DISTORTION COMPENSATION OF NONLINEAR SYSTEMS BASED ON INDIRECT LEARNING ARCHITECTURE

Emad Abd-Elrady, Li Gan and Gernot Kubin

Christian Doppler Laboratory for Nonlinear Signal Processing
Institute of Signal Processing and Speech Communication
Graz University of Technology. Inffeldgasse 16b, A-8010 Graz, Austria.
E-mail: emad.abd-elrady@tugraz.at, li.gan@tugraz.at, gernot.kubin@tugraz.at

ABSTRACT

Distortion compensation of nonlinear systems is an important topic in many practical applications. This paper concerns with linearization of nonlinear systems which can be modeled using Volterra series by connecting two adaptive nonlinear Volterra filters. The first one is a training filter connected in parallel with the nonlinear system and its kernels are estimated recursively. The second adaptive filter is a predistorter connected tandemly with the nonlinear system and its kernels are a copy from the training filter. Three recursive algorithms, namely: the Recursive Least Squares (RLS), the Kalman Filter (KF), and the Recursive Prediction Error Method (RPEM) algorithms, are developed and studied using numerical simulations. Simulation studies for time-invariant and time-varying nonlinear systems show that the KF and RPEM algorithms provide lower nonlinear distortion as compared to the RLS algorithm.

1. INTRODUCTION

Digital compensation of nonlinear distortion is an essential requirement in many applications. In wireless communication systems, the nonlinearity of high power amplifiers is an obstacle to increase the transfer data rate and mobility. In Hi-Fi systems, small distortion produced by nonlinear components dominates the overall performance. Further more examples can be found in communication systems, speech processing, and control engineering, see [1-3]

In this paper, the Indirect Learning Architecture (ILA) method [4, 5] shown in Fig. 1 is considered for distortion compensation of nonlinear systems. The coefficients of the predistorter are a copy of the coefficients of a training filter connected in parallel with the nonlinear system. These coefficients can be estimated recursively, as done in [4], using the Recursive Least Squares (RLS) algorithm, see [6, 7]. Also, in this paper, the Kalman Filter (KF) and the Recursive Prediction Error Method (RPEM) algorithms [6, 7] are developed for this approach. Moreover, the performance of these three algorithms is studied in case the nonlinear system is time-varying.

This paper is organized as follows. In Sec. 2, a review for the ILA method is given. Sec. 3 discusses the RLS algorithm for estimating the kernels of the training Volterra filter and hence the kernels of the Volterra predistorter. The KF and RPEM algorithms are developed in Sec. 4 and Sec. 5, respectively. Some simulation examples are given in Sec. 6. Conclusions are presented in Sec. 7.

2. THE INDIRECT LEARNING ARCHITECTURE

Assume that the nonlinear system $\boldsymbol{H}_{(q)}$ to be compensated in Fig. 1 is a discrete-time (possibly time-varying) causal system. Also, the

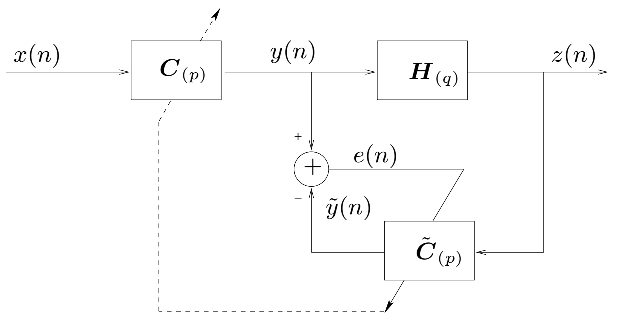

Fig. 1. Compensation of nonlinear distortion using the ILA method.

system $\boldsymbol{H}_{(q)}$ with input and output signals $y(n)$ and $z(n)$ can be modeled by qth-order Volterra series with M-tap memories. Hence

$$z(n) = \sum_{k=1}^{q} \Big(\sum_{i_1=0}^{M-1} .. \sum_{i_k=0}^{M-1} h_k(i_1, \cdots, i_k; n) y(n-i_1)..y(n-i_k) \Big) \tag{1}$$

where $h_k(i_1, \cdots, i_k; n)$ are the kth-order kernels.

Similarly, the relation between the input and output of the predistorter $\boldsymbol{C}_{(p)}$ is given by

$$y(n) = \sum_{k=1}^{p} \Big(\sum_{i_1=0}^{N-1} .. \sum_{i_k=0}^{N-1} c_k(i_1, \cdots, i_k; n) x(n-i_1)..x(n-i_k) \Big) \tag{2}$$

where N is the number of memories and $c_k(i_1, \cdots, i_k; n)$ are the kth-order kernels. According to the pth-order Volterra theorem [8], $\boldsymbol{C}_{(p)}$ can remove nonlinearities up to pth-order provided that the inverse of the first-order Volterra system is causal and stable.

As a direct approach, the predistorter $\boldsymbol{C}_{(p)}$ is an adaptive filter whose kernels $c_k(i_1, \cdots, i_k; n)$ can be estimated recursively as done in [3] using the nonlinear filtered-x least mean squares algorithm. In this paper, the kernels $c_k(i_1, \cdots, i_k; n)$ are estimated *indirectly* as a copy of the kernels of the training Volterra filter $\tilde{\boldsymbol{C}}_{(p)}$, see Fig. 1. The same approach was used in [4, 5]

The input-output relation of the training filter is given as

$$\tilde{y}(n) = \sum_{k=1}^{p} \Big(\sum_{i_1=0}^{N-1} .. \sum_{i_k=0}^{N-1} \tilde{c}_k(i_1, \cdots, i_k; n) z(n-i_1)..z(n-i_k) \Big) \tag{3}$$

where $\tilde{c}_k(i_1, \cdots, i_k; n)$ are the kth-order kernels.

Let us define an error signal $e(n)$ to be given by

$$e(n) = y(n) - \tilde{y}(n) \tag{4}$$

As this error signal approaches zero, the overall output of the system $z(n)$ (input to the training filter $\tilde{\boldsymbol{C}}_{(p)}$) approaches the total system input $x(n)$ (input to the predistorter $\boldsymbol{C}_{(p)}$) since the outputs of the two Volterra models, *i.e.* $y(n)$ and $\tilde{y}(n)$, approach each other. When the kernels $\tilde{c}_k(i_1, \cdots, i_k; n)$ and indirectly $c_k(i_1, \cdots, i_k; n)$ have been found and it is believed that the nonlinear system characteristics are not changing, the training branch is shutted down. The training branch can be reconnected in case of significant change in the characteristics of $\boldsymbol{H}_{(q)}$ and hence high nonlinear distortion level investigated at the receiver end. On the other hand, in case of time-varying nonlinear systems, the training branch should be always connected.

The ILA method has two drawbacks that degrade its performance. Namely: the convergence of the training filter to a biased estimates due to the noisy measurement of $z(n)$ and the fact that the postinverse and the preinverse of the nonlinear system are not always equal. Some new techniques are proposed in [9, 10] in order to improve the performance of the ILA method.

In this paper, the predistorter kernels are estimated recursively using the RLS, KF and RPEM algorithms [6, 7]. This is the topic of the next sections.

3. THE RLS ALGORITHM

Let us define the kernels vector $\tilde{\boldsymbol{C}}$ of the training filter $\tilde{\boldsymbol{C}}_{(p)}$ as

$$\tilde{\boldsymbol{C}}^T = \begin{pmatrix} \tilde{\boldsymbol{C}}_1 & \cdots & \tilde{\boldsymbol{C}}_p \end{pmatrix}, \tag{5}$$

where $\tilde{\boldsymbol{C}}_k$ is given by

$$\tilde{\boldsymbol{C}}_k^T(n) = \begin{pmatrix} \tilde{c}_k(0, \cdots, 0; n) & \cdots & \tilde{c}_k(N-1, \cdots, N-1; n) \end{pmatrix}. \tag{6}$$

Writing Eq. (2) and Eq. (3) in vector form, we have

$$y(n) = \boldsymbol{C}^T(n)\boldsymbol{X}(n) \tag{7}$$

$$\tilde{y}(n) = \tilde{\boldsymbol{C}}^T(n)\boldsymbol{Z}(n) \tag{8}$$

where $\boldsymbol{C}^T(n) = \tilde{\boldsymbol{C}}^T(n)$ and

$$\boldsymbol{X}(n) = \begin{pmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-N+1) \\ \vdots \\ x(n)x(n)\cdots x(n) \\ \vdots \\ x(n-N+1)x(n-N+1)\cdots x(n-N+1) \end{pmatrix} \tag{9}$$

$$\boldsymbol{Z}(n) = \begin{pmatrix} z(n) \\ z(n-1) \\ \vdots \\ z(n-N+1) \\ \vdots \\ z(n)z(n)\cdots z(n) \\ \vdots \\ z(n-N+1)z(n-N+1)\cdots z(n-N+1) \end{pmatrix}. \tag{10}$$

Hence using Eq. (7) and Eq. (8), the error $e(n)$ can be written as

$$e(n) = \tilde{\boldsymbol{C}}^T(n)(\boldsymbol{X}(n) - \boldsymbol{Z}(n)). \tag{11}$$

Therefore, if the Volterra models satisfy the conditions [4]

$$\begin{aligned} \boldsymbol{X}(n) \neq \boldsymbol{Z}(n) &\Longrightarrow y(n) \neq \tilde{y}(n) \\ \boldsymbol{X}(n) = \boldsymbol{Z}(n) &\Longrightarrow y(n) = \tilde{y}(n) \end{aligned} \tag{12}$$

the error signal $e(n)$ approaches zero, $\boldsymbol{Z}(n)$ approaches $\boldsymbol{X}(n)$, and hence $z(n)$ approaches $x(n)$. These conditions are simply mean that the nonlinearity of the system $\boldsymbol{H}_{(q)}$ is invertible.

The kernel vector $\tilde{\boldsymbol{C}}$ can be estimated as done in [4] using the RLS algorithm [6, 7]. The RLS algorithm minimizes the cost function $V_{RLS}(\tilde{\boldsymbol{C}})$ given by

$$V_{RLS}(\tilde{\boldsymbol{C}}) = \sum_{k=1}^{n} \lambda^{n-k} e^2(k) \tag{13}$$

where $\lambda \leq 1$ is the forgetting factor and $e(n)$ is given by Eq. (11). The smaller the value of λ, the quicker the information in previous data will be forgotten (*cf.* Sec. 4). Therefor, the choice of λ controls the ability of the algorithm to track time-varying parameters. The RLS algorithm takes the following form [6, 7]:

$$\begin{aligned} e(n) &= \tilde{\boldsymbol{C}}^T(n-1)(\boldsymbol{X}(n) - \boldsymbol{Z}(n)) \\ \boldsymbol{K}(n) &= (\lambda + \boldsymbol{Z}^T(n)\boldsymbol{P}(n-1)\boldsymbol{Z}(n))^{-1}\boldsymbol{P}(n-1)\boldsymbol{Z}(n) \\ \boldsymbol{P}(n) &= \lambda^{-1}\boldsymbol{P}(n-1) - \lambda^{-1}\boldsymbol{K}(n)\boldsymbol{Z}^T(n)\boldsymbol{P}(n-1) \\ \tilde{\boldsymbol{C}}(n) &= \tilde{\boldsymbol{C}}(n-1) + \boldsymbol{K}(n)e(n) \end{aligned} \tag{14}$$

The most common choice for the initial condition of $\boldsymbol{P}(n)$ is $\boldsymbol{P}(0) = \rho\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix and ρ is a constant reflects our trust in the initial parameter vector $\tilde{\boldsymbol{C}}(0)$.

4. THE KF ALGORITHM

The Kalman filter (KF) algorithm [6, 7] is a well studied algorithm that provides the optimal (mean square) estimate of the system state vector and also has the ability to track time-varying parameters. The KF is usually presented for state-space equation whose matrices may be time varying.

In order to construct the state space model for the ILA method, the kernel vector $\tilde{\boldsymbol{C}}$ is modeled as a random walk or a drift. Hence, the training filter has the following state-space model

$$\begin{aligned} \tilde{\boldsymbol{C}}(n+1) &= \tilde{\boldsymbol{C}}(n) + \boldsymbol{v}(n) \\ \tilde{y}(n) &= \tilde{\boldsymbol{C}}^T(n)\boldsymbol{Z}(n) \end{aligned} \tag{15}$$

where $E\{\boldsymbol{v}(n)\boldsymbol{v}^T(m)\} = \boldsymbol{R}_1\delta_{n,m}$ and the covariance matrix $\boldsymbol{R}_1$ describes how fast different components of $\tilde{\boldsymbol{C}}$ are expected to vary.

Applying Kalman filter to the state-space model (15) gives the following recursive algorithm [6, 7]:

$$\begin{aligned} e(n) &= \tilde{\boldsymbol{C}}^T(n-1)(\boldsymbol{X}(n) - \boldsymbol{Z}(n)) \\ \boldsymbol{K}(n) &= (1 + \boldsymbol{Z}^T(n)\boldsymbol{P}(n-1)\boldsymbol{Z}(n))^{-1}\boldsymbol{P}(n-1)\boldsymbol{Z}(n) \\ \boldsymbol{P}(n) &= \boldsymbol{P}(n-1) - \boldsymbol{K}(n)\boldsymbol{Z}^T(n)\boldsymbol{P}(n-1) + \boldsymbol{R}_1 \\ \tilde{\boldsymbol{C}}(n) &= \tilde{\boldsymbol{C}}(n-1) + \boldsymbol{K}(n)e(n) \end{aligned} \tag{16}$$

The design variable $\boldsymbol{R}_1$ plays a similar role to that of the forgetting factor λ in the RLS algorithm (14). In case of tracking time variations of the Volterra kernels, λ should be small or $\boldsymbol{R}_1$ is large. On the other hand, for good convergence properties and small variances of time-invariant kernels, λ should be close to 1 or $\boldsymbol{R}_1$ close to zero.

5. THE RPEM ALGORITHM

The RPEM algorithm [6, 7] is derived by a minimization of the cost function

$$V(\tilde{\boldsymbol{C}}) = \lim_{\mathcal{N}\to\infty} \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} E\left[\varepsilon^2(n,\tilde{\boldsymbol{C}})\right] \tag{17}$$

where $\varepsilon(n,\tilde{\boldsymbol{C}})$ is the prediction error defined as

$$\varepsilon(n,\tilde{\boldsymbol{C}}) = y(n) - \tilde{y}(n,\tilde{\boldsymbol{C}}) \tag{18}$$

The RPEM algorithm gives consistent parameter estimates under weak conditions in case the asymptotic cost function has a unique stationary point represents the true parameter vector [6, 7].

The formulation of the RPEM algorithm requires the negative gradient of $\varepsilon(n,\tilde{\boldsymbol{C}})$. Thus using Eqs. (8) and (18), we have

$$\boldsymbol{\psi}(n) = -\frac{d\varepsilon(n,\tilde{\boldsymbol{C}})}{d\tilde{\boldsymbol{C}}} = \frac{d\tilde{y}(n,\tilde{\boldsymbol{C}})}{d\tilde{\boldsymbol{C}}} = \boldsymbol{Z}(n). \tag{19}$$

Hence, the RPEM algorithm [6, 7] follows as

$$\begin{aligned}
&\varepsilon(n) = y(n) - \tilde{\boldsymbol{C}}^T(n-1)\boldsymbol{Z}(n)\\
&\lambda(n) = \lambda_o\lambda(n-1) + 1 - \lambda_o\\
&S(n) = \boldsymbol{\psi}^T(n)\boldsymbol{P}(n-1)\boldsymbol{\psi}(n) + \lambda(n)\\
&\boldsymbol{P}(n) = \big(\boldsymbol{P}(n-1)\\
&\qquad - \boldsymbol{P}(n-1)\boldsymbol{\psi}(n)S^{-1}(n)\boldsymbol{\psi}(n)^T(t)\boldsymbol{P}(n-1)\big)/\lambda(n)\\
&\tilde{\boldsymbol{C}}(n) = \tilde{\boldsymbol{C}}(n-1) + \boldsymbol{P}(n)\boldsymbol{\psi}(n)\varepsilon(n).
\end{aligned} \tag{20}$$

Here $\lambda(n)$ is a forgetting factor grows exponentially to 1 as $n \to \infty$ where the rate λ_o and the initial value $\lambda(0)$ are design variables.
Remark 1: In case of predistortion of time-varying nonlinear systems, the RPEM algorithm of (20) can be modified to:

$$\begin{aligned}
&\varepsilon(n) = y(n) - \tilde{\boldsymbol{C}}^T(n-1)\boldsymbol{Z}(n)\\
&S(n) = \boldsymbol{\psi}^T(n)\boldsymbol{P}(n-1)\boldsymbol{\psi}(n) + r_2\\
&\boldsymbol{P}(n) = \big(\boldsymbol{P}(n-1)\\
&\qquad - \boldsymbol{P}(n-1)\boldsymbol{\psi}(n)S^{-1}(n)\boldsymbol{\psi}(n)^T(t)\boldsymbol{P}(n-1)\big) + \boldsymbol{R}_3\\
&\tilde{\boldsymbol{C}}(n) = \tilde{\boldsymbol{C}}(n-1) + \boldsymbol{P}(n)\boldsymbol{\psi}(n)\varepsilon(n)
\end{aligned} \tag{21}$$

where r_2 and $\boldsymbol{R}_3$ are the gain design variables, see [7]. This modification transforms the problem into Kalman Filter (KF) formulation.
Remark 2: Since our system model is linear in parameters, the gradient calculation in Eq. (19) leads to $\boldsymbol{\psi}(n) = \boldsymbol{Z}(n)$. Hence, the two algorithms (16) and (21) are expected to perform similarly. An identical performance of these two algorithms is obtained in case the design variables are chosen as $r_2 = 1$ and $\boldsymbol{R}_3 = \boldsymbol{R}_1$.

6. SIMULATION STUDY

In this section, a comparison study between the RLS, KF, and RPEM algorithms is given using computer simulations. Two examples are given. In both examples, the nonlinear system $\boldsymbol{H}_{(q)}$ is a known second-order Volterra system. The predistorter $\boldsymbol{C}_{(p)}$ and the training filter $\tilde{\boldsymbol{C}}_{(p)}$ are also assumed to be second-order Volterra filters. This means that $q = p = 2$. Also, the number of memories is chosen to be $N = 3$. The input signal to the predistorter is chosen as a Gaussian random signal of 5×10^3 samples with variance 1 whose frequency band is limited to prevent aliasing [3, 8].

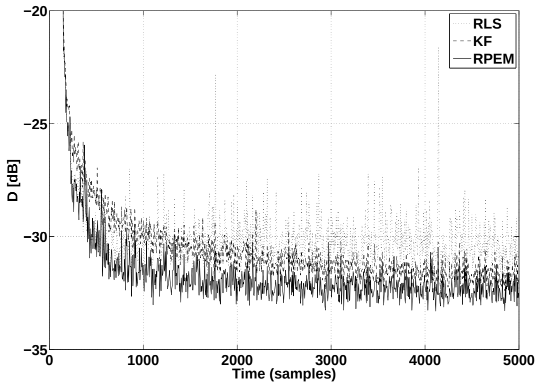

Fig. 2. Comparison between the RLS, KF and RPEM algorithms for a time-invariant nonlinear system.

As a measure of performance, the mean square distortion of the total system consists of the predistorter plus the nonlinear system is evaluated as [3]

$$D(n) = 10\,\log_{10}\left(\frac{\widehat{E}\{(z(n)-d(n))^2\}}{\widehat{E}\{d^2(n)\}}\right) \tag{22}$$

where $\widehat{E}\{.\}$ is the mean obtained by 10^3 independent realizations and $d(n)$ is the desired signal defined as

$$d(n) = x(n-\tau) + v(t). \tag{23}$$

Here τ is the time delay necessary to have a causal predistorter and $v(t)$ is zero-mean additive white Gaussian noise (AWGN). In this simulations, the desired signal $d(n)$ is chosen such that a signal to noise ratio (SNR) of 40 dB is achieved.
Remark 3: The delay time τ equals zero in case the system to be compensated is minimum phase [3].

Example 1: Compensation of distortion for a *time-invariant* nonlinear system.
In this example, a comparison study between the RLS, KF, and RPEM algorithms is given. The input-output relation of the nonlinear system $\boldsymbol{H}_{(2)}$ is chosen as

$$z(n) = \boldsymbol{H}_{(2)}\,[y(n)] = \boldsymbol{H}_1\,[y(n)] + \boldsymbol{H}_2\,[(y(n)] \tag{24}$$

where the first-order kernel matrix $\boldsymbol{H}_1$ is given as

$$\boldsymbol{H}_1 = \begin{pmatrix} 0.5625 & 0.4810 & 0.1124 & -0.1669 \end{pmatrix} \tag{25}$$

and the second-order kernel matrix $\boldsymbol{H}_2$ is

$$\boldsymbol{H}_2 = \begin{pmatrix} 0.1749 & 0 & 0 \\ 0 & 0 & -0.0875 \\ 0 & -0.0875 & \end{pmatrix}. \tag{26}$$

The simulation results are shown in Fig. 2. All the three algorithms were initialized with $\boldsymbol{C}(0) = \boldsymbol{1}$ and $\boldsymbol{P}(0) = 10\boldsymbol{I}$. Also, $\lambda = 0.99$, $\boldsymbol{R}_1 = 10^{-6}\boldsymbol{I}$, $\lambda_o = 0.99$, and $\lambda(0) = 0.95$. The results of Fig. 2 show that the KF and the RPEM algorithms provide a lower total distortion than the RLS algorithm. On average, the achieved values of D are -30.53 dB, -31.64 dB and -32.36 for the RLS, KF and RPEM algorithms, respectively.

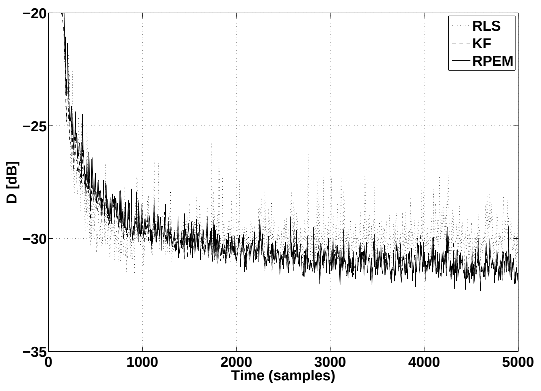

Fig. 3. Comparison between the RLS, KF and RPEM algorithms for a time-varying nonlinear system.

Example 2: Compensation of distortion for a *time-varying* nonlinear system.
In order to investigate the ability of the three recursive algorithms studied in this paper to track time variations in the kernels of the nonlinear system $\boldsymbol{H}_{(q)}$, the following simulation study was performed.

First, a zero-mean white Gaussian noise with a covariance matrix of $10^{-6}\boldsymbol{I}$ was added to the elements of the kernel matrices described by Eqs. (24)-(26) and the data were generated as done in Example 1. In this case, the RPEM algorithm (21) was considered. Also, the three algorithms were initialized as in Example 1 except $\boldsymbol{R}_1 = 10^{-8}\boldsymbol{I}$, $r_2 = 1.2$ and $\boldsymbol{R}_3 = 10^{-6}\boldsymbol{I}$. The simulation results are shown in Fig. 3. Again, it can be noticed from this figure that the KF and RPEM algorithms achieve lower total distortion than the RLS algorithm. On average, the achieved values of D are -30.08 dB, -31.31 dB and -31.26 for the RLS, KF and RPEM algorithms, respectively.

Finally, it is assumed to have an abrupt change in the nonlinear system at $n = 500$ and the elements of the kernel matrices are increased by 10% from its original value. The simulation results are given in Fig. 4. The three algorithms were initialized with $\lambda = 0.99$, $\boldsymbol{R}_1 = 10^{-6}\boldsymbol{I}$, $r_2 = 1.2$ and $\boldsymbol{R}_3 = 10^{-6}\boldsymbol{I}$. The results show that the RLS algorithm is the fastest to recover from this abrupt change as compared to the KF and RPEM algorithms which perform quite similar and achieve a lower total distortion than the RLS algorithm. On average, the achieved values of D are -31.36 dB, -32.32 dB and -32.66 for the RLS, KF and RPEM algorithms, respectively.

7. CONCLUSIONS

The ILA method has been considered in this paper to estimate the parameters of the predistorter that is used to linearize nonlinear systems. The parameters of the training filter are estimated recursively using the RLS, KF, and RPEM algorithms. These parameters are copied simultaneously to the predistorter that has the same order and memory length as the training filter. These different recursive algorithms have been studied for time-invariant and time-varying second-order nonlinear Volterra systems. Simulation studies show that the KF and RPEM algorithms achieve lower total distortion as compared to the RLS algorithm. On the other hand, the RLS algorithm shows a faster convergence speed in case of abrupt changes in the kernel values of the nonlinear system.

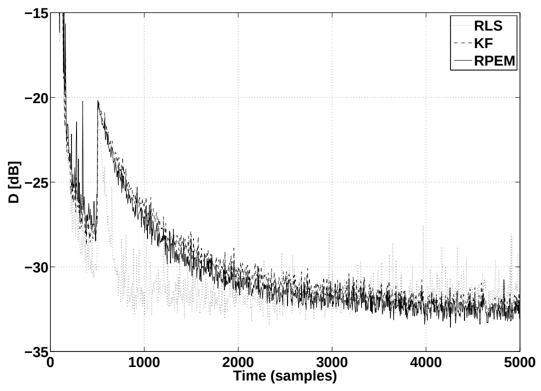

Fig. 4. Comparison between the RLS, KF and RPEM algorithms for an abrupt change.

Acknowledgements

This work was supported by Infineon Technologies Austria AG. The authors would also like to thank Dr. David Schwingshackl and Dr. Gerhard Paoli for valuable discussions during this work.

8. REFERENCES

[1] X. Y. Gao and W. M. Snelgrove, “Adaptive linearization schemes for weakly nonlinear systems using adaptive linear and nonlinear FIR filters,” in *Proc. of the 33rd Midwest Symposium on Circuits and Systems*, 1990.

[2] P. Singerl and H. Koeppl, “A low-rate identification method for digital predistorters based on volterra kernel interpolation,” in *Proc. of The 48th Midwest Symposium on Circuits and Systems*, Ohio, USA, 2005, pp. 1533–1536.

[3] Y. H. Lim, Y. S. Cho, I. W. Cha, and D. H. Youn, “An adaptive nonlinear prefilter for compensation of distortion in nonlinear systems,” *IEEE Tran. on Signal Processing*, vol. 46, no. 6, 1998.

[4] C. Eun abd E. J. Powers, “A new Volterra predistorter based on indirect learning architecture,” *IEEE Trans. on Signal Processing*, vol. 45, no. 1, 1997.

[5] L. Ding, R. Raich, and G. T. Zhou, “A Hammerstein predistortion linearization design based on the indirect learning architecture,” in *Proc. of The IEEE International Conference on Acoustics, Speech, and Signal Processing*, Orlando, Florida, 2002.

[6] T. Söderström and P. Stoica, *System Identification*, Prentice-Hall International, Hemel Hempstead, United Kingdom, 1989.

[7] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*, M.I.T. Press, Cambridge, MA, USA, 1983.

[8] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*, R. E. Krieger, Florida, USA, 1989.

[9] D. R. Morgan, Z. Ma, and L. Ding, “Reducing measurement noise effects in digital predistortion of RF power amplifiers,” in *Proc. of The IEEE ICC*, Alaska, USA, 2003, pp. 2436–2439.

[10] Y. Qian and T. Yao, “Structure for adaptive predistortion suitable for efficient adaptive algorithm application,” *Electronic Letters*, vol. 38, pp. 1282–1283, 2002.