An efficient implementation of the kernel affine projection algorithm

Felix Albu, Dinu Coltuc
Faculty of Electronics
Valahia University of Targoviste
Targoviste, Romania
felix.albu@valahia.ro

Kiyoshi Nishikawa
Faculty of System Design
Tokyo Metropolitan University
Tokyo, Japan
knishikawa@m.ieice.org

Marius Rotaru
Dept. of Telecommunications
University Politehnica of Bucharest
Bucharest, Romania
marius.rotaru@gmail.com

***Abstract*— In this paper an efficient kernel affine projection algorithm using dichotomous coordinate descent iterations is proposed. The effectiveness of the proposed algorithm for nonlinear system identification and forward prediction is confirmed by computer simulations.**

Keywords—kernel affine projection algorithm; dichotomous coordinate descent; nonlinear system identification; forward prediction;

I. INTRODUCTION

In system identification applications, the main goal is to identify an unknown system using an adaptive filter [1]. Linear adaptive filters have been used in a variety of applications, e.g., echo [2-4], active noise control [5-6], equalization in wireless communication channels [7] etc. One of the most promising algorithms is the affine projection (AP) algorithm [8] and its efficient implementations (e.g. [3-4]).

Recently, as an extension of the linear counterparts, kernel adaptive filters have been proposed that enable us to adaptively identify non-linear systems [8]. Kernel adaptive filters [9] are derived by applying the kernel method to linear adaptive filters, and several algorithms were proposed, i.e., the kernel recursive least squares (KRLS) [10], the kernel least mean square (KLMS) [11], the kernel normalized LMS (KNLMS) [12], the kernel proportionate NLMS (KPNLMS) [13], kernel affine projection (KAPA) [12], the kernel ERLS-DCD [14] algorithms etc.

The paper proposes an efficient implementation of the kernel affine projection algorithm inspired from the methods used for the affine projection algorithms. There is a resemblance between KAPA and the variable order affine projection algorithms [3]. The matrix updating procedure for a variable projection order affine projection was firstly presented in [2]. Also, the dichotomous coordinate descent (DCD) iterations [15] were firstly used for solving the linear systems of the fast affine projection algorithm in [3]. To the best of our knowledge, the DCD technique hasn't been applied yet to the kernel affine projection based algorithms.

This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS - UEFISCDI, project number PN-II-ID-PCE-2011-3-0097 and by the Sectorial Operational Programme "Human Resources Development" 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/107/1.5/S/76813.

The paper is organized as follows. Section II presents the proposed implementation, while in Section III its numerical complexity is investigated. The simulation results presented in Section IV compare the proposed algorithm with KAPA in different scenarios. Finally, the conclusions are given.

II. DCD-KAPA

A. A review of the kernel affine projection algorithm

The conventional kernel filters were described in many papers (e.g. [9], [12], [16]). The input signal $x(n)$ at moment n is transformed into a high dimensional feature space F by a transformation function $\Phi(x)$ and the output of the adaptive filter is given by

$$f\left(\mathbf{x}(n)\right) = \Phi^T\left(\mathbf{x}(n)\right)\mathbf{w}(n), \tag{1}$$

where $\mathbf{w}(n) = \left[w_0(n), w_1(n), ..., w_{M-1}(n)\right]$, is the filter coefficient vector of the adaptive filter, $w_i(n)$, and M are the i-th coefficient of the filter at moment n and the length of the filter respectively, and $\mathbf{x}(n) = \left[x(n), x(n-1), ..., x(n-M+1)\right]$. We assume that the filter vector $\mathbf{w}(n)$ can be expressed as a linear combination of m vectors $\Phi(\mathbf{y}(j))$ as

$$\mathbf{w}(n)=\sum_{j=1}^{m}\alpha_j\Phi(\mathbf{y}(j)). \tag{2}$$

The vectors $\mathbf{y}(j)$ are a subset of $\mathbf{x}(l), l=0,1,...,n-1$ and α_j is the weight corresponding to $\mathbf{y}(j)$. Then, the output in (1) is expressed [9] as

$$f(\mathbf{x}(n))=\sum_{j=1}^{m}\alpha_j\left(\Phi^T(\mathbf{x}(n))\Phi(\mathbf{y}(j))\right). \tag{3}$$

In the kernel adaptive filter, $\boldsymbol{\alpha}(n)=[\alpha_1,\alpha_2,...,\alpha_m]^T$ is the coefficient vector of the adaptive filter instead of $\mathbf{w}(n)$ [11-12]. In these algorithms the inner product $\Phi^T(\mathbf{x}(n))\Phi(\mathbf{y}(j))$ in Eq. (3) is obtained via the kernel function $k(\cdot,\cdot)$ used to calculate the inner product in the space F [11]:

$$\forall\mathbf{a},\mathbf{b}\in X \qquad k(\mathbf{a},\mathbf{b})=\Phi^T(\mathbf{a})\Phi(\mathbf{b}) \tag{4}$$

The Gaussian kernel is defined as:

$$k(\mathbf{a},\mathbf{b})=\exp\left(-\zeta\|\mathbf{a}-\mathbf{b}\|^2\right), \tag{5}$$

where $\|\cdot\|$ is the Euclidean norm and ζ is the kernel parameter. Another kernel is the Laplace kernel:

$$k(\mathbf{a},\mathbf{b})=\exp\left(-\zeta\|\mathbf{a}-\mathbf{b}\|\right), \tag{6}$$

The kernel AP algorithm (KAPA) was proposed in [12]. First, we rewrite Eq. (3) as $f(\mathbf{x}(n))=\mathbf{h}(n)\boldsymbol{\alpha}(n)$ where

$$\mathbf{h}(n)=\left[k(\mathbf{x}(n),\mathbf{y}(1)),...,k(\mathbf{x}(n),\mathbf{y}(m))\right]^T \\ =[h_1,...,h_m]^T. \tag{7}$$

We define the matrix *D*, called the dictionary, as $\mathbf{D}=[\mathbf{y}(1),...,\mathbf{y}(m)]$. The vectors stored in the dictionary $\mathbf{D}$ are *m* ($m\le n$) input vectors of the previous time, where *m* is a variable determined by the algorithm below. Let us denote $\mathbf{D}$ at time *n* by $\mathbf{D}_n$. Then, $\mathbf{D}_n$ and $\mathbf{h}(n)$ are updated according to the pseudo algorithm from [16]. The size of $\mathbf{D}_n$ is increased if $\max_{j=1,...,m}|k(\mathbf{x}(n),\mathbf{y}(j))|<\gamma_0$ [12]. The value of the threshold $\gamma_0\in[0..1]$ is determined according to the sparseness of the signal [16]. The kernel output error vector is

$$\mathbf{e}(n)=\mathbf{d}(n)-\mathbf{H}(n)\boldsymbol{\alpha}(n-1) \tag{8}$$

where $\mathbf{d}(n)=[d(n),...d(n-p+1)]^T$, p is the order and $\mathbf{H}(n)$ denotes the *p*-by-*m* matrix whose (*i,j*)th entry is $k(\mathbf{x}(n-i+1),\mathbf{y}(j))$. If δ is a small regularization and we note

$$\mathbf{s}(n)=\left(\mathbf{H}(n)\mathbf{H}^T(n)+\delta\mathbf{I}\right)^{-1}\mathbf{e}(n) \tag{9}$$

the filter coefficients $\boldsymbol{\alpha}(n)$ of KAPA are updated as follows: [12]:

$$\boldsymbol{\alpha}(n)=\boldsymbol{\alpha}(n-1)+\mu\mathbf{H}^T(n)\mathbf{s}(n) \tag{10}$$

where μ is the normalized step-size parameter in the range $0<\mu<2$.

B. The proposed DCD-KAPA

It can be noticed that, if we note $\mathbf{R}(n)=\mathbf{H}(n)\mathbf{H}^T(n)+\delta\mathbf{I}$ Eq. (9) is equivalent with solving the linear system

$$\mathbf{R}(n)\mathbf{s}(n)=\mathbf{e}(n) \tag{11}$$

This linear system can be solved by using the DCD iterations. The DCD algorithm is a multiplication-free and division-free algorithm based on binary representation of elements of the solution vector with M_b bits within an amplitude range $[-H_S,H_S]$, where H_S is chosen as a power of two [15]. The divisions and multiplications are replaced by bit-shift operations. The DCD iterations start by updating the most significant bit of its elements and proceeds to less significant bits. The algorithm complexity is limited by N_u, the maximum number of “successful” iterations. The peak complexity of the DCD algorithm for the linear system of Eq. (11), M_b and N_u, is $p(2N_u+M_b)$ additions. Another efficient DCD version was proposed in [17] and finds a ‘leading’ element of the solution to be updated. We will concentrate on the first option implemented as follows [3]:

Initialization: $\mathbf{s}(n)=0, d_s=H_s, q=0.$

```
for $o = 1 : M_b$
    $d_s = d_s / 2$
(j) Flag = 0
for $i = 0 : m-1$
    if $|e_i(n)| > (d_s/2)[\mathbf{R}(n)]_{ii}$ ,then
        Flag = 1, $q = q+1$
            $s_i(n) = s_i(n) + \mathrm{sgn}(e_i(n)) \cdot d_s$
            $\mathbf{e}(n) = \mathrm{e}(n) - \mathrm{sgn}(e_i(n)) \cdot d_s \cdot \mathbf{R}(:,i)$
        if $q > N_u$, stop
    end of the $i-$loop
    if Flag = 1, go to (j)
 end of the $o-$loop
```

C. *Efficient implementation of the proposed method*

The computation of the matrix $\mathbf{R}(n)$ can be made in an efficient way taking into account its structure. The size of this square matrix varies depending on the projection order. A similar situation is encountered in variable projection affine projection algorithms [2]. There are two possible situations. In the case without order increase, the matrix $\mathbf{R}(n)$ is updated by replacing only the first row and column with the vector $\mathbf{r}(n)$ defined below, while the bottom $(m-1)\times m$ sub-matrix is replaced with the top $(m-1)\times m$ sub-matrix of $\mathbf{R}(n-1)$. For the first m iterations the full $\mathbf{R}(n-1)$ is used. The first element of $\mathbf{r}(n)$ is $\mathbf{h}(n)\mathbf{h}^T(n)+\delta$ while the remaining m – 1 elements are given by $\underline{\mathbf{H}}(n)\mathbf{h}(n)$ where $\underline{\mathbf{H}}(n)$ represents the lower $(p-1)\times m$ sub-matrix of $\mathbf{H}(n)$.

In the second situation, when there is an order increase, the matrix $\mathbf{R}(n)$ is updated by replacing the first row and column with a modified vector. The last m – 1 elements of the last column of $\mathbf{H}^T(n)$ are taken into account in order to update $\mathbf{R}(n)$ and denoted by $\mathbf{q}(n)$. For this case, remaining m – 1 elements of $\mathbf{r}(n)$ are given by $\underline{\mathbf{H}}(n)\mathbf{h}(n)+\mathbf{q}(n)$, while the bottom $(m-1)\times m$ sub-matrix is replaced with the sum between the top $(m-1)\times m$ sub-matrix of $\mathbf{R}(n-1)$ and $\mathbf{q}^T(n)\mathbf{q}(n)$. This update leads to important complexity reduction, as it is shown in the next section.

III. NUMERICAL COMPLEXITY

Table 1 reports the computational costs of KAPA and DCD-KAPA. It can be seen that the number of multiplications and additions of DCD-KAPA increase linearly with m, while the numerical complexity of KAPA is $O(m^2)$ [12]. The number of additions of DCD-KAPA depends on the M_b and N_u. Fig. 1 shows the ratios of numerical complexities of DCD-KAPA and KAPA for different N_u values, p = 5 and $M_b = 16$. It can be seen that the savings depend on the N_u value and if there is or not an order increase.

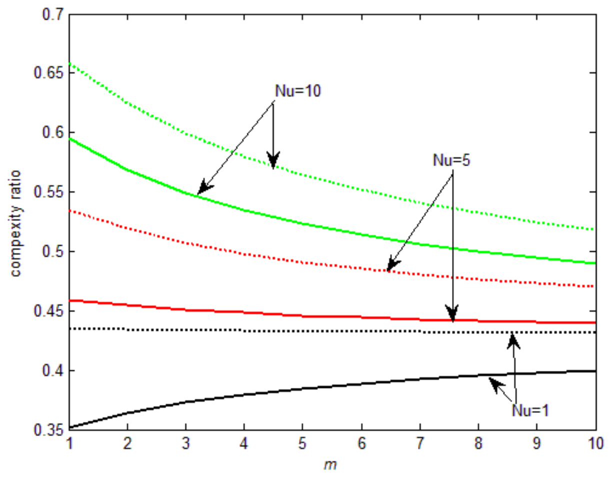

Fig. 1: A comparison of the complexity ratios of DCD-KAPA and KAPA in terms of multiplications and additions and different N_u values; without order increase (solid line), with order increase (dotted line).

TABLE I. COMPUTATIONAL COST PER ITERATION OF KAPA AND DCD-KAPA

		without order increase	with order increase
KAPA[12]	x	$(p^2+2p)m+p^3+p$	$(p^2+2p)m+p^3+2p^2+p$
	+	$(p^2+2p)m+p^3+p^2$	$(p^2+2p)m+p^3+p^2+p-1$
DCD-KAPA	x	$3pm+p$	$3pm+p^2+2p$
	+	$3pm+p(2N_u+M_b+1)$	$3pm+p^2+1+p(2N_u+M_b)$

The savings are substantial (e.g. around 50 % for $N_u = 5$). As expected, a lower N_u value leads to higher complexity savings. The computational cost of $\mathbf{h}(n)$ depends on the selected kernel and it is not taken into account [12]. The final size of a dictionary of kernel functions is finite and after a transient period during which the order of the model increases, computational complexity is reduced to that without order increase [12].

IV. SIMULATION RESULTS

The performance of the proposed algorithm for system identification and forward prediction problems were investigated by computer simulations. For all the simulations a white Gaussian noise of SNR =40 dB with zero mean was added, $\delta = 0.07$, $\gamma_0 = 0.3$, and $\mu = 0.01$ were used. The data were generated from the initial condition $v(0) = 0.5$ [12]. The input was sampled from a zero-mean Gaussian distribution with standard deviation 0.25.

$$v(n) = 1.1\exp\left(-\left|v(n-1)\right|\right) + u(n); \qquad (12)$$
$$d(n) = v^2(n)$$

The system output was corrupted by an additive zero-mean white Gaussian noise with standard deviation equal to 1 and the signal-to-noise ratio was −4.0 dB [12]. The Laplace kernel with $\zeta = 0.35$ is used. Fig. 2 shows the comparison of the performance of KAPA and DCD-KAPA in terms of the mean squared errors (MSEs) averaged over 100 independent trials. The filter length was M = 5. It can be seen from Fig. 2 than DCD-KAPA with 10 DCD iterations obtains a close performance to KAPA. As expected, increasing the number of updates lead to improved approximation (around 2 dB for 1 update and less than 0.3 dB for 10 updates).

Fig. 3 examines the performance of the investigated algorithms for a forward prediction problem. The Gaussian kernel with $\zeta = 0.13$ was used. For the forward prediction example the following equation was used:

$$x(n) = \left(0.8 - 0.5\exp\left(-x(n-1)^{-2}\right)\right)x(n-1) - \qquad (13)$$
$$\left(0.3 + 0.9\exp\left(-x(n-1)^{2}\right)\right)x(n-2)$$
$$+0.1\sin\left(x(n-1)\pi\right).$$

It can be noticed from Fig. 3 that DCD-KAPA with $N_u = 5$ has almost identical performance with KAPA. The same conclusions were obtained for previous situations for higher filter lengths.

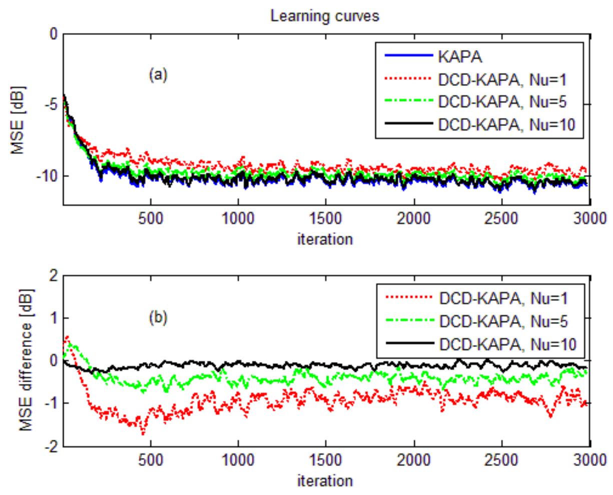

Fig. 2: a) Comparison of convergence characteristics of KAPA and DCD-KAPA with different number of updates applied to system identification; b) the MSE difference.

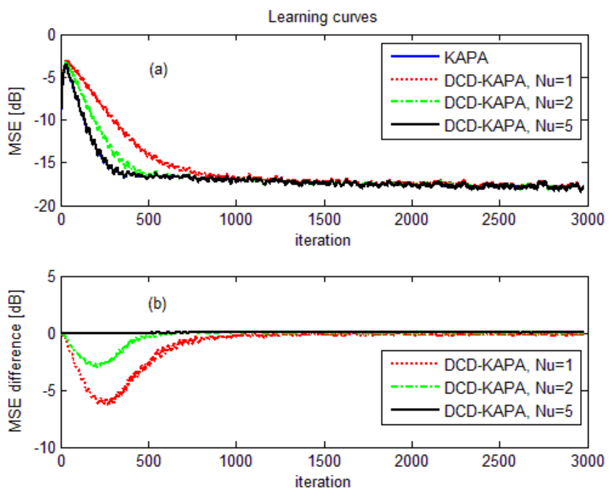

Fig. 3: a) Comparison of convergence characteristics of KAPA and DCD-KAPA with different number of updates applied to a forward prediction; b) the MSE difference.

The influence of the coherence value γ_0 is examined in Fig. 4 for the previous test cases. Steady-state performance of the algorithms was measured by the mean-square prediction error over the last 500 samples of each time series averaged over 100 independent runs. The value of γ_0 was varied from 0.05 to 0.95 in increments of 0.05. It can be seen from Fig. 4 that the best coherence value depends on the application and the filter length. For example in case of the first example, the optimum values for γ_0 are 0.7, 0.2 and 0.35 for p = 2, p = 5 and p = 10 respectively. In the forward prediction case, the optimum values for γ_0 are 0.9, 0.75 and 0.8 for p = 2, p = 5 and p = 10 respectively. These parameters are the same for DCD-KAPA for sufficient number of updates.

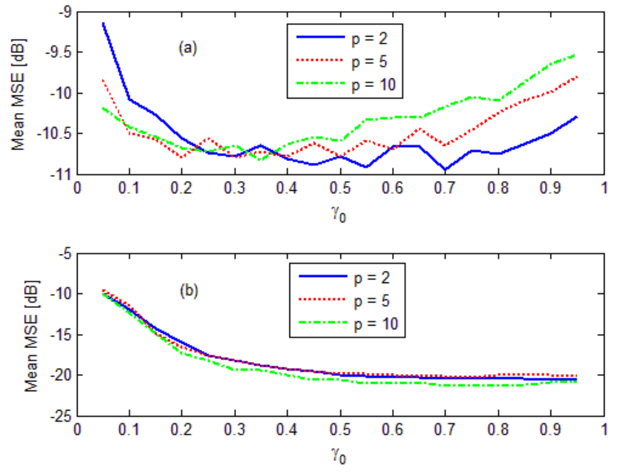

Fig. 4: Mean MSE of KAPA for the last 500 iterations for different γ_0 values in three cases: $p = 2$, $p = 5$ and $p = 10$; a) nonlinear system identification case; b) forward prediction case

V. CONCLUSIONS

In this paper, an efficient implementation of KAPA using dichotomous coordinate descent iterations is proposed. The suitable number of DCD updates was investigated for system identification and forward prediction situations. The influence of the coherence parameter has been studied too. It is shown that the proposed DCD-KAPA implementation can achieve an important complexity reduction over KAPA.

REFERENCES

[1] A.H. Sayed, Fundamentals of Adaptive Filtering, John Wiley & Sons, 2003.

[2] F. Albu, C. Paleologu, J. Benesty, "A Variable Step Size Evolutionary Affine Projection Algorithm", in Proc. of ICASSP 2011, Prague, Czech Republic, May 2011, pp. 429-432.

[3] Y. Zakharov and F. Albu, " Coordinate descent iterations in fast affine projection algorithm ", IEEE Signal Processing Letters, Vol. 12, Issue 5, May 2005, pp: 353-356.

[4] F. Albu, D. Coltuc, D. Comminiello, M. Scarpiniti, "The variable step size regularized block exact affine projection algorithm", " in Proc. of ISETC 2012, Nov. 2012, pp. 283-286.

[5] A. Gonzales, F. Albu, M. Ferrer, M. Diego, "Evolutionary and variable step size affine projection algorithms for active noise control ", IET Signal Processing, Special issue on active noise control, 2013, in press.

[6] F. Albu, "Leading Element Dichotomous Coordinate Descent Exponential Recursive Least Squares Algorithm for Multichannel Active Noise Control", in Proc. of AAS Acoustics 2012, Nov. 2012, pp. 1-4.

[7] G. E. Bottomley, "Channel Equalization for Wireless Communications: From Concepts to Detailed Mathematics", John Wiley & Sons, 2011

[8] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties", Electron. Commun. Jpn., vol. 67-A, no. 5, pp. 19–27, 1984.

[9] W. Liu, J.C. Principe, and S. Haykin, Kernel Adaptive Filtering, Wiley, 2010.

[10] Y. Engel, S. Mannor, and R. Meir, The Kernel Recursive Least-Squares Algorithm, IEEE Transactions on Signal Processing, vol. 52, pp. 2275–2285, 2004.

[11] W. Liu, P.P. Pokharel, and J. C. Principe, The Kernel Least-Mean-Square Algorithm, IEEE Transactions on Signal Processing, vol. 56, pp. 543–554, 2008.

[12] C. Richard, J. C. M. Bermudez, and P. Honeine, Online Prediction of Time Series Data With Kernels, IEEE Transactions on Signal Processing, vol. 57, pp. 1058–1067, 2009.

[13] F. Albu, K. Nishikawa, "The Kernel Proportionate NLMS Algorithm", accepted at EUSIPCO 2013.

[14] Y. Ogawa and K. Nishikawa, A Kernel Adaptive Filter based on ERLS-DCD Algorithm, in Proc. of Intl Tech. Conf. Circuits Systems, Computer, Communications 2011, no.P4-13, pp.1228–1231, 2011.

[15] Y. Zakharov, T. Tozer, Multiplication-free iterative algorithm for LS problem, Electron. Lett., 2004, 40, (9), pp. 567-569.

[16] H. Nishikawa, and H. Nakazato, Mixture structure of kernel adaptive filters for improving the convergence characteristics, in Proc. of APSIPA 2012, 2012.

[17] J. Liu, Y. Zakharov, B. Weaver, Architecture and FPGA Design of Dichotomous Coordinate Descent Algorithms, IEEE Trans. Circuits and Systems. Part I: Regular Papers, 56(11):2425-2438, Nov., 2009.

The codes for the proposed algorithms can be obtained from http://falbu.50webs.com/List_of_publications_ka.htm

The reference for the paper is:
F. Albu, D. Coltuc, K. Nishikawa, M. Rotaru, "An efficient implementation of the kernel affine projection algorithm", in *Proc. of ISPA 2013*, September 2013, Trieste, Italy, pp. 342-346