A COMPARATIVE STUDY OF KERNEL ADAPTIVE FILTERING ALGORITHMS

Steven Van Vaerenbergh and Ignacio Santamaría

Dept. of Communications Engineering
University of Cantabria, Spain

ABSTRACT

Kernel adaptive filtering is a growing field of signal processing that is concerned with nonlinear adaptive filtering. When implemented naïvely, the time and memory complexities of these algorithms grow at least linearly with the amount of data processed. A large number of practical solutions have been proposed throughout the last decade, based on sparsification or pruning mechanisms. Nevertheless, there is a lack of understanding of their relative merits, which often depend on the data they operate on. We propose to study the quality of the solution as a function of either the time or the memory complexity. We empirically test six different kernel adaptive filtering algorithms on three different benchmark data sets. We make our code available through an open source toolbox that includes additional algorithms and allows to measure the complexities explicitly in number of floating point operations and bytes needed, respectively.

***Index Terms*—** Kernel adaptive filtering, nonlinear filtering, comparison, benchmarks.

1. INTRODUCTION

Kernel adaptive filtering (KAF) methods are rapidly gaining popularity to solve a wide variety of prediction, identification and regression problems, since they provide state-of-the-art performance in many real-world applications. Relying on the "kernel-trick" as a basic building block, many kernel-based versions of the popular least-mean squares (LMS) and recursive least-squares (RLS) algorithms have been proposed over the last years in both the signal processing and machine learning fields, see for instance [1, 2, 3, 4, 5, 6, 7]

A bottleneck typically encountered in online kernel-based learning, is that KAF methods suffer from growing complexity, since the number of kernels required to represent their solution can grow linearly or faster with the number of processed data. To overcome this drawback and to limit the complexity of the kernel-based filters, several different sparsification or fixed-budget approaches can be applied, thus multiplying the number of possible KAF methods.

This work has been partially supported by TEC2010-19545-C04-03 (COSIMA) and Consolider-Ingenio 2010 CSD2008-00010 (COMONSENS).

Unlike their linear counterparts, for kernel adaptive filters the trade-offs among speed of convergence, tracking ability and misadjustment noise are difficult to characterize theoretically, with only a few exceptions [8]. Therefore, to understand the relative merits of the different KAF techniques, one typically has to resort to simulation studies. Nevertheless, the studies conducted so far in the literature are typically focused on a particular algorithm and they use specific metrics and datasets in order to highlight the benefits of this algorithm in comparison to its close competitors. As a result, it is difficult to have a fair view of the real merits of the most popular KAF techniques.

In an attempt to fill this gap, we conduct a comparative study on the performance of six representative KAF techniques applied to three different regression problems, specifically predicting the chaotic Lorenz time series, predicting respiratory motion traces, and identifying a time-varying wireless channel. The algorithms we consider are four variants of the KRLS algorithm [2, 9, 10, 6] and two KLMS-like algorithms [1, 5], which together form a representative sampling of the current state-of-the-art KAF techniques. The selected benchmark problems are also representative of different scenarios encountered in practice, e.g., stationary and non-stationary (dynamical) systems.

As we have mentioned previously, the complexity of KAF algorithms strongly depends on the size of the dictionary used to approximate the desired output. Most kernel adaptive filtering algorithms have a single parameter that controls the growth of the dictionary. For algorithms based on a sparsification criterion, this is typically a threshold that determines if a new datum is accepted into the dictionary. For algorithms on a budget this is typically a fixed dictionary size. We will assess the quality of the predictions as a function of the compute time or memory as this parameter is varied.

The rest of this paper is structured as follows. In Section 2, we review the kernel adaptive filtering problem and then we briefly outline the algorithms compared in this study. In Section 3, we describe a set of performance measures that allow us to explore the existing performance/complexity trade-offs. In Section 4 we introduce three benchmark data sets on which we perform either prediction or system identification, and we discuss the relative merits of the different KAF techniques. Section 5 summarizes the main conclusions of this work.

2. KERNEL-BASED ADAPTIVE FILTERING

2.1. Problem statement

Kernel adaptive filtering is the subfield of online kernel-based learning that deals with the problem of nonlinear regression. Specifically, given an input-output stream of data pairs $(\mathbf{x}_i, y_i)$, the task consists in estimating the nonlinear function $f_i(\cdot)$ that relates input and output,

$$y_i = f_i(\mathbf{x}_i), \tag{1}$$

and to update this estimate efficiently every time a new data pair is received. The unknown function $f_i(\cdot)$ may be changing over time, as opposed to standard regression settings that assume static underlying models $f(\cdot)$. This problem is encountered for instance in adaptive signal processing theory, which classically employs linear techniques, i.e. assuming a solution of the form, $f_i(\mathbf{x}_i) = \mathbf{w}_i^\top \mathbf{x}_i$, on which there exists a vast body of literature [11].

Kernel methods are based on a nonlinear transformation of the data $\mathbf{x}_i$ into a high-dimensional *feature space*. In this feature space, inner products can be calculated by using a positive definite kernel function satisfying Mercer's condition [12]: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. This simple and elegant idea, also known as the "kernel trick", allows to perform inner-product based algorithms implicitly in feature space by replacing all inner products by kernels. Many kernel functions exist, though the most commonly used is the Gaussian kernel

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2). \tag{2}$$

Thanks to the Representer Theorem [13], the nonlinearities of a wide range of problems can be represented sufficiently well as a kernel expansion in terms of the training data

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}). \tag{3}$$

Kernel recursive least-squares (KRLS) algorithms calculate the coefficients α_i by solving a least-squares problem which involves the inversion of a *kernel matrix* whose dimensions depend on the number of stored data, M. Therefore, they have quadratic computational and memory complexity in terms of M, i.e. $\mathcal{O}(M^2)$. Kernel least-mean squares (KLMS) algorithms, on the other hand, use stochastic gradient descent to obtain α_i, and they have linear complexity, $\mathcal{O}(M)$. As mentioned above, in online scenarios the amount of processed data, M, grows in time, and practical algorithms therefore need to restrict the amount of data that will be stored. While the complexities of KLMS and KRLS algorithms differ an order of magnitude, an interesting question is whether KLMS with high M can reach comparable performance as KRLS with low M. This will become clear in the experiments.

In the sequel we outline the six state-of-the-art KAF algorithms that are included in this study. We distinguish between algorithms that allow for tracking and algorithms that rather assume a stationary data model.

2.2. NORMA

Naive Online regularized Risk Minimization Algorithm (NORMA) is a kernel-based version of leaky least-mean squares (LMS) [1]. It is closely related to the KLMS algorithm proposed in [3] but it includes regularization. As a result, its coefficients shrink over time, allowing it to discard the oldest bases in a sliding-window fashion.

2.3. QKLMS

Quantized KLMS (QKLMS) [5] constructs a dictionary according to a "quantization" process in which data points are mapped onto the closest dictionary point. A similar criterion was followed in [4]. By allowing to update previously calculated coefficients, QKLMS obtains tracking capability.

2.4. ALD-KRLS

Approximate Linear Dependency KRLS (ALD-KRLS) [2] is the first KRLS-type algorithm proposed in the literature. It slows down dictionary growth by using a sparsity criterion based on linear dependency. Its only parameter, apart from the kernel, is a sensitivity threshold ν which determines if a basis will be accepted into the dictionary. An important characteristic of ALD-KRLS in the context of this study is that it assumes the model underlying the data to be stationary.

2.5. SW-KRLS

Sliding-Window KRLS (SW-KRLS) achieves tracking by performing regression only on the M last observed data, [9]. It is a conceptually very simple algorithm that obtains reasonable performance in a wide range of scenarios.

2.6. FB-KRLS

Fixed-Budget KRLS (FB-KRLS) is a modification of SW-KRLS: Instead of discarding the oldest data point in each iteration, it discards the data point that causes the least error upon being discarded [10]. This approach yields performance improvements in stationary scenarios, though its pruning criterion is not optimized for non-stationary environments.

2.7. KRLS-T

The KRLS Tracker (KRLS-T) algorithm [6] was devised by deriving standard KRLS theory from a Bayesian point of view. Along with each prediction it also provides confidence intervals, which are exploited to operate a forgetting mechanism that can handle non-stationary scenarios. In order to control its adaptation rate it uses a forgetting factor λ, while its dictionary size M is a budget-dependent parameter.

3. METRICS FOR PERFORMANCE COMPARISONS

For each algorithm, we study the trade-off between the MSE and the complexity, in terms computation and memory. By fixing a performance goal, such as available memory or maximum allowable error, the trade-off figures allow the practitioner to determine which algorithm shows the most favorable remaining properties. There exist several other performance measures, such as convergence speed and robustness w.r.t. quantification, though we leave these for future studies.

3.1. Computational complexity

In order to quantify the computational complexity of an algorithm, the CPU time is often chosen as the cost. Though CPU time is easy to measure, it depends on several environmental parameters, such as the machine on which the algorithm is run and the processes this machine is running in parallel. Furthermore, comparing the CPU times of different algorithms is only fair if these algorithms are implemented with a similar level of code optimization.

An alternative is to explicitly measure the mathematical operations required by each of the algorithms. KAF algorithms perform only algebraically simple calculations, such as multiplications of matrices and vectors, and therefore this is a viable option. In the experiments we combine these numbers into a single measure, the amount of floating point operations (FLOPS) required, and we plot the average amount of FLOPS needed per iteration.

3.2. Memory

In the KAF literature it is common practice to measure the memory as the amount of bases stored in the dictionary. Nevertheless, this measure loses sense when comparing KLMS and KRLS-based algorithms, as their dictionary sizes have different impacts on the true memory usage. We measure the memory usage of each algorithm explicitly as the number of bytes required for all stored variables, including dictionaries, matrices and coefficients. In the experiments we report the maximum number of bytes used per iteration, as this corresponds to a physical limit.

4. EXPERIMENTS

We conduct experiments on three different data sets. The parameter that controls the dictionary size for each algorithm determines in turn the computational and memory complexities. For each experiment, we vary this parameter over a wide range and measure the MSE. The so obtained "performance curves" are used for comparison. The remaining parameters are chosen by an exhaustive search in order to optimize the position of the performance curve. For the initial set of parameters in each search we relied on the technique described in [14]. Note that the optimal selection of the parameters is an important open problem in the KAF literature, and it falls outside the scope of this paper.

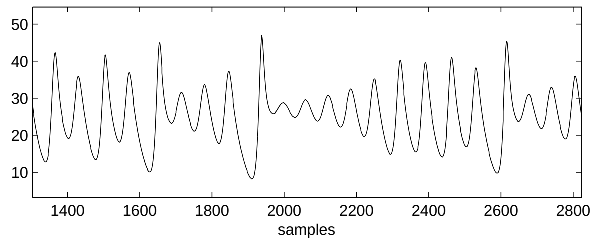


Fig. 1. A snapshot of the Lorenz time series.

The implementations of all the compared algorithms are available through the Kernel Adaptive Filtering Toolbox, `http://sourceforge.net/projects/kafbox/`.

4.1. Prediction of the Lorenz attractor time series

The Lorenz attractor is a nonlinear dynamical system corresponding to the long-term behavior of a chaotic flow. Due to its nonlinear and chaotic behavior, it is one of the standard benchmark data sets in the kernel adaptive filtering literature [15]. We sample 10.000 points of the x-component of the series under its standard parameter setting, using the first order approximation with step size 0.01. A snapshot of this time series is shown in Fig. 1.

The experiment consists in performing one-step ahead prediction, i.e. given all samples x_i up till time $i = t$, the task is to predict the sample x_{t+1}. To select the filter order we can either rely on Takens embedding theorem [16], which is a principled approach to selecting the filter order of a deterministic series, or use the empirical method described in [14]. We choose filter order 6, i.e. prediction is performed using inputs $\mathbf{x}_i = [x_{i-5}, x_{i-4}, \ldots, x_i]^\top$. The parameters used for each algorithm are shown in Table 2.

The obtained results are displayed in Fig. 2. NORMA was not included as it did not obtain MSE results in the range of interest. A first conclusion is that KRLS-type algorithms typically obtain lower steady state MSEs than KLMS, which is in accordance with the literature. By adding the FLOPS or memory into the picture, we get an idea of the resources required. If, for instance, we are working in a scenario with a restriction on computational complexity, we should select the algorithm that performs best under this restriction by looking which performance curve is most to the left for the amount of FLOPS available. In the same manner, by fixing a maximum on MSE we obtain the FLOPS and memory required by each algorithm. In this case, if the computational resources are limited, it is more interesting to use SW-KRLS. On the other hand, if the goal is to minimize the MSE at any cost, FB-KRLS does so at minimum cost.

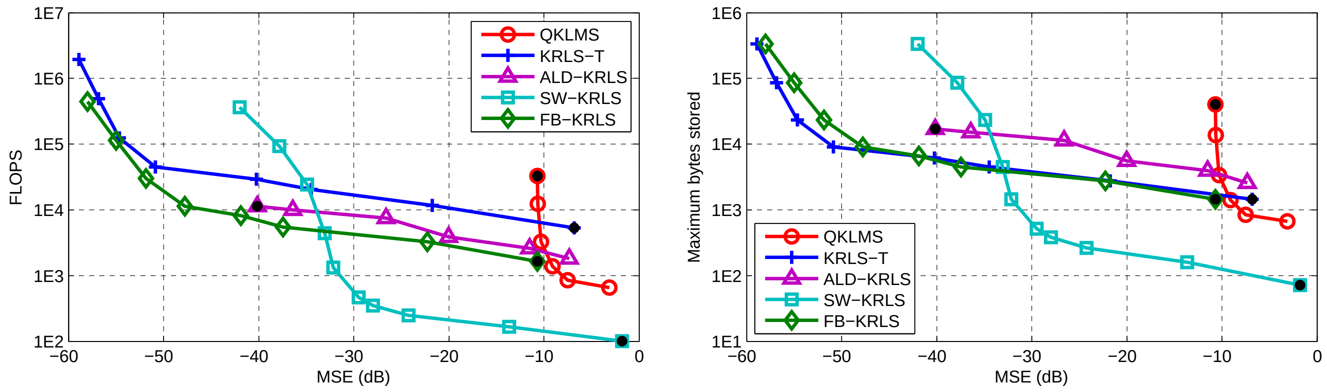


Fig. 2. Average number of FLOPS (left) and maximum memory usage (right), as a function of MSE, for the Lorenz time series prediction. Each marker represents a single run of one of the algorithms with a single set of parameters. The start of each parameter sweep is indicated by a black dot.

Table 1. Parameters used in the Lorenz time series prediction. A Gaussian kernel with $\sigma = 32$ was used.

Method	Fixed parameters	Varying parameter
QKLMS	$\eta = 0.5$	$\epsilon_{\mathbb{U}} = 1, 2, 5, 10, 15, 18$
KRLS-T	$\lambda = 1$, $\sigma_n^2 = 10^{-6}$	$M = 10, 15, 20, 24, 30, 50, 100, 200$
ALD-KRLS	—	$\nu = 0.0001, 0.0002, 0.001, .01 .05 .1$
SW-KRLS	$c = 10^{-6}$	$M = 1, 2, 3, 4, 5, 10, 20, 50, 100, 200$
FB-KRLS	$c = 10^{-6}$, $\mu = 0$	$M = 10, 15, 20, 25, 30, 50, 100, 200$

4.2. Prediction of respiratory motion traces

In robotic radiosurgery, a photon beam source is used to ablate tumors. It is operated by a robot arm that aims to move the beam source to compensate for the motion of internal organs. Traditionally, this is achieved by recording the motion of markers applied to the body surface and by using this motion to draw conclusions about the tumor position. Although this method significantly increases the targeting accuracy, the system delay arising from data processing and positioning of the beam results in a systematic error. This error can be decreased by predicting the motion of the body surface [17].

In this experiment we apply KAF algorithms to predict a respiratory motion trace. The data was recorded at Georgetown University Hospital using CyberKnife® equipment, and it represents the recorded position of one of the markers attached to the body surface[1]. A snapshot of this motion trace is shown in Fig. 3. The delay to compensate totals 115 ms, which, at a sampling frequency of 26 Hz, corresponds to 3 samples. The task thus consists in three-step ahead prediction. We use a time-embedding of 8 samples, and the remaining parameters are listed in Table 2. The results, for four of the algorithms, are found in Fig. 3. Since the breathing pattern may change over time, tracking algorithms should perform better in this experiment. The highest accuracy is obtained

[1]Data available at `http://signals.rob.uni-luebeck.de/`

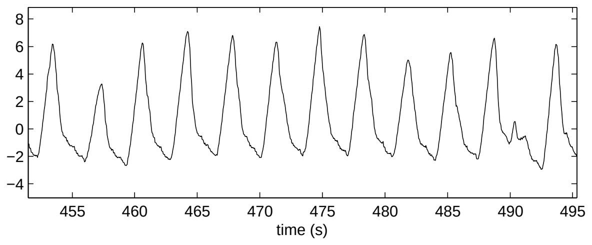
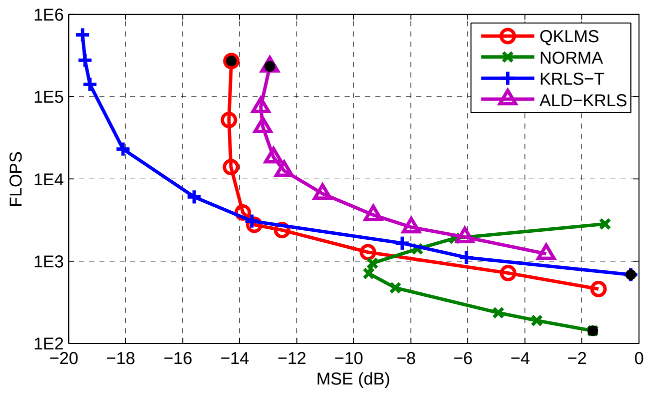


Fig. 3. A snapshot of the respiratory motion trace.

by KRLS-T, in return for the most computational and memory resources. For MSE performances of -10 dB or more, NORMA is faster, while the required memory is similar for KRLS-T and NORMA at high MSE values. The figure of merit for ALD-KRLS is relatively weak, most likely because the non-stationarity in this experiment requires tracking.

4.3. Identification of a time-varying channel

We acquired data from a wireless communication test bed that is used to evaluate the performance of digital communication systems in realistic indoor environments. The platform is composed of several transmit and receive nodes, each one including a radio-frequency front-end and baseband hardware for signal generation and acquisition. The front-end also in-



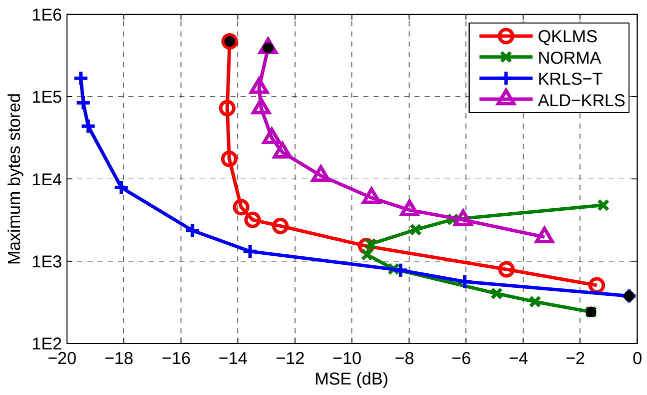


Fig. 4. Average FLOPS (left) and maximum memory usage (right), as a function of MSE, for the respiratory motion prediction.

Table 2. Parameters used in the respiratory motion prediction. A Gaussian kernel with $\sigma = 7$ was used.

Method	Fixed parameters	Varying parameter
QKLMS	$\eta = 0.99$	$\epsilon_{\mathbb{U}} = 0.2, 0.5, 1, 2, 2.5, 2.75, 4, 6, 7$
NORMA	$\eta = 0.99$, $\lambda = 10^{-6}$	$\tau = 3, 4, 5, 10, 15, 20, 30, 40, 60$
KRLS-T	$\lambda = 0.999$, $\sigma_n^2 = 10^{-6}$	$M = 3, 4, 5, 7, 10, 20, 50, 70, 100$
ALD-KRLS	—	$\nu = 0.0001, 0.001, 0.003, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5$

corporates a programmable variable attenuator to control the transmit power and therefore the signal saturation. A more detailed description of the test bed can be found in [18]. Using the hardware platform, we transmitted clipped orthogonal frequency-division multiplexing (OFDM) signals centered at 5.4 GHz over real frequency-selective and time-varying channels, with normalized Doppler frequency around 10^{-3}. The transmit amplifier was operated close to saturation. We collected 8000 inputs x_i and outputs y_i in order to identify the nonlinear channel and to track its changes over time.

The time-embedding is chosen as 4, i.e. we use the inputs up till $\mathbf{x}_i = [x_{i-3}, \dots, x_i]^\top$ to predict the output y_i, and the remaining parameter are listed in Table 3. The results of four tracking algorithms on the online identification problem can be found in Fig. 5. A first observation is that the best MSE is substantially higher than in the previous experiments, due to the high amount of noise and variability in the measurements. Without any limitation on the available resources, KRLS-T obtains better MSE values. If the amount of computation or memory is limited, for instance when implementing the KAF method on wireless sensor network consisting of low-cost nodes, QKLMS obtains similarly good performance.

5. CONCLUSIONS

We have studied the trade-offs between the MSE performance and the complexity for several state-of-the-art kernel adaptive filtering algorithms, on three benchmark data sets. The proposed figures of merit are meaningful indicators of the relative performances of these algorithms: Since they allow us to highlight advantages and disadvantages of each algorithm in different scenarios, they constitute an interesting tool for the practitioner.

As expected, we observed that there is not a single best-performing algorithm for all scenarios. Rather, the optimal choice of algorithm depends on the target MSE range, the available computational resources and the particular data set.

In future work we plan to include additional measures, such as as the convergence speed, which may be of interest in scenarios with less restrictions on complexity.

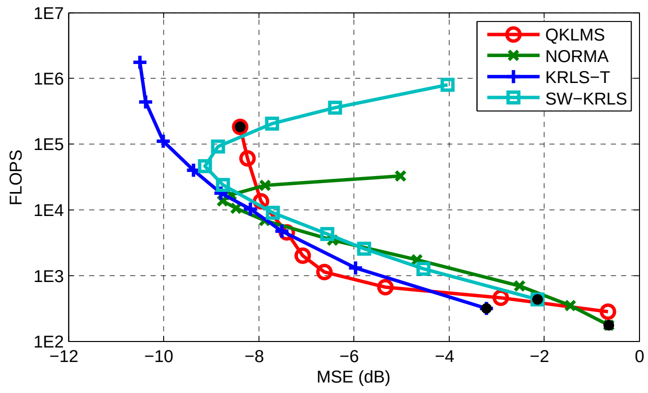
6. REFERENCES

[1] J. Kivinen, A. J. Smola, and R. C. Williamson, “Online learning with kernels,” *IEEE Trans. on Sig. Proc.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[2] Y. Engel, S. Mannor, and R. Meir, “The kernel recursive least squares algorithm,” *IEEE Trans. on Sig. Proc.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[3] W. Liu, P. P. Pokharel, and J. C. Príncipe, “The kernel least-mean-square algorithm,” *IEEE Trans. on Sig. Proc.*, vol. 56, no. 2, pp. 543–554, 2008.

[4] C. Richard, J. C. M. Bermudez, and P. Honeine, “Online prediction of time series data with kernels,” *IEEE Trans. on Sig. Proc.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.



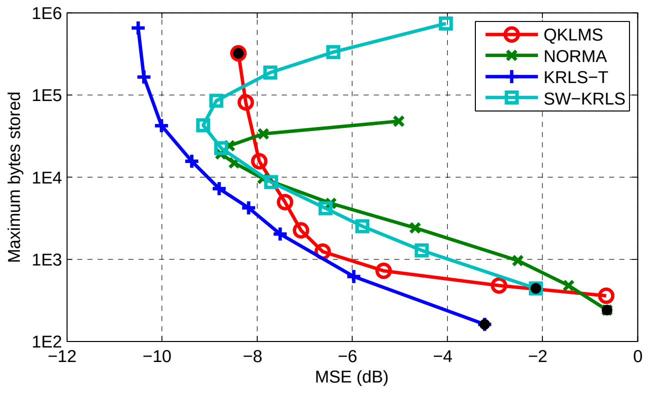


Fig. 5. Average FLOPS (left) and maximum memory usage (right), as a function of MSE, for the channel identification problem.

Table 3. Parameters used in the MIMO test bed identification. A Gaussian kernel with $\sigma = 3.1$ was used.

Method	Fixed parameters	Varying parameter
QKLMS	$\eta = 0.6$	$\epsilon_{\mathbb{U}} = 0.1, 1, 2, 3, 4, 5, 6, 7, 8$
NORMA	$\eta = 0.4, \lambda = 10^{-4}$	$\tau = 5, 10, 20, 50, 100, 200, 310, 400, 500, 700, 1000$
KRLS-T	$\lambda = 0.995, \sigma_n^2 = 0.015$	$M = 2, 5, 10, 15, 20, 30, 50, 100, 200$
SW-KRLS	$c = 0.015$	$M = 5, 10, 15, 20, 30, 50, 70, 100, 150, 200, 300$

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

[6] S. Van Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría, “Kernel recursive least-squares tracker for time-varying regression,” *IEEE Trans. on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1313–1326, Aug. 2012.

[7] M. Yukawa, “Multikernel adaptive filtering,” *IEEE Trans. on Sig. Proc.*, vol. 60, no. 9, pp. 4672–4682, 2012.

[8] W. D. Parreira, J. C. M. Bermudez, C. Richard, and J.-Y. Tourneret, “Stochastic behavior analysis of the Gaussian kernel least-mean-square algorithm,” *IEEE Trans. on Sig. Proc.*, vol. 60, no. 5, pp. 2208–2222, 2012.

[9] S. Van Vaerenbergh, J. Vía, and I. Santamaría, “A sliding-window kernel RLS algorithm and its application to nonlinear channel identification,” in *2006 IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, Toulouse, France, May 2006.

[10] S. Van Vaerenbergh, I. Santamaría, W. Liu, and J. C. Príncipe, “Fixed-budget kernel recursive least-squares,” in *2010 IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, Dallas, USA, Apr. 2010.

[11] A. Sayed, *Fundamentals of adaptive filtering*, Wiley-IEEE Press, 2003.

[12] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[13] B. Schölkopf and A. J. Smola, *Learning with Kernels*, The MIT Press, Cambridge, MA, USA, 2002.

[14] S. Van Vaerenbergh, I. Santamaría, and M. Lázaro-Gredilla, “Estimation of the forgetting factor in kernel recursive least squares,” in *2012 IEEE Int. Workshop on Mach. Learn. for Sig. Proc. (MLSP)*, Sept. 2012.

[15] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, Wiley, 2010.

[16] F. Takens, “Detecting strange attractors in turbulence,” *Dynamical Systems and Turbulence*, vol. 898, pp. 366–381, 1981.

[17] F. Ernst, *Compensating for quasi-periodic motion in robotic radiosurgery*, Springer, 2012.

[18] J. Gutiérrez, Ó. González, Pérez, D. Ramírez, L. Vielva, J. Ibáñez, and I. Santamaría, “Frequency-domain methodology for measuring MIMO channels using a generic test bed,” *IEEE Trans. on Instrumentation and Measurement*, vol. 60, no. 3, pp. 827–838, Mar. 2011.