Distributed Adaptive Learning With Multiple Kernels in Diffusion Networks

Ban-Sok Shin, *Student Member, IEEE*, Masahiro Yukawa, *Member, IEEE*, Renato Luís Garrido Cavalcante, *Member, IEEE*, and Armin Dekorsy, *Senior Member, IEEE*

***Abstract*—We propose an adaptive scheme for distributed learning of nonlinear functions by a network of nodes. The proposed algorithm consists of a local adaptation stage utilizing multiple kernels with projections onto hyperslabs and a diffusion stage to achieve consensus on the estimates over the whole network. Multiple kernels are incorporated to enhance the approximation of functions with several high- and low-frequency components common in practical scenarios. We provide a thorough convergence analysis of the proposed scheme based on the metric of the Cartesian product of multiple reproducing kernel Hilbert spaces. To this end, we introduce a modified consensus matrix considering this specific metric and prove its equivalence to the ordinary consensus matrix. Besides, the use of hyperslabs enables a significant reduction of the computational demand with only a minor loss in the performance. Numerical evaluations with synthetic and real data are conducted showing the efficacy of the proposed algorithm compared to the state-of-the-art schemes.**

***Index Terms*—Distributed adaptive learning, kernel adaptive filter, multiple kernels, consensus, spatial reconstruction, nonlinear regression.**

I. Introduction

A. Background

DISTRIBUTED learning within networks is a topic of high relevance due to its applicability in various areas such as environmental monitoring, social networks and big data [1]–[3]. In such applications, observed data are usually spread over the nodes, and thus, they are unavailable at a central entity. In environmental monitoring applications, for instance, nodes observe a common physical quantity of interest such as temperature, gas or humidity at each specific location. For a spatial reconstruction of the physical quantity over the area covered by the network non-cooperative strategies will not deliver a satisfactory performance. Rather distributed learning algorithms relying on information exchanges among neighboring nodes are required to fully exploit the observations available in the network.

Distributed learning of linear functions has been addressed by a variety of algorithms in the past decade, e.g. [4]–[12]. In contrast to these works, we address the problem of distributed learning of *nonlinear functions/systems*. To this end, we exploit kernel methods, which have been used to solve e.g. nonlinear regression tasks [13], [14]. Based on a problem formulation in a reproducing kernel Hilbert space (RKHS) linear techniques can be applied to approximate an unknown nonlinear function. This function is then modeled as an element of the RKHS, and corresponding kernel functions are utilized for its approximation. This methodology has been exploited to derive a variety of kernel adaptive filters [15]–[25]. In particular, the naive online regularized risk minimization [15], the kernel normalized least-mean-squares, the kernel affine projection [20] or the hyperplane projection along affine subspace (HYPASS) [23], [26] enjoy significant attention due to their limited complexity and their applicability in online learning scenarios. The HYPASS algorithm has been derived from a functional space approach based on the adaptive projected subgradient method (APSM) [27] in the set-theoretic estimation framework [28], [29]. It exploits a metric with regard to the kernel Gram matrix showing faster convergence and improved steady-state performance. The kernel Gram matrix determines the metric of a subspace of an RKHS and is decisive for the convergence behavior of gradient-descent algorithms [30]. In [31], [32] kernel adaptive filters have been extended by multiple kernels to increase the degree of freedom in the estimation process. By this, a more accurate approximation of functions with high and low frequency components is possible with a smaller number of dictionary samples compared to using a single kernel.

Regarding distributed kernel-based estimation algorithms, several schemes have been derived [33]–[42]. In [33] a distributed consensus-based regression algorithm based on kernel least squares has been proposed and extended by multiple kernels in [34]. Both schemes utilize alternating direction method of multipliers (ADMM) [43] for distributed consensus-based processing. Recent works in [35]–[37] apply diffusion-based schemes to the kernel least-mean-squares (KLMS) to derive distributed kernel adaptive filters where nodes process information in parallel. The functional adapt-then-combine

Manuscript received January 19, 2018; revised July 10, 2018 and August 13, 2018; accepted August 21, 2018. Date of publication August 31, 2018; date of current version September 24, 2018. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Gesualdo Scutari. The work of M. Yukawa was supported by the Japan Society for the Promotion of Science Grants-in-Aid (15K06081, 15K13986, and 15H02757). *(Corresponding author: Ban-Sok Shin.)*

B.-S. Shin and A. Dekorsy are with the Department of Communications Engineering, University of Bremen, 28359 Bremen, Germany (e-mail: shin@ant.uni-bremen.de; dekorsy@ant.uni-bremen.de).

M. Yukawa is with the Department of Electronics and Electrical Engineering, Keio University, Yokohama 223-8522, Japan (e-mail: yukawa@elec.keio.ac.jp).

R. L. G. Cavalcante is with the Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany (e-mail: renato.cavalcante@hhi.fraunhofer.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2018.2868040

KLMS (FATC-KLMS) proposed in [35] is a kernelized version of the algorithm derived in [9]. The random Fourier features diffusion KLMS (RFF-DKLMS) proposed in [36] uses random Fourier features to achieve a fixed-size coefficient vector and to avoid an a priori design of a dictionary set. However, the achievable performance strongly depends on the number of utilized Fourier features. Besides, the aforementioned schemes incorporate update equations in the ordinary Euclidean space and thus, do not exploit the metric induced by the kernel Gram matrix. Furthermore, the majority of these schemes do not consider multiple kernels in their adaptation mechanism.

B. Main Contributions

For the derivation of the proposed algorithm we rely on the previous work of [10]. However, while [10] only considers distributed learning for linear functions in a Euclidean space we specifically derive a kernel-based learning scheme in an RKHS and its isomorphic Euclidean space, respectively. More specifically, we propose a distributed algorithm completely operating in the Cartesian product space of multiple RKHSs. The Cartesian product space has been exploited by the Cartesian HYPASS (CHYPASS) algorithm for adaptive learning with multiple kernels proposed in [32]. When operating in the corresponding Euclidean parameter space a metric based on the kernel Gram matrix of each employed kernel needs to be considered. This metric is determined by a block diagonal matrix of which the block diagonals are given by kernel Gram matrices. To derive a distributed learning scheme we rely on average consensus on the coefficient vectors for each kernel. The key idea of our proposed scheme is to fully conduct distributed learning in a Euclidean space considering the metric of the Cartesian product space. This metric is responsible for an enhanced convergence speed of the adaptive algorithm. Operating with this metric implies that the consensus matrix used for diffusion of information within the network needs to be adapted to it. To this end, we introduce a *modified consensus matrix* operating in the metric of the product space. In fact, we show that the modified consensus matrix coincides with the consensus matrix operating in the ordinary Euclidean space as used in [10]. This finding actually implies that the metric of the product space does not alter the convergence properties of the average consensus scheme. This is particularly important in proving the monotone approximation property of our proposed scheme. We provide a thorough convergence analysis considering the metric of the product space. Specifically, we prove monotone approximation, asymptotic optimization, asymptotic consensus, convergence and characterization of the limit point within the framework of APSM. As a practical implication we demonstrate that by projecting the current estimate onto a *hyperslab* instead of the ordinary hyperplane we can significantly reduce the computational demand per node. By varying the hyperslab thickness (similar to an error bound), a trade-off between error performance and complexity per node can be adjusted. We corroborate our findings by extensive numerical evaluations on synthetic as well as real data and by mathematical proofs given in the appendices.

II. Preliminaries

A. Basic Definitions

We denote the inner product and the norm of the Euclidean space $\mathbb{R}^M$ by $\langle\cdot,\cdot\rangle_{\mathbb{R}^M}$ and $||\cdot||_{\mathbb{R}^M}$, respectively, and those in the RKHS $\mathcal{H}$ by $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ and $||\cdot||_{\mathcal{H}}$, respectively. Given a positive definite matrix $\boldsymbol{K}\in\mathbb{R}^{M\times M}$, $\langle\boldsymbol{x},\boldsymbol{y}\rangle_{\boldsymbol{K}}:=\boldsymbol{x}^{\mathsf{T}}\boldsymbol{K}\boldsymbol{y}$, $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^M$, defines an inner product with the norm $||\boldsymbol{x}||_{\boldsymbol{K}}:=\sqrt{\langle\boldsymbol{x},\boldsymbol{x}\rangle_{\boldsymbol{K}}}$. The norm of a matrix $\boldsymbol{X}\in\mathbb{R}^{M\times M}$ induced by the vector norm $||\cdot||_{\boldsymbol{K}}$ is defined as $||\boldsymbol{X}||_{\boldsymbol{K}}:=\max_{\boldsymbol{y}\neq\boldsymbol{0}}||\boldsymbol{X}\boldsymbol{y}||_{\boldsymbol{K}}/||\boldsymbol{y}||_{\boldsymbol{K}}$. The spectral norm of a matrix is denoted as $||\boldsymbol{X}||_2$ when we choose $\boldsymbol{K}=\boldsymbol{I}_M$ as the $M\times M$ identity matrix [44]. A set $C\subset\mathbb{R}^M$ is said to be convex if $\alpha\boldsymbol{x}+(1-\alpha)\boldsymbol{y}\in C$, $\forall\boldsymbol{x},\boldsymbol{y}\in C$, $\forall\alpha\in(0,1)$. If in addition the set C is closed, we call it a closed convex set. The $\boldsymbol{K}$-projection of a vector $\boldsymbol{w}\in\mathbb{R}^M$ onto a closed convex set C is defined by [45], [46]

$$P_C^{\boldsymbol{K}}(\boldsymbol{w}):=\min_{\boldsymbol{v}\in C}||\boldsymbol{w}-\boldsymbol{v}||_{\boldsymbol{K}}. \tag{1}$$

B. Multikernel Adaptive Filter

In the following we present the basics regarding multikernel adaptive filters which have been applied to online regression of nonlinear functions [31], [32]. We denote a multikernel adaptive filter by $\varphi:\mathcal{X}\to\mathbb{R}$ where $\mathcal{X}\subseteq\mathbb{R}^L$ is the input space of dimension L and $\mathbb{R}$ the output space. The filter/function φ employs Q positive definite kernels $\kappa_q:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ with $q\in\mathcal{Q}=\{1,2,\ldots,Q\}$. Each kernel κ_q induces an RKHS $\mathcal{H}_q$ [13], and φ uses corresponding dictionaries $\mathcal{D}_q=\{\kappa_q(\cdot,\bar{\boldsymbol{x}}_\ell)\}_{\ell=1}^r$, each of cardinality r. Here, each dictionary $\mathcal{D}_q$ contains kernel functions κ_q centered at samples $\bar{\boldsymbol{x}}_\ell\in\mathcal{X}$. For simplicity, we assume that each dictionary $\mathcal{D}_q$ uses the same centers $\{\bar{\boldsymbol{x}}_\ell\}_{\ell=1}^r$ although this assumption is not required. The multikernel adaptive filter φ is then given by

$$\varphi:=\sum_{q\in\mathcal{Q}}\sum_{\ell=1}^{r}w_{q,\ell}\kappa_q(\cdot,\bar{\boldsymbol{x}}_\ell). \tag{2}$$

The output of φ for arbitrary input samples $\boldsymbol{x}$ can be computed via

$$\varphi(\boldsymbol{x})=\sum_{q\in\mathcal{Q}}\sum_{\ell=1}^{r}w_{q,\ell}\kappa_q(\boldsymbol{x},\bar{\boldsymbol{x}}_\ell)=\langle\boldsymbol{w},\boldsymbol{\kappa}(\boldsymbol{x})\rangle_{\mathbb{R}^{rQ}}\ . \tag{3}$$

Here, vectors $\boldsymbol{w}$ and $\boldsymbol{\kappa}(\boldsymbol{x})$ are defined as

$$\begin{aligned}\boldsymbol{w}_{(q)}&:=[w_{q,1},\ldots,w_{q,r}]^{\mathsf{T}}\in\mathbb{R}^r,\\ \boldsymbol{w}&:=[\boldsymbol{w}_{(1)}^{\mathsf{T}},\ldots,\boldsymbol{w}_{(Q)}^{\mathsf{T}}]^{\mathsf{T}}\in\mathbb{R}^{rQ},\\ \boldsymbol{\kappa}_q(\boldsymbol{x})&:=[\kappa_q(\boldsymbol{x},\bar{\boldsymbol{x}}_1),\ldots,\kappa_q(\boldsymbol{x},\bar{\boldsymbol{x}}_r)]^{\mathsf{T}}\in\mathbb{R}^r,\\ \boldsymbol{\kappa}(\boldsymbol{x})&:=[\boldsymbol{\kappa}_1^{\mathsf{T}}(\boldsymbol{x}),\ldots,\boldsymbol{\kappa}_Q^{\mathsf{T}}(\boldsymbol{x})]^{\mathsf{T}}\in\mathbb{R}^{rQ}.\end{aligned}$$

A commonly used kernel function is the Gaussian kernel defined as

$$\kappa_q(\boldsymbol{x}_1,\boldsymbol{x}_2)=\exp\left(-\frac{||\boldsymbol{x}_1-\boldsymbol{x}_2||_{\mathbb{R}^L}^2}{2\zeta_q^2}\right),\ \boldsymbol{x}_1,\boldsymbol{x}_2\in\mathcal{X}, \tag{4}$$

where $\zeta_q>0$ is the kernel bandwidth. The metric of an RKHS is determined by the kernel Gram matrix. It contains the inherent

correlations of a dictionary $\mathcal{D}_q$ with respect to (w.r.t.) the kernel κ_q and is defined as

$$\boldsymbol{K}_q := \begin{bmatrix} \kappa_q(\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_1) & \dots & \kappa_q(\bar{\boldsymbol{x}}_1, \bar{\boldsymbol{x}}_r) \\ \vdots & \ddots & \vdots \\ \kappa_q(\bar{\boldsymbol{x}}_r, \bar{\boldsymbol{x}}_1) & \dots & \kappa_q(\bar{\boldsymbol{x}}_r, \bar{\boldsymbol{x}}_r) \end{bmatrix} \in \mathbb{R}^{r \times r}. \quad (5)$$

Assuming that each dictionary $\mathcal{D}_q$ is linearly independent it follows that each $\boldsymbol{K}_q$ is positive-definite [46]. Moreover, we introduce the multikernel Gram matrix $\boldsymbol{K} := \mathrm{blkdiag}\{\boldsymbol{K}_1, \boldsymbol{K}_2, \dots, \boldsymbol{K}_Q\} \in \mathbb{R}^{rQ \times rQ}$ being the block-diagonal matrix of the Gram matrices of all kernels. Then, by virtue of Lemma 1 from [47] we can parameterize φ by $\boldsymbol{w}$ in the Euclidean space $\mathbb{R}^{rQ}$ using the $\boldsymbol{K}$ inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{K}}$. In fact, the $\boldsymbol{K}$-metric in the Euclidean space corresponds to the metric of a functional subspace in the Cartesian product of multiple RKHSs [32]. Indeed, we can express (3) equivalently by

$$\varphi(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{\kappa}(\boldsymbol{x}) \rangle_{\mathbb{R}^{rQ}} = \langle \boldsymbol{w}, \boldsymbol{K}^{-1}\boldsymbol{\kappa}(\boldsymbol{x}) \rangle_{\boldsymbol{K}}. \quad (6)$$

Instead of applying a learning method to the function φ in the Cartesian product space we can directly apply it to the coefficient vector $\boldsymbol{w} \in \mathbb{R}^{rQ}$ in $\left(\mathbb{R}^{rQ}, \langle \cdot, \cdot \rangle_{\boldsymbol{K}}\right)$. This representation is based on the *parameter space approach* from the kernel adaptive filtering literature with the *functional space approach* as its equivalent counterpart, see [31, Appendix A]. In the following, we will formulate the distributed learning problem in the parameter space $\left(\mathbb{R}^{rQ}, \langle \cdot, \cdot \rangle_{\boldsymbol{K}}\right)$ to facilitate an easy understanding. However, we emphasize that this formulation originates from considerations in an isomorphic functional space. The interested reader is referred to Appendix A for a problem formulation in the functional space.

III. Problem Formulation and Objective

A. System Model

We address the problem of distributed adaptive learning of a continuous, nonlinear function $\psi : \mathcal{X} \to \mathbb{R}$ by a network of J nodes. The function ψ is assumed to lie in the sum space of Q RKHSs defined as $\mathcal{H}^+ := \mathcal{H}_1 + \mathcal{H}_2 + \dots + \mathcal{H}_Q := \left\{ \sum_{q \in \mathcal{Q}} f_q \mid f_q \in \mathcal{H}_q \right\}$. We label a node by index j and the time by index k. Each node j observes the nonlinear function $\psi \in \mathcal{H}^+$ by sequentially feeding it with inputs $\boldsymbol{x}_{j,k} \in \mathbb{R}^L$. Then each node j acquires the measurement $d_{j,k} \in \mathbb{R}$ per time index k via

$$d_{j,k} = \psi(\boldsymbol{x}_{j,k}) + n_{j,k}, \quad (7)$$

where $n_{j,k} \in \mathbb{R}$ is a noise sample. Based on the nodes' observations, at each time index k we have a set of J acquired input-output samples $\{(\boldsymbol{x}_{j,k}, d_{j,k})\}_{j=1}^{J}$ available within the network.

To describe the connections among the nodes in the network we employ a graph $\mathcal{G} = (\mathcal{J}, \mathcal{E})$ with a set of nodes $\mathcal{J} = \{1, \dots, J\}$ and a set of edges $\mathcal{E} \subseteq \mathcal{J} \times \mathcal{J}$. Each edge in the network represents a connection between two nodes j and i given by $(j, i) \in \mathcal{E}$ where each node j is connected to itself, i.e., $(j, j) \in \mathcal{E}$. We further assume that the graph is *undirected*, i.e., edges (j, i) and (i, j) are equivalent to each other. The set of neighbors for each node j is given as $\mathcal{N}_j = \{i \in \mathcal{J} \mid (j, i) \in \mathcal{E}\}$ containing all nodes connected to node j (including node j itself). Furthermore, we consider the graph to be *connected*, i.e., each node can be reached by any other node over multiple hops. The objective of the nodes is to learn the nonlinear function ψ based on the acquired input-output samples $\{(\boldsymbol{x}_{j,k}, d_{j,k})\}_{j \in \mathcal{J}}$ in a distributed fashion. To this end, nodes are able to exchange information with their neighboring nodes to enhance their individual estimate of the unknown function ψ.

B. Problem Formulation in Parameter Space

Based on the parametrization of the multikernel adaptive filter φ by the coefficient vector $\boldsymbol{w}$ we formulate an optimization problem in the parameter space of $\boldsymbol{w}$. The objective is to find a $\boldsymbol{w}$ such that the estimated output $\varphi(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{K}^{-1}\boldsymbol{\kappa}(\boldsymbol{x}) \rangle_{\boldsymbol{K}}$ is close to the function output $\psi(\boldsymbol{x})$ for arbitrary input samples $\boldsymbol{x} \in \mathcal{X}$. This has to be achieved in a distributed fashion for each node j in the network based on the acquired data pairs $\{(\boldsymbol{x}_{j,k}, d_{j,k})\}_{j \in \mathcal{J}}$. Thus, we equip each node j with a multikernel adaptive filter (2) parameterized by its individual coefficient vector $\boldsymbol{w}_j$. Furthermore, each node j is assumed to rely on the same dictionaries $\mathcal{D}_q, q \in \mathcal{Q}$, i.e., they are globally known and common to all nodes. To specify the coefficient vectors which result in an estimate close to the node's measurement, we introduce the closed convex set $\mathcal{S}_{j,k}$ per node j and time index k:

$$\mathcal{S}_{j,k} := \left\{ \boldsymbol{w}_j \in \mathbb{R}^{rQ} : |\langle \boldsymbol{w}_j, \boldsymbol{K}^{-1}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k}) \rangle_{\boldsymbol{K}} - d_{j,k}| \leq \varepsilon_j \right\},$$

where $\varepsilon_j \geq 0$ is a design parameter. The set $\mathcal{S}_{j,k}$ is a *hyperslab* containing those vectors $\boldsymbol{w}_j$ which provide an estimate $\varphi(\boldsymbol{x}_{j,k}) = \left\langle \boldsymbol{w}_j, \boldsymbol{K}^{-1}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k}) \right\rangle_{\boldsymbol{K}}$ with a maximum distance of ε_j to the desired output $d_{j,k}$ [48]. The parameter ε_j controls the thickness of the hyperslab $\mathcal{S}_{j,k}$, and is introduced to consider the uncertainty caused by measurement noise $n_{j,k}$. The key issue is to find an optimal $\boldsymbol{w}_j \in \mathcal{S}_{j,k}$. To this end, we define a local cost function $\Theta_{j,k}$ at time k per node j as the metric distance between its coefficient vector $\boldsymbol{w}_j$ and the hyperslab $\mathcal{S}_{j,k}$ in the $\boldsymbol{K}$-norm sense:

$$\Theta_{j,k}(\boldsymbol{w}_j) := ||\boldsymbol{w}_j - P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_j)||_{\boldsymbol{K}}. \quad (8)$$

This cost function gives the residual between $\boldsymbol{w}_j$ and its $\boldsymbol{K}$-projection onto $\mathcal{S}_{j,k}$. Due to the distance metric $\Theta_{j,k}(\boldsymbol{w}_j)$ is a non-negative, convex function with minimum value $\Theta^{\star}_{j,k} := \min_{\boldsymbol{w}_j} \Theta_{j,k}(\boldsymbol{w}_j) = 0$. Then we define the global cost of the network at time k to be the sum of all local costs by

$$\Theta_k := \sum_{j \in \mathcal{J}} \Theta_{j,k}(\boldsymbol{w}_j) \quad (9)$$

where each individual cost $\Theta_{j,k}$ can be time-varying. The objective is to minimize the sequence $(\Theta_k)_{k \in \mathbb{N}}$ of global costs (9) over all nodes in the network where due to convexity of $\Theta_{j,k}$ the global cost Θ_k is also convex. Simultaneously, the coefficient vectors $\boldsymbol{w}_j$ of all nodes have to converge to the same solution, which guarantees consensus in the network. To this end, we consider the following optimization problem at time k as in [8],

[10], [49]:

$$\min_{\{\boldsymbol{w}_j\,|j\in\mathcal{J}\}} \Theta_k := \sum_{j\in\mathcal{J}} \Theta_{j,k}(\boldsymbol{w}_j) \tag{10a}$$

$$\text{s.t.} \quad \boldsymbol{w}_j = \boldsymbol{w}_i, \quad \forall i \in \mathcal{N}_j. \tag{10b}$$

Constraint (10b) enforces all coefficient vectors to converge to the same solution, i.e., $\boldsymbol{w}_1 = \boldsymbol{w}_2 = \cdots = \boldsymbol{w}_J$ guaranteeing consensus within the network.

C. Optimal Solution Set

From the definition (8) of the local cost $\Theta_{j,k}$ we directly see that its minimizers are given by points in the hyperslab $\mathcal{S}_{j,k}$. Since each local cost $\Theta_{j,k}$ is a metric distance with minimum value zero the minimizers of the global cost Θ_k at time k are given by the intersection $\Upsilon_k := \bigcap_{j\in\mathcal{J}} \mathcal{S}_{j,k}$. Points in Υ_k minimize each local cost $\Theta_{j,k}$ and therefore also their sum $\Theta_k = \sum_{j\in\mathcal{J}} \Theta_{j,k}(\boldsymbol{w}_j)$. Thus, a point minimizing each local cost $\Theta_{j,k}, \forall j \in \mathcal{J}$, is also a minimizer of the global cost Θ_k. To consider arbitrary many time instants $k \geq 0$ we can now define the optimal solution set to problem (10):

$$\Upsilon^\star := \bigcap_{k\geq 0} \bigcap_{j\in\mathcal{J}} \mathcal{S}_{j,k}. \tag{11}$$

Points in the set $\Upsilon^\star$ minimize the global cost Θ_k for any time instant $k \geq 0$ and at any node j. We therefore call a point $\boldsymbol{w}^\star \in \Upsilon^\star$ *ideal estimate*. However, finding $\boldsymbol{w}^\star$ is a challenging task particularly under practical considerations. Due to limited memory, for instance, not all measurements can be stored over time at each node. Hence, information about the set $\Upsilon^\star$ is unavailable and thus an ideal estimate $\boldsymbol{w}^\star$ cannot be acquired. An alternative, feasible task is the minimization of all but finitely many global costs Θ_k. This approach stems from the intuition that a good estimate should minimize as many costs Θ_k as possible. To acquire such an estimate the nodes should agree on a point contained in the set

$$\Upsilon := \overline{\liminf_{k\to\infty} \Upsilon_k} = \overline{\bigcup_{k=0}^{\infty} \bigcap_{m\geq k} \Upsilon_m} \supset \Upsilon^\star \tag{12}$$

where the overbar gives the closure of a set. Finding a point in Υ is clearly a less restrictive task than finding one in $\Upsilon^\star$ since all global costs Θ_k excluding finitely many ones need to be minimized. Therefore, our proposed algorithm should achieve estimates in the set Υ. It has been shown that the APSM converges to points in the set Υ [27], [50].

Remark 1: For the above considerations we need to assume that $\Upsilon^\star \neq \emptyset$. To enable $\Upsilon^\star \neq \emptyset$ the hyperslab threshold ε_j of $\mathcal{S}_{j,k}$ should be chosen sufficiently large depending on the noise distribution and its variance. Examples on how to choose ε_j in noisy environments have been proposed in [48]. For impulsive noise occurring finitely many times one can regard the time instant of the final impulse as $k = 0$ to guarantee $\Upsilon^\star \neq \emptyset$. If however impulsive noise occurs infinitely many times on the measurements it is not straightforward to ensure $\Upsilon^\star \neq \emptyset$ and convergence of the APSM which will be introduced later on. Nevertheless, whenever the impulsive noise occurs the error signal in the APSM will abruptly change. Based on this change those noisy measurements can be detected and discarded in practice so that $\Upsilon^\star \neq \emptyset$ is satisfied.

IV. Proposed Algorithm: Diffusion-Based Multikernel Adaptive Filter

To solve (10) in a distributed way we employ a two-step scheme consisting of a local adaptation and a diffusion stage which has been commonly used in the literature, see e.g. [4], [35], [49]:

1) a *local APSM update* per node j on the coefficient vector $\boldsymbol{w}_j$ giving an intermediate coefficient vector $\boldsymbol{w}'_j$;
2) a *diffusion stage* to fuse vectors $\boldsymbol{w}'_i$ from neighboring nodes $i \in \mathcal{N}_j$ to update $\boldsymbol{w}_j$.

Step 1) ensures that each local cost $\Theta_{j,k}$ is reduced, and, hence the global cost Θ_k is reduced as well. Step 2) seeks for a consensus among all coefficient vectors $\{\boldsymbol{w}_j\}_{j\in\mathcal{J}}$ through information exchange among neighboring nodes to satisfy constraint (10b). By this exchange each node inherently obtains the property sets from its neighbors which can be exploited to improve the convergence behavior of the learning algorithm.

A. Local APSM Update

The APSM asymptotically minimizes a sequence of nonnegative convex (not necessarily differentiable) functions [27] and can thus be used to minimize the local cost function $\Theta_{j,k}(\boldsymbol{w}_j)$ in (8) per node j. For the coefficient vector $\boldsymbol{w}_{j,k} \in \mathbb{R}^{rQ}$ at node j and time k a particular case of the APSM update with the $\boldsymbol{K}$-norm reads

$$\boldsymbol{w}'_{j,k+1} := \begin{cases} \boldsymbol{w}_{j,k} - \mu_{j,k} \dfrac{\Theta_{j,k}(\boldsymbol{w}_{j,k}) - \Theta^\star_{j,k}}{||\Theta'_{j,k}(\boldsymbol{w}_{j,k})||^2_{\boldsymbol{K}}} \Theta'_{j,k}(\boldsymbol{w}_{j,k}) \\ \qquad\qquad \text{if } \Theta'_{j,k}(\boldsymbol{w}_{j,k}) \neq \boldsymbol{0} \\ \boldsymbol{w}_{j,k} \qquad\qquad \text{otherwise} \end{cases} \tag{13}$$

where $\Theta'_{j,k}(\boldsymbol{w}_{j,k})$ is a subgradient[1] of $\Theta_{j,k}(\boldsymbol{w}_{j,k})$ at $\boldsymbol{w}_{j,k}$. The parameter $\mu_{j,k} \in (0, 2)$ is the step size. Since the learning scheme is to operate with the $\boldsymbol{K}$-metric it is used for the squared norm in the denominator. A subgradient for (8) is given by [27]

$$\Theta'_{j,k}(\boldsymbol{w}_{j,k}) = \frac{\boldsymbol{w}_{j,k} - P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_{j,k})}{||\boldsymbol{w}_{j,k} - P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_{j,k})||_{\boldsymbol{K}}}, \quad \text{for } \boldsymbol{w}_{j,k} \notin \mathcal{S}_{j,k}. \tag{14}$$

This subgradient gives $||\Theta'_{j,k}(\boldsymbol{w}_{j,k})||^2_{\boldsymbol{K}} = 1$ and thus we arrive at the following APSM update per node j:

$$\boldsymbol{w}'_{j,k+1} := \boldsymbol{w}_{j,k} - \mu_{j,k} \left(\boldsymbol{w}_{j,k} - P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_{j,k}) \right). \tag{15}$$

As we can see, the difference vector $\boldsymbol{w}_{j,k} - P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_{j,k})$ is used to move the coefficient vector $\boldsymbol{w}_{j,k}$ into the direction of the hyperslab $\mathcal{S}_{j,k}$ controlled by the step size $\mu_{j,k}$. Note that this update solely relies on local information, i.e., no information

[1] A vector $\Theta'(\boldsymbol{y}) \in \mathbb{R}^M$ is a subgradient of a function $\Theta : \mathbb{R}^M \to \mathbb{R}$ at $\boldsymbol{y} \in \mathbb{R}^M$ if $\Theta(\boldsymbol{y}) + \langle \boldsymbol{x} - \boldsymbol{y}, \Theta'(\boldsymbol{y}) \rangle \leq \Theta(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^M$.

from neighboring nodes is needed. The projection $P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_{j,k})$ is calculated by [45]

$$P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}) = \begin{cases} \boldsymbol{w}, & \text{if } \boldsymbol{w} \in \mathcal{S}_{j,k} \\ \boldsymbol{w} - \dfrac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k}) - d_{j,k} - \varepsilon_j}{||\boldsymbol{K}^{-1}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k})||^2_{\boldsymbol{K}}}\boldsymbol{K}^{-1}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k}), \\ \quad \text{if } \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k}) > d_{j,k} + \varepsilon_j \\ \boldsymbol{w} - \dfrac{\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k}) - d_{j,k} + \varepsilon_j}{||\boldsymbol{K}^{-1}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k})||^2_{\boldsymbol{K}}}\boldsymbol{K}^{-1}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k}), \\ \quad \text{if } \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k}) < d_{j,k} - \varepsilon_j. \end{cases} \tag{16}$$

B. Diffusion Stage

To satisfy constraint (10b) and reach consensus on the coefficient vectors $\boldsymbol{w}_j$, each node j fuses its own vector $\boldsymbol{w}'_j$ with those of its neighbors $\{\boldsymbol{w}'_i\}_{i\in\mathcal{N}_j}$. To this end, we employ a symmetric matrix $\boldsymbol{G} \in \mathbb{R}^{J\times J}$ assigning weights to the edges in the network. The (j,i)-entry of $\boldsymbol{G}$ is denoted by g_{ji} and gives the weight on the edge between nodes j and i. If no connection is present among both nodes, the entry will be zero. The fusion step per node j at time k follows

$$\boldsymbol{w}_{j,k} := \sum_{i\in\mathcal{N}_j} g_{ji}\boldsymbol{w}'_{i,k}. \tag{17}$$

To guarantee that all nodes converge to the same coefficient vector, $\boldsymbol{G}$ needs to fulfill the following conditions [10]:

$$||\boldsymbol{G} - (1/J)\mathbf{1}_J\mathbf{1}_J^{\mathsf{T}}||_2 < 1, \quad \boldsymbol{G}\mathbf{1}_J = \mathbf{1}_J, \tag{18}$$

where $\mathbf{1}_J$ is the vector of J ones. The first condition guarantees convergence to the average of all states in the network while the second condition keeps the network at a stable state if consensus has been reached. Such matrices have been vastly applied in literature for consensus averaging problems, see e.g. [4], [51], [52]. Our proposed algorithm to solve (10) is then given by the following update equations per node j and time index k:

$$\boldsymbol{w}'_{j,k+1} := \boldsymbol{w}_{j,k} - \mu_{j,k}\left(\boldsymbol{w}_{j,k} - P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_{j,k})\right) \tag{19a}$$

$$\boldsymbol{w}_{j,k+1} := \sum_{i\in\mathcal{N}_j} g_{ji}\boldsymbol{w}'_{i,k+1} \tag{19b}$$

where the projection $P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_{j,k})$ is given in (16). In each iteration k each node j performs a local APSM update and transmits its intermediate coefficient vector $\boldsymbol{w}'_{j,k}$ to its neighbors $i \in \mathcal{N}_j$. After receiving the intermediate coefficient vectors $\boldsymbol{w}'_{i,k}$ from its neighbors, each node j fuses these with its own vector $\boldsymbol{w}'_{j,k}$ by a weighted average step.

In fact, (19a) comprises the projection in the Cartesian product of Q RKHSs which is used by the CHYPASS algorithm [32]. Therefore, we call our proposed scheme diffusion-based CHYPASS (D-CHYPASS) being a distributed implementation of CHYPASS.

Remark 2: If the diffusion stage (19b) in D-CHYPASS is omitted the algorithm reduces to a local adaptation or non-cooperative scheme where each node individually approximates ψ based on its node-specific measurement data. However, in this case each node j has access to its individual property set $\mathcal{S}_{j,k}$ only per time instant k. In contrast, by diffusing the coefficient vectors among neighboring nodes each node j inherently obtains information about the property sets $\{\mathcal{S}_{i,k}\}_{i\in\mathcal{N}_j}$ of its neighbors. This can be simply observed when inserting (19a) into (19b). Therefore, compared to local adaptation D-CHYPASS will show a faster convergence speed and a lower steady-state error due to a cooperation within the network. Several works have shown the benefit of distributed approaches over non-cooperative strategies in the context of diffusion-based adaptive learning, see [4] and references therein.

V. Theoretical Analysis

A. Consensus Matrix

To analyze the theoretical properties of the D-CHYPASS algorithm, let us first introduce the definition of the consensus matrix.

Definition 1 (Consensus Matrix [10]): A consensus matrix $\boldsymbol{P} \in \mathbb{R}^{rQJ\times rQJ}$ is a square matrix satisfying the following two properties.

1) $\boldsymbol{P}\boldsymbol{z} = \boldsymbol{z}$ and $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{z} = \boldsymbol{z}$ for any vector $\boldsymbol{z} \in \mathcal{C} := \left\{\mathbf{1}_J \otimes \boldsymbol{a} \in \mathbb{R}^{rQJ} \,|\, \boldsymbol{a} \in \mathbb{R}^{rQ}\right\}$.
2) The rQ largest singular values of $\boldsymbol{P}$ are equal to one and the remaining $rQJ - rQ$ singular values are strictly less than one.

We denote by $\otimes$ the Kronecker product. We can further establish the following properties of the consensus matrix $\boldsymbol{P}$:

Lemma 1 (Properties of Consensus Matrix [10]): Let $\boldsymbol{e}_n \in \mathbb{R}^{rQ}$ be a unit vector with its n-th entry being one and $\boldsymbol{b}_n = (\mathbf{1}_J \otimes \boldsymbol{e}_n)/\sqrt{J} \in \mathbb{R}^{rQJ}$. Further, we define the *consensus subspace* $\mathcal{C} := \operatorname{span}\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{rQ}\}$ and the stacked vector of all coefficient vectors in the network $\boldsymbol{z}_k = [\boldsymbol{w}_{1,k}^{\mathsf{T}}, \ldots, \boldsymbol{w}_{J,k}^{\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{rQJ}$. Then, we have the following properties.

1) The consensus matrix $\boldsymbol{P}$ can be decomposed into $\boldsymbol{P} = \boldsymbol{B}\boldsymbol{B}^{\mathsf{T}} + \boldsymbol{X}$ with $\boldsymbol{B} := [\boldsymbol{b}_1 \ldots \boldsymbol{b}_{rQ}] \in \mathbb{R}^{rQJ\times rQ}$ and $\boldsymbol{X} \in \mathbb{R}^{rQJ\times rQJ}$ satisfying $\boldsymbol{X}\boldsymbol{B}\boldsymbol{B}^{\mathsf{T}} = \boldsymbol{B}\boldsymbol{B}^{\mathsf{T}}\boldsymbol{X} = \mathbf{0}$ and $||\boldsymbol{X}||_2 < 1$.
2) The nodes have reached consensus at time index k if and only if $(\boldsymbol{I}_{rQJ} - \boldsymbol{B}\boldsymbol{B}^{\mathsf{T}})\boldsymbol{z}_k = \mathbf{0}$, i.e., $\boldsymbol{z}_k \in \mathcal{C}$.

A consensus matrix can be constructed by matrix $\boldsymbol{G}$ as $\boldsymbol{P} = \boldsymbol{G} \otimes \boldsymbol{I}_{rQ}$ where $\boldsymbol{I}_{rQ}$ is the $rQ \times rQ$ identity matrix. The matrix $\boldsymbol{P}$ is then said to be compatible to the graph $\mathcal{G}$ since $\boldsymbol{z}_{k+1} = \boldsymbol{P}\boldsymbol{z}_k$ can be equivalently calculated by $\boldsymbol{w}_{j,k+1} = \sum_{i\in\mathcal{N}_j} g_{ji}\boldsymbol{w}_{i,k}$ (see (17)) [10]. By definition of the consensus matrix we know that $||\boldsymbol{P}||_2 = 1$ holds. However, for further analysis of the D-CHYPASS algorithm, we need to know the norm w.r.t. matrix $\boldsymbol{K}$ since D-CHYPASS operates with the $\boldsymbol{K}$-metric. Therefore, we introduce a modified consensus matrix $\widehat{\boldsymbol{P}}$ satisfying $||\widehat{\boldsymbol{P}}||_{\mathcal{K}} = 1$, where $\mathcal{K} := \boldsymbol{I}_J \otimes \boldsymbol{K}$.

Lemma 2 (Modified Consensus Matrix): Suppose that $\boldsymbol{P}$ is a consensus matrix defined as in Definition 1. Let

$$\widehat{\boldsymbol{P}} := \mathcal{K}^{-1/2}\boldsymbol{P}\mathcal{K}^{1/2}$$

be the *modified consensus matrix* where $\mathcal{K}$ is the block-diagonal matrix with J copies of $\boldsymbol{K}$:

$$\mathcal{K} := \boldsymbol{I}_J \otimes \boldsymbol{K} \in \mathbb{R}^{rQJ\times rQJ}. \tag{20}$$

Assume further, that the dictionary $\mathcal{D}_q = \{\kappa_q(\cdot, \bar{\boldsymbol{x}}_\ell)\}_{\ell=1}^r$ for each $q \in \mathcal{Q}$ is linearly independent, i.e., its corresponding kernel Gram matrix $\boldsymbol{K}_q$ is of full rank, and thus $\mathcal{K}$ is also linearly independent. Then, the $\mathcal{K}$-norm of $\widehat{\boldsymbol{P}}$ is given by $||\widehat{\boldsymbol{P}}||_{\mathcal{K}} = 1$. In particular, it holds that both consensus matrices are identical to each other, i.e., $\widehat{\boldsymbol{P}} = \boldsymbol{P}$.

Proof: The proof is given in Appendix B. ■

Due to Lemma 2, for further analysis we are free to use either $\boldsymbol{P}$ or $\widehat{\boldsymbol{P}}$ and it holds that $||\boldsymbol{P}||_{\mathcal{K}} = ||\widehat{\boldsymbol{P}}||_{\mathcal{K}} = 1$.

B. Convergence Analysis

From (19) we can summarize both update equations of the D-CHYPASS in terms of all coefficient vectors in the network by defining

$$\boldsymbol{z}_k := \begin{bmatrix} \boldsymbol{w}_{1,k} \\ \vdots \\ \boldsymbol{w}_{J,k} \end{bmatrix}, \; \boldsymbol{y}_k := \begin{bmatrix} \mu_{1,k}(\boldsymbol{w}_{1,k} - P^{\boldsymbol{K}}_{\mathcal{S}_{1,k}}(\boldsymbol{w}_{1,k})) \\ \vdots \\ \mu_{J,k}(\boldsymbol{w}_{J,k} - P^{\boldsymbol{K}}_{\mathcal{S}_{J,k}}(\boldsymbol{w}_{J,k})) \end{bmatrix}$$

and rewriting (19a) and (19b) into

$$\boldsymbol{z}_{k+1} = (\boldsymbol{G} \otimes \boldsymbol{I}_{rQ})(\boldsymbol{z}_k - \boldsymbol{y}_k). \tag{21}$$

We show the convergence properties of D-CHYPASS for fixed and deterministic network topologies. Although the space under study is the $\boldsymbol{K}$-metric space unlike [8], [10] we can still prove the properties due to Lemmas 1 and 2.

Theorem 1: The sequence $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$ generated by (21) satisfies the following.

1) *Monotone approximation:* Assume that $\boldsymbol{w}_{j,k} \notin \mathcal{S}_{j,k}$ with $\mu_{j,k} \in (0, 2)$ for at least one node j and that $\mu_{i,k} \in [0, 2]$ $(i \neq j)$. Then, for every $\boldsymbol{w}^\star_k \in \Upsilon_k$ and $\boldsymbol{z}^\star_k := [(\boldsymbol{w}^\star_k)^\mathsf{T}, (\boldsymbol{w}^\star_k)^\mathsf{T}, \ldots, (\boldsymbol{w}^\star_k)^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{rQJ}$ it holds that
$$||\boldsymbol{z}_{k+1} - \boldsymbol{z}^\star_k||_{\mathcal{K}} < ||\boldsymbol{z}_k - \boldsymbol{z}^\star_k||_{\mathcal{K}} \tag{22}$$
where $\Upsilon_k \neq \emptyset$ since we assume that $\Upsilon^\star \neq \emptyset$.

For the remaining properties we assume that $\mu_{j,k} \in [\epsilon_1, 2 - \epsilon_2]$ with $\epsilon_1, \epsilon_2 > 0$ and that a sufficiently large hyperslab threshold ε_j per node j has been chosen such that $\boldsymbol{w}^\star \in \Upsilon^\star \neq \emptyset$. We further define $\boldsymbol{z}^\star := [(\boldsymbol{w}^\star)^\mathsf{T}, (\boldsymbol{w}^\star)^\mathsf{T}, \ldots, (\boldsymbol{w}^\star)^\mathsf{T}]^\mathsf{T}$. Then the following holds:

2) *Asymptotic minimization of local costs:* For every $\boldsymbol{z}^\star$ the local costs $\Theta_{j,k}(\boldsymbol{w}_{j,k}) = ||\boldsymbol{w}_{j,k} - P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_{j,k})||_{\boldsymbol{K}}$ are asymptotically minimized, i.e.,
$$\lim_{k\to\infty} \Theta_{j,k}(\boldsymbol{w}_{j,k}) = 0, \forall j \in \mathcal{J}. \tag{23}$$

3) *Asymptotic consensus:* With the decomposition $\boldsymbol{P} = \boldsymbol{B}\boldsymbol{B}^\mathsf{T} + \boldsymbol{X}$ and $||\boldsymbol{X}||_2 < 1$ the sequence $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$ asymptotically achieves consensus such that
$$\lim_{k\to\infty} (\boldsymbol{I}_{rQJ} - \boldsymbol{B}\boldsymbol{B}^\mathsf{T})\boldsymbol{z}_k = \boldsymbol{0}. \tag{24}$$

4) *Convergence of* $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$*:* Suppose that $\Upsilon^\star$ has a nonempty interior, i.e., there exists $\rho > 0$ and interior point $\tilde{\boldsymbol{u}}$ such that $\{\boldsymbol{v} \in \mathbb{R}^{rQ} \,|\, ||\boldsymbol{v} - \tilde{\boldsymbol{u}}||_{\boldsymbol{K}} \leq \rho\} \subset \Upsilon^\star$. Then, the sequence $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$ converges to a vector $\widehat{\boldsymbol{z}} = [\widehat{\boldsymbol{w}}^\mathsf{T}, \ldots, \widehat{\boldsymbol{w}}^\mathsf{T}]^\mathsf{T} \in \mathcal{C}$ satisfying $(\boldsymbol{I}_{rQJ} - \boldsymbol{B}\boldsymbol{B}^\mathsf{T})\widehat{\boldsymbol{z}} = \boldsymbol{0}$.

5) *Characterization of limit point* $\widehat{\boldsymbol{z}}$*:* Suppose for an interior $\tilde{\boldsymbol{u}} \in \Upsilon^\star$ that for any $\epsilon > 0$ and any $\eta > 0$ there exists a $\xi > 0$ such that
$$\min_{k\in\mathcal{I}} \sum_{j\in\mathcal{J}} ||\boldsymbol{w}_{j,k} - P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_{j,k})||_{\boldsymbol{K}} \geq \xi, \tag{25}$$
where
$$\mathcal{I} := \Bigg\{ k \in \mathbb{N} \,|\, \sum_{j\in\mathcal{J}} \mathrm{d}_{\boldsymbol{K}}(\boldsymbol{w}_{j,k}, \mathrm{lev}_{\leq 0}\Theta_{j,k}) > \epsilon \text{ and } \sum_{j\in\mathcal{J}} ||\tilde{\boldsymbol{u}} - \boldsymbol{w}_{j,k}||_{\boldsymbol{K}} \leq \eta \Bigg\}$$
and $\mathrm{lev}_{\leq 0}\Theta_{j,k} := \{\boldsymbol{w} \in \mathbb{R}^{rQ} \,|\, \Theta_{j,k}(\boldsymbol{w}) \leq 0\}$. Then it holds that $\widehat{\boldsymbol{w}} \in \Upsilon$ with Υ defined as in (12).

Proof: The proofs of Theorems 1.1–1.3 can be directly deduced from the corresponding proofs of Theorems 1a)–1c) in [10, Appendix III] under the consideration that $||\boldsymbol{P}||_2 = ||\boldsymbol{P}||_{\mathcal{K}} = 1$ (see Lemma 2) and that $\Theta_{j,k}(\boldsymbol{w}_j)$ is a non-negative convex function. Note that the proof of Theorem 1.1 needs to be derived considering the $\mathcal{K}$-metric and not the ordinary Euclidean metric as in [10]. The proofs of Theorems 1.4 and 1.5 are given in Appendix C. ■

VI. Numerical Evaluation

In the following section, we evaluate the performance of the D-CHYPASS by applying it to the spatial reconstruction of multiple Gaussian functions, real altitude data and the tracking of a time-varying nonlinear function by a network of nodes. The nodes are distributed over the unit-square area $A = [0, 1]^2$ and each node j uses its Cartesian position vector $\boldsymbol{x}_j = [x_{j,1}, x_{j,2}]^\mathsf{T} \in \mathcal{X}$ as its regressor. We assume that the positions of the nodes stay fixed, i.e., $\boldsymbol{x}_{j,k}$ does not change over time. This is not necessary for the D-CHYPASS to be applicable, e.g. it can be applied to a mobile network where the positions change over time as investigated in [42]. Per time index k the nodes take a new measurement $d_{j,k}$ of the function ψ at their position $\boldsymbol{x}_j$. Hence, the network constantly monitors the function ψ. For all experiments we assume model (7) with zero-mean white Gaussian noise of variance σ_n^2. Since in this scenario measurements of the function ψ are spatially spread over the nodes a collaboration among the nodes is inevitable for a good regression performance. Thus, it is an appropriate application example where the benefit of distributed learning becomes clear.

We compare the performance of the D-CHYPASS to the RFF-DKLMS [36], the FATC-KLMS [35] and the multikernel distributed consensus-based estimation (MKDiCE) [34] which are state of the art algorithms for distributed kernel-based estimation. Both RFF-DKLMS and FATC-KLMS are single kernel approaches based on a diffusion mechanism. Assuming that the FATC-KLMS only considers local data in its adaptation step, both schemes exhibit the same number of transmissions per node as the D-CHYPASS. To enable a fair comparison we restrict the adaptation step of the FATC-KLMS to use local data only and extend the algorithm by multiple kernels as in

D-CHYPASS. We call this scheme the diffusion-based multikernel least-mean-squares (DMKLMS). Its update equation per node j is given by

$$\boldsymbol{w}'_{j,k+1} := \boldsymbol{w}_{j,k} + \mu_{j,k}\left(d_{j,k} - \boldsymbol{w}_{j,k}^{\mathsf{T}}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k})\right)\boldsymbol{\kappa}(\boldsymbol{x}_{j,k}) \quad (26a)$$

$$\boldsymbol{w}_{j,k+1} := \sum_{i\in\mathcal{N}_j} g_{ji}\boldsymbol{w}'_{i,k+1}. \quad (26b)$$

The RFF-DKLMS approximates kernel evaluations by random Fourier features such that no design of a specific dictionary set is necessary. However, its performance is highly dependent on the number of the utilized Fourier features which determines the dimension of the vectors to be exchanged. The MKDiCE is a distributed regression scheme based on kernel least squares with multiple kernels using the ADMM for its distributed mechanism. The number of transmissions per iteration is higher compared to the D-CHYPASS, RFF-DKLMS and DMKLMS. Naturally, it is not an adaptive scheme but is included here for reference purposes. As benchmark performance, we consider the central CHYPASS given by

$$\boldsymbol{w}_{k+1} := \boldsymbol{w}_k - \mu\sum_{j\in\mathcal{J}}\left(\boldsymbol{w}_k - P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_k)\right). \quad (27)$$

The central CHYPASS requires all node positions and measurements $\{(\boldsymbol{x}_{j,k}, d_{j,k})\}_{j\in\mathcal{J}}$ per time index k at a single node to perform the projection $P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_k)$ onto each set $\mathcal{S}_{j,k}$.

Regarding the dictionaries we assume that each $\mathcal{D}_q$ uses the same samples $\{\bar{\boldsymbol{x}}_\ell\}_{\ell=1}^r$. These samples are a subset of the node positions $\{\boldsymbol{x}_j\}_{j\in\mathcal{J}}$ in the network and are selected following the coherence criterion: A node position $\boldsymbol{x}_j$ is compared to every dictionary entry $\{\bar{\boldsymbol{x}}_\ell\}_{\ell=1}^r$ and is included as dictionary sample $\bar{\boldsymbol{x}}_{r+1}$ if it satisfies

$$\max_{q\in\mathcal{Q}}\max_{\ell=1,\ldots,r}|\kappa_q(\boldsymbol{x}_j, \bar{\boldsymbol{x}}_\ell)| \le \tau. \quad (28)$$

Here, $0 < \tau \le 1$ is the coherence threshold controlling the cardinality of $\mathcal{D}_q$. The dictionary $\mathcal{D}_q$ is generated a priori over all node positions before the algorithm iterates. After that it stays fixed throughout the reconstruction process for the specific algorithm.

As error metric we consider the network NMSE_k per time k over the area A. It evaluates the normalized squared-difference between reconstructed field $\varphi_j(\boldsymbol{x})$ and the true field $\psi(\boldsymbol{x})$ averaged over all nodes:

$$\mathrm{NMSE}_k := \frac{1}{J}\sum_{j\in\mathcal{J}}\frac{\mathrm{E}\left\{\int_A|\psi(\boldsymbol{x}) - \boldsymbol{w}_{j,k}^{\mathsf{T}}\boldsymbol{\kappa}(\boldsymbol{x})|^2 d\boldsymbol{x}\right\}}{\int_A|\psi(\boldsymbol{x})|^2 d\boldsymbol{x}}. \quad (29)$$

The expectation in the numerator is approximated by averaging over sufficiently many independent trials. The integrals are approximated by a sum over regularly positioned grid points which sample the area A.

A. Multiple Gaussian Functions

As a first example we apply the D-CHYPASS algorithm to the reconstruction of two Gaussian functions with different bandwidths given as follows:

$$\psi(\boldsymbol{x}) := 2\exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{p}_1||^2_{\mathbb{R}^2}}{2\cdot 0.1^2}\right) + \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{p}_2||^2_{\mathbb{R}^2}}{2\cdot 0.3^2}\right)$$

with $\boldsymbol{p}_1 = [0.5, 0.7]^{\mathsf{T}}, \boldsymbol{p}_2 = [0.3, 0.1]^{\mathsf{T}}$, and the Cartesian coordinate vector $\boldsymbol{x} = [x_1, x_2]^{\mathsf{T}}$. We use $J = 60$ nodes randomly placed over A following a uniform distribution where nodes share a connection if their distance to each other satisfies $D < 0.3$. We assume a noise variance of $\sigma_n^2 = 0.3$ at the nodes and average the performance over 200 trials with a new network realization in each trial. Regarding the kernel choice we use two Gaussian kernels ($Q = 2$) with bandwidths $\zeta_1 = 0.1$ and $\zeta_2 = 0.3$. For all diffusion-based algorithms we use the Metropolis-Hastings weights [53] where each entry g_{ji} is determined by

$$g_{ji} = \begin{cases} \dfrac{1}{\max\{\delta_j, \delta_i\}} & \text{if } j \ne i \text{ and } (j,i)\in\mathcal{E} \\ 1 - \sum_{i\in\mathcal{N}_j\setminus\{j\}}\dfrac{1}{\max\{\delta_j,\delta_i\}} & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

and $\delta_j = |\mathcal{N}_j|$ denotes the degree of a node j. For all algorithms we set the coherence threshold τ such that the same average dictionary size of $\bar{r} = 33$ is utilized. Single kernel approaches use the arithmetic average of the bandwidths chosen for the multikernel schemes as their kernel bandwidth. We evaluate the D-CHYPASS (I) with a hyperplane projection, i.e., $\varepsilon_j = 0$, and the D-CHYPASS (II) with a hyperslab projection with $\varepsilon_j = 0.5$. The chosen parameter values for the considered algorithms are listed in Table I.

TABLE I
PARAMETER VALUES FOR EXPERIMENT IN SECTION VI-A

Algorithm	Parameters		
D-CHYPASS (I)	$\mu_{j,k} = 0.2$ $\varepsilon_j = 0$	$\tau = 0.95$	$\zeta_1 = 0.1$ $\zeta_2 = 0.3$
D-CHYPASS (II)	$\mu_{j,k} = 0.5$ $\varepsilon_j = 0.5$		
DMKLMS	$\mu_{j,k} = 0.1$		
MKDiCE	$\mu_{j,k} = 0.5$		
Central CHYPASS	$\mu = 3.3\cdot 10^{-3}$ $\varepsilon_j = 0$		
FATC-KLMS	$\mu_{j,k} = 0.07$	$\tau = 0.9$	$\zeta = 0.2$
RFF-DKLMS (I)	$\mu_{j,k} = 0.1$	$r_{\mathrm{RFF}} = 100$	
RFF-DKLMS (II)	$\mu_{j,k} = 0.1$	$r_{\mathrm{RFF}} = 500$	

Fig. 1 compares the NMSE learning curves of D-CHYPASS (I) and D-CHYPASS (II) to a local adaptation and the central CHYPASS. Clearly, the local adaptation completely fails to approximate ψ while both D-CHYPASS (I) and D-CHYPASS (II) perform close to the central CHYPASS. Fig. 2 compares the performance of D-CHYPASS (I) to state of the art schemes. D-CHYPASS (I) significantly outperforms the compared algorithms in terms of convergence speed and steady-state error. Regarding monokernel approaches, FATC-KLMS outperforms RFF-DKLMS (I) in its steady-state error although it uses a dictionary of only $\bar{r} = 33$ samples compared to $r_{\mathrm{RFF}} = 100$

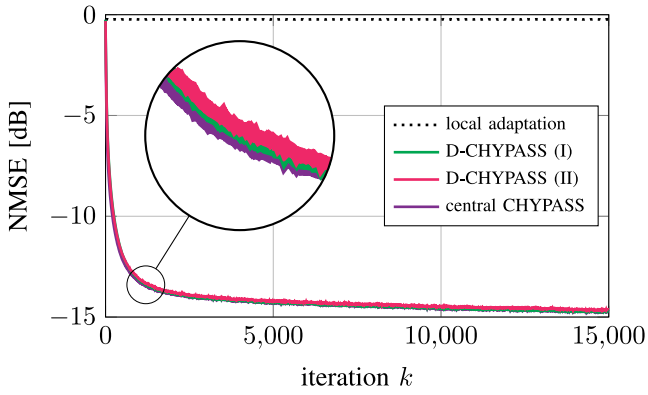

Fig. 1. Comparing learning curves of D-CHYPASS to central and local adaptation.

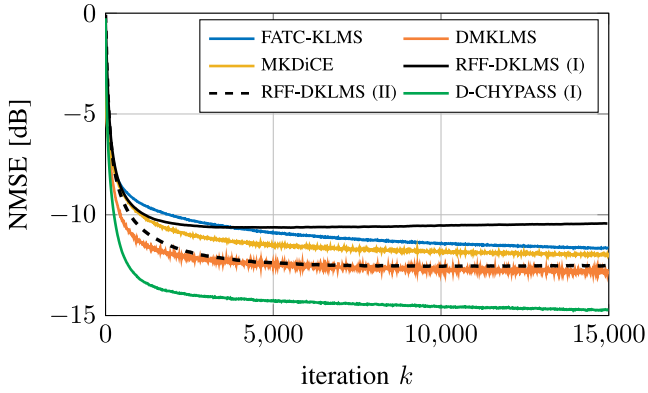

Fig. 2. Learning curves for the reconstruction of multiple Gaussian functions.

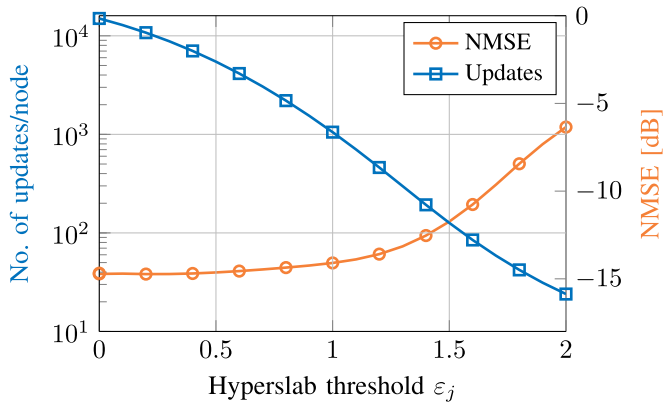

Fig. 3. Number of updates per node and NMSE for different values of the hyperslab threshold ε_j for the D-CHYPASS.

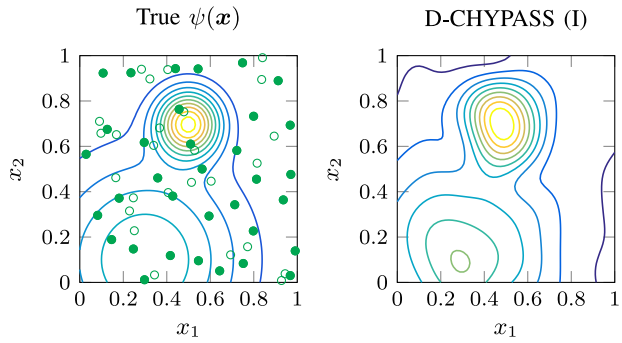

Fig. 4. Contour plots of the true $\psi(\boldsymbol{x})$ (left) and its reconstruction $\varphi(\boldsymbol{x})$ (right) at one node using the D-CHYPASS at steady state. Green circles show the node positions and filled circles the chosen dictionary entries.

random Fourier features. By increasing the number of Fourier features to $r_{\mathrm{RFF}} = 500$ the performance can be significantly improved, cf. RFF-DKLMS (II). Nevertheless, this improvement comes with a huge increase in communication overhead since the number of Fourier features is equal to the dimension of the coefficient vectors to be exchanged. While DMKLMS exchanges vectors with $\bar{r}Q = 66$ entries only, the coefficient vectors in RFF-DKLMS (II) have $r_{\mathrm{RFF}} = 500$ entries. Thus, by relying on an a priori designed dictionary as in DMKLMS and D-CHYPASS, huge savings in communication overhead and computational complexity can be achieved. The enhanced performance by D-CHYPASS compared to the other multikernel approaches is due to a better metric in form of the $\boldsymbol{K}$-norm and the normalization factor $||\boldsymbol{K}^{-1}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k})||^2_{\boldsymbol{K}}$ in the projection $P^{\boldsymbol{K}}_{\mathcal{S}_{j,k}}(\boldsymbol{w}_{j,k})$ which adapts the step size $\mu_{j,k}$. By exploiting the projection w.r.t. the $\boldsymbol{K}$-norm the shape of the cost function $\Theta_{j,k}(\boldsymbol{w}_{j,k})$ is changed such that convergence speed is improved [54].

From Fig. 1, D-CHYPASS (I) and (II) show a similar performance with a negligible loss for D-CHYPASS (II). However, this minor loss comes with a huge reduction in complexity per node j. Since D-CHYPASS (II) projects onto a hyperslab with $\varepsilon_j = 0.5$ there is a higher probability that $\boldsymbol{w}_{j,k}$ is contained in $\mathcal{S}_{j,k}$ than in D-CHYPASS (I) where $\varepsilon_j = 0$. If $\boldsymbol{w}_{j,k} \in \mathcal{S}_{j,k}$, the vector $\boldsymbol{w}_{j,k}$ is not updated saving a significant amount of computations. In contrast, when using a hyperplane ($\varepsilon_j = 0$) the vector $\boldsymbol{w}_{j,k}$ has to be updated in each iteration. Fig. 3 shows the number of local APSM updates (19a) per node in logarithmic scale over the hyperslab threshold ε_j. Additionally, the NMSE averaged over the last 200 iterations in relation to the threshold is depicted. For thresholds $\varepsilon_j > 0$ a step size of $\mu_{j,k} = 0.5$ is used. We can observe that using hyperslab thresholds up to $\varepsilon_j = 0.5$ saves a huge amount of complexity while keeping the error performance constant. E.g. for D-CHYPASS (II) with $\varepsilon_j = 0.5$ in average 5,468 updates are executed per node. Compared to 15,000 updates for D-CHYPASS (I) a reduction of approximately 64% in computations can be achieved. This is crucial especially for sensors with low computational capability and limited battery life. However, from Fig. 3 it is also clear that the computational load cannot be arbitrarily reduced without degrading the reconstruction performance. This is visible especially for thresholds $\varepsilon_j > 1$.

In Fig. 4 we depict the contour plot of the true function $\psi(\boldsymbol{x})$ together with an exemplary set of node positions and the reconstructed function $\varphi(\boldsymbol{x})$ by D-CHYPASS (I). The reconstruction is shown for one node in the network at steady state. By virtue of the consensus averaging step each node in the network will have the same reconstruction. We can observe that both Gaussian functions are approximated with good accuracy. The peaks

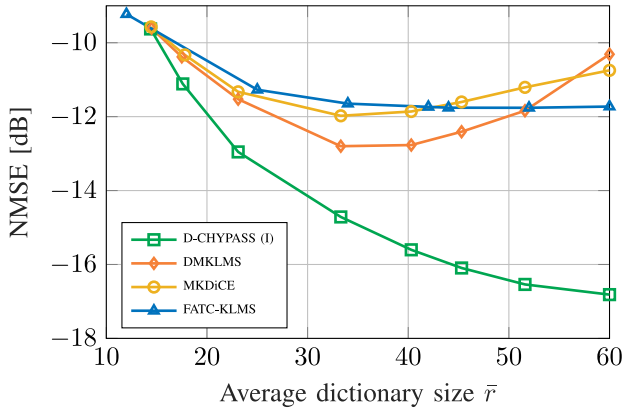

Fig. 5. NMSE over dictionary size for the reconstruction of multiple Gaussian functions.

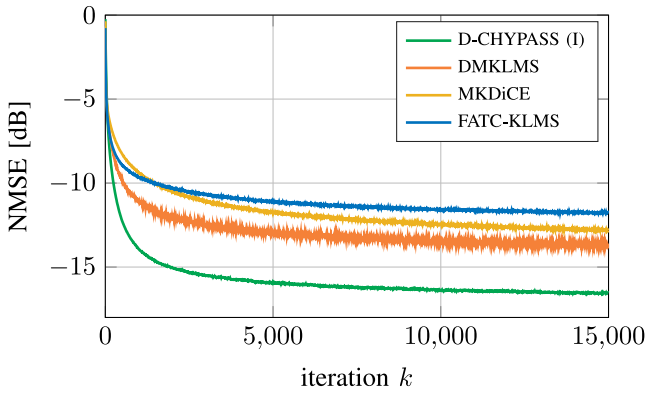

Fig. 6. Learning curves for the reconstruction of multiple Gaussian functions for a coherence threshold $\tau = 0.99$ corresponding to an average dictionary size of $\bar{r} = 53$ samples.

of both functions can be clearly distinguished. However, at outer regions some inaccuracies can still be seen. These are expected to be reduced when increasing the dictionary size.

Fig. 5 shows the error performance of the algorithms over the averaged dictionary size $\bar{r}$ for 200 trials. The NMSE values are calculated as an average over the last 200 iterations with again 15,000 iterations for each algorithm. We observe that D-CHYPASS (I) outperforms its competitors with growing dictionary size. In particular, DMKLMS and MKDiCE lose in performance for dictionary sizes $\bar{r} > 33$ while D-CHYPASS steadily improves its reconstruction. This is due to the reason that for DMKLMS and MKDiCE the step size has to be adjusted to the growing dictionary size to avoid an increasing steady-state error. In DMKLMS the step size is not normalized to the squared norm of the kernel vector $\boldsymbol{\kappa}(\boldsymbol{x})$ as in D-CHYPASS which can lead to divergence. Regarding FATC-KLMS a similar effect is expected to appear for higher dictionary sizes $\bar{r} > 60$ since it uses one kernel only. Therefore, in the range of 40 to 60 dictionary samples it performs better than DMKLMS and MKDiCE. To show that the performance of MKDiCE and DMKLMS can be improved, in Fig. 6 we depict the NMSE performance for an adapted step size over the iteration with $\tau = 0.99$. This coherence threshold results in an average dictionary size of

TABLE II
PARAMETER VALUES FOR EXPERIMENT IN SECTION VI-B

Algorithm	Parameters		
D-CHYPASS	$\mu_{j,k} = 0.5, \varepsilon_j = 0$	$\tau = 0.85$	$\zeta_1 = 0.06$
DMKLMS	$\mu_{j,k} = 0.05$		$\zeta_2 = 0.1$
MKDiCE	$\mu_{j,k} = 0.7$		
FATC-KLMS	$\mu_{j,k} = 0.05$	$\tau = 0.78$	$\zeta = 0.08$
RFF-DKLMS	$\mu_{j,k} = 0.2$	$r_{\mathrm{RFF}} = 200$	$\zeta = 0.08$

$\bar{r} = 53$, a point where MKDiCE and DMKLMS show degrading performance according to Fig. 5. The step sizes are chosen as $\mu_{j,k} = 0.05$ for DMKLMS and $\mu_{j,k} = 0.2$ for MKDiCE, respectively. We observe that by adjusting the step size to the dictionary size the steady-state performance at $\bar{r} = 53$ is improved compared to Fig. 5. Now, both MKDiCE and DMKLMS outperform FATC-KLMS.

Remark 3: Regarding D-CHYPASS (I) it should be noted that for $\tau > 0.98$ divergence was observed. This is caused by the inversion of an ill-conditioned kernel Gram matrix $\boldsymbol{K}$. This occurs if the dictionary employs node positions close to each other leading to linear dependency in $\boldsymbol{K}$. With a higher coherence threshold the probability of such a case increases. To numerically stabilize the inversion of $\boldsymbol{K}$ a scaled identity matrix $\gamma \boldsymbol{I}_{rQ}$ is added to the matrix as regularization. The matrix $\boldsymbol{K}$ in (16) is then substituted by $\boldsymbol{K} + \gamma \boldsymbol{I}_{rQ}$. For thresholds $\tau > 0.98$ a regularization parameter of $\gamma = 0.01$ was used in this experiment to achieve a stable performance.

B. Real Altitude Data

We apply D-CHYPASS to the reconstruction of real altitude data where each node measures the altitude at its position $\boldsymbol{x}_j$. For the data we use the ETOPO1 global relief model which is provided by the National Oceanic and Atmospheric Administration [55] and which exhibits several low/high frequency components. In the original data the position is given by the longitude and latitude and the corresponding altitude $\psi(\boldsymbol{x})$ is delivered for each such position. As in [56], we choose an area of 31×31 points with longitudes $\{138.5, 138.5 + \frac{1}{60}, \ldots, 139\}$ and latitudes $\{34.5, 34.5 + \frac{1}{60}, \ldots, 35\}$. However, for easier handling we map longitudes and latitudes to Cartesian coordinates in the unit-square area such that $\boldsymbol{x} \in [0, 1]^2$. We consider $J = 200$ nodes randomly placed over the described area. Nodes with a distance $D < 0.2$ to each other share a connection. We assume noise with $\sigma_n^2 = 0.3$. The coherence threshold is set such that each algorithm employs a dictionary of average size $\bar{r} = 105$ while the RFF-DKLMS uses $r_{\mathrm{RFF}} = 200$ Fourier features. The performances are averaged over 200 independent trials. Table II lists the chosen parameter values for the considered algorithms.

Fig. 7 depicts the NMSE performance over the iteration. Again D-CHYPASS outperforms the other algorithms in terms of convergence speed and steady-state error. Although DMKLMS performs very close to D-CHYPASS it can be observed that the convergence speed of D-CHYPASS is faster. FATC-KLMS and RFF-DKLMS perform worst since their

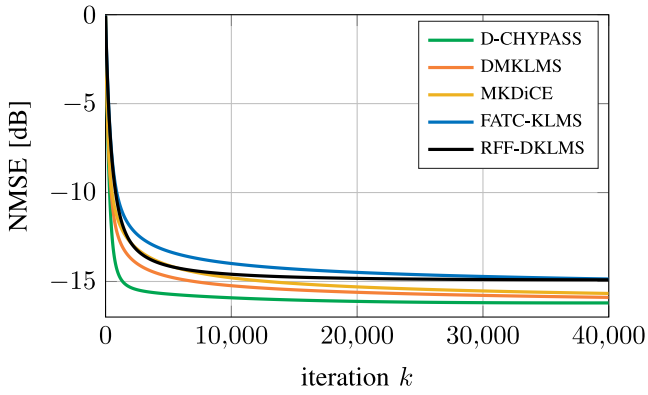

Fig. 7. Learning curves for the reconstruction of altitude data.

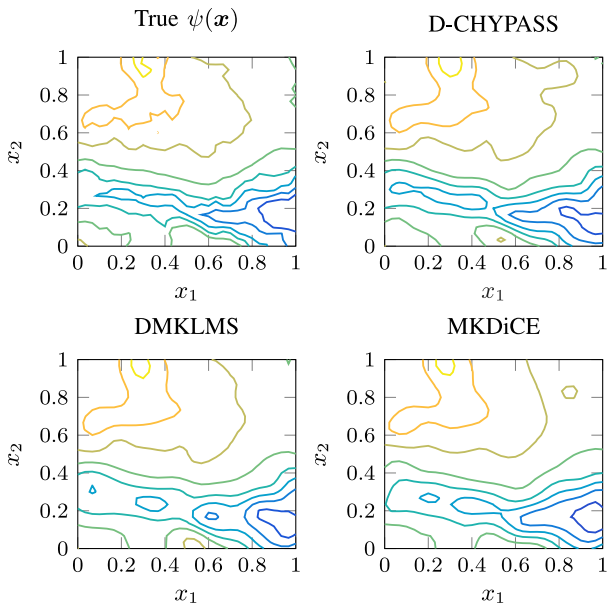

Fig. 8. Contour plots of the altitude reconstruction by one node for the D-CHYPASS, DMKLMS, and MKDiCE.

reconstruction capability is limited by the use of one kernel only. While RFF-DKLMS converges faster than FATC-KLMS it should be noted that it produces a higher communication overhead due to the use of $r_{\mathrm{RFF}} = 200$ Fourier features compared to $\bar{r} = 105$ dictionary samples in FATC-KLMS. The contour plots for the multikernel approaches at steady-state at one node are shown in Fig. 8. For the D-CHYPASS we can observe a good reconstruction of the original $\psi(\boldsymbol{x})$ although details in the area around $[0.4, 0.7]^{\mathsf{T}}$ and $[0.4, 0.3]^{\mathsf{T}}$ are missing. The reconstructions by DMKLMS and MKDiCE show a less accurate approximation especially in the areas around the valley $[0.4, 0.3]^{\mathsf{T}}$.

C. Time-Varying Nonlinear Function

In the following, we examine the tracking performance of the D-CHYPASS w.r.t. time-varying functions. To this end, we consider the following function being dependent on both the position $\boldsymbol{x}$ and time k:

$$\psi(\boldsymbol{x}, k) = 0.8 \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{p}_1||^2_{\mathbb{R}^2}}{2(1 - 0.5\sin(2\pi 10^{-3}k)) \cdot 0.3^2}\right) + \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{p}_2||^2_{\mathbb{R}^2}}{2(1 + 0.5\sin(2\pi 10^{-3}k)) \cdot 0.1^2}\right)$$

with $\boldsymbol{p}_1 = [0.6, 0.5]^{\mathsf{T}}$ and $\boldsymbol{p}_2 = [0.25, 0.3]^{\mathsf{T}}$. This function contains two Gaussian shapes whose bandwidths are expanding and shrinking over time k. We apply the D-CHYPASS to the reconstruction of the time-varying function $\psi(\boldsymbol{x}, k)$ and compare it to the MKDiCE and DMKLMS. We use a network of $J = 80$ nodes randomly distributed over the unit-square area and average the performance over 200 trials with a new network realization in each trial. The noise variance is $\sigma_n^2 = 0.3$. For the considered algorithms we set τ such that an average dictionary size of $\bar{r} = 36$ samples is achieved. We evaluate the D-CHYPASS with one and two kernels. Table III lists the chosen parameter values for the considered algorithms.

TABLE III
PARAMETER VALUES FOR EXPERIMENT IN SECTION VI-C

Algorithm	Parameters		
D-CHYPASS (I)	$\mu_{j,k} = 0.5$	$\tau = 0.95$	$\zeta_1 = 0.1, \zeta_2 = 0.3$
	$\varepsilon_j = 0$		
D-CHYPASS (II)	$\mu_{j,k} = 0.5$	$\tau = 0.9$	$\zeta_1 = 0.2$
	$\varepsilon_j = 0$		
DMKLMS	$\mu_{j,k} = 0.1$	$\tau = 0.95$	$\zeta_1 = 0.1, \zeta_2 = 0.3$
MKDICE	$\mu_{j,k} = 0.5$	$\tau = 0.95$	$\zeta_1 = 0.1, \zeta_2 = 0.3$

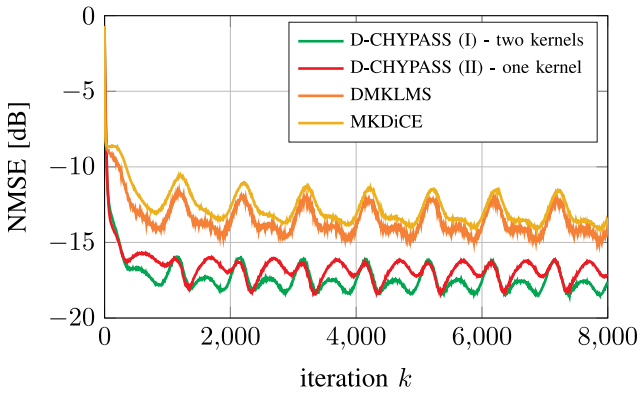

Fig. 9. NMSE performance over iteration number for the tracking of a time-varying function.

Fig. 9 shows the NMSE over the iteration number k. The fluctuations in the error curves are due to the time-varying bandwidths in $\psi(\boldsymbol{x}, k)$. For all algorithms these fluctuations stay in a specific error range illustrating that the function $\psi(\boldsymbol{x}, k)$ can be tracked within a certain range of accuracy. We observe that D-CHYPASS (I) and (II) significantly outperform the remaining algorithms. Additionally, the range of the fluctuations in the error is lower for D-CHYPASS compared to the other algorithms. It is also visible that utilizing two kernels in D-CHYPASS (I) improves the tracking performance compared to using one kernel as in D-CHYPASS (II). Nevertheless it is worth noting, that

TABLE IV
COMPUTATIONAL COMPLEXITY AND OVERHEAD OF ALGORITHMS

Algorithm	Complexity	Overhead
D-CHYPASS	$(2\|\mathcal{E}\| + J(L+4))\,Qr$ $+(Qr^2+2)J$	JQr
D-CHYPASS (selective update)	$\left((L+1)Qr + v_{\text{inv}}(s) + s^2 + 2\right)J$ $+(2\|\mathcal{E}\| + 3J)s$	
DMKLMS	$(2\|\mathcal{E}\| + J(L+4))\,Qr + J$	
FATC-KLMS	$(2\|\mathcal{E}\| + J(L+4))\,r + J$	Jr
MKDiCE	$(6\|\mathcal{E}\| + 4J + L + 2)\,Qr$ $+J\left(1 + (Qr)^2 + v_{\text{inv}}(Qr)\right)$	$2JQr$ $+2\|\mathcal{E}\|Qr$
RFF-DKLMS	$J(4r_{\text{RFF}} + 1) + (2\|\mathcal{E}\| + J)r_{\text{RFF}}$	Jr_{RFF}

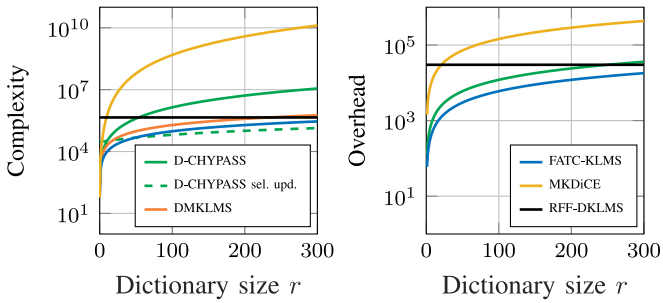

Fig. 10. Computational complexity and communication overhead of the algorithms per iteration k over the dictionary size r.

even with only one kernel the D-CHYPASS (II) outperforms the multikernel approaches DMKLMS and MKDiCE illustrating the significant gain by employing the $\boldsymbol{K}$-norm in the algorithm.

D. Computational Complexity and Communication Overhead

We analyze the complexities and communication overhead per iteration of the algorithms. For the complexities we consider the number of multiplications and assume that Gaussian kernels are used. Each dictionary $\mathcal{D}_q$ is designed a priori, stays fixed over time k and is common to all nodes. Therefore, the kernel Gram matrix $\boldsymbol{K}$ can be computed offline before the iterative process of D-CHYPASS avoiding an inversion in each iteration. Note that $\boldsymbol{K}$ is block diagonal such that Q inversions of $r \times r$ matrices have to be computed. This results in a complexity of order $\mathcal{O}(JQr^3)$ in the network before D-CHYPASS starts iterating. To further reduce the complexity of D-CHYPASS the selective-update strategy can be applied. It selects the s most coherent dictionary samples such that only s entries of the coefficient vector $\boldsymbol{w}_{j,k}$ are updated [47]. Usually, $s \leq 5$ so that $s \ll r$. Then per iteration k the inverse of an $s \times s$ matrix has to be computed while reducing the number of multiplications. For the overhead we count the number of transmitted scalars among all nodes. All algorithms except the MKDiCE use a consensus averaging step that causes broadcast transmissions only. The MKDiCE comprises also unicast transmissions of vectors which depend on the receiving node and which increase the overhead significantly. Table IV lists the complexities and overhead of the algorithms where the complexity for an inversion of a $p \times p$ matrix is denoted by $v_{\text{inv}}(p) := p^3$. Fig. 10 depicts the complexity and the overhead over the dictionary size r for $L = 2, s = 7, Q = 2$ and a network of $J = 60$ nodes with $|\mathcal{E}| = 300$ edges. The RFF-DKLMS with $r_{\text{RFF}} = 500$ is included as reference. Clearly, the complexity and overhead of the ADMM-based MKDiCE are highest among the algorithms due to the inversion of a $Qr \times Qr$ matrix per iteration k and the transmission of unicast vectors, respectively. Furthermore, for dictionary sizes up to $r = 50$ the D-CHYPASS has lower complexity than the RFF-DKLMS. By including the selective-update strategy the complexity of D-CHYPASS is significantly reduced and is even lower than the FATC-KLMS. D-CHYPASS and DMKLMS exhibit the same overhead per iteration which is lower compared to RFF-DKLMS for $r < 200$.

VII. CONCLUSION

We proposed an adaptive learning algorithm exploiting multiple kernels and projections onto hyperslabs for the regression of nonlinear functions in diffusion networks. We provided a thorough convergence analysis regarding monotone approximation, asymptotic minimization, consensus and the limit point of the algorithm. To this end, we introduced a novel modified consensus matrix which we proved to be identical to the ordinary consensus matrix. As an application example we investigated the proposed scheme for the reconstruction of spatial distributions by a network of nodes with both synthetic and real data. Note that it is not restricted to such a scenario and can be applied in general to any distributed nonlinear system identification task. Compared to the state of the art algorithms we could observe significant gains in error performance, convergence speed and stability over the employed dictionary size. In particular, our proposed APSM-based algorithm significantly outperformed an ADMM-based multikernel scheme (MKDiCE) in terms of error performance with highly decreased complexity and communication overhead. By embedding the hyperslab projection the computational demand per node could be drastically reduced over a certain range of thresholds while keeping the error performance constant.

APPENDIX A
DERIVATION IN CARTESIAN PRODUCT SPACE OF MULTIPLE RKHSS

A. Equivalent Functional Problem Formulation

Since ψ lies in the sum space $\mathcal{H}^+$ of Q RKHSs it is decomposable into the sum $\psi := \sum_{q\in\mathcal{Q}} \psi_{(q)}$ where $\psi_{(q)} \in \mathcal{H}_q$. To estimate ψ we utilize a multikernel adaptive filter φ following (2). Then φ can be expressed as a decomposable sum $\varphi := \sum_{q\in\mathcal{Q}} \varphi_{(q)} \in \mathcal{H}^+$ with $\varphi_{(q)} := \sum_{\ell=1}^{r} w_{q,\ell}\kappa_q(\cdot, \bar{\boldsymbol{x}}_\ell) \in \mathcal{H}_q$. Each monokernel filter $\varphi_{(q)}$ is an element of the *dictionary subspace* $\mathcal{M}_q := \operatorname{span}\{\kappa_q(\cdot, \bar{\boldsymbol{x}}_\ell)\}_{\ell=1}^{r}$. Hence, φ lies in the sum of Q dictionary subspaces

$$\mathcal{M}^+ := \mathcal{M}_1 + \mathcal{M}_2 + \cdots + \mathcal{M}_Q \subset \mathcal{H}^+. \tag{30}$$

To estimate ψ by each node j in the network, we first define the following hyperslab containing all functions φ of bounded local instantaneous error to the current measurement $d_{j,k}$ at node j

and time k:

$$\breve{\mathcal{S}}_{j,k} := \{\varphi \in \mathcal{M}^+ : |\langle \varphi, \kappa(\cdot, \boldsymbol{x}_{j,k})\rangle_{\mathcal{H}^+} - d_{j,k}| \leq \varepsilon_j\}, \quad (31)$$

where here $\kappa := \sum_{q=1}^{Q} \kappa_q$ is the reproducing kernel of the sum space $\mathcal{H}^+$. To each node j we assign a multikernel adaptive filter $\varphi_j \in \mathcal{H}^+$ as an individual estimate of ψ. To pursue an adaptive learning approach of ψ in the sum space $\mathcal{H}^+$ we define the local cost to be the metric distance between the estimate φ_j and its projection $P^{\mathcal{H}^+}_{\breve{\mathcal{S}}_{j,k}}(\varphi_j)$ onto the hyperslab $\breve{\mathcal{S}}_{j,k}$:

$$\breve{\Theta}_{j,k}(\varphi_j) = ||\varphi_j - P^{\mathcal{H}^+}_{\breve{\mathcal{S}}_{j,k}}(\varphi_j)||_{\mathcal{H}^+}. \quad (32)$$

Then we can formulate the following consensus-based optimization problem w.r.t. the estimates $\{\varphi_j\}_{j\in\mathcal{J}}$ in the sum space $\mathcal{H}^+$:

$$\min_{\{\varphi_j | j\in\mathcal{J}\}} \sum_{j\in\mathcal{J}} \breve{\Theta}_{j,k}(\varphi_j) \quad (33a)$$

$$\text{s.t. } \varphi_j = \varphi_i, \quad \forall i \in \mathcal{N}_j. \quad (33b)$$

However, in the sum space $\mathcal{H}^+$ the norm and inner product have no closed-form expressions, since functions might not be uniquely decomposable depending on the chosen kernels [32]. Thus, solving (33) with (32) in closed form in $\mathcal{H}^+$ is not possible. An alternative approach is to formulate the problem in the Cartesian product space $\mathcal{H}^\times$ defined as

$$\mathcal{H}^\times := \mathcal{H}_1 \times \mathcal{H}_2 \times \cdots \times \mathcal{H}_Q \quad (34a)$$

$$:= \{(f_1, f_2, \ldots, f_Q) : f_q \in \mathcal{H}_q, q \in \mathcal{Q}\}. \quad (34b)$$

In $\mathcal{H}^\times$ the non-unique decomposability issue does never occur since the tuple of functions is considered rather than their sum [32]. Its norm has the closed-form expression

$$||F||_{\mathcal{H}^\times} := \sqrt{\sum_{q\in\mathcal{Q}} ||f_{(q)}||^2_{\mathcal{H}_q}}, \forall F := (f_{(q)})_{q\in\mathcal{Q}} \in \mathcal{H}^\times. \quad (35)$$

Instead of the sum $\varphi = \sum_{q\in\mathcal{Q}} \varphi_{(q)}$ we consider the Q-tuple $\Phi := (\varphi_{(q)})_{q\in\mathcal{Q}}$ in $\mathcal{H}^\times$ with $\Phi : \mathcal{X} \to \mathbb{R}^Q$. In the same way for ψ we consider the Q-tuple $\Psi := (\psi_{(q)})_{q\in\mathcal{Q}}$. Since each monokernel filter $\varphi_{(q)}$ is an element of the dictionary subspace $\mathcal{M}_q$, the tuple Φ lies in the Cartesian product of Q dictionary subspaces

$$\mathcal{M}^\times := \mathcal{M}_1 \times \mathcal{M}_2 \times \cdots \times \mathcal{M}_Q \subset \mathcal{H}^\times. \quad (36)$$

To formulate (33) in the product space $\mathcal{H}^\times$, we design the hyperslab

$$\widetilde{\mathcal{S}}_{j,k} := \{\Phi \in \mathcal{M}^\times : |\langle \Phi, \tilde{\kappa}(\cdot, \boldsymbol{x}_{j,k})\rangle_{\mathcal{H}^\times} - d_{j,k}| \leq \varepsilon_j\} \quad (37)$$

for each node j containing all Φ s of bounded local instantaneous error at time instant k. Here, $\tilde{\kappa} : \mathcal{X}\times\mathcal{X} \to \mathbb{R}^Q, (\boldsymbol{x}, \boldsymbol{x}') \mapsto (\kappa_q(\boldsymbol{x}, \boldsymbol{x}'))_{q\in\mathcal{Q}}$ with $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ and $\tilde{\kappa}(\cdot, \boldsymbol{x}) := (\kappa_q(\cdot, \boldsymbol{x}))_{q\in\mathcal{Q}} \in \mathcal{H}^\times$. The estimated output for $\boldsymbol{x}_{j,k}$ is then given by $\varphi(\boldsymbol{x}_{j,k}) = \langle \Phi, \tilde{\kappa}(\cdot, \boldsymbol{x}_{j,k})\rangle_{\mathcal{H}^\times} = \sum_{q\in\mathcal{Q}} \langle \varphi_{(q)}, \kappa_q(\cdot, \boldsymbol{x}_{j,k})\rangle_{\mathcal{H}_q}$ using the reproducing property of each kernel. To each node j we assign a corresponding tuple Φ_j. By analogy with (32) we define the local cost function $\widetilde{\Theta}_{j,k}$ per node j in $\mathcal{H}^\times$ as

$$\widetilde{\Theta}_{j,k}(\Phi_j) = ||\Phi_j - P^{\mathcal{H}^\times}_{\widetilde{\mathcal{S}}_{j,k}}(\Phi_j)||_{\mathcal{H}^\times}. \quad (38)$$

The respective global optimization problem w.r.t. the functions $\{\Phi_j\}_{j\in\mathcal{J}}$ in the network reads

$$\min_{\{\Phi_j | j\in\mathcal{J}\}} \sum_{j\in\mathcal{J}} \widetilde{\Theta}_{j,k}(\Phi_j) \quad (39a)$$

$$\text{s.t. } \Phi_j = \Phi_i, \quad \forall i \in \mathcal{N}_j. \quad (39b)$$

Since the norm in $\mathcal{H}^\times$ has the closed-form expression (35), problem (39) is tractable. In general, an optimal solution of (33) in the sum space $\mathcal{H}^+$ corresponds to multiple different tuples in $\mathcal{H}^\times$ as points in $\mathcal{H}^+$ are not uniquely decomposable. In other words, every such optimal tuple in $\mathcal{H}^\times$ sums up to the same optimal solution in $\mathcal{H}^+$. Regarding the relation between the product and parameter spaces we note that the finite-dimensional dictionary subspace $(\mathcal{M}^\times, \langle\cdot,\cdot\rangle_{\mathcal{H}^\times})$ is isomorphic to the Euclidean parameter space $(\mathbb{R}^{rQ}, \langle\cdot,\cdot\rangle_{\boldsymbol{K}})$, see [47, Lemma 1]. In the parameter space the inner product of $\mathcal{H}^\times$ is preserved by $\langle\cdot,\cdot\rangle_{\boldsymbol{K}}$ and Φ_j is equivalent to the respective coefficient vector $\boldsymbol{w}_j \in \mathbb{R}^{rQ}$. Then, under the correspondence $\mathcal{M}^\times \ni \Phi_j \longleftrightarrow \boldsymbol{w}_j \in \mathbb{R}^{rQ}$ problem (39) is the equivalent formulation in the product space $\mathcal{H}^\times$ of problem (10) such that $\widetilde{\Theta}_{j,k}(\Phi_j) = \Theta_{j,k}(\boldsymbol{w}_j)$ holds.

B. Derivation of Local APSM Update

For the derivation of the local update in D-CHYPASS based on (39) we first consider the local APSM update for $\Phi_{j,k}$ under the assumption that $\Phi_{j,k} \notin \widetilde{\mathcal{S}}_{j,k}$:

$$\Phi'_{j,k+1} = \Phi_{j,k} - \mu_{j,k} \frac{\widetilde{\Theta}_{j,k}(\Phi_{j,k}) - \widetilde{\Theta}^\star_{j,k}}{||\widetilde{\Theta}'_{j,k}(\Phi_{j,k})||^2_{\mathcal{H}^\times}} \widetilde{\Theta}'_{j,k}(\Phi_{j,k}) \quad (40)$$

A subgradient of $\widetilde{\Theta}_{j,k}(\Phi_{j,k})$ is given by

$$\widetilde{\Theta}'_{j,k}(\Phi_{j,k}) := \frac{\Phi_{j,k} - P^{\mathcal{H}^\times}_{\widetilde{\mathcal{S}}_{j,k}}(\Phi_{j,k})}{||\Phi_{j,k} - P^{\mathcal{H}^\times}_{\widetilde{\mathcal{S}}_{j,k}}(\Phi_{j,k})||_{\mathcal{H}^\times}}. \quad (41)$$

Inserting (38) and (41) into the APSM update (40) we achieve

$$\Phi'_{j,k+1} = \Phi_{j,k} - \mu_{j,k}\left(\Phi_{j,k} - P^{\mathcal{H}^\times}_{\widetilde{\mathcal{S}}_{j,k}}(\Phi_{j,k})\right). \quad (42)$$

The projection $P^{\mathcal{H}^\times}_{\widetilde{\mathcal{S}}_{j,k}}(\Phi_{j,k})$ can be calculated by [32]

$$P^{\mathcal{H}^\times}_{\widetilde{\mathcal{S}}_{j,k}}(\Phi) = \begin{cases} \Phi, & \text{if } \Phi \in \widetilde{\mathcal{S}}_{j,k} \\ \Phi - \dfrac{\Phi(\boldsymbol{x}_{j,k}) - d_{j,k} - \varepsilon_j}{\sum_{q\in\mathcal{Q}} ||P^{\mathcal{H}_q}_{\mathcal{M}_q}(\kappa_q(\cdot, \boldsymbol{x}_{j,k}))||^2_{\mathcal{H}_q}} \\ \times \sum_{q\in\mathcal{Q}} P^{\mathcal{H}_q}_{\mathcal{M}_q}(\kappa_q(\cdot, \boldsymbol{x}_{j,k})), \\ \quad \text{if} \quad \Phi(\boldsymbol{x}_{j,k}) > d_{j,k} + \varepsilon_j \\ \Phi - \dfrac{\Phi(\boldsymbol{x}_{j,k}) - d_{j,k} + \varepsilon_j}{\sum_{q\in\mathcal{Q}} ||P^{\mathcal{H}_q}_{\mathcal{M}_q}(\kappa_q(\cdot, \boldsymbol{x}_{j,k}))||^2_{\mathcal{H}_q}} \\ \times \sum_{q\in\mathcal{Q}} P^{\mathcal{H}_q}_{\mathcal{M}_q}(\kappa_q(\cdot, \boldsymbol{x}_{j,k})), \\ \quad \text{if} \quad \Phi(\boldsymbol{x}_{j,k}) < d_{j,k} - \varepsilon_j \end{cases} \quad (43)$$

where $P^{\mathcal{H}_q}_{\mathcal{M}_q}$ is the projection operator onto the dictionary subspace $\mathcal{M}_q$. By virtue of [32, Lemma 1] it holds that $P^{\mathcal{H}_q}_{\mathcal{M}_q}(\kappa_q(\cdot, \boldsymbol{x}_{j,k})) = \sum_{\ell=1}^{r} \alpha^{(q)}_{j,\ell} \kappa_q(\bar{\boldsymbol{x}}_\ell, \cdot)$ with the coefficients

$\boldsymbol{\alpha}_j^{(q)} := [\alpha_{j,1}^{(q)}, \ldots, \alpha_{j,r}^{(q)}]^\mathsf{T} = \boldsymbol{K}_q^{-1}\boldsymbol{\kappa}_q(\boldsymbol{x}_{j,k})$. The squared norm in the denominator of the projection $P_{\widetilde{\mathcal{S}}_{j,k}}^{\mathcal{H}^\times}(\Phi)$ yields

$$||P_{\mathcal{M}_q}^{\mathcal{H}_q}(\kappa_q(\cdot,\boldsymbol{x}_{j,k}))||_{\mathcal{H}_q}^2 = \langle P_{\mathcal{M}_q}^{\mathcal{H}_q}(\kappa_q(\cdot,\boldsymbol{x}_{j,k})), \kappa_q(\cdot,\boldsymbol{x}_{j,k})\rangle_{\mathcal{H}_q} = \sum_{\ell=1}^{r}\alpha_{j,\ell}^{(q)}\kappa_q(\bar{\boldsymbol{x}}_\ell,\boldsymbol{x}_{j,k}).$$

Thus, the denominators in (43) can be computed by

$$\sum_{q\in\mathcal{Q}}||P_{\mathcal{M}_q}^{\mathcal{H}_q}(\kappa_q(\cdot,\boldsymbol{x}_{j,k}))||_{\mathcal{H}_q}^2 = \sum_{q\in\mathcal{Q}}\sum_{\ell=1}^{r}\alpha_{j,\ell}^{(q)}\kappa_q(\bar{\boldsymbol{x}}_\ell,\boldsymbol{x}_{j,k}) = ||\boldsymbol{K}^{-1}\boldsymbol{\kappa}(\boldsymbol{x}_{j,k})||_{\boldsymbol{K}}^2.$$

By parameterizing each Φ_j in terms of $\boldsymbol{w}_j$ and assuming a fixed dictionary $\mathcal{D}_q$ for each kernel κ_q we arrive at the update equation presented in (19a) with the projection (16).

APPENDIX B
PROOF OF LEMMA 2

Let us consider the squared $\mathcal{K}$-norm of $\widehat{\boldsymbol{P}}$:

$$||\widehat{\boldsymbol{P}}||_{\mathcal{K}}^2 := \max_{\boldsymbol{x}\neq\boldsymbol{0}}\frac{||\widehat{\boldsymbol{P}}\boldsymbol{x}||_{\mathcal{K}}^2}{||\boldsymbol{x}||_{\mathcal{K}}^2} = \max_{\boldsymbol{x}\neq\boldsymbol{0}}\frac{\boldsymbol{x}^\mathsf{T}\widehat{\boldsymbol{P}}^\mathsf{T}\mathcal{K}\widehat{\boldsymbol{P}}\boldsymbol{x}}{\boldsymbol{x}^\mathsf{T}\mathcal{K}\boldsymbol{x}}$$

We assume $\mathcal{K}$ to be non-singular. Hence, $\mathcal{K}^{-1/2}$ exists and we can insert $\widehat{\boldsymbol{P}} = \mathcal{K}^{-1/2}\boldsymbol{P}\mathcal{K}^{1/2}$:

$$||\widehat{\boldsymbol{P}}||_{\mathcal{K}}^2 = \max_{\boldsymbol{x}\neq\boldsymbol{0}}\frac{\boldsymbol{x}^\mathsf{T}\mathcal{K}^{1/2}\boldsymbol{P}^\mathsf{T}\mathcal{K}^{-1/2}\mathcal{K}\mathcal{K}^{-1/2}\boldsymbol{P}\mathcal{K}^{1/2}\boldsymbol{x}}{\boldsymbol{x}^\mathsf{T}\mathcal{K}\boldsymbol{x}} = \max_{\boldsymbol{y}\neq\boldsymbol{0}}\frac{\boldsymbol{y}^\mathsf{T}\boldsymbol{P}^\mathsf{T}\boldsymbol{P}\boldsymbol{y}}{\boldsymbol{y}^\mathsf{T}\boldsymbol{y}} \text{ with } \boldsymbol{y} = \mathcal{K}^{1/2}\boldsymbol{x}.$$

By definition of the consensus matrix $\boldsymbol{P}$ it follows that $||\widehat{\boldsymbol{P}}||_{\mathcal{K}}^2 = 1$. We now show that the modified consensus matrix $\widehat{\boldsymbol{P}}$ is identical to $\boldsymbol{P}$. Assume that $\boldsymbol{P}$ is compatible to the graph $\mathcal{G}$ of any connected, undirected network via the matrix $\boldsymbol{G}\in\mathbb{R}^{J\times J}$. Then, it holds that $\boldsymbol{P} = \boldsymbol{G}\otimes\boldsymbol{I}_{rQ}$. By examining the definition of $\widehat{\boldsymbol{P}}$ we find

$$\begin{aligned}\widehat{\boldsymbol{P}} &= \mathcal{K}^{-1/2}\boldsymbol{P}\mathcal{K}^{1/2} = \mathcal{K}^{-1/2}(\boldsymbol{G}\otimes\boldsymbol{I}_{rQ})\mathcal{K}^{1/2}\\ &= \begin{bmatrix}\boldsymbol{K}^{-1/2} & & \boldsymbol{0}\\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{K}^{-1/2}\end{bmatrix}\\ &\quad\times\begin{bmatrix}g_{11}\boldsymbol{I}_{rQ} & \ldots & g_{1J}\boldsymbol{I}_{rQ}\\ \vdots & \ddots & \vdots\\ g_{J1}\boldsymbol{I}_{rQ} & \ldots & g_{JJ}\boldsymbol{I}_{rQ}\end{bmatrix}\begin{bmatrix}\boldsymbol{K}^{1/2} & & \boldsymbol{0}\\ & \ddots & \\ \boldsymbol{0} & & \boldsymbol{K}^{1/2}\end{bmatrix}\\ &= \begin{bmatrix}g_{11}\boldsymbol{K}^{-1/2}\boldsymbol{I}_{rQ}\boldsymbol{K}^{1/2} & \ldots & g_{1J}\boldsymbol{K}^{-1/2}\boldsymbol{I}_{rQ}\boldsymbol{K}^{1/2}\\ \vdots & \ddots & \vdots\\ g_{J1}\boldsymbol{K}^{-1/2}\boldsymbol{I}_{rQ}\boldsymbol{K}^{1/2} & \ldots & g_{JJ}\boldsymbol{K}^{-1/2}\boldsymbol{I}_{rQ}\boldsymbol{K}^{1/2}\end{bmatrix}\\ &= \boldsymbol{G}\otimes\boldsymbol{I}_{rQ} = \boldsymbol{P}\end{aligned}$$

Thus, matrices $\widehat{\boldsymbol{P}}$ and $\boldsymbol{P}$ are equivalent to each other.

APPENDIX C
PROOF OF THEOREM 1

A. Proof of Theorem 1.4

We mimic the proof of [49, Th. 2.3] to show the convergence of $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$. From Theorem 1.1 we know that the sequence $(||\boldsymbol{z}_k - \boldsymbol{z}^\star||_{\mathcal{K}}^2)_{k\in\mathbb{N}}$ converges for every $\boldsymbol{z}^\star = [(\boldsymbol{w}^\star)^\mathsf{T},\ldots,(\boldsymbol{w}^\star)^\mathsf{T}]^\mathsf{T}$ where $\boldsymbol{w}^\star\in\Upsilon^\star$. Thus, the sequence $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$ is bounded and every subsequence of $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$ has an accumulation point. Then, according to the Bolzano-Weierstrass Theorem the bounded real sequence $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$ has a convergent subsequence $(\boldsymbol{z}_{k_l})_{k_l\in\mathbb{N}}$. Let $\widehat{\boldsymbol{z}}$ be the unique accumulation point of $(\boldsymbol{z}_{k_l})_{k_l\in\mathbb{N}}$. With $\lim_{k\to\infty}(\boldsymbol{I}_{rQJ} - \boldsymbol{B}\boldsymbol{B}^\mathsf{T})\boldsymbol{z}_k = \boldsymbol{0}$ it follows that

$$\lim_{k\to\infty}(\boldsymbol{I}_{rQJ} - \boldsymbol{B}\boldsymbol{B}^\mathsf{T})\boldsymbol{z}_{k_l} = (\boldsymbol{I}_{rQJ} - \boldsymbol{B}\boldsymbol{B}^\mathsf{T})\widehat{\boldsymbol{z}} = \boldsymbol{0}.$$

Hence, $\widehat{\boldsymbol{z}}$ lies in the consensus subspace $\mathcal{C}$. To show that this point is a unique accumulation point suppose the contrary, i.e., $\widehat{\boldsymbol{z}} = [\widehat{\boldsymbol{w}}^\mathsf{T},\ldots,\widehat{\boldsymbol{w}}^\mathsf{T}]^\mathsf{T}\in\mathcal{C}$ and $\tilde{\boldsymbol{z}} = [\tilde{\boldsymbol{w}}^\mathsf{T},\ldots,\tilde{\boldsymbol{w}}^\mathsf{T}]^\mathsf{T}\in\mathcal{C}$ are two different accumulation points. For every $\boldsymbol{z}^\star$ the sequence $(||\boldsymbol{z}_k - \boldsymbol{z}^\star||_{\mathcal{K}}^2)_{k\in\mathbb{N}}$ converges and hence it follows that

$$\begin{aligned}0 &= ||\widehat{\boldsymbol{z}} - \boldsymbol{z}^\star||_{\mathcal{K}}^2 - ||\tilde{\boldsymbol{z}} - \boldsymbol{z}^\star||_{\mathcal{K}}^2\\ &= ||\widehat{\boldsymbol{z}}||_{\mathcal{K}}^2 - ||\tilde{\boldsymbol{z}}||_{\mathcal{K}}^2 - 2(\widehat{\boldsymbol{z}} - \tilde{\boldsymbol{z}})^\mathsf{T}\mathcal{K}\boldsymbol{z}^\star\\ &= ||\widehat{\boldsymbol{z}}||_{\mathcal{K}}^2 - ||\tilde{\boldsymbol{z}}||_{\mathcal{K}}^2 - 2J(\widehat{\boldsymbol{w}} - \tilde{\boldsymbol{w}})^\mathsf{T}\boldsymbol{K}\boldsymbol{w}^\star.\end{aligned}$$

It thus holds that $\boldsymbol{w}^\star\in H := \{\boldsymbol{w}\,|\,2J(\widehat{\boldsymbol{w}} - \tilde{\boldsymbol{w}})\boldsymbol{K}\boldsymbol{w} = ||\widehat{\boldsymbol{z}}||_{\mathcal{K}}^2 - ||\tilde{\boldsymbol{z}}||_{\mathcal{K}}^2\}$ where $\widehat{\boldsymbol{w}} - \tilde{\boldsymbol{w}}\neq 0\,(\Leftrightarrow\widehat{\boldsymbol{z}}\neq\tilde{\boldsymbol{z}})$. Since we assume that $\boldsymbol{w}^\star\in\Upsilon^\star$ this implies that $\Upsilon^\star$ is a subset of the hyperplane H. This contradicts the assumption of a nonempty interior of $\Upsilon^\star$. Hence, the bounded sequence $(\boldsymbol{z}_k)_{k\in\mathbb{N}}$ has a unique accumulation point, and so it converges.

B. Proof of Theorem 1.5

In this proof we mimic the proof of [27, Th. 2(d)], [50, Th. 3.1.4] and [8, Th. 2(e)] to characterize the limit point $\widehat{\boldsymbol{w}}$ of the sequence $(\boldsymbol{w}_{j,k})_{k\in\mathbb{N}}, \forall j\in\mathcal{J}$. Furthermore, we need [27, Claim 2] which is proven for any real Hilbert space and thus, also holds for the $\boldsymbol{K}$-metric Euclidean space.

Fact 1 ([27, Claim 2]): Let $C\subset\mathbb{R}^{rQ}$ be a nonempty closed convex set. Suppose that $\rho>0$ and $\tilde{\boldsymbol{u}}$ satisfies $\{\boldsymbol{v}\in\mathbb{R}^{rQ}\,|\,||\boldsymbol{v} - \tilde{\boldsymbol{u}}||_{\boldsymbol{K}}\leq\rho\}\subset C$. Assume $\boldsymbol{w}\in\mathbb{R}^{rQ}/C$ and $t\in(0,1)$ such that $\boldsymbol{u}_t := t\boldsymbol{w} + (1-t)\tilde{\boldsymbol{u}}\notin C$. Then $\mathrm{d}_{\boldsymbol{K}}(\boldsymbol{w},C) > \rho\frac{1-t}{t} = \rho\frac{||\boldsymbol{u}_t - \boldsymbol{w}||_{\boldsymbol{K}}}{||\boldsymbol{u}_t - \tilde{\boldsymbol{u}}||_{\boldsymbol{K}}} > 0$ with $\mathrm{d}_{\boldsymbol{K}}(\boldsymbol{w},C) := ||\boldsymbol{w} - P_C^{\boldsymbol{K}}(\boldsymbol{w})||_{\boldsymbol{K}}$.

Assume the contrary of our statement, i.e., $\widehat{\boldsymbol{w}}\notin\liminf_{k\to\infty}\Upsilon_k$. Denote by $\tilde{\boldsymbol{u}}$ an interior point of $\Upsilon^\star$. Therefore, there exists $\rho>0$ such that $\{\boldsymbol{v}\in\mathbb{R}^{rQ}\,|\,||\boldsymbol{v} - \tilde{\boldsymbol{u}}||_{\boldsymbol{K}}\leq\rho\}\subset\Upsilon^\star$. Furthermore, there exists $t\in(0,1)$ such that $\boldsymbol{u}_t := t\widehat{\boldsymbol{w}} + (1-t)\tilde{\boldsymbol{u}}\notin\Upsilon\supset\liminf_{k\to\infty}\Upsilon_k$. Since $\lim_{k\to\infty}\boldsymbol{w}_{j,k} = \widehat{\boldsymbol{w}}\,(\forall j\in\mathcal{J})$ there exists $N_1\in\mathbb{N}$ such that $||\boldsymbol{w}_{j,k} - \widehat{\boldsymbol{w}}||_{\boldsymbol{K}}\leq\rho\frac{1-t}{2t}, \forall k\geq N_1, \forall j\in\mathcal{J}$. Then, by $\boldsymbol{u}_t\notin\liminf_{k\to\infty}\Upsilon_k$ for any $L_1 > N_1$ there exists $k_1\geq L_1$ satisfying $\boldsymbol{u}_t\notin\Upsilon_{k_1} = \bigcap_{j\in\mathcal{J}}(\mathrm{lev}_{\leq 0}\Theta_{j,k_1})$. It follows that there exists a node $i\in\mathcal{J}$ such that $\boldsymbol{u}_t\notin\mathrm{lev}_{\leq 0}\Theta_{i,k_1}$. By $\Upsilon\subset\Upsilon_k\subset\mathrm{lev}_{\leq 0}\Theta_{i,k_1}$ and Fact 1

for node i it holds that

$$\begin{aligned}\mathrm{d}_{\boldsymbol{K}}(\boldsymbol{w}_{i,k_1}, \mathrm{lev}_{\leq 0}\Theta_{i,k_1}) &\geq \mathrm{d}_{\boldsymbol{K}}(\widehat{\boldsymbol{w}}, \mathrm{lev}_{\leq 0}\Theta_{i,k_1}) \\ &\quad - ||\boldsymbol{w}_{i,k_1} - \widehat{\boldsymbol{w}}||_{\boldsymbol{K}} \\ &\geq \rho\frac{1-t}{t} - \frac{\rho}{2}\frac{1-t}{t} \\ &= \frac{\rho}{2}\frac{1-t}{t} =: \epsilon > 0\end{aligned}$$

Thus, it follows that $\sum_{j\in\mathcal{J}} \mathrm{d}_{\boldsymbol{K}}(\boldsymbol{w}_{j,k_1}, \mathrm{lev}_{\leq 0}\Theta_{j,k_1}) \geq \epsilon$. By the triangle inequality we have

$$\begin{aligned}||\tilde{\boldsymbol{u}} - \boldsymbol{w}_{j,k_1}||_{\boldsymbol{K}} &\leq ||\tilde{\boldsymbol{u}} - \widehat{\boldsymbol{w}}||_{\boldsymbol{K}} + ||\boldsymbol{w}_{j,k_1} - \widehat{\boldsymbol{w}}||_{\boldsymbol{K}} \\ &\leq ||\tilde{\boldsymbol{u}} - \widehat{\boldsymbol{w}}||_{\boldsymbol{K}} + \frac{\rho}{2}\frac{1-t}{t} \quad (j \in \mathcal{J})\end{aligned}$$

so that

$$\sum_{j\in\mathcal{J}} ||\tilde{\boldsymbol{u}} - \boldsymbol{w}_{j,k_1}||_{\boldsymbol{K}} \leq J||\tilde{\boldsymbol{u}} - \widehat{\boldsymbol{w}}||_{\boldsymbol{K}} + J\frac{\rho}{2}\frac{1-t}{t} =: \eta > 0.$$

Given a fixed $L_2 > k_1$, we can find a $k_2 \geq L_2$ such that $\sum_{j\in\mathcal{J}} \mathrm{d}_{\boldsymbol{K}}(\boldsymbol{w}_{j,k_2}, \mathrm{lev}_{\leq 0}\Theta_{j,k_2}) \geq \epsilon$ and $\sum_{j\in\mathcal{J}} ||\tilde{\boldsymbol{u}} - \boldsymbol{w}_{j,k_2}||_{\boldsymbol{K}} \leq \eta$. Thus, we can construct a subsequence $\{k_l\}_{l=1}^{\infty}$ satisfying

$$\sum_{j\in\mathcal{J}} \mathrm{d}_{\boldsymbol{K}}(\boldsymbol{w}_{j,k_l}, \mathrm{lev}_{\leq 0}\Theta_{j,k_l}) \geq \epsilon \text{ and } \sum_{j\in\mathcal{J}} ||\tilde{\boldsymbol{u}} - \boldsymbol{w}_{j,k_l}||_{\boldsymbol{K}} \leq \eta.$$

With the assumptions of the theorem there exists a $\xi > 0$ such that $\sum_{j\in\mathcal{J}} \Theta_{j,k_l}(\boldsymbol{w}_{j,k_l}) \geq \xi$ for every $l \geq 1$. However, this contradicts $\lim_{k\to\infty} \Theta_{j,k}(\boldsymbol{w}_{j,k}) = 0, \forall j \in \mathcal{J}$ from Theorem 1.2. Thus, it follows that $\widehat{\boldsymbol{w}} \in \overline{\liminf_{k\to\infty} \Upsilon_k}$ and the proof is complete.

REFERENCES

[1] M. U. Ilyas, M. Zubair Shafiq, A. X. Liu, and H. Radha, "A distributed algorithm for identifying information hubs in social networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 9, pp. 629–640, Sep. 2013.

[2] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "SensorScope," *ACM Trans. Sens. Netw.*, vol. 6, no. 2, pp. 1–32, 2010.

[3] F. Facchinei, S. Sagratella, and G. Scutari, "Parallel selective algorithms for nonconvex big data optimization," *IEEE Trans. Signal Process.*, vol. 63, no. 7, pp. 1874–1889, Apr. 2015.

[4] A. H. Sayed, "Adaptive networks," *Proc. IEEE*, vol. 102, no. 4, pp. 460–497, Apr. 2014.

[5] H. Paul, J. Fliege, and A. Dekorsy, "In-network-processing: Distributed consensus-based linear estimation," *IEEE Commun. Lett.*, vol. 17, no. 1, pp. 59–62, Jan. 2013.

[6] G. Mateos and G. B. Giannakis, "Distributed recursive least-squares: Stability and performance analysis," *IEEE Trans. Signal Process.*, vol. 60, no. 7, pp. 3740–3754, Jul. 2012.

[7] S. Tu and A. H. Sayed, "Mobile adaptive networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 649–664, Aug. 2011.

[8] R. L. G. Cavalcante, A. Rogers, N. R. Jennings, and I. Yamada, "Distributed asymptotic minimization of sequences of convex functions by a broadcast adaptive subgradient method," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 1–37, Aug. 2011.

[9] F. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1035–1048, Mar. 2010.

[10] R. L. G. Cavalcante, I. Yamada, and B. Mulgrew, "An adaptive projected subgradient approach to learning in diffusion networks," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2762–2774, Jul. 2009.

[11] H. Zhu, A. Cano, and G. Giannakis, "Distributed consensus-based demodulation: Algorithms and error analysis," *IEEE Trans. Wireless Commun.*, vol. 9, no. 6, pp. 2044–2054, Jun. 2010.

[12] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Distributed LMS for consensus-based in-network adaptive processing," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2365–2382, Jun. 2009.

[13] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[14] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *Ann. Statist.*, vol. 36, no. 3, pp. 1171–1220, 2008.

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

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

[17] S. Van Vaerenbergh, J. Via, and I. Santamaria, "A sliding-window kernel RLS algorithm and its application to nonlinear channel identification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2006, pp. 789–772.

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

[19] W. Liu, I. M. Park, Y. Wang, and J. C. Príncipe, "Extended kernel recursive least squares algorithm," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 3801–3814, Oct. 2009.

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

[21] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering*. New York, NY, USA: Wiley, 2010.

[22] S. Van Vaerenbergh, M. Lazaro-Gredilla, and I. Santamaria, "Kernel recursive least-squares tracker for time-varying regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1313–1326, Aug. 2012.

[23] M. Yukawa and R.-I. Ishii, "An efficient kernel adaptive filtering algorithm using hyperplane projection along affine subspace," in *Proc. 20th Eur. Signal Process. Conf.*, 2012, pp. 2183–2187.

[24] W. Gao, J. Chen, C. Richard, J. Huang, and R. Flamary, "Kernel LMS algorithm with forward-backward splitting for dictionary learning," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 5735–5739.

[25] W. Gao and C. Richard, "Convex combinations of kernel adaptive filters," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, 2014, pp. 1–5.

[26] M. Takizawa and M. Yukawa, "Adaptive nonlinear estimation based on parallel projection along affine subspaces in reproducing kernel Hilbert space," *IEEE Trans. Signal Process.*, vol. 63, no. 16, pp. 4257–4269, Aug. 2015.

[27] 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.

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

[29] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 97–123, Jan. 2011.

[30] M. Yukawa and K. R. Müller, "Why does a Hilbertian metric work efficiently in online learning with kernels?" *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1424–1428, Oct. 2016.

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

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

[33] B.-S. Shin, H. Paul, and A. Dekorsy, "Distributed kernel least squares for nonlinear regression applied to sensor networks," in *Proc. 24th Eur. Signal Process. Conf.*, 2016, pp. 1588–1592.

[34] B.-S. Shin, H. Paul, M. Yukawa, and A. Dekorsy, "Distributed nonlinear regression using in-network processing with multiple Gaussian kernels," in *Proc. IEEE 18th Int. Workshop Signal Process. Adv. Wireless Commun.*, 2017, pp. 1–5.

[35] W. Gao, J. Chen, C. Richard, and J. Huang, "Diffusion adaptation over networks with kernel least-mean-square," in *Proc. IEEE 6th Int. Workshop Comput. Adv. Multi-Sens. Adapt. Process.*, 2015, pp. 217–220.

[36] P. Bouboulis, S. Chouvardas, and S. Theodoridis, "Online distributed learning over networks in RKH spaces using random Fourier features," *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1920–1932, Apr. 2018.

[37] S. Chouvardas and M. Draief, "A diffusion kernel LMS algorithm for nonlinear adaptive networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2016, pp. 4164–4168.
[38] P. Honeine, C. Richard, J. C. M. Bermudez, H. Snoussi, M. Essoloh, and F. Vincent, "Functional estimation in Hilbert space for distributed learning in wireless sensor networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2009, pp. 2861–2864.
[39] P. Honeine, C. Richard, J. C. M. Bermudez, J. Chen, and H. Snoussi, "A decentralized approach for nonlinear prediction of time series data in sensor networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2010, 2010, Art. no. 12.
[40] J. Predd, S. Kulkarni, and H. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 56–69, Jul. 2006.
[41] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *J. Mach. Learn. Res.*, vol. 11, pp. 1663–1707, 2010.
[42] B.-S. Shin, M. Yukawa, R. L. G. Cavalcante, and A. Dekorsy, "A hybrid dictionary approach for distributed kernel adaptive filtering in diffusion networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 3414–3418.
[43] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.
[44] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2013.
[45] H. Stark and Y. Yang, *Vector Space Projections: A Numerical Approach to Signal and Image Processing, Neural Nets, and Optics*. New York, NY, USA: Wiley, 1998.
[46] D. G. Luenberger, *Optimization by Vector Space Methods*, vol. 41. New York, NY, USA: Wiley, 1969.
[47] M. Takizawa and M. Yukawa, "Efficient dictionary-refining kernel adaptive filter with fundamental insights," *IEEE Trans. Signal Process.*, vol. 64, no. 16, pp. 4337–4350, Aug. 2016.
[48] I. Yamada, K. Slavakis, and K. Yamada, "An efficient robust adaptive filtering algorithm based on parallel subgradient projection techniques," *IEEE Trans. Signal Process.*, vol. 50, no. 5, pp. 1091–1101, May 2002.
[49] R. L. G. Cavalcante and S. Stanczak, "A distributed subgradient method for dynamic convex optimization problems under noisy information exchange," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 2, pp. 243–256, Apr. 2013.
[50] K. Slavakis, I. Yamada, and N. Ogura, "The adaptive projected subgradient method over the fixed point set of strongly attracting nonexpansive mappings," *Numer. Funct. Anal. Optim.*, vol. 27, nos. 7/8, pp. 905–930, 2006.
[51] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Syst. Control Lett.*, vol. 53, no. 1, pp. 65–78, 2004.
[52] S. Boyd and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. 4th Int. Symp. Inf. Process. Sens. Netw.*, 2005, pp. 63–70.
[53] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *J. Parallel Distrib. Comput.*, vol. 67, no. 1, pp. 33–46, 2007.
[54] M. Yukawa, K. Slavakis, and I. Yamada, "Adaptive parallel quadratic-metric projection algorithms," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 5, pp. 1665–1680, Jul. 2007.
[55] C. Amante and B. Eakins, "ETOPO1 1 arc-minute global relief model: Procedures, data sources and analysis," NOAA Tech. Memorandum NESDIS NGDC-24, Nat. Geophys. Data Center, NOAA, Boulder, CO, USA, 2009.
[56] O. Toda and M. Yukawa, "Online model-selection and learning for nonlinear estimation based on multikernel adaptive filtering," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E100-A, no. 1, pp. 236–250, 2017.

Ban-Sok Shin (S'13) received the Dipl.-Ing. (M.Sc.) degree from the University of Bremen, Bremen, Germany, in 2013. Since then, he has been a Research Assistant with the Department of Communications Engineering, University of Bremen, where he is currently working toward the Ph.D. degree. His research interests include distributed inference/estimation, adaptive signal processing, machine learning, and their application to sensor networks.

Masahiro Yukawa (S'05–M'06) received the B.E., M.E., and Ph.D. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 2002, 2004, and 2006, respectively. He was a Visiting Researcher/Professor with the University of York, U.K., from October 2006 to March 2007, with the Technical University of Munich, Germany, from July 2008 to November 2007, and with the Technical University of Berlin, Germany, from April 2016 to February 2017. He was a Special Postdoctoral Researcher with RIKEN, Japan, in 2007–2010, and an Associate Professor with Niigata University, Japan, in 2010–2013. He is currently an Associate Professor with the Department of Electronics and Electrical Engineering, Keio University, Yokohama, Japan. Since July 2017, he has been a Visiting Scientist with AIP Center, RIKEN, Japan. His research interests include mathematical adaptive signal processing, convex/sparse optimization, and machine learning.

Dr. Yukawa was a recipient of the Research Fellowship of the Japan Society for the Promotion of Science from April 2005 to March 2007. He received the Excellent Paper Award and the Young Researcher Award from the IEICE in 2006 and in 2010, respectively, the Yasujiro Niwa Outstanding Paper Award in 2007, the Ericsson Young Scientist Award in 2009, the TELECOM System Technology Award in 2014, the Young Scientists' Prize, the Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology in 2014, the KDDI Foundation Research Award in 2015, and the FFIT Academic Award in 2016. He has been an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING since 2015. He was an Associate Editor for *Multidimensional Systems and Signal Processing* from 2012 to 2016, and the *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* from 2009 to 2013. He is a member of the Institute of Electronics, Information and Communication Engineers.

Renato Luís Garrido Cavalcante (M'09) received the Electronics Engineering degree from the Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil, in 2002, and the M.E. and Ph.D. degrees in communications and integrated systems from the Tokyo Institute of Technology, Tokyo, Japan, in 2006 and 2008, respectively.

He is currently a Research Fellow with the Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, Berlin, Germany, and a Lecturer with the Technical University of Berlin. Previously, he held appointments as a Research Fellow with the University of Southampton, Southampton, U.K., and as a Research Associate with the University of Edinburgh, Edinburgh, U.K. His current interests include signal processing for distributed systems, multiagent systems, convex analysis, machine learning, and wireless communications.

Dr. Cavalcante received the Excellent Paper Award from the Institute of Electronics, Information and Communication Engineers in 2006 and the IEEE Signal Processing Society (Japan Chapter) Student Paper Award in 2008. He also co-authored a study that received a Best Student Paper Award at the 13th IEEE International Workshop on Signal Processing Advances in Wireless Communications in 2012. From April 2003 to April 2008, he was a recipient of the Japanese Government (MEXT) Scholarship.

Armin Dekorsy (SM'18) received the B.Sc. degree from Fachhochschule Konstanz, Konstanz, Germany, the M.Sc. degree from the University of Paderborn, Paderborn, Germany, and the Ph.D. degree from the University of Bremen, Bremen, Germany, all in communications engineering.

From 2000 to 2007, he was a Research Engineer with Deutsche Telekom AG and a Distinguished Member of Technical Staff at Bell Labs Europe, Lucent Technologies. In 2007, he joined Qualcomm GmbH as a European Research Coordinator, conducting Qualcomms internal and external European research activities. He is currently the Head of the Department of Communications Engineering, University of Bremen. He has authored or coauthored more than 150 journal and conference publications. He holds more than 17 patents in the area of wireless communications. He has long-term expertise in the research of wireless communication systems, baseband algorithms, and signal processing.

Prof. Dekorsy is a senior member of the IEEE Communications and Signal Processing Society and the VDE/ITG Expert Committee "Information and System Theory."