# Distributed Kernel Least Squares for Nonlinear Regression Applied to Sensor Networks

Ban-Sok Shin, Henning Paul, and Armin Dekorsy
Department of Communications Engineering
University of Bremen, Bremen, Germany
Email: {shin, paul, dekorsy}@ant.uni-bremen.de

***Abstract*—In this paper, we address the task of distributed nonlinear regression. For this, we exploit kernel methods which can cope with nonlinear regression tasks and a consensus-based approach to derive a distributed scheme. Both techniques are combined and a distributed kernel-based least squares algorithm for nonlinear function regression is proposed. We apply our algorithm to sensor networks and the distributed estimation of diffusion fields which are known to be highly nonlinear. Performance evaluations regarding static and time-varying fields with multiple sources and arbitrary network topologies are provided showing a successful reconstruction. For the tracking of time-varying fields our proposed algorithm outperforms the state of the art.**

## I. Introduction

Nonlinear functions are commonly used to model physical phenomena as, e.g., the spatial distribution of temperature. Often, these functions need to be approximated to predict the behavior of the phenomenon. This can be done by regression based on input and measured output samples of the nonlinear function. However, due to the nonlinearity of the function this is a difficult task demanding nonlinear regression methods. It becomes even more challenging if the measurement is done in a distributed fashion. An example is a sensor network (SN) observing a diffusion field modeled by a nonlinear function which is common in environmental monitoring [1]. Often, in such applications there is no central unit or fusion center (FC). Thus, nonlinear regression combined with a distributed scheme based on an information exchange among sensors is desired. In the past, kernel methods have been successfully applied to solve nonlinear estimation and regression tasks [2]. Kernel methods enable the implementation of nonlinear regression algorithms by transforming the input samples into a high-dimensional space where the problem can be modeled linearly. E.g. in [3], nonlinear adaptive filters have been developed by combining kernels with adaptive filtering. In [4], a kernel-based least squares (LS) algorithm has been proposed for the regression of nonlinear functions. Kernel-based algorithms have been also used for distributed estimation of diffusion fields in SNs. In [5], a decentralized kernel normalized least-mean-squares (KNLMS) algorithm is proposed being able to estimate a temperature distribution. However, the algorithm implements the KNLMS update step from sensor to sensor such that several steps through the network are required until the same sensor can again update its estimate. A distributed algorithm based on diffusion of information in the network is proposed in [6]. Here, a distributed least-mean-squares (LMS) algorithm from [7] is combined with kernel methods resulting in an adaptive algorithm capable of approximating nonlinear functions. However, it is not guaranteed that all sensors converge to the same function approximation.

The contribution of this paper is twofold: First, our proposed algorithm performs a kernel LS regression in a distributed way. Second, we use a consensus-based estimation from [8] for distributed processing which has the benefit that all sensors perform the same LS regression. By this, we obtain the kernel DiCE (KDiCE) algorithm which is able to approximate nonlinear functions distributedly.

## II. System Model

We consider a SN of $N_s$ connected sensors deployed over a specific area. The SN is described by a graph with a set of nodes $\mathcal{J} := \{1, \dots, N_s\}$ representing the sensors and a set of undirected edges $\mathcal{E}$ connecting the nodes. We assume that the graph is connected, i.e., each node can be reached by any other node over multiple hops. Besides, we denote the neighborhood of node $j$ by the set $\mathcal{N}_j$ containing all nodes connected to node $j$. The SN observes a nonlinear function $f : \mathcal{X} \to \mathbb{R}$ mapping samples from the input space $\mathcal{X} \subset \mathbb{R}^L$ into the output space $\mathbb{R}$. The function can model, e.g., the spatial distribution of temperature. Then each sensor $j$ performs a measurement $d_j$ at its Cartesian position $\mathbf{x}_j \in \mathcal{X}$ of $f(\mathbf{x}_j)$ by

$$d_j = f(\mathbf{x}_j) + n_j. \tag{1}$$

Here, $n_j$ is white Gaussian noise of variance $\sigma_n^2$. We further assume that the sensors know their positions resulting in $N_s$ data pairs $\{(\mathbf{x}_j, d_j)\}_{j=1}^{N_s}$ with positions and measurements. Based on these data pairs a regression problem can be formulated where the objective is to approximate the unknown nonlinear function $f(\mathbf{x})$ to predict the field for arbitrary positions $\mathbf{x}$.

As an example for $f(\mathbf{x})$, we consider a two-dimensional diffusion field ($\mathcal{X} \subset \mathbb{R}^2$) with $M$ instantaneous and localized sources in an isotropic medium. According to [9], the diffusion field is described by the following function at Cartesian coordinate $\mathbf{x}$ and time $t$ where each diffusion source $m$ has intensity $c_m$, activation time $t_m$ and Cartesian position vector $\mathbf{p}_m$:

$$f(\mathbf{x}, t) = \sum_{m=1}^{M} \frac{c_m}{4\pi\nu(t - t_m)} \exp\left(-\frac{||\mathbf{x} - \mathbf{p}_m||^2}{4\nu(t - t_m)}\right) \cdot h(t - t_m). \tag{2}$$

Here, $h(t)$ is the Heaviside-function and $\nu$ the diffusion constant of the medium. If the diffusion field is static, the time dependence in (2) is dropped resulting in

$$f(\mathbf{x}) = \sum_{m=1}^{M} \frac{c_m}{4\pi\nu} \exp\left(-\frac{||\mathbf{x}-\mathbf{p}_m||^2}{4\nu}\right). \tag{3}$$

We implicitly assume that all $M$ sources have activation time $t_m = 0$ and that the field is measured once at time $t = 1$ by the SN. Then the function is dependent on the coordinate $\mathbf{x}$ only. Since $f(\mathbf{x})$ is nonlinear, linear methods will not provide satisfactory regression performance.

## III. Problem Formulation

Assume a positive definite kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that corresponds to the dot product $\kappa(\mathbf{x}_j, \mathbf{x}_n) = \Phi^{\mathrm{T}}(\mathbf{x}_j)\Phi(\mathbf{x}_n)$ between two samples $\mathbf{x}_j$ and $\mathbf{x}_n$ in a high-dimensional space, called reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ [2]. The function $\Phi : \mathcal{X} \to \mathcal{H}$ maps input samples $\mathbf{x}$, i.e. the sensor positions, to the function $\kappa(\,\cdot\,, \mathbf{x})$ in the RKHS $\mathcal{H}$. Using such a kernel allows to represent the nonlinear function $f(\mathbf{x})$ as a linear combination of such kernels. Thus, we have a linear representation of $f(\mathbf{x})$ in $\mathcal{H}$ and can use linear regression methods in $\mathcal{H}$ to approximate the function $f(\mathbf{x})$. The objective is then to find the optimal function $f^*(\,\cdot\,)$ in the RKHS $\mathcal{H}$ that minimizes the sum of squared residuals between sensor measurements $d_j$ and estimated measurements $\hat{f}(\mathbf{x}_j)$ [10]:

$$f^* = \underset{\hat{f}\in\mathcal{H}}{\arg\min} \sum_{j=1}^{N_s} (d_j - \hat{f}(\mathbf{x}_j))^2. \tag{4}$$

This is a LS regression problem in $\mathcal{H}$ based on $N_s$ data pairs $\{(\mathbf{x}_j, d_j)\}_{j=1}^{N_s}$. In general, the function $f^*(\,\cdot\,)$ will differ from the true function $f(\,\cdot\,)$ since its calculation uses the noisy measurements $d_j$. We can reformulate problem (4) by means of the *representer theorem* [2] stating that the function $\hat{f}(\,\cdot\,)$ can be expressed by linear combinations of kernel evaluations based on all $N_s$ sensor positions weighted by coefficients $w_j$:

$$\hat{f}(\,\cdot\,) = \sum_{j=1}^{N_s} w_j \kappa(\mathbf{x}_j, \,\cdot\,). \tag{5}$$

The number of required sensor positions $\mathbf{x}_j$ and thus the number of weighting coefficients $w_j$ can be reduced by dictionary learning methods. These methods aim at generating a set with the most significant sensor positions out of all $N_s$ sensor positions while guaranteeing a satisfactory regression performance, see e.g. [11]. By this, complexity and required storage are reduced. We denote $\tilde{\mathbf{x}}_\ell$ to be the $\ell$-th vector being selected by dictionary learning out of the set of all sensor positions $\{\mathbf{x}_j\}_{j=1}^{N_s}$. Hence, we choose $N_d$ such vectors to get the set $\{\tilde{\mathbf{x}}_\ell\}_{\ell=1}^{N_d}$. We call the set $\mathcal{D} = \{\kappa(\tilde{\mathbf{x}}_\ell, \,\cdot\,)\}_{\ell=1}^{N_d}$ the dictionary of size $N_d$ containing kernel evaluations with respect to (w.r.t.) the selected vectors $\{\tilde{\mathbf{x}}_\ell\}_{\ell=1}^{N_d}$. With a slight abuse of notation we get a new representation of the function $\hat{f}(\,\cdot\,)$ using the elements of the dictionary $\mathcal{D}$:

$$\hat{f}(\,\cdot\,) = \sum_{\ell=1}^{N_d} w_\ell \kappa(\tilde{\mathbf{x}}_\ell, \,\cdot\,). \tag{6}$$

Then with (6), we can reformulate the optimization problem in (4) in terms of the weight vector $\mathbf{w} = [w_1, \ldots, w_{N_d}]^{\mathrm{T}}$ and the vector $\boldsymbol{\kappa}(\mathbf{x}_j) = [\kappa(\tilde{\mathbf{x}}_1, \mathbf{x}_j), \ldots, \kappa(\tilde{\mathbf{x}}_{N_d}, \mathbf{x}_j)]^{\mathrm{T}}$:

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^{N_d}}{\arg\min} \sum_{j=1}^{N_s} (d_j - \mathbf{w}^{\mathrm{T}}\boldsymbol{\kappa}(\mathbf{x}_j))^2. \tag{7}$$

A commonly used kernel is the Gaussian kernel defined as [2]

$$\kappa(\mathbf{x}_j, \mathbf{x}_n) = \exp\left(-\frac{||\mathbf{x}_j - \mathbf{x}_n||^2}{2\zeta^2}\right), \tag{8}$$

with $\zeta$ being the kernel bandwidth. Inserting the Gaussian kernel into (6) and comparing it to (3) we observe that approximating $f(\mathbf{x})$ with $N_d$ Gaussian kernels centered at the positions $\tilde{\mathbf{x}}_\ell$ of the dictionary $\mathcal{D}$ matches the diffusion field function. With (7), problem (4) reduces to finding the coefficient vector $\mathbf{w}^*$ instead of the function $f^*(\,\cdot\,)$. This formulation builds the foundation for the following approaches.

## IV. Kernel Least Squares Regression

### A. Central Kernel LS Solution

In the following, we assume that all $N_s$ sensor positions are contained in the dictionary $\mathcal{D}$ such that the dictionary size $N_d$ equals the number of sensors in the network $N_s$. Vector $\mathbf{d}$ contains the stacked measurements $\mathbf{d} = [d_1, \ldots, d_{N_s}]^{\mathrm{T}}$ while $\mathbf{K}$ is the kernel Gram matrix [2] with $\mathbf{K} = [\boldsymbol{\kappa}(\mathbf{x}_1), \ldots, \boldsymbol{\kappa}(\mathbf{x}_{N_s})]^{\mathrm{T}}$. Then, problem (7) can be written as

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^{N_s}}{\arg\min} ||\mathbf{d} - \mathbf{K}\mathbf{w}||^2. \tag{9}$$

The solution can be directly given by the Pseudo-Inverse of $\mathbf{K}$ and $\mathbf{d}$ with

$$\mathbf{w}^*_{\mathrm{LS}} = (\mathbf{K}^{\mathrm{T}}\mathbf{K})^{-1}\mathbf{K}^{\mathrm{T}}\mathbf{d}. \tag{10}$$

The inversion of $\mathbf{K}^{\mathrm{T}}\mathbf{K}$ can lead to numerical instabilities if the matrix $\mathbf{K}$ does not have a full column rank. Such a case can appear in a SN where sensors with similar positions have similar distances to all other sensors. This leads to linear dependencies among the columns of $\mathbf{K}$ and to numerical instability for the inversion if the Gaussian kernel (8) is used since it considers the squared distance between sensors in its argument. To improve the stability we can adjust the cost function (9) by an additional regularization term $\epsilon||\mathbf{w}||^2$ [12]. Then we obtain the following solution to the regularized cost function:

$$\mathbf{w}^*_{\mathrm{LS,reg}} = (\mathbf{K}^{\mathrm{T}}\mathbf{K} + \epsilon\mathbf{I}_{N_s})^{-1}\mathbf{K}^{\mathrm{T}}\mathbf{d}, \tag{11}$$

where $\epsilon$ is the regularization parameter used for tuning between an exact fitting for the data and numerical stability for the inversion. The kernel least squares (KLS) solutions (10) and (11) require all measurements and sensor positions to be available at the FC to calculate the optimal weights $\mathbf{w}^*$. Thus, we call them the central KLS solutions and together with (5) a prediction of the function $f(\mathbf{x})$ for arbitrary inputs $\mathbf{x}$ is obtained. However, in most applications distributed solutions are necessary to avoid single-point of failure risks by the FC and to enable each sensor to act autonomously based on its local estimate. For these approaches, the central solution (11) acts as a benchmark.

## B. *Kernel DiCE (KDiCE) Algorithm*

Many approaches exist to solve a LS problem as (9) in a distributed way. Among these, consensus-based algorithms [8], [13], [14] have been proposed in the past. The distributed consensus-based estimation (DiCE) [8] solves a LS problem in a distributed fashion by using a consensus constraint on the estimates and utilizing the alternating direction method of multipliers (ADMM). We follow this idea and derive a kernel-based regression algorithm calculating the optimal weighting coefficients $\mathbf{w}^*$ in a distributed fashion. For this, the optimization problem (7) is modified by introducing local weight vectors $\mathbf{w}_j \in \mathbb{R}^{N_d}$ at each node $j$ and by adding a consensus constraint on these vectors. Then we formulate the optimization problem w.r.t. the set $\{\mathbf{w}_j | j \in \mathcal{J}\}$ containing the weight vectors of all nodes in the network:

$$\{\mathbf{w}_j^* | j \in \mathcal{J}\} = \underset{\{\mathbf{w}_j | j \in \mathcal{J}\}}{\arg\min} \sum_{j=1}^{N_s} (d_j - \mathbf{w}_j^{\mathrm{T}} \boldsymbol{\kappa}(\mathbf{x}_j))^2 \tag{12a}$$

$$\text{s.t.} \quad \mathbf{w}_j = \mathbf{w}_i, \quad \forall j \in \mathcal{J}, i \in \mathcal{N}_j. \tag{12b}$$

The additional constraint (12b) forces a consensus on $\mathbf{w}_j$ among neighboring nodes $i \in \mathcal{N}_j$ across the whole network. By this, each node $j$ achieves the same solution for $\mathbf{w}_j$, namely the central KLS solution (10). However, due to direct coupling of weight vectors among neighboring nodes solving (12a) with (12b) will not result in a parallel processing among the nodes. Thus, we introduce an auxiliary variable $\mathbf{z}_j \in \mathbb{R}^{N_d}$ per node $j$ such that constraint (12b) can be decoupled by

$$\mathbf{w}_j = \mathbf{z}_i, \quad \mathbf{z}_j = \mathbf{w}_j, \qquad \forall j \in \mathcal{J}, i \in \mathcal{N}_j. \tag{13}$$

With these constraints problem (12a) can be solved using the augmented Lagrangian method and the ADMM. We build the augmented Lagrangian cost function over all nodes with Lagrange multipliers $\boldsymbol{\lambda}_{ji} \in \mathbb{R}^{N_d}$:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{\lambda}) = \sum_{j=1}^{N_s} \left[ \frac{1}{2}(d_j - \mathbf{w}_j^{\mathrm{T}} \boldsymbol{\kappa}(\mathbf{x}_j))^2 - \sum_{i \in \mathcal{N}_j'} \boldsymbol{\lambda}_{ji}^{\mathrm{T}} (\mathbf{w}_j - \mathbf{z}_i) + \sum_{i \in \mathcal{N}_j'} \frac{1}{2\mu} ||\mathbf{w}_j - \mathbf{z}_i||^2 \right] \tag{14a}$$

$$= \sum_{j=1}^{N_s} \mathcal{L}_j(\mathbf{w}_j, \boldsymbol{z}, \boldsymbol{\lambda}), \tag{14b}$$

where $\mathcal{N}_j' = \mathcal{N}_j \cup \{j\}$ is the neighborhood set including node $j$. Now, we can minimize the cost function $\mathcal{L}_j(\mathbf{w}_j, \boldsymbol{z}, \boldsymbol{\lambda})$ individually for each node $j$ w.r.t. $\mathbf{w}_j$ and $\mathbf{z}_j$. Following the derivations in [8] we obtain equations for $\mathbf{w}_j, \mathbf{z}_j$ and Lagrange multipliers $\boldsymbol{\lambda}_{ji}$ per sensor $j$. We can calculate these variables in an iterative fashion with iteration index $k$ resulting in

$$\mathbf{z}_j^{k+1} = \frac{\mu}{|\mathcal{N}_j'|} \sum_{i \in \mathcal{N}_j'} \frac{1}{\mu} \mathbf{w}_i^k - \boldsymbol{\lambda}_{ij}^k \tag{15a}$$

$$\boldsymbol{\lambda}_{ji}^{k+1} = \boldsymbol{\lambda}_{ji}^k - \frac{1}{\mu} \left( \mathbf{w}_j^k - \mathbf{z}_i^{k+1} \right) \tag{15b}$$

$$\mathbf{w}_j^{k+1} = \left( \boldsymbol{\kappa}(\mathbf{x}_j) \boldsymbol{\kappa}(\mathbf{x}_j)^{\mathrm{T}} + \frac{|\mathcal{N}_j'|}{\mu} \mathbf{I}_{N_d} \right)^{-1} \cdot \left( d_j \boldsymbol{\kappa}(\mathbf{x}_j) + \sum_{i \in \mathcal{N}_j'} \frac{1}{\mu} \mathbf{z}_i^{k+1} + \boldsymbol{\lambda}_{ji}^{k+1} \right). \tag{15c}$$

Here, $\mu$ is a positive step size and the algorithm is initialized with $\mathbf{w}_j^0 = \boldsymbol{\lambda}_{ij}^0 = \mathbf{0}$ for all nodes. (15a)-(15c) constitute our proposed KDiCE algorithm for distributed kernel-based LS regression. Each node $j$ calculates its auxiliary variable $\mathbf{z}_j^{k+1}$ and broadcasts it to its neighbors. After receiving the auxiliary variables $\mathbf{z}_i^{k+1}$ from their neighbors, each node updates the Lagrange multipliers $\boldsymbol{\lambda}_{ji}^{k+1}$ and transmits them to its neighbors. These need to be transmitted in a unicast fashion since they are dependent on the edge between node $j$ and $i$. After receiving the Lagrange multipliers from its neighbors, each node updates the weight vector $\mathbf{w}_j^{k+1}$ and broadcasts it to its neighbors initiating the next iteration.

Since each sensor $j$ possesses its own weight vector $\mathbf{w}_j^k$, it can predict the function $\hat{f}_j^k(\mathbf{x})$ for arbitrary positions $\mathbf{x}$ per iteration $k$ via (6) as follows:

$$\hat{f}_j^k(\mathbf{x}) = \sum_{\ell=1}^{N_d} w_{j,\ell}^k \, \kappa(\tilde{\mathbf{x}}_\ell, \mathbf{x}) = (\mathbf{w}_j^k)^{\mathrm{T}} \boldsymbol{\kappa}(\mathbf{x}), \tag{16}$$

with $w_{j,\ell}^k$ being the $\ell$-th entry of the vector $\mathbf{w}_j^k$ and $\boldsymbol{\kappa}(\mathbf{x}) = [\kappa(\tilde{\mathbf{x}}_1, \mathbf{x}), \ldots, \kappa(\tilde{\mathbf{x}}_{N_d}, \mathbf{x})]^{\mathrm{T}}$. To calculate $\boldsymbol{\kappa}(\mathbf{x})$ each sensor $j$ needs to know the sensor positions $\{\tilde{\mathbf{x}}_\ell\}_{\ell=1}^{N_d}$ contained in the dictionary $\mathcal{D}$. With (16) each sensor is able to predict the function $f(\mathbf{x})$ over the complete area covered by the SN.

## V. Performance Evaluation

## A. *Static Diffusion Field*

For the evaluation of the KDiCE algorithm we consider the regression of a static diffusion field $f(\mathbf{x})$ given by (3). The KDiCE is not restricted to such functions but can be used for other nonlinear functions. We generate a field with diffusion constant $\nu = 0.01$ by three sources with intensities $c_1 = 1, c_2 = 0.7, c_3 = 0.5$ and positions $\mathbf{p}_1 = [0.3, 0.3]^{\mathrm{T}}, \mathbf{p}_2 = [0.8, 0.6]^{\mathrm{T}}, \mathbf{p}_3 = [0.2, 0.8]^{\mathrm{T}}$. Furthermore, we place $N_s = 100$ sensors randomly over the unit square area. Sensors having a distance less than 0.22 to each other are connected by an error-free communication link. Moreover, white Gaussian noise with power $\sigma_n^2 = 0.01$ is added to the measurements of the sensors. We use the Gaussian kernel (8) with a bandwidth of $\zeta = \sqrt{2\nu}$ assuming that the diffusion constant $\nu$ is known. By this choice the kernel is matched to the field function (3) to achieve a high regression performance. If the bandwidth is

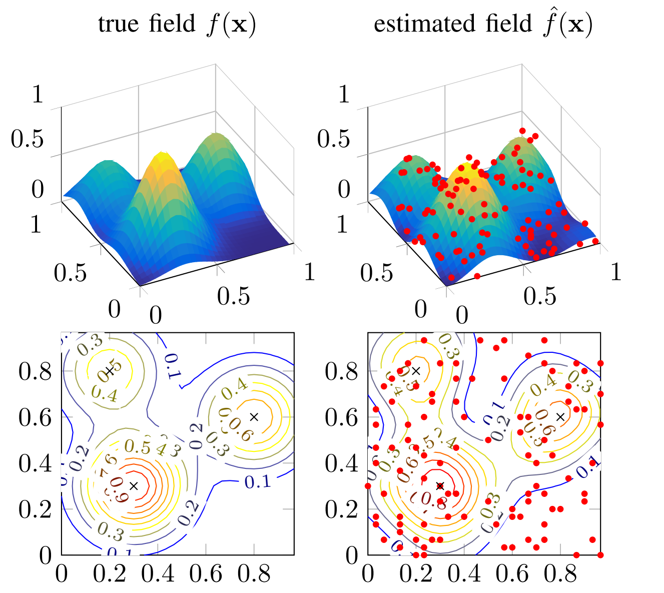

Fig. 1. Surface and contour plots of the true and estimated diffusion field after 300 iterations for one sensor. Black circles indicate the sensor positions and the measured field values. Black crosses indicate the position of the diffusion sources.

either too low or too high sensor positions will be interpreted by the kernel as either completely distinct or similar samples and the performance will decrease. Thus, the choice of the bandwidth is crucial for the regression performance of the algorithm. For the step size we choose $\mu_{\text{KDiCE}} = 8$ and stop the algorithm after 300 iterations. Regarding the dictionary no learning method is applied such that $\mathcal{D} = \{\kappa(\mathbf{x}_j, \cdot)\}_{j=1}^{N_s}$.

Fig. 1 depicts surface and contour plots of the true and estimated diffusion field of one sensor. The position and the height of the black circles indicate the sensor location and the measured field value. We show the estimated diffusion field after 300 iterations of the KDiCE for one network topology. In both plots we observe a high similarity between estimated and true field indicating that the KDiCE is able to reconstruct the complete diffusion field successfully. In order to evaluate the regression performance of the KDiCE we determine the error between the estimated field $\hat{f}_j^k(\mathbf{x})$ of each sensor $j$ and the true field $f(\mathbf{x})$. The error between estimated and true field is averaged over $N_g$ considered grid points of the unit square area and over all $N_s$ nodes in the network to give the following mean square error (MSE) per trial $r$ and iteration $k$:

$$\begin{aligned}\text{MSE}_r^k &= \frac{1}{N_s}\frac{1}{N_g}\sum_{j=1}^{N_s}\sum_{n=1}^{N_g}|f(\mathbf{x}_n) - \hat{f}_j^k(\mathbf{x}_n)|^2 \\ &= \frac{1}{N_s}\frac{1}{N_g}\sum_{j=1}^{N_s}\sum_{n=1}^{N_g}|f(\mathbf{x}_n) - (\mathbf{w}_j^k)^{\mathrm{T}}\boldsymbol{\kappa}(\mathbf{x}_n)|^2. \quad (17)\end{aligned}$$

Fig. 2 shows the MSE averaged over 100 trials for different realizations of noise and network topology for the KDiCE algorithm and the central regularized KLS solution (11). For the central solution we set the regularization parameter to

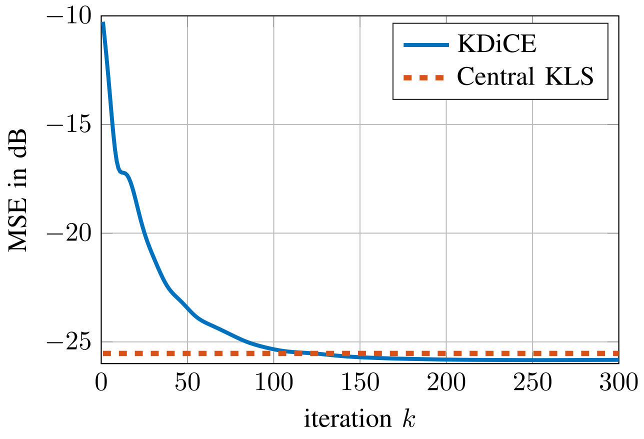

Fig. 2. Comparison of the field estimation MSE averaged over all sensor nodes between the KDiCE and the central kernel LS solution.

$\epsilon = 0.1$. One can see that the KDiCE achieves the central regression performance approximately after 130 iterations and even outperforms it afterwards. This is due to the regularization term introduced in the central LS solution (11) which shifts the solution away from the pure LS solution. Since in problem (12a) considered by the KDiCE no regularization term is contained, the weight vectors $\mathbf{w}_j^*$ will differ from the one calculated by the central LS estimator. Hence, it is possible that the KDiCE achieves a better regression than the central LS estimator. Our evaluation illustrates that the KDiCE is an appropriate distributed estimator for static diffusion fields.

## B. *Time-Varying Diffusion Field*

Due to its iterative structure, we can use the KDiCE to track time-varying diffusion fields. To evaluate its tracking performance, we generate a field according to (2) with $M = 2$ sources. The sources have intensities $c_1 = 1, c_2 = 0.7$, positions $\mathbf{p}_1 = [0.3, 0.3]^{\mathrm{T}}, \mathbf{p}_2 = [0.8, 0.6]^{\mathrm{T}}$ and activation times $t_1 = 0, t_2 = 10$, respectively. Since the field is time-varying, the SN needs to sample it in intervals to be able to track it. Therefore, each sensor measures the field at its position every $\Delta t = 0.2$ time instant up to a maximum measurement time of 20 time instants resulting in a total of 100 measured field samples over time. Again we assume noise with power $\sigma_n^2 = 0.01$ during each measurement. For the evaluation of the KDiCE we compare its MSE over time to the Functional Adapt-then-Combine Kernel Least-Mean-Square (FATC-KLMS) [6]. This algorithm is a distributed kernelized LMS scheme where each sensor $j$ updates its local weight vector $\mathbf{w}_j$ in a two-step manner. The first step calculates an intermediate weight vector based on an LMS adaptation rule. In the second step intermediate weight vectors from neighboring sensors are received by each sensor $j$ and a weighted average of these vectors is calculated to update the weight vector $\mathbf{w}_j$. For this weighted averaging step we use the relative-degree rule according to [7].

Both algorithms apply the coherence criterion [11] to determine the dictionary $\mathcal{D}$ which is then used by all sensors in the network. This criterion adds a sensor position $\mathbf{x}_j$ into the dictionary $\mathcal{D}$ if the condition

$$\max_{\ell=1,\ldots,N_d} |\kappa(\tilde{\mathbf{x}}_\ell, \mathbf{x}_j)| \leq \tau \tag{18}$$

is met, where $\tau$ is the coherence threshold which we choose to 0.8. The criterion uses the kernel function in order to compare the current sensor position $\mathbf{x}_j$ to the sensors contained in the dictionary in terms of their similarity in the RKHS. For a sensor position close to a position already contained in the dictionary the kernel evaluation gives a high value. If this value is greater than $\tau$ the sensor position is not included into the dictionary. We then perform the coherence criterion over all $N_s$ sensor positions of the specific network topology before we run the algorithms. After the dictionary is determined it stays fixed for the specific algorithm. The step sizes of the algorithms are $\mu_{\text{KDiCE}} = 1$ for the KDiCE and $\mu_{\text{KLMS}} = 0.05$ for the FATC-KLMS, respectively. Furthermore, in order to improve the tracking ability of the algorithms the bandwidth of the Gaussian kernel is adapted in each time interval $\Delta t$ via $\zeta = \sqrt{2\nu t}$. By this, we take the time-varying property of the diffusion field according to (2) into account. We assume that the sensor measurements start at $t = 0$ when the first source is activated and that the network tracks the time $t$ afterwards. Regarding the communication overhead, we note that the KDiCE exchanges variables $\mathbf{z}_j, \boldsymbol{\lambda}_{ji}$ and $\mathbf{w}_j$ all of dimension $N_d$ whereas the FATC-KLMS only exchanges its intermediate weight vectors of the same dimension per iteration $k$. To compare both algorithms the KDiCE runs $k_{\text{KDiCE}} = 1$ iteration whereas the FATC-KLMS runs $k_{\text{KLMS}} = 3$ iterations per measurement. By this, we make sure that both algorithms have a comparable amount of information exchange per time interval $\Delta t$. Fig. 3 depicts the MSE for both algorithms averaged over 100 trials. One can clearly see that the KDiCE outperforms the FATC-KLMS in terms of convergence speed and estimation performance. The KDiCE has converged after approximately $t = 7$ time instants for the first source and its MSE is 3 dB lower compared to the FATC-KLMS. After $t = 10$ time instants the FATC-KLMS has approximately reached the same MSE as the KDiCE. At this time instant the activation of the second source can be observed by an increased MSE for both algorithms. Also after the second source is activated the KDiCE exhibits a faster adaptation and a lower MSE. Although the KDiCE is no adaptive algorithm, our evaluation illustrates that it is able to track time-varying fields and even outperforms an LMS-based algorithm.

## VI. Conclusion

In this paper, we presented the KDiCE as a distributed kernel-based algorithm for nonlinear LS regression. We evaluated its ability to estimate and track diffusion fields with multiple sources for arbitrary network topologies. For the estimation of time-varying fields the KDiCE outperformed the state of the art [6] w.r.t. convergence speed and estimation accuracy. Future work involves the investigation of an adaptive scheme to determine the kernel bandwidth and the optimization of dictionary samples to improve the regression performance.

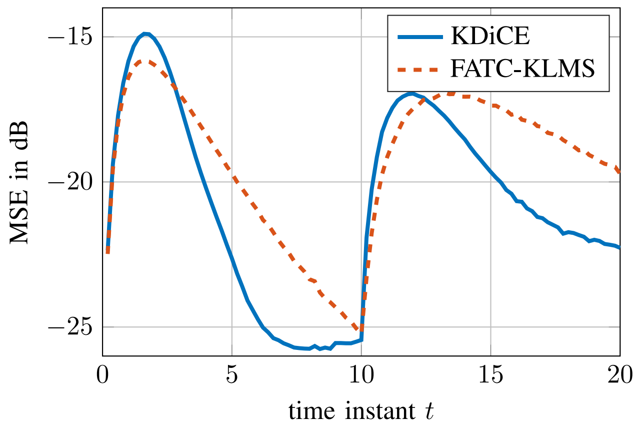

Fig. 3. MSE of the KDiCE (1 iteration) and the FATC-KLMS (3 iterations) for a time-varying diffusion field with $M = 2$ sources.

## Acknowledgment

The work leading to this publication was partially funded by the German Research Foundation (DFG) under grant Pa2507/1.

## References

[1] S. N. Simic and S. Sastry, "Distributed environmental monitoring using random sensor networks," in *Proc. of the 2nd Int. Workshop on Information Processing in Sensor Networks*, 2003, pp. 582–592.

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

[3] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering*. John Wiley & Sons, 2010.

[4] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," *Proc. of the 15th Int. Conf. on Machine Learning*, pp. 515–521, 1998.

[5] P. Honeine, C. Richard, J. Bermudez, J. Chen, and H. Snoussi, "A decentralized approach for nonlinear prediction of time series data in sensor networks," *EURASIP Journal on Wireless Communications and Networking*, no. 1, 2010.

[6] W. Gao, J. Chen, C. Richard, and J. Huang, "Diffusion adaptation over networks with kernel least-mean-square," in *IEEE CAMSAP*, 2015.

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

[8] 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, 2013.

[9] Y. M. Lu, P. L. Dragotti, and M. Vetterli, "Localizing point sources in diffusion fields from spatiotemporal samples," in *Proc. of Int. Conf. on Sampling Theory and Applications (SampTA)*, May 2011.

[10] W. Gao, J. Chen, C. Richard, J. Huang, and R. Flamary, "Kernel LMS algorithm with forward-backward splitting for dictionary learning," in *IEEE ICASSP*, 2013, pp. 5735–5739.

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

[12] F. A. Tobar and D. P. Mandic, "The quaternion kernel least squares," in *IEEE ICASSP*, 2013, pp. 6128–6132.

[13] G. Xu, H. Paul, D. Wübben, and A. Dekorsy, "Fast distributed consensus-based estimation (Fast-DiCE) for cooperative networks," in *International ITG Workshop on Smart Antennas*, 2014.

[14] R. Lopez-Valcarce, S. S. Pereira, and A. Pages-Zamora, "Distributed total least squares estimation over networks," in *IEEE ICASSP*, 2014, pp. 7580–7584.