xi^{(j)}}(u^{(i)}-c^{(i)})\left(\check{d}^{(j)} - h^{(j)}\exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{c}\|^2}{2\xi^{(j)}}\right)\right)\exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{c}\|^2}{2\xi^{(j)}}\right),\end{aligned} \tag{C.4}$$

where $u^{(i)}$ and $c^{(i)}$ denote the ith components of $\boldsymbol{u}$ and $\boldsymbol{c}$, respectively. By (C.4) and $\exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{c}\|^2}{2\xi^{(j)}}\right) = \delta^{(i)}\exp\left(-\frac{(u^{(i)}-c^{(i)})^2}{2\xi^{(j)}}\right)$, we can verify, for $\boldsymbol{c}, \tilde{\boldsymbol{c}} \in \mathbb{R}^L$, that

$$\begin{aligned}&\left|\left[\frac{\partial J^{(\boldsymbol{c}^{(j)})}(\boldsymbol{c})}{\partial \boldsymbol{c}^{(j)}}\right]_i - \left[\frac{\partial J^{(\boldsymbol{c}^{(j)})}(\tilde{\boldsymbol{c}})}{\partial \boldsymbol{c}^{(j)}}\right]_i\right| \\ &\le \frac{\left|h^{(j)}\check{d}^{(j)}\delta^{(i)}\right|}{\left|\xi^{(j)}\right|}\left|(u^{(i)}-c^{(i)})\exp\left(-\frac{(u^{(i)}-c^{(i)})^2}{2\xi^{(j)}}\right)\right. \\ &\quad \left.-(u^{(i)}-\tilde{c}^{(i)})\exp\left(-\frac{(u^{(i)}-\tilde{c}^{(i)})^2}{2\xi^{(j)}}\right)\right| \\ &\quad + \frac{\left|h^{(j)}\delta^{(i)}\right|^2}{\left|\xi^{(j)}\right|}\left|(u^{(i)}-c^{(i)})\exp\left(-\frac{(u^{(i)}-c^{(i)})^2}{\xi^{(j)}}\right)\right. \\ &\quad \left.-(u^{(i)}-\tilde{c}^{(i)})\exp\left(-\frac{(u^{(i)}-\tilde{c}^{(i)})^2}{\xi^{(j)}}\right)\right|.\end{aligned} \tag{C.5}$$

Direct applications of Lemma C.1 to the two terms in the right side of (C.5) for $c = (u^{(i)}-c^{(i)})$ and $c = \frac{(u^{(i)}-c^{(i)})}{\sqrt{2}}$ yields

$$\begin{aligned}&\left|\left[\frac{\partial J^{(\boldsymbol{c}^{(j)})}(\boldsymbol{c})}{\partial \boldsymbol{c}^{(j)}}\right]_i - \left[\frac{\partial J^{(\boldsymbol{c}^{(j)})}(\tilde{\boldsymbol{c}})}{\partial \boldsymbol{c}^{(j)}}\right]_i\right| \\ &\le \frac{\left|h^{(j)}\check{d}^{(j)}\delta^{(i)}\right|}{\left|\xi^{(j)}\right|}\left|c^{(i)}-\tilde{c}^{(i)}\right| + \frac{\left|h^{(j)}\delta^{(i)}\right|^2}{\left|\xi^{(j)}\right|}\left|c^{(i)}-\tilde{c}^{(i)}\right| \\ &= \frac{\delta^{(i)}\left|h^{(j)}\right|}{\xi^{(j)}}\left(\left|\check{d}^{(j)}\right| + \delta^{(i)}\left|h^{(j)}\right|\right)\left|c^{(i)}-\tilde{c}^{(i)}\right|.\end{aligned} \tag{C.6}$$

By (C.6), we finally obtain the following bound:

$$\begin{aligned}
&\left\| \frac{\partial J^{(\boldsymbol{c}^{(j)})}(\boldsymbol{c})}{\partial \boldsymbol{c}^{(j)}} - \frac{\partial J^{(\boldsymbol{c}^{(j)})}(\tilde{\boldsymbol{c}})}{\partial \boldsymbol{c}^{(j)}} \right\|^2 \\
&= \sum_{i=1}^{L} \left(\frac{\delta^{(i)} \left| h^{(j)} \right|}{\xi^{(j)}} \left(\left| \check{d}^{(j)} \right| + \delta^{(i)} \left| h^{(j)} \right| \right) \left| c^{(i)} - \tilde{c}^{(i)} \right| \right)^2 \\
&\leq \left(\delta^* \frac{\left| h^{(j)} \right|}{\xi^{(j)}} \left(\left| \check{d}^{(j)} \right| + \delta^* \left| h^{(j)} \right| \right) \right)^2 \sum_{i=1}^{L} \left| c^{(i)} - \tilde{c}^{(i)} \right|^2 \\
&= \left(\delta^* \frac{\left| h^{(j)} \right|}{\xi^{(j)}} \left(\left| \check{d}^{(j)} \right| + \delta^* \left| h^{(j)} \right| \right) \right)^2 \left\| \boldsymbol{c} - \tilde{\boldsymbol{c}} \right\|^2, \qquad \text{(C.7)}
\end{aligned}$$

where $\delta^* := \max_{i=1,\cdots,L} \delta^{(i)}$.

Bibliography

[1] Mohinder Grewal and Angus Andrews, "Kalman filtering: Theory and applications," Jan. 1985.

[2] Simon J. Julier and Jeffrey K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, Ivan Kadar, Ed. International Society for Optics and Photonics, 1997, vol. 3068, pp. 182 – 193, SPIE.

[3] Y. Shi, K. Sun, L. Huang, and Y. Li, "Online identification of permanent magnet flux based on extended Kalman filter for IPMSM drive with position sensorless control," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4169–4178, 2012.

[4] E. Laroche, E. Sedda, and C. Durieu, "Methodological insights for online estimation of induction motor parameters," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 5, pp. 1021–1028, 2008.

[5] E. Ghahremani and I. Kamwa, "Dynamic state estimation in power system by applying the extended Kalman filter with unknown inputs to phasor measurements," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2556–2566, 2011.

[6] E. Ghahremani and I. Kamwa, "Online state estimation of a synchronous generator using unscented Kalman filter from phasor measurements units," *IEEE Transactions on Energy Conversion*, vol. 26, no. 4, pp. 1099–1108, 2011.

[7] M. Partovibakhsh and G. Liu, "An adaptive unscented Kalman filtering approach for online estimation of model parameters and state-of-charge of lithium-ion batteries for autonomous mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 357–363, 2015.

[8] C. Wang, Z. Wang, L. Zhang, D. Cao, and D. G. Dorrell, "A vehicle rollover evaluation system based on enabling state and parameter estimation," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[9] X. Ding, Z. Wang, L. Zhang, and C. Wang, "Longitudinal vehicle speed estimation for four-wheel-independently-actuated electric vehicles based on multi-sensor fusion," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12797–12806, 2020.

[10] V. J. Mathews, "Adaptive polynomial filters," *IEEE Signal Processing Magazine*, vol. 8, no. 3, pp. 10–26, 1991.

[11] Li Tan and J. Jiang, "Adaptive volterra filters for active control of nonlinear noise processes," *IEEE Transactions on Signal Processing*, vol. 49, no. 8, pp. 1667–1676, 2001.

[12] Taiho Koh and E. Powers, "Second-order volterra filtering and its application to nonlinear system identification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 6, pp. 1445–1455, December 1985.

[13] H. Al-Duwaish, M. N. Karim, and V. Chandrasekar, "Use of multilayer feedforward neural networks in identification and control of wiener model," in *Proc. IEEE Control Theory Appl.*, 1996, vol. 143, pp. 255–258.

[14] Kurt Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251 – 257, 1991.

[15] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 76–86, 1992.

[16] Balázs Csanád Csáji et al., "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, no. 48, pp. 7, 2001.

[17] B. Schölkopf and A. J. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2001.

[18] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic, New York, 4th edition, 2008.

[19] Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*, The MIT Press, 2005.

[20] A. O'Hagan, "Curve fitting and optimal design for prediction," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 40, no. 1, pp. 1–42, 1978.

[21] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Processing*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[22] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[23] A. V. Malipatil, Y.-F. Huang, S. Andra, and K. Bennett, "Kernelized set-membership approach to nonlinear adaptive filtering," in *Proc. IEEE ICASSP*, 2005, pp. 149–152.

[24] W. Liu and J. Príncipe, "Kernel affine projection algorithms," *EURASIP J. Adv. Signal Process.*, vol. 2008, pp. 1–12, 2008.

[25] W. Liu, P. P. Pokharel, and J. C. Príncipe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Processing*, vol. 56, no. 2, pp. 543–554, Feb. 2008.

[26] K. Slavakis, S. Theodoridis, and I. Yamada, "Online kernel-based classification using adaptive projection algorithms," *IEEE Trans. Signal Processing*, vol. 56, no. 7, pp. 2781–2796, July 2008.

[27] C. Richard, J.-C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Processing*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[28] M. Yukawa, "Multikernel adaptive filtering," *IEEE Trans. Signal Processing*, vol. 60, no. 9, pp. 4672–4682, Sep. 2012.

[29] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Networks and Learning Systems*, vol. 23, no. 1, pp. 22–32, Dec. 2012.

[30] Steven Van Vaerenbergh, Miguel Lázaro-Gredilla, and Ignacio Santamaría, "Kernel recursive least-squares tracker for time-varying regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1313–1326, Aug. 2012.

[31] A. H. Sayed, *Adaptive Filters*, John Wiley & Sons, 2008.

[32] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, India, 1996.

[33] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering*, Wiley, New Jersey, 2010.

[34] W. Liu, I. Park, and J. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Network and Learning Systems*, vol. 20, no. 12, pp. 1950–1961, Dec. 2009.

[35] M. Yukawa, "On use of multiple kernels in adaptive learning — Extended reproducing kernel Hilbert space with Cartesian product," in *Proc. IEICE Signal Processing Symposium*, Nov. 2010, pp. 59–64.

[36] M. Yukawa, "Nonlinear adaptive filtering techniques with multiple kernels," in *Proc. EUSIPCO*, 2011, pp. 136–140.

[37] Michael Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer Publishing Company, 2010.

[38] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A survey of sparse representation: Algorithms and applications," *IEEE Access*, vol. 3, pp. 490–530, 2015.

[39] J. Lesouple, T. Robert, M. Sahmoudi, J. Tourneret, and W. Vigneau, "Multipath mitigation for GNSS positioning in an urban environment using sparse estimation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 4, pp. 1316–1328, 2019.

[40] D. Meng, X. Wang, M. Huang, L. Wan, and B. Zhang, "Robust weighted subspace fitting for DOA estimation via block sparse recovery," *IEEE Communications Letters*, vol. 24, no. 3, pp. 563–567, 2020.

[41] W. Gao, J. Chen, C. Richard, and J. Huang, "Online dictionary learning for kernel LMS," *IEEE Trans. Signal Processing*, vol. 62, no. 11, pp. 2765–2777, June 2014.

[42] B. Chen, S. Zhao, P. Zhu, S. Seth, and J. C. Príncipe, "Online efficient learning with quantized KLMS and L_1 regularization," in *Proc. Int. Joint Conf. Neural Networks*, 2012.

[43] Songlin Zhao, Badong Chen, Pingping Zhu, and José C. Príncipe, "Fixed budget quantized kernel least-mean-square algorithm," *Signal Processing*, vol. 93, no. 9, pp. 2759–2770, 2013.

[44] I. Yamada and N. Ogura, "Adaptive projected subgradient method for asymptotic minimization of sequence of nonnegative convex functions," *Numer. Funct. Anal. Optim.*, vol. 25, no. 7&8, pp. 593–617, 2004.

[45] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," in *Proc. Int. Conf. Machine Learning*. 2004, ACM.

[46] M. Yukawa, "Adaptive learning in cartesian product of reproducing kernel Hilbert spaces," *IEEE Trans. Signal Processing*, vol. 63, no. 22, pp. 6037–6048, Nov. 2015.

[47] M. Kasparick, R. L. G. Cavalcante, S. Valentin, S. Stańczak, and M. Yukawa, "Kernel-based adaptive online reconstruction of coverage maps with side information," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 7, pp. 5461–5473, July 2016.

[48] D. A. Awan, R. L. G. Cavalcante, M. Yukawa, and S. Stanczak, "Detection for 5g-noma: An online adaptive machine learning approach," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[49] B. Shin, M. Yukawa, R. L. G. Cavalcante, and A. Dekorsy, "Distributed adaptive learning with multiple kernels in diffusion networks," *IEEE Transactions on Signal Processing*, vol. 66, no. 21, pp. 5505–5519, Nov 2018.

[50] D. A. Awan, R. L. G. Cavalcante, M Yukawa, and S. Stanczak, *Adaptive Learning for Symbol Detection: A Reproducing Kernel Hilbert Space Approach*, chapter 11, pp. 197–211, Machine Learning for Future Wireless Communications. Wiley, New York, 2020.

[51] M. Yukawa and R. ishii, "Online model selection and learning by multikernel adaptive filtering," in *Proc. EUSIPCO*, 2013, pp. 1–5.

[52] O. Toda and M. Yukawa, "Online model-selection and learning for nonlinear estimation based on multikernel adaptive filtering," *IEICE Trans. Fundamentals*, vol. 1, no. E100-A, pp. 236–250, Jan. 2017.

[53] B. Chen, J. Liang, N. Zheng, and J. C. Principe, "Kernel least mean square with adaptive kernel size," *Neurocomputing*, vol. 191, pp. 95–105, 2013.

[54] T. Wada and T. Tanaka, "Doubly adaptive kernel adaptive filtering," in *Proc. APSIPA*, 2017, TA-P3.6.

[55] T. Wada, K.Fukumori, and T. Tanaka, "Dictionary learning for Gaussian kernel adaptive filtering with variable kernel center and width," in *Proc. IEEE ICASSP*, April 2018.

[56] C. Saide, R. Lengelle, P. Honeine, C. Richard, and R. Achkar, "Dictionary adaptation for online prediction of time series data with kernels," in *Proc. IEEE Statistical Signal Process- ing Workshop (SSP)*, 2012, pp. 604–607.

[57] C. Saide, R. Lengelle, P. Honeine, and R. Achkar, "Onlinekernel adaptive algorithms with dictionary adaptation for mimo models," *IEEE Signal Processing Letter*, vol. 20, no. 5, pp. 535–538, 2013.

[58] H. Chen, Y. Gong, X. Hong, and S. Chen, "A fast adaptive tunable rbf network for nonstationary systems," *IEEE Transactions on Cybernetics*, vol. 46, no. 12, pp. 2683–2692, Dec 2016.

[59] M. Takizawa and M. Yukawa, "Steepening squared error function facilitates online adaptation of Gaussian scales," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 5450–5454.

[60] Y. Murakami, M. Yamagishi, M. Yukawa, and I. Yamada, "A sparse adaptive filtering using time-varying soft-thresholding techniques," in *Proc. IEEE ICASSP*, 2010, pp. 3734–3737.

[61] Ingo Steinwart, "On the influence of the kernel on the consistency of support vector machines," *Journal of machine learning research*, vol. 2, no. Nov, pp. 67–93, 2001.

[62] Yurii Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87, Springer Science & Business Media, 2013.

[63] I. Yamada, K. Slavakis, and K. Yamada, "An efficient robust adaptive filtering algorithm based on parallel subgradient projection techniques," *IEEE Trans. Signal Processing*, vol. 50, no. 5, pp. 1091–1101, May 2002.

[64] M. Yukawa and I. Yamada, "Pairwise optimal weight realization — Acceleration technique for set-theoretic adaptive parallel subgradient projection algorithm," *IEEE Trans. Signal Processing*, vol. 54, no. 12, pp. 4557–4571, Dec. 2006.

[65] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections: a unifying framework for linear and nonlinear classification and regression tasks," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 97–123, Jan. 2011.

[66] P. L. Combettes, "The foundations of set theoretic estimation," *Proc. IEEE*, vol. 81, no. 2, pp. 182–208, Feb. 1993.

[67] M. Yukawa, K. Slavakis, and I. Yamada, "Adaptive parallel quadratic-metric projection algorithms," *IEEE Trans. Audio, Speech and Language processing*, vol. 15, no. 5, pp. 1665–1680, July 2007.

[68] M. Yukawa and I. Yamada, "A unified view of adaptive variable-metric projection algorithms," *EURASIP Journal on Advances in Signal Processing,*, vol. 2009, no. 34, pp. 1–13, July 2009.

[69] D. G. Luenberger, Ed., *Optimization by Vector Space Methods*, New York: Wiley, 1969.

[70] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, New York, 1985.

[71] T. J. Dodd, V. Kadirkamanathan, and R. F. Harrison, "Function estimation in Hilbert space using sequential projections," in *IFAC Conf. Intell. Control Syst. Signal Process.*, 2003, pp. 113–118.

[72] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Tracking the best hyperplane with a simple budget perceptron," *Journal of Machine Learning Research*, vol. 69, no. 2-3, pp. 143–167, 2007.

[73] Francesco Orabona, Joseph Keshet, and Barbara Caputo, "The projectron: A bounded kernel-based perceptron," in *Proc. ICML*, 2008, pp. 720–727.

[74] Peilin Zhao, Jialei Wang, Pengcheng Wu, Rong Jin, and Steven C. H. Hoi, "Fast bounded online gradient descent algorithms for scalable kernel-based online learning," in *Proc. ICML*, 2012.

[75] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 67–93, 2001.

[76] M. Yukawa and I. Yamada, "A deterministic analysis of linearly constrained adaptive filtering algorithms," in *Proc. EUSIPCO*, 2011, pp. 131–135.

[77] Masahiro Yukawa, Youngchul Sung, and Gilwon Lee, "Dual-domain adaptive beamformer under linearly and quadratically constrained minimum variance," *IEEE Trans. Signal Process.*, vol. 61, no. 11, pp. 2874–2886, June 2013.

[78] M. Yukawa and Y. Saito, "Widely linear LQCMV beamformer and augmented dual-domain adaptive algorithm," in *Proc. IEEE ICICS*, 2013, pp. 1–5.

[79] M. Ohnishi and M. Yukawa, "Online nonlinear estimation via iterative L^2-space projections: Reproducing kernel of subspace," *IEEE Trans. Signal Processing*, vol. 66, 2018.

[80] M. Takizawa, M. Yukawa, and C. Richard, "A stochastic behavior analysis of stochastic restricted-gradient descent algorithm in reproducing kernel Hilbert spaces," pp. 2001–2005, 2015.

[81] B. Widrow and M.E. Hoff, "Adaptive switching circuits.," *IRE WESCON Convention Record*, pp. 96–104, Aug. 1960.

[82] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, New York, 2011.

[83] Jie Chen, Wei Gao, Cédric Richard, and J.-C. M. Bermudez, "Convergence analysis of kernel LMS algorithm with pre-tuned dictionary," in *Proc. IEEE ICASSP*, 2014, pp. 7243–7247.

[84] A. S. Weigend and EDS N. A. Gershenfeld, Eds., *Time Series Prediction: Forecasting the Future and Understanding the Past Reading*, MA, Addition-Weasly, 1994.

[85] J. Platt, "A resourse-allocating network for function interpolation," *Neural comput.*, vol. 3, no. 2, pp. 213–225, 1991.

[86] Masahiro Yukawa and Klaus-Robert Müller, "Why does a Hilbertian metric work efficiently in online learning with kernels?," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1424–1428, 2016.

[87] W. Gao, J. Chen, C. Richard, and J. Huang, "Online dictionary learning for kernel LMS analysis and forward-backward splitting algorithm," in *IEEE Trans. Signal Process.*, 2013, submitted.

[88] Masaaki Takizawa and Masahiro Yukawa, "Efficient dictionary-refining kernel adaptive filter with fundamental insights," *IEEE Trans. Signal Processing*, vol. 64, no. 16, pp. 4337–4350, Aug. 2016.

[89] S. J. Wright, *Coordinate descent algorithms*, Springer Berlin Heidelberg, 2015.

[90] Paulo Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, 01 2008.

[91] Wallace A Martins, Markus VS Lima, Paulo SR Diniz, and Tadeu N Ferreira, "Optimal constraint vectors for set-membership affine projection algorithms," *Signal Processing*, vol. 134, pp. 285–294, 2017.

[92] Andre Flores and Rodrigo C de Lamare, "Set-membership kernel adaptive algorithms," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2676–2680.

[93] Amaresh V Malipatil, Yih-Fang Huang, Srinivas Andra, and Kristin Bennett, "Kernelized set-membership approach to nonlinear adaptive filtering," in *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* IEEE, 2005, vol. 4, pp. iv–149.

[94] K. Chen, S. Werner, A. Kuh, and Y. Huang, "Nonlinear adaptive filtering with kernel set-membership approach," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1515–1528, 2020.

[95] A. Nemirovski and D. Yudin, *Problem complexity and Method Efficiency in Optimization*, Wiley, 1983.

[96] M. Takizawa and M. Yukawa, "Online learning with self-tuned Gaussian kernels: Good kernel-initialization by multiscale screening," in *IEEE ICASSP*, 2019, pp. 4863–4867.

[97] Amir Beck, *First-order methods in optimization*, SIAM, 2017.

[98] H. Fan, Q. Song, and S. B. Shrestha, "Kernel online learning with adaptive kernel width," *Neurocomputing*, vol. 175, pp. 233–242, 2016.

[99] M. Yukawa, Y. Tawara, M. Yamagishi, and I. Yamada, "Sparsity-aware adaptive filters based on lp-norm inspired soft-thresholding techniqu," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2012, pp. 2749–2752.

[100] MC Mackey and L Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[101] T. Ergen and S. S. Kozat, "Efficient online learning algorithms based on LSTM neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3772–3783, Aug. 2018.

[102] Christopher M Bishop, *Pattern recognition and machine learning*, springer, 2006.

Publication Related to Dissertation

Articles in Journals

1. Masa-aki Takizawa and Masahiro Yukawa, "Adaptive nonlinear estimation based on parallel projection along affine subspaces in reproducing kernel Hilbert space", IEEE Trans. Signal Processing, vol.63, no.16, pp.4257–4269, August 2015.

2. Masa-aki Takizawa and Masahiro Yukawa, "Efficient dictionary-refining kernel adaptive filter with fundamental insight", IEEE Trans. Signal Processing, vol.64, no.16, pp.4337–4350, August 2016.

3. Masa-aki Takizawa and Masahiro Yukawa, "Joint learning of model parameters and coefficients for online nonlinear estimation", IEEE Access, vol.9, pp.24026–24040, January 2021. (Article DOI: 10.1109/ACCESS.2021.3053651)

Conference Presentations

4. Masa-aki Takizawa and Masahiro Yukawa, "An efficient data-reusing kernel adaptive filtering algorithm based on parallel hyperslab projection along affine subspaces," in Proceedings of 38th IEEE ICASSP, pp.3557–3561, Vancouver: Canada, May 2013.

5. Masa-aki Takizawa and Masahiro Yukawa, "An efficient sparse kernel adaptive filtering algorithm based on isomorphism between functional subspace and Euclidean space," in Proceedings of 39th IEEE ICASSP, pp.4541–4545, Florence: Italy, May 2014.

6. Masa-aki Takizawa and Masahiro Yukawa, "Online learning with self-tuned Gaussian kernels: good kernel-initialization by multiscale screening" in Proceedings of 44th IEEE ICASSP, pp.4863–4867, Brighton, U.K., May 2019.

7. Masa-aki Takizawa and Masahiro Yukawa,"Steepening squared error function facilitates online adaptation of Gaussian scales" in Proceedings of 45th IEEE ICASSP, pp.5450–5454, Barcelona, Spain, May 2020.

Peer-Reviewed Articles in Conference Proceedings

8. Masa-aki Takizawa, Masahiro Yukawa, and Cedric Richard, "A stochastic behavior analysis of stochastic restricted-gradient descent algorithm in reproducing kernel Hilbert spaces," in Proceedings of 40th IEEE ICASSP, pp.2001–2005, Brisbane: Australia, April 2015.

Articles in Domestic Conferences

9. Masa-aki Takizawa and Masahiro Yukawa, An efficient online learning method based on self-tuned Gaussian kernels, Technical Report of IEICE, vol.118, no.496, IEICE-SIP2018-123, pp.105-111, Iojima island, March 2019.

10. Masa-aki Takizawa, Masahiro Yukawa, and Cedric Richard, "Performance analysis of the stochastic restricted gradient descent algorithm in RKHS," in Proceedings of IEICE SIP Symposium, pp.259-264, Kyoto, Nov. 2014.

11. Masa-aki Takizawa and Masahiro Yukawa, "A note on kernel adaptive filtering algorithms : RKHS projection or parameter-space projection?," in Proceedings of IEICE SIP Symposium, pp.33-38, Shimonoseki, Nov. 2013.

12. Masa-aki Takizawa and Masahiro Yukawa, "A data-reusing kernel adaptive filter based on parallel hyperplane projection along subspace," in Proceedings of IEICE SIP Symposium, pp.506-510, Ishigaki, Nov. 2012.