Identification of Third-order Volterra-PARAFAC models based on PARAFAC decomposition using a tensor approach

ZOUHOUR BEN AHMED
Laboratoire CEM
Sfax Engineering School
BP 1173, 3038 Sfax
TUNISIA
zouhour.ben.ahmed@gmail.com

NABIL DERBEL
Laboratoire CEM
Sfax Engineering School
BP 1173, 3038 Sfax
TUNISIA
n.derbel@enis.rnu.tn

Abstract: Volterra models are very useful for representing nonlinear systems with vanishing memory. The main drawback of these models is their huge number of parameters to be estimated. In this paper, we present a new class of Volterra models, called Volterra-Parafac models, with a reduced parametric complexity, by considering Volterra kernels of order ($p > 2$) as symmetric tensors and by using a parallel factor (PARAFAC) decomposition. This paper is concerned with the problem of identification of third-order Volterra-PARAFAC models. Two types of algorithms are proposed for estimating the parameters of these models when input-output signals and kernel coefficients are real valued. The first is called Levenberg-Marquardt algorithm and the second is the Partial Update LMS algorithms. Some simulation results illustrate the proposed identification methods.

Key–Words: Volterra models, identification, tensors, PARAFAC, Levenberg-Marquardt, Partial Update LMS

1 Introduction

Volterra models represent a nonlinear extension of the popular finite impulse response (FIR) linear model, with guaranteed stability in the boundedinput bounded-output sense. Moreover, they are commutly used because of their property of linearity with respect to parameters, the kernels coefficients. The main drawback of Volterra models consists in their parametric complexity that imply to estimate a huge number of parameters. This number increases with the system nonlinearity order and memory. This models are now widely used in various application areas like nonlinear acoustic echo cancellation [3], speech modeling [1], loudspeaker linearization [2], nonlinear communication channel identification and equalization [5, 4], and many others.

There are many approaches to reduce parametric complexity in Volterra models. One approach is to consider the block-structured nonlinear models constituted by a cascade of linear dynamic and nonlinear static subsystems. These models are characterized by a smaller number of parameters and they are nonlinear with respect to their parameters. Another approach, consists in expanding the Volterra kernels onto orthonormal basis functions (OBFs).

In the present paper, by considering Volterra kernels as symmetric tensors, a parallel factor (PARAFAC) decomposition is used to reduce parameter complexity and derive the Volterra-PARAFAC model. Then, adaptive algorithms such as Levenberg-Marquardt and Partial Update Least Mean Squares algorithms (PU-LMS) are proposed for estimating the parameters of such a model and then compared with existing algorithms.

The rest of this paper is organized as follows. Section 2 describes the Volterra model and then we apply the PARAFAC decomposition to symmetric Volterra kernels to construct the Volterra-PARAFAC model. In Section 3, we propose a Levenberg-Marquardt method for identifying Volterra-PARAFAC models. In Section 4, we propose two types of PU-LMS algorithm for identifying Volterra-PARAFAC models. The proposed identification methods are illustrated by means of some simulation results in Section 5, before concluding the paper in Section 6.

2 Volterra-PARAFAC models

A p^{th}-order Volterra model for a causal, stable, finite memory, time-invariant, SISO system is described by the following input-output relationship:

$$y(k) = h_0 + \sum_{p=1}^{P} y_p(k) \quad (1)$$

with:

$$y_p(k) = \sum_{m_1,\cdots,m_p=1}^{M} h_p(m_1,\cdots,m_p)\prod_{i=1}^{p} u(k-m_i)$$

where $u(k)$ and $y(k)$ denote respectively the input and output signals, P is the nonlinearity degree of the Volterra model, M is the memory of the p^{th}-order homogeneous term $y_p(k)$ and $h_p(m_1,\cdots,m_p)$ is a coefficient of the p^{th}-order kernel. This coefficient can be viewed as an element of a tensor $\mathbb{H}_p \in \mathcal{K}^{M\times M\times\cdots\times M}$, with $\mathcal{K}=\mathcal{R}$ or $\mathcal{C}$ depending on whether the kernel coefficients are real-valued or complex-valued.

Any symmetric tensor $\mathbb{H} \in \mathcal{K}^{M\times M\times\cdots\times M}$ with rang R can be decomposed using the symmetric PARAFAC decomposition [6]:

$$h(m_1,\cdots,m_p)=\sum_{r=1}^{R}\prod_{p=1}^{P} a_{m_p,r}^{(p)},\ m_p=1,\cdots,M$$

where $a_{m_p,r}^{(p)}$ is an element of the matrix factor $\mathbf{A}^{(p)} \in \mathcal{K}^{M\times R}$.

Replacing the Volterra kernel of the p^{th}-order homogeneous term in (1) by its symmetric Parafac decomposition, we obtain the following input-output relationship :

$$y(k)=h_0+\sum_{p=1}^{P}\sum_{r=1}^{r_p}\prod_{i=1}^{p}\left(\sum_{m_i=0}^{M} a_{m_i,r}^{(p)}u(k-m_i)\right) \quad (2)$$

Let define the linear input regression vector $\mathbf{u}^T(k)=[u(k),\cdots,u(k-M+1)]$, expression (2) becomes:

$$y(k)=h_0+\sum_{p=1}^{P}\sum_{r=1}^{r_p}\left(\mathbf{u}^T(k)\mathbf{A}_{.r}^{(p)}\right)^p \quad (3)$$

Let us consider a third-order Volterra-PARAFAC model with memory M, rang R and real-valued parameters. The input-output relationship is given by:

$$y(k)=h_0+\sum_{p=1}^{3}\sum_{r=1}^{R}\left(\mathbf{u}^T(k)\mathbf{A}_{.r}^{(p)}\right)^p \quad (4)$$

Defining following vectors:

$$\boldsymbol{\theta}^T=[\boldsymbol{\theta}^{(0)}\ \boldsymbol{\theta}^{(1)^T}\ \boldsymbol{\theta}^{(2)^T}\ \boldsymbol{\theta}^{(3)^T}] \quad (5)$$

with:

$\boldsymbol{\theta}^{(0)}=h_0$

$\boldsymbol{\theta}^{(1)}=\mathbf{A}_{.1}^{(1)}=[h_1(1)\ h_1(2)\ \cdots\ h_1(M)]^T \in \mathcal{R}^{M\times 1}$

$\boldsymbol{\theta}^{(p)}=vec(\mathbf{A}^{(p)}) \in \mathcal{R}^{MR\times 1},\ p=2,3$

The input-output relation (4) can be rewritten as:

$$y(k)=f(k,\boldsymbol{\theta}) \quad (6)$$

The measured output signal of nonlinear system to be identified can be deduced as:

$$s(k)=f(k,\boldsymbol{\theta})+\nu(k) \quad (7)$$

where $\nu(k)$ is a white noise sequence with covariance σ^2.

3 Levenberg-Marquardt estimation algorithm for Volterra-PARAFAC models

3.1 Levenberg-Marquardt algorithm

The Levenberg-Marquardt (LM) method is a standard technique used to solve nonlinear least squares problems. The nonlinear least squares methods involve an iterative improvement of parameters values in order to minimize the sum-squared errors between function and measured data. With the LM method, it is not necessary to calculate the second derivatives to find estimated parameters. However, at each iteration, the algorithm solves a set of linear equations to calculate the gradient, which is easier and faster than other optimization techniques, in terms of the calculations. For a given iteration, all the unknowns are estimated jointly, contrary to the ALS (Alternating Least Square) which updates the components alternately [10]. This algorithm has been proposed particularly in [11] for fitting a PARAFAC model.

Given a set of data points ($\mathbf{x}_i$, y_i) related by the function f such that $y_i=f(\mathbf{x}_i,\mathbf{a})$, the LM method find the parameters $\mathbf{a}$ that minimize the sum-squared error:

$$\begin{aligned} S(\mathbf{a}) &= \frac{1}{2}\sum_{i=1}^{m} e_i(\mathbf{a})^2 \\ &= \frac{1}{2}\sum_{i=1}^{m}(y_i-f(\mathbf{x}_i,\mathbf{a}))^2 \end{aligned} \quad (8)$$

Given an estimation $\hat{\mathbf{a}}^{(k)}$ of $\mathbf{a}$ at iteration k, $\hat{\mathbf{a}}^{(k+1)}$ is obtained by $\hat{\mathbf{a}}^{(k+1)}=\hat{\mathbf{a}}^{(k)}+\Delta\mathbf{a}$, where $\Delta\mathbf{a}$ mast be calculated such to ensure a direction of steepest descent for S.

Let's define $\mathbf{J}$ the jacobian matrix of the function $f(\mathbf{x}_i,\mathbf{a})$ with respect to the parameters vector $\mathbf{a}$. The update equation of $\Delta\mathbf{a}$ using Gauss-Newton (GN) method is obtained by solving the following normal equations:

$$\mathbf{J}^T\mathbf{J}\Delta\mathbf{a}=\mathbf{J}^T\mathbf{e}(\mathbf{a}) \quad (9)$$

A restriction on this method is the requirement that $\mathbf{J}$ must have full rank. GN is notable for its fast convergence close to the solution, but its efficiency depends on having an accurate initial guess. To ensure the global convergence of GN method and guarantee that $\mathbf{J}^T\mathbf{J}$ remains positive definite, Levenberg proposed to replace the $\mathbf{J}^T\mathbf{J}$ matrix by $\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}$ where $\lambda > 0$ is the damping factor. LM is also especially useful when $\mathbf{J}$ is rank deficient, in which case GN would not be effective. The method is then described by:

$$\left(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}\right)\Delta\mathbf{a}=\mathbf{J}^T\mathbf{e}(\mathbf{a}) \tag{10}$$

where small values of λ result in a Gauss-Newton update and large values of λ result in a gradient descent update. The parameter λ is initialized to be large. If an iteration happens to result in a worse approximation, λ is increased. As the solution approaches the minimum, λ is decreased, the Levenberg-Marquardt method approaches the Gauss-Newton method, and the solution typically converges rapidly to the local minimum [12].

LM'disadvantage algorithm is that if the value of damping factor λ is large, inverting $\mathbf{J}^T\mathbf{J}$ is not used at all. Marquardt's contribution is to replaced the identity matrix, $\mathbf{I}$, with the diagonal matrix consisting of the diagonal elements of $\mathbf{J}^T\mathbf{J}$, resulting in the Levenberg-Marquardt algorithm:

$$\left(\mathbf{J}^T\mathbf{J} + \lambda\text{diag}(\mathbf{J}^T\mathbf{J})\right)\Delta\mathbf{a}=\mathbf{J}^T\mathbf{e}(\mathbf{a}) \tag{11}$$

The estimated parameter update equation of the LM algorithm is:

$$\hat{\mathbf{a}}^{(k+1)} = \hat{\mathbf{a}}^{(k)} + \left(\mathbf{J}^T\mathbf{J} + \lambda\text{diag}(\mathbf{J}^T\mathbf{J})\right)^{-1}\mathbf{J}^T\mathbf{e}(\hat{\mathbf{a}}^{(k)})$$

where $\mathbf{J}$ is the Jacobian of the nonlinear function $f(\mathbf{x}_i, \mathbf{a})$ with respect to the parameter $\mathbf{a}$ calculated at point $\mathbf{a} = \hat{\mathbf{a}}^{(k)}$ and $\mathbf{e}(\hat{\mathbf{a}}^{(k)}) = \mathbf{y} - f(\mathbf{x}, \hat{\mathbf{a}}^{(k)})$.

3.2 Levenberg-Marquardt algorithm for identifying Volterra-PARAFAC models

Whereas the Volterra-PARAFAC model is nonlinear in its parameters, the iterative algorithms can be used to estimate PARAFAC coefficients. Let consider the LM algorithm that minimize the following function:

$$E(k) = \frac{1}{2}e(k)^2 = \frac{1}{2}\left(s(k) - f(k, \boldsymbol{\theta})\right)^2 \tag{12}$$

where $\boldsymbol{\theta}$, $f(k, \boldsymbol{\theta})$ and $s(k)$ are defined in (5), (6) et (7) respectively.

Table 1: LM algorithm for identifying the Volterra-PARAFAC model

1. $k = 0$, Randomly initialize $\hat{\boldsymbol{\theta}}(0)$,
2. $k = k + 1$,
3. The estimated parameter update equations of the LM algorithm are :

$$e(k)=s(k) - f(k, \hat{\boldsymbol{\theta}}(k-1))$$
$$\hat{\boldsymbol{\theta}}^{(p)}(k)=\hat{\boldsymbol{\theta}}^{(p)}(k-1) + \left[\hat{\mathbf{J}}_k^{(p)^T}\hat{\mathbf{J}}_k^{(p)} + \lambda_p\text{diag}\left(\hat{\mathbf{J}}_k^{(p)^T}\hat{\mathbf{J}}_k^{(p)}\right)\right]^{-1} \times\hat{\mathbf{J}}_k^{(p)^T}e(k).$$

4. Return to Step (2) until $k = K$, where K is the number of input-output data to be processed.

The estimated parameter update equations of the LM algorithm, where the input-output signals and kernel coefficients are real valued, is given by:

$$\hat{\boldsymbol{\theta}}^{(p)}(k)=\hat{\boldsymbol{\theta}}^{(p)}(k-1) + \left[\hat{\mathbf{J}}_k^{(p)^T}\hat{\mathbf{J}}_k^{(p)} + \lambda_p\text{diag}(\hat{\mathbf{J}}_k^{(p)^T}\hat{\mathbf{J}}_k^{(p)})\right]^{-1} \times\hat{\mathbf{J}}_k^{(p)^T}e(k) \tag{13}$$

where λ_p is non-negative damping factor, $\hat{\mathbf{J}}_k^{(p)}$ define the Jacobian matrix of nonlinear function $f(k, \boldsymbol{\theta})$ with respect to the parameters vector $\boldsymbol{\theta}$ calculated at the point $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}^{(p)}(k-1)$.

By deriving $f(k, \boldsymbol{\theta})$, we obtain:

$$\frac{\partial f(k, \boldsymbol{\theta})}{\partial a_{mr}} = \begin{cases} 1, & \text{if } p = 0 \\ u(k-m), & \text{if } p = 1 \\ p\,u(k-m)\left(\mathbf{u}^T(k)\mathbf{A}_{.r}^{(p)}\right)^{p-1}, & \text{if } p \geq 2 \end{cases} \tag{14}$$

Which give :

$$\mathbf{J}_k^{(p)} = \begin{cases} 1, & \text{if } p = 0 \\ \mathbf{u}^T(k), & \text{if } p = 1 \\ p\left(\mathbf{u}^T(k)\mathbf{A}^{(p)}\right)^{\bullet(p-1)} \otimes \mathbf{u}_p^T(k), & \text{if } p \geq 2 \end{cases} \tag{15}$$

where $\bullet$ and $\otimes$ denote, respectively, Hadamard and Kronecker products. The estimated parameter update equations of the LM algorithm for Volterra-PARAFAC models identification are then summarized in table 1.

4 Partial update LMS estimation algorithms for Volterra-PARAFAC models

4.1 Partial Update LMS (PU-LMS) algorithms

The least mean-squares (LMS) algorithm is a popular algorithm for updating of weights in adaptive filter. Although there exist algorithms with faster convergence rates like RLS, LMS is popular because of its ease of implementation and low computational resources and memory [15].

They have been employed in various areas, including interference cancellation, echo cancellation, space time modulation and coding, signal copy in surveillance and wireless communications [16].

Two types of partial update LMS algorithms are in use in the literature and have been described in [14]. "Periodic LMS algorithm" and "Sequential LMS algorithm". The Periodic PU-LMS algorithm updates all the filter coefficients every P^{th} iteration instead of every iteration. The Sequential PU-LMS algorithm updates only a fraction of coefficients every iteration.

4.1.1 Sequential Partial Update LMS algorithms (SPU-LMS)

Let consider $\mathbf{x}(k)$ the linear input regression vector and $\mathbf{w}(k)$ the vector containing coefficients of adaptive filter of odd length L, for the instant k. Define:

$$\mathbf{w}(k)=[w_1\ w_2\ \cdots\ w_L]^T \tag{16}$$

$$\mathbf{x}(k)=[x_1\ x_2\ \cdots\ x_L]^T \tag{17}$$

Assume that the filter length L is a multiple of Q. Let define the index set $S = 1, 2, \cdots, L$. Let $S_1, S_2, \cdots, S_Q$ a Q mutually exclusive subsets of equal size z ($z = \frac{L}{Q}$) of S. The SPU-LMS algorithm is described as follows. At a given iteration, k, one of the sets S_q, $q = 1, \cdots, Q$, is chosen and there is z coefficients to be updated. Without loss of generality, it can be assumed that at iteration k, the set S_q is chosen for update, such as $q = (k \bmod Q) + 1$. Therefore, the equation of SPU-LMS algorithm is described by [14, 15]:

$$w_j(k+1) = \begin{cases} w_j(k) + \mu e(k) x_j(k) & \text{if } j \in S_q \\ w_j(k) & \text{else} \end{cases} \tag{18}$$

where $e(k) = d(k) - \mathbf{w}(k)^T\mathbf{x}(k)$ is the error signal at iteration k and mod denotes the modulo operator.

Define $\mathcal{I}_q$ is the identity matrix of dimension L such as the j^{th} row is equal to zero if $j \notin S_q$. In that case, $\mathcal{I}_q\mathbf{x}_k$ will have $\frac{L}{Q}$ non-zero entries of $\mathbf{x}_k$. Moreover, if the subset S_q is chosen at iteration k, that mean the weights with their indices in S_q will be chosen for update at iteration k. Then, the update equation of SPU-LMS algorithm can be written as :

$$\mathbf{w}(k+1)=\mathbf{w}(k) + \mu e(k)\mathcal{I}_q\mathbf{x}(k). \tag{19}$$

$\mathcal{I}_q$ is the coefficient selection matrix and is given by:

$$\mathcal{I}_q = \begin{pmatrix} i_1\ 0\ \ldots 0 \\ 0\ i_2\ \ddots\ 0 \\ \vdots\ \ddots\ \ddots\ \vdots \\ 0 \ldots\ 0\ i_L \end{pmatrix} \tag{20}$$

where $i_j = \begin{cases} 1 \text{ if } j \in S_q \text{ such as } q = (k \bmod Q) + 1 \\ 0 \text{ else} \end{cases}$

4.1.2 Periodic Partial LMS algorithm (PPU-LMS)

The update equation of PPU-LMS algorithm is given by [14, 15]:

$$w_j(k+1) = \begin{cases} w_j(k) + \mu e(l) x_j(l) & \text{if } (k+j) \bmod P = 0 \\ & \text{and } l = P\lfloor k/P \rfloor \\ w_j(k) & \text{else} \end{cases} \tag{21}$$

where $\lfloor . \rfloor$ denotes the truncation operation. The PPU-LMS algorithm consist in updating all the coefficients at periodic intervals (every P^{th} iteration). The update equation (21) is mathematically equivalent to the following coefficient vector update:

$$\mathbf{w}(k+P)=\mathbf{w}(k) + \mu e(k)\mathbf{x}(k). \tag{22}$$

Table 2 resume the different update equations, at iteration k, of PU-LMS algorithms for a FIR filter of length L with impulse response of filter coefficients $\mathbf{w}(k) = [w_1\ w_2\ \cdots\ w_L]^T$. $\mathbf{x}(k) = [x_1\ x_2\ \cdots\ x_L]^T$ denotes the linear regression vector, $e(k) = d(k) - \mathbf{w}(k)^T\mathbf{x}(k)$ is the error signal and $d(k)$ represents the desired response.

Table 3 shows the computational complexity of the partial update FIR filters. These table present number of multiplications, divisions, additions, and comparisons in each iteration for filter vector update equation. As we can see, for partial update FIR filters, the computational complexity is lower than ordinary FIR filters.

4.2 SPU-LMS and PPU-LMS algorithms for identifying Volterra-PARAFAC model

Whereas the Volterra-PARAFAC model is nonlinear in its parameters, the adaptive algorithms can be used to estimate PARAFAC coefficients [8].

Table 2: Update equations of PU-LMS algorithms.

Algorithms	Update equations
LMS	$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k)$
SPU-LMS	$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathcal{I}_q\mathbf{x}(k)$, where $\mathcal{I}_q$ is a matrix defined in (20)
PPU-LMS	$\mathbf{w}(k+P) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k)$, where P is the period

Table 3: Computational complexity of ordinary and partial update LMS algorithms.

Algorithms	$\times$	$+$
LMS	$2L+1$	$2L$
SPU-LMS	$L+Q+1$	$L+Q$
PPU-LMS	$L+(L+1)/P$	$L+L/P$

The LMS algorithm (also called stochastic gradient algorithm) has been proposed in [9, 7] for estimating the parameters of the p^{th}-order Volterra-PARAFAC model, then its normalized version called NLMS algorithm.

The gradient of the nonlinear function $f(k,\boldsymbol{\theta})$, presented in (6), can be calculated by means of the following formulae [6, 7]:

$$\boldsymbol{\varphi}^T(k) = [1\ \boldsymbol{\varphi}_1^T\ \boldsymbol{\varphi}_2^T\ \boldsymbol{\varphi}_3^T] \tag{23}$$

with:

$$\boldsymbol{\varphi}_1 = \mathbf{u}(k)$$

and:

$$\boldsymbol{\varphi}_p^T(k) = p\left[\alpha_{p,1}^{p-1}(k)\cdots\alpha_{p,R}^{p-1}(k)\right]\otimes\mathbf{u}^T(k)$$

where:

$$\alpha_{p,r}(k) = \left(\mathbf{u}^T(k)\hat{\mathbf{A}}_{:r}^{(p)}(k-1)\right),\ r=1,\cdots,R,\ p=2,3,$$

The estimated parameter update equations of the LMS algorithm, where the input-output signals and kernel coefficients are real valued, is given by:

$$\hat{\boldsymbol{\theta}}^{(p)}(k)=\hat{\boldsymbol{\theta}}^{(p)}(k-1)+\mu_p\hat{\boldsymbol{\varphi}}_p(k)e(k) \tag{24}$$

where μ_p is the step size that controls the convergence speed and the steady-state properties of the algorithm. Then, by using equation (19), we obtain the update equation of SPU-LMS for Volterra-PARAFAC model [13]:

$$\hat{\boldsymbol{\theta}}^{(p)}(k)=\hat{\boldsymbol{\theta}}^{(p)}(k-1)+\mu_p\mathcal{I}_q^{(p)}\hat{\boldsymbol{\varphi}}_p(k)e(k) \tag{25}$$

Table 4: SPU-LMS algorithm for identifying the Volterra-PARAFAC model

1. $k=0$, Randomly initialize $\hat{\boldsymbol{\theta}}(0)$,
2. $k=k+1$,
3. The estimated parameter update equations of the SPU-LMS algorithm are :
$$e(k)=s(k)-f(k,\hat{\boldsymbol{\theta}}(k-1))$$
$$\hat{\boldsymbol{\theta}}^{(p)}(k)=\hat{\boldsymbol{\theta}}^{(p)}(k-1)+\mu_p\mathcal{I}_q^{(p)}\hat{\boldsymbol{\varphi}}_p(k)e(k)$$
4. Return to Step (2) until $k=K$, where K is the number of input-output data to be processed.

with $\mathcal{I}_q^{(p)}$ is the coefficient selection matrix defined in (20) and the length of vector $\hat{\boldsymbol{\theta}}^{(p)}$ is a multiple of Q_p. For each iteration k, only the coefficients that belong to the chosen set are updated.

The estimated parameter update equations of the SPU-LMS algorithm for Volterra-PARAFAC model identification are then summarized in table 3.
For $p=0$, we have $Q_0=1$ and the update equation (25) becomes:

$$\hat{\boldsymbol{\theta}}^{(0)}(k)=\hat{\boldsymbol{\theta}}^{(0)}(k-1)+\mu_0 e(k)$$

For $p=1$, we have $M=\alpha Q_1$ such as α is positive constant, the update equation (25) becomes in this case:

$$\hat{\boldsymbol{\theta}}^{(1)}(k)=\hat{\boldsymbol{\theta}}^{(1)}(k-1)+\mu_1\mathcal{I}_q^{(1)}\mathbf{u}(k)e(k)$$

and

$$\mathcal{I}_q^{(1)} = \begin{pmatrix} i_1 & 0 & \dots & 0 \\ 0 & i_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & i_M \end{pmatrix} \in \mathcal{R}^{M\times M}$$

where $i_j = \begin{cases} 1 \text{ if } j\in S_q \text{ such as } q=(k\,mod\,Q_1)+1 \\ 0 \text{ else} \end{cases}$
For $p\geq 2$, we have $MR=\alpha Q_p$ such as α is positive constant, the update equation (25) becomes then:

$$\hat{\boldsymbol{\theta}}^{(p)}(k)=\hat{\boldsymbol{\theta}}^{(p)}(k-1)+\mu_p\mathcal{I}_q^{(p)}\hat{\boldsymbol{\varphi}}_p(k)e(k)$$

and

$$\mathcal{I}_q^{(p)} = \begin{pmatrix} i_1 & 0 & \dots & 0 \\ 0 & i_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & i_{MR} \end{pmatrix} \in \mathcal{R}^{MR\times MR}$$

Table 5: PPU-LMS algorithm for identifying the Volterra-PARAFAC model

1. $k = 0$, Randomly initialize $\hat{\boldsymbol{\theta}}(0)$,
2. $k = k + 1$,
3. The estimated parameter update equations of the PPU-LMS algorithm are :

$$e(k)=s(k) - f(k, \hat{\boldsymbol{\theta}}(k-1))$$
$$\hat{\boldsymbol{\theta}}^{(p)}(k)=\hat{\boldsymbol{\theta}}^{(p)}(k-P) + \mu_p \hat{\boldsymbol{\varphi}}_p(k)e(k)$$

4. Return to Step (2) until $k = K$, where K is the number of input-output data to be processed.

where $i_j = \begin{cases} 1 \text{ if } j \in S_q \text{ such as } q = (k \, mod \, Q_p) + 1 \\ 0 \text{ else} \end{cases}$

By using equation (22), we obtain the update equation of PPU-LMS for Volterra-PARAFAC model [13]:

$$\hat{\boldsymbol{\theta}}^{(p)}(k)=\hat{\boldsymbol{\theta}}^{(p)}(k-P) + \mu_p \hat{\boldsymbol{\varphi}}_p(k)e(k) \quad (26)$$

PPU-LMS algorithm updates all coefficients of vector $\hat{\boldsymbol{\theta}}^{(p)}$ every P iteration instead of every iteration.

The estimated parameter update equations of the PPU-LMS algorithm for Volterra-PARAFAC model identification are then summarized in table 4.

5 Simulation Results

In this section, we present some Monte Carlo simulation results for illustrating performance of the proposed methods for parametric estimation of a third-order Volterra-Parafac model. $N_m = 10$ Volterra models were simulated by drawing the coefficients of the linear kernel, the quadratic and of the cubic kernel Parafac factors from a Gaussian distribution, with $M = 10$ and $R = 3$. Moreover, $N_b = 10$ additive white Gaussian noises ν (defined in (7)) with zero-mean were added to each model output with fixed SNR (Signal-to-Noise Ratio). The noise sequence ν can be written as $\nu = \sigma \mathbf{e}$, where $\mathbf{e} = [e_1 \cdots e_N]^T$ is a white gaussian noise which the mean is equal to zero and the variance is equal to 1. The value of σ is chosen such that the SNR is equal to the one desired. The SNR equation is then given by :

$$\text{SNR}_{fixed}=20 \log \left(\frac{\|\mathbf{y}\|_2}{\|\nu\|_2} \right) = 20 \log \left(\frac{\|\mathbf{y}\|_2}{\sigma \|\mathbf{e}\|_2} \right) \quad (27)$$

Determination of the σ value: For the white gaussian noise $\mathbf{e}$, we consider the following expression of SNR :

$$\text{SNR}=20 \log \left(\frac{\|\mathbf{y}\|_2}{\|\mathbf{e}\|_2} \right) \quad (28)$$

From (27) and (28), we get:

$$\text{SNR}_{fixed}=\text{SNR} - 20 \log(\sigma)$$
$$20 \log(\sigma)=\text{SNR} - \text{SNR}_{fixed}$$
$$\sigma=10^{\frac{\text{SNR} - \text{SNR}_{fixed}}{20}}$$

Performances are evaluated by means of Normalized Mean Square Error (NMSE) on the output signal at time k :

$$\text{NMSE}_{\mathbf{s},k}=10 \log \left(\frac{1}{N_m N_b} \sum_{m=1}^{N_m} \sum_{b=1}^{N_b} \frac{\|\hat{\mathbf{s}}_{k,m,b} - \mathbf{s}_{k,m}\|_F^2}{\|\mathbf{s}_{k,m}\|_F^2} \right)$$

where $\mathbf{s}_{k,m} = [s_{k-\tau+1,m} \cdots s_{k,m}]^T$ denotes the output vector associated with the $m^{ème}$ simulated model and $\hat{\mathbf{s}}_{k,m,b} = [\hat{s}_{k-\tau+1,m,b} \cdots \hat{s}_{k,m,b}]^T$ denotes the corresponding vector of reconstructed outputs calculated in sliding window of length $\tau = 2000$, by using the estimated Volterra-PARAFAC model for the b^{th} converged experiment.

The input sequence has been a 6-RMS (six-Random Multilevel Sequence), whose values were uniformly drawn from the alphabet $\{\pm 1, \pm 2/3, \pm 1/3\}$.

The damping factor values of the LM algorithm and the adaptation step sizes of the PU-LMS algorithms are given in table 6. We have to notice that the performances of the LM and the PU-LMS algorithms are strongly dependent on the choice of this factors.

Table 6: Damping factor values

p	1	2	3
λ_p	$2 \cdot 10^{-3}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-6}$
μ_p	$1.2 \cdot 10^{-5}$	$4.5 \cdot 10^{-6}$	$1.4 \cdot 10^{-7}$

In subsection 5.1, given noisy input-output measurements, we evaluate the parameter estimation algorithm derived in Section 3. Then, we compare it with the EKF (Extended Kalman Filter;) algorithm developed in [6].

In subsection 5.2, given noisy input-output measurements, we evaluate the parameter estimation algorithms derived in Section 4. Then, we compare it with the standard LMS algorithm developed in [7].

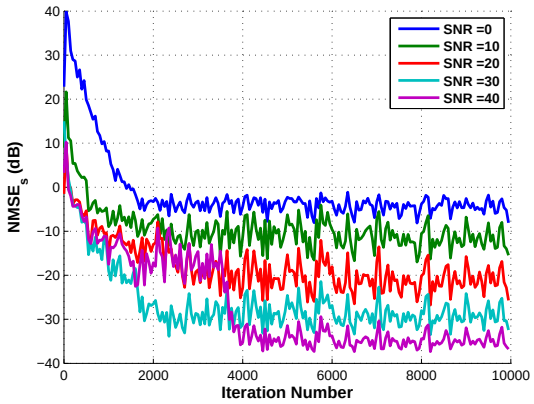

Figure 1: $\mathrm{NMSE_s}$ vs iterations

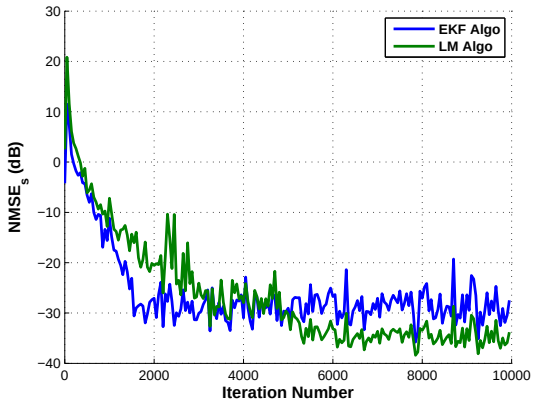

Figure 3: $\mathrm{NMSE_s}$ vs iterations for LM and EKF algorithms

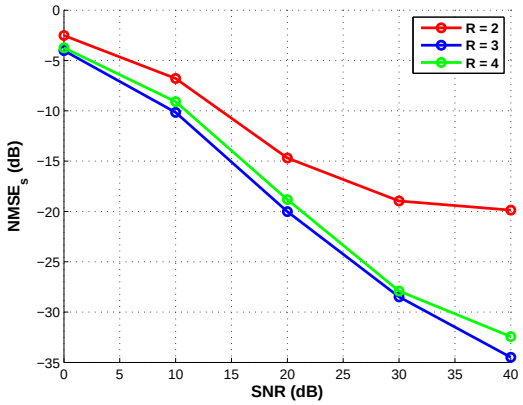

Figure 2: $\mathrm{NMSE_s}$ vs SNR for different values of $R_2 = R_3 = R$

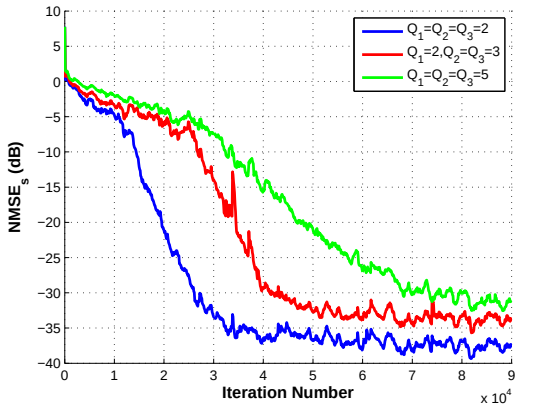

Figure 4: $\mathrm{NMSE_s}$ vs iterations for different Q_p value of SPU-LMS algorithm

5.1 Parameter estimation using the LM algorithm

Figure 1 shows the $\mathrm{NMSE_s}$ versus the iterations for different values of SNR. Figure 2 presents the $\mathrm{NMSE_s}$ versus SNR for different values of approximation ranks $R_2 = R_3 = R = 2, 3, 4$.

From these simulation results, we can conclude that, as expected, the $\mathrm{NMSE_s}$ decreases when the SNR increases. Moreover, it can be observed that the $\mathrm{NMSE_s}$ increase when the rank is underestimated, and that is the more so as the SNR increases. However, when the rank is overestimated, $\mathrm{NMSE_s}$ does not change.

Figure 3 shows the $\mathrm{NMSE_s}$ versus the iterations, with SNR=40 dB, for the proposed LM algorithm and the EKF algorithm. Although the LM method gives more precise estimation in terms of $\mathrm{NMSE_s}$, the EKF result a convergence rate 3 times faster than the LM.

5.2 Parameter estimation using the PU-LMS algorithms

Figures 4, 5 show the $\mathrm{NMSE_s}$ versus the iterations for the adaptive SPU-LMS and PPU-LMS algorithms, for different value of Q_p, $p = 1, 2, 3$ for SPU-LMS algorithm and different value of period for PPU-LMS algorithm, in the case of SNR=40 dB.
Figures 6, 7 show the $\mathrm{NMSE_s}$ versus the iterations, with SNR=40 dB and SNR respectively, for the three adaptive algorithms (ordinary LMS, SPU-LMS and PPU-LMS).

From these simulation results, we can conclude that:

- As expected, the $\mathrm{NMSE_s}$ decreases when the SNR increases, whatever the method we consider.
- SPU-LMS gives more precise estimation than

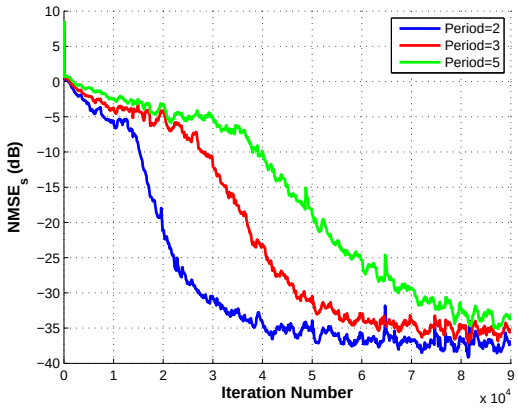

Figure 5: $\mathrm{NMSE_s}$ vs iterations for different P value of PPU-LMS algorithm

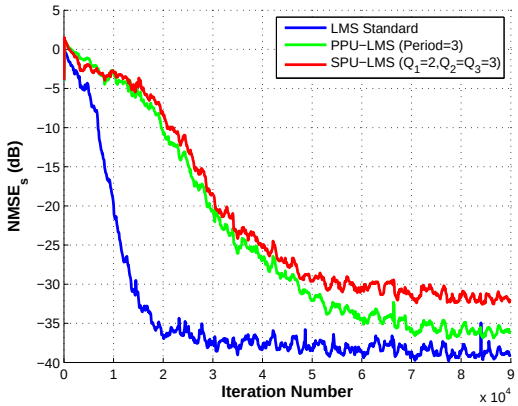

Figure 6: $\mathrm{NMSE_s}$ vs iterations for the three adaptive algorithms

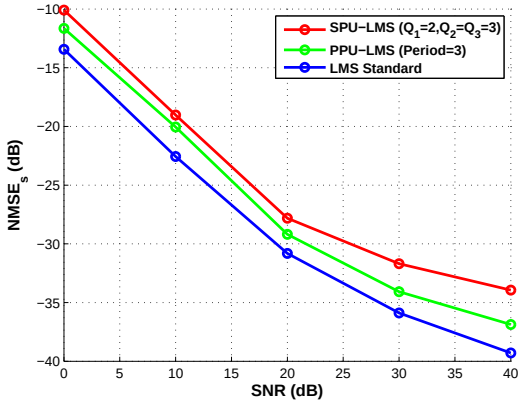

Figure 7: $\mathrm{NMSE_s}$ vs SNR for the three adaptive algorithms

PPU-LMS in terms of $\mathrm{NMSE_s}$, with difference 1-2 dB. Ordinary LMS algorithm gives the best estimations, with difference 2-3 dB in terms of $\mathrm{NMSE_s}$ compared to SPU-LMS and PPU-LMS, whatever the value of SNR.

- The more we increase Q_p, i.e. to reduce the number of updated coefficients of estimated vector $\boldsymbol{\theta}^{(p)}$, the convergence rate of SPU-LMS algorithm becomes slower.
- The more we increase the period P of PPU-LMS algorithm, i.e. to update all the coefficients of estimated vector $\boldsymbol{\theta}^{(p)}$ after several iterations, the convergence rate becomes slower.
- Ordinary LMS algorithm converges 2 times faster than PPU-LMS and SPU-LMS. In conclusion, that itself converges much more rapidly than CLMS.

6 Conclusion

In this paper, we have presented Levenberg-Marquardt algorithm and Partial Update LMS algorithms for identifying a nonlinear third-order Volterra-PARAFAC models based on PARAFAC decomposition of its kernels considered as symmetric tensors. The performance of these algorithms have been presented by means of computer simulations. The proposed algorithms are able to provide a good identification of Volterra-PARAFAC coefficients. Then, we presented a performance comparison between Levenberg-Marquardt and Extended Kalman Filter methods, in one hand, and between Partial Update LMS and ordinary LMS, in other hand. In a future work, robustness to noise, extension to higher-order tensors, and a comparison of Volterra-Parafac models with pruning Volterra models recently proposed will be considered.

References:

[1] E. Mumolo and D. Francescato, *Adaptive predictive coding of speech by means of Volterra predictors*, In Proc. of IEEE Winter Workshop on Nonlinear digital signal processing, Tampere, Finland 1993

[2] Y. Kajikawa, *Subband parallel cascade Volterra filter for linearization of loudspeaker systems*, In EUSIPCO, Lausanne, Switzerland 2008

[3] L. Tan and J. Jiang, Adaptive Volterra filters for active control of nonlinear noise processes, *IEEE Tr. on Signal Processing* 49, 2001, pp. 1667–1676.

[4] A.Y. Kibangou and G. Favier, Blind equalization of nonlinear channels using tensor decompositions with code/space/time diversities, *Signal Processing* 89, 2009, pp. 133–143.

[5] C.A.R.. Fernandes and G. Favier and J.C.M. Mota , Blind identification of multiuser nonlinear channels using tensor decomposition and precoding, *Signal Processing* 89, 2009, pp. 2644–2656.

[6] G. Favier and T. Bouilloc, *Identification de modle de Volterra base sur la decomposition PARAFAC*, GRETSI, Dijon, France 2009

[7] G. Favier and A.Y. Kibangou and T. Bouilloc , Nonlinear system modeling and identification using Volterra-PARAFAC models, *Int Journal on Adaptive Control and Signal Processing*, 2011.

[8] A.Y. Kibangou, Modles de Volterra complexit rduite : Estimation paramtrique et application l'galisation des canaux de communication, *Universit Nice-Sophia Antipolis-UFR Sciences*, 2005.

[9] T. Bouilloc, Applications de dcompositions tensorielles l'identification de modles de Volterra et aux systmes de communication MIMO-CDMA, *Universit Nice-Sophia Antipolis-UFR Sciences*, 2011.

[10] D. Nion and L. De Lathauwer, *Sparation et galisation aveugles de signaux CDMA par la dcomposition en blocs d'un tenseur au moyen de l'algorithme de Levenberg-Marquardt*, 21th Colloque GRETSI, Troyes, France 2007

[11] G. Tomasi and R. Bro, A Comparison of Algorithms for Fitting the PARAFAC Model, *Computational Statistics and Data Analysis* 50, 2006, pp. 1700–1734.

[12] H. Gavin, *The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems*, Dept. Civil and Environmental Engineering, Duke Univ 2011

[13] Z. Ben Ahmed and G. Favier and N. Derbel, *Identification of Volterra-PARAFAC models using Partial Update LMS algorithms*, 7th International Conference on Modelling, Identification and Control, Sousse, Tunisia 2015

[14] S.C. Douglas, Adaptive filters employing partial updates, *IEEE Tr. on Circuits and Systems-II: Analog and digital signal processing* 44, 1997, pp. 209–216.

[15] M. Godavarti and A.O. Hero, Partial Update LMS Algorithms, *IEEE Tr. on Signal Processing* 53, 2005, pp. 2382–2399.

[16] K. Dogancay and P.A. Naylor, *Recent advances in partial update and sparse adaptive filters*, in Proc. European Signal Processing Conference 2005