Enhanced Line Search : A Novel Method to Accelerate PARAFAC

Myriam Rajih, Pierre Comon, Richard Harshman

SIAM J. MATRIX ANAL. APPL.
Vol. 0, No. 0, pp. 000–000

ENHANCED LINE SEARCH: A NOVEL METHOD TO ACCELERATE PARAFAC*

MYRIAM RAJIH†, PIERRE COMON†, AND RICHARD A. HARSHMAN‡

Abstract. Several modifications have been proposed to speed up the alternating least squares (ALS) method of fitting the PARAFAC model. The most widely used is line search, which extrapolates from linear trends in the parameter changes over prior iterations to estimate the parameter values that would be obtained after many additional ALS iterations. We propose some extensions of this approach that incorporate a more sophisticated extrapolation, using information on nonlinear trends in the parameters and changing all the parameter sets simultaneously. The new method, called "enhanced line search (ELS)," can be implemented at different levels of complexity, depending on how many different extrapolation parameters (for different modes) are jointly optimized during each iteration. We report some tests of the simplest parameter version, using simulated data. The performance of this lowest-level of ELS depends on the nature of the convergence difficulty. It significantly outperforms standard LS when there is a "convergence bottleneck," a situation where some modes have almost collinear factors but others do not, but is somewhat less effective in classic "swamp" situations where factors are highly collinear in all modes. This is illustrated by examples. To demonstrate how ELS can be adapted to different N-way decompositions, we also apply it to a four-way array to perform a blind identification of an under-determined mixture (UDM). Since analysis of this dataset happens to involve a serious convergence "bottleneck" (collinear factors in two of the four modes), it provides another example of a situation in which ELS dramatically outperforms standard line search.

Key words. PARAFAC, alternating least squares (ALS), line search, enhanced line search (ELS), acceleration, swamps, bottlenecks, collinear factors, degeneracy

AMS subject classifications. 65B99, 15A69, 15A21

DOI. 10.1137/06065577

1. Introduction.

PARAFAC can be seen as a generalization of two-way factor analysis to multiway data. It was first introduced by Harshman in 1970 [9] based on the principle of parallel proportional profiles (PP) proposed by Cattell in 1944 [4]. The PP principle states that if two (or more) different two-way models are described by the same set of loading vectors but their relative proportions or weights change from one model to the other, then those loading vectors lead to a new model which is unambiguous with respect to (w.r.t.) rotation [4, 5, 2]. In other words, suppose that the matrix $\mathbf{X}_1$ can be modeled:

$$\mathbf{X}_1 = \mathbf{a}_1\mathbf{b}_1^T c_{11} + \mathbf{a}_2\mathbf{b}_2^T c_{12} + \cdots + \mathbf{a}_F\mathbf{b}_F^T c_{1F} + \mathbf{E}_1,$$

where $\mathbf{a}_f$ and $\mathbf{b}_f$ $(1 \le f \le F)$ are the columns of matrices $\mathbf{A}$ and $\mathbf{B}$, respectively, and $\mathbf{E}_1$ is a matrix of random disturbances (and/or other unmodeled variation). And suppose that another matrix $\mathbf{X}_2$ can be modeled using the same set of loading vectors only in different proportions (i.e., $\frac{c_{11}}{c_{21}} \neq \frac{c_{12}}{c_{22}} \neq \cdots \frac{c_{1F}}{c_{2F}}$):

$$\mathbf{X}_2 = \mathbf{a}_1\mathbf{b}_1^T c_{21} + \mathbf{a}_2\mathbf{b}_2^T c_{22} + \cdots + \mathbf{a}_F\mathbf{b}_F^T c_{2F} + \mathbf{E}_2.$$

*Received by the editors March 29, 2006; accepted for publication (in revised form) by L. De Lathauwer July 5, 2007; published electronically DATE. This work has been partially supported by the IST Programme of the European Community, under the PASCAL Network of Excellence IST-2002-506778, and by the contract ANR-06-BLAN-0074 "DECOTES."
http://www.siam.org/journals/simax/x-x/65537.html

†I3S Laboratory, UNSA-CNRS, 2000 route des Lucioles, B.P. 121, F-06903 Sophia-Antipolis, France (rajih@i3s.unice.fr, comon@i3s.unice.fr).

‡Department of Psychology, University of Western Ontario, London, Ontario, N6A 5C2 Canada (harshman@uwo.ca, http://publish.uwo.ca/∼harshman).

Then, we can build a combined model:

$$\mathbf{X}_k = \mathbf{A}\mathbf{C}_k\mathbf{B}^T + \mathbf{E}_k, k = 1, 2, \tag{1}$$

where $\mathbf{C}_k$ is a diagonal matrix with the elements of vector $\mathbf{c}_k$ in its diagonal, where $\mathbf{c}_k$ denotes the kth row of the slice weighting or "occasion weights" matrix $\mathbf{C}$ [13]. The trilinear decomposition used in the model is also known as CANDECOMP for CANonical DECOMPosition; it was introduced by Caroll and Chang in 1970 [3] to provide a basis for fitting INDSCAL, an important generalization of multidimensional scaling that provides unique dimensions and allows the estimation of dimension weights for individual subjects. Alternatively, the model can be written in scalar form as

$$X_{ijk} = \sum_f A_{if}B_{jf}C_{kf} + E_{ijk}.$$

Matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are called loading matrices.

The three-way PARAFAC model, along with its extension to higher orders [9, 3], has most often been applied in psychometrics and chemometrics [26, 27], and in the signal processing area [18, 6, 7]. While the two-way factor model suffers a rotational indeterminacy that yields an infinite set of solutions, the PARAFAC model enjoys a uniqueness property under conditions that often can be met in real data situations. Uniqueness properties have been studied by multiple authors, with some of the most important general results found in [10] and its recent generalization [15], the Kruskal theorems in, e.g., [16], and the extensions in [26], and elsewhere. Progress in this area is ongoing—for example, in the case of "tall" arrays, a significantly more relaxed condition has been derived in [19]. A relatively complete list of relevant articles up to 2006-07 can be found in [15].

A variety of algorithms have been used to fit the PARAFAC model (for a detailed summary and discussion; see, e.g., [28]. The most widely used is the alternating least squares (ALS) algorithm. The convergence of ALS was found to be very slow in some cases, typically when two factors are almost collinear. Line search [2, 24] is one of the most important solutions proposed to cope with the problem of slow convergence. We focus in this paper on the line search solution and present a generalization of this method for speeding up ALS; we discuss its simplest version and demonstrate that it can exhibit very good performance in some circumstances, yet perform less successfully in others—opening interesting directions for further exploration. We call this method enhanced line search (ELS).

A regularized (ridge) regression was proposed by Rayens and Mitchell in [23] to speed up the ALS algorithm in case of ill-posed problems. While the estimates produced by ridge regression are biased, they suggested ways of dealing with this, including a switch back to regular ALS estimation at the end of the fitting procedure, when the approximate solution has been reached. They designed their method to avoid difficulties which they called convergence "swamps," characterized by high factor collinearity in all three modes. We will see that ELS (at least the simple version tested here) is most successful with a different kind of convergence difficulty.

In [22], Paatero proposed the multilinear engine (ME) program to accelerate the fit of the PARAFAC model. ME changes all of the sets of parameters at once, whereas ELS is based on ALS, and updates alternatively each of the loading factors.

In [8], Franc proposed an acceleration to the convergence of PARAFAC based on a gradient method. In fact, the loading matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are updated using the gradient descent.

A closed-form solution to fit the PARAFAC model was proposed by Sanchez and Kowalski in [25]. It reduces the problem to a rectangular eigenvalue-eigenvector equation, but it needs at least two of the loading matrices to be linearly independent.

Another closed-form solution for three-way arrays, and based on a single matrix eigenvalue decomposition (EVD) was presented in [21] by Leurgans, Ross, and Abel. It also requires that two of the loading matrices are linearly independent and that every pair of columns of the last loading matrix is linearly independent. Previous approaches are made more robust in [20] by taking all matrix slices into account, which leads to a simultaneous matrix decomposition. All of these methods require that the array rank (as defined in [17], for instance), F, is less than or equal to two of the array dimensions. In [19] De Lathauwer generalizes the approach presented in [20] to the case where F is less than or equal to one of the array's dimensions, and subject to a condition involving the product of the remaining array dimensions. One advantage of ELS is that it can be applied even if the previous conditions are not met.

2. Model and notation. We consider the three-way PARAFAC model of expression (1). This model can be written in a compact form using the Khatri–Rao product $\odot$ (columnwise Kronecker product) as, possibly up to an error term,

$$\mathbf{X}^{(I\times JK)} \approx \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T,$$

where matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are matrices of size $I\times F$, $J\times F$, and $K\times F$, and $\mathbf{X}^{(I\times JK)}$ is the matrix of size $I\times JK$ obtained by unfolding the array $\mathbf{X}$ of size $I\times J\times K$ in the first mode. There exist several algorithms that fit the PARAFAC model. We focus on the most widely used among all: the ALS algorithm. ALS consists of estimating one of the three matrices at each step by minimizing in the least squares sense the error

$$\Upsilon = \| \mathbf{X}^{(I\times JK)} - \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T \|_F^2,$$

where $\| \bullet \|_F$ denotes the Frobenius norm. With matrices $\mathbf{B}$ and $\mathbf{C}$ fixed to initial values, the estimate of $\mathbf{A}$ in the least square sense is given by

$$\widehat{\mathbf{A}} = \mathbf{X}^{(I\times JK)}(\mathbf{Z}_a^+)^T, \tag{2}$$

where $\mathbf{Z}_a = \mathbf{C}\odot\mathbf{B}$ and $(^+)$ is the Moore–Penrose pseudoinverse. We estimate matrices $\mathbf{B}$ and $\mathbf{C}$ in an equivalent way, with $\mathbf{Z}_b = \mathbf{A}\odot\mathbf{C}$ and $\mathbf{Z}_c = \mathbf{B}\odot\mathbf{A}$, and repeat the same steps until a convergence criterion is reached—typically when the error Υ exhibits, between two iterations, a change smaller than a predefined threshold, which varies depending on the data. For simple data it can be set to 10^{-6}, for example, but it should be smaller for difficult data, 10^{-10}, for example. Note that, in order to avoid a threshold that is scale dependent, a relative error can be used instead, or array $\mathbf{X}$ can be prenormalized by its Frobenius norm.

We summarize the steps of the ALS algorithm in Figure 1.

It sometimes happens that the convergence needs a very large number of iterations. Choosing good starting values will, in some cases, help to reach the global minimum very quickly. Sometimes, however, it is impossible to reach a global minimum quickly by ALS from any starting point because the solution is embedded in a deep swamp, or is in fact unreachable at the solution rank, and can only be approached through an infinite series of diverging better fitting sets of loadings, as described by Kruskal.

3. Line search. Bro (in [2, p. 95–96]) and Harshman (in [9, p. 32–33]) have pointed out the important fact that, when the convergence is slow, there exist cycles of convergence defined by a unique direction. Within a given cycle, the loading factors evolve in the same direction to the final solution of that cycle. The following cycles exhibit the same scenario. The convergence within the cycle can take several iterations. To limit the number of iterations of a given cycle, Harshman and Bro propose

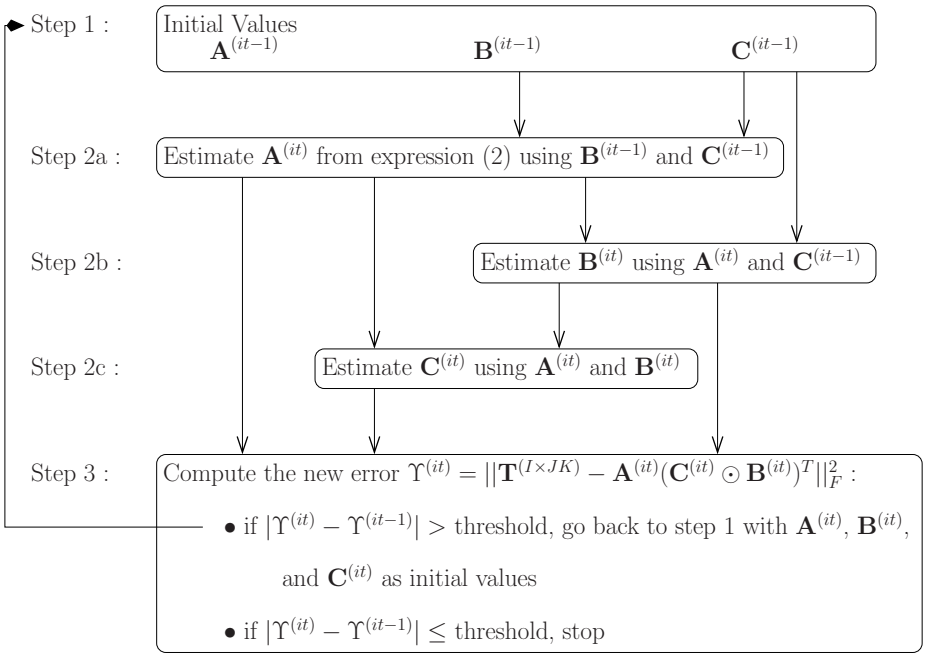

FIG. 1. *Steps of the ALS algorithm.*

to extrapolate, or more precisely, they propose to predict the value of the loading factors a certain number of iterations ahead by computing a sort of linear regression:

$$\mathbf{A}^{(new)} = \mathbf{A}^{(it-2)} + R_{LS}(\mathbf{A}^{(it-1)} - \mathbf{A}^{(it-2)}) \tag{3}$$

$\mathbf{A}^{(it-1)}$ is the estimate of matrix $\mathbf{A}$ obtained in the ALS iteration $(it-1)$, and $\mathbf{A}^{(new)}$ is the matrix that will be used in the itth iteration instead of $\mathbf{A}^{(it-1)}$. $(\mathbf{A}^{(it-1)} - \mathbf{A}^{(it-2)})$ defines the direction of the cycle. Matrices $\mathbf{B}^{(new)}$ and $\mathbf{C}^{(new)}$ are obtained in an equivalent way using the same relaxation factor R_{LS}. Of course, extrapolation should be very simple and does not make sense if it requires more time than the corresponding iterations. The simplest case is, of course, when R_{LS} is given a fixed value (between 1.2 and 1.3) [9], or is set to $it^{1/3}$ [2].

At every iteration it, the "new" loading factors are used to compute the error

$$\Upsilon^{(new)} = \| \mathbf{X}^{(I\times JK)} - \mathbf{A}^{(new)}(\mathbf{C}^{(new)} \odot \mathbf{B}^{(new)})^T \|_F^2 . \tag{4}$$

If $\Upsilon^{(new)} \geq \Upsilon^{(it-1)}$, then this means that we went too far in the extrapolation because R_{LS} is too large; R_{LS} is decreased, and we take the loading factors of iteration $(it-1)$ instead of the "new" ones. However, if $\Upsilon^{(new)} < \Upsilon^{(it-1)}$ then acceleration is accomplished and we gain some iterations.

The steps of the ALS algorithm with line search, as proposed by Andersson and Bro in [1], are summarized in Figure 2. The dashed area corresponds to the line search part.

Line search is executed after a few iterations of the ALS algorithm in order to wait for the system to stabilize. In [1] "few" is set to 6 but it could be higher depending on the data. The relaxation factor R_{LS} is defined for iteration (it) by : $R_{LS} = it^{1/n}$, with n fixed to 3 at the beginning of the simulation. When the acceleration fails several times (5 times in [1]), R_{LS} is decreased to $it^{1/(n+1)}$ and $\mathbf{A}^{(it-1)}$, $\mathbf{B}^{(it-1)}$, and $\mathbf{C}^{(it-1)}$

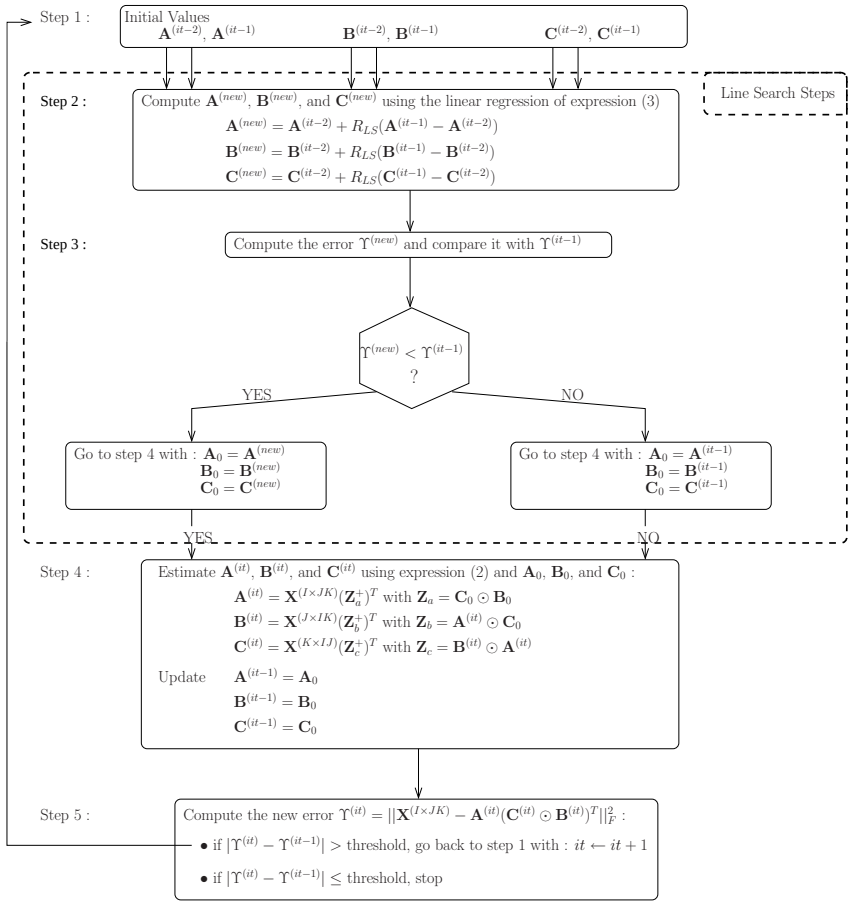

FIG. 2. *Steps of the ALS algorithm with LS.*

are used to update the loading factors of the current iteration (it) as described by the graph in Figure 2 at the end of the third step. However, when $\Upsilon^{(new)} < \Upsilon^{(it-1)}$, matrices $\mathbf{A}_0$, $\mathbf{B}_0$, and $\mathbf{C}_0$ are set to $\mathbf{A}^{(new)}$, $\mathbf{B}^{(new)}$, and $\mathbf{C}^{(new)}$, respectively. After estimating the loading factors at step 4, we update the loading matrices of iteration $(it-1)$ to $\mathbf{A}_0$, $\mathbf{B}_0$, and $\mathbf{C}_0$, and use them with those of iteration (it) for the next iteration (unless the algorithm has converged).

The fact that R_{LS} has a small value would suggest that the acceleration is not very efficient. This is not true since the effect of R_{LS} is compounded from one iteration to the next, leading eventually to a noticeable reduction of the number of iterations, as shown in Figure 10.

This linear extrapolation was applied to our synthetic data in section 5.6, as a basis for comparison with our ELS method. We will see that it can produce a clear improvement, for example reducing iterations in one example from about 10,000 to 5,000. However, since the number required is still high, it is still of interest to look for a novel method to reduce the number of iterations even further.

4. Enhanced line search (ELS). The idea of the enhanced line search (ELS) consists of seeking the optimal relaxation factor R_{LS} that leads to the final solution of a given cycle in only one step. For iteration (it), define $\mathbf{G}_a^{(it)} = \mathbf{A}^{(it-1)} - \mathbf{A}^{(it-2)}$ as the

direction of the cycle for loading matrix $\mathbf{A}$. $\mathbf{G}_b^{(it)}$ and $\mathbf{G}_c^{(it)}$ are defined equivalently. Instead of fixing a single value R_{LS} for the three modes as in expression (3), we may look for the optimal triplet (R_a, R_b, R_c) that minimizes

$$\begin{aligned}\Upsilon_{ELS} =& \| \mathbf{X}^{(I\times JK)} - (\mathbf{A}^{(it-2)} + R_a\mathbf{G}_a^{(it)}) \\ & \Big((\mathbf{C}^{(it-2)} + R_c\mathbf{G}_c^{(it)}) \odot (\mathbf{B}^{(it-2)} + R_b\mathbf{G}_b^{(it)})\Big)^T \|_F^2 \,. \end{aligned} \tag{5}$$

ELS is performed at the beginning of the ALS algorithm as shown in Figure 3, where step $1'$ corresponds to the ELS part. Relaxation factors applied to the loadings are no longer fixed as for the line search method in Figure 2, but they are computed at step $1'$ of Figure 3 as the optimal values that provide the smallest error Υ_{ELS}. At step 3, after estimating the loading matrices of iteration (it), we update those of iteration $(it-1)$ to $\mathbf{A}^{(new)}$, $\mathbf{B}^{(new)}$, and $\mathbf{C}^{(new)}$. Loading matrices of both iterations $(it-1)$ and (it) will then be used in the next iteration if the algorithm does not reach convergence.

The optimal solution is obtained when we jointly minimize Υ_{ELS} w.r.t. the three different factors R_a, R_b, and R_c. In this case the problem consists of solving a system of three polynomials in three unknowns, which leads to a high numerical complexity. Solutions with a smaller complexity are obtained by taking only two unknowns, or the same factor for all the modes $R = R_a = R_b = R_c$. Some of the possible optimizations are listed below:

- (R_a, R_b, R_c) which gives the optimal solution and involves a polynomial in three unknowns of degree 6.
- (R, R, R_c) where we use the same factor for $\mathbf{A}$ and $\mathbf{B}$ and we minimize Υ_{ELS} w.r.t. two variables R and R_c. This involves a polynomial in two unknowns of degree 6.
- (R, R, R) where we use the same factor for all matrices and involves a polynomial in a single unknown of degree 6.
- $R(R_b, R_c)$ where we use the relaxation factor of line search $R = it^{1/3}$ for matrix $\mathbf{A}$, and minimize (5) w.r.t. R_b and R_c. This involves a polynomial in two unknowns of degree 4.
- $R(R, R)$ which is the same as $R(R_b, R_c)$ with $R_b = R_c$, and involves a polynomial in a single unknown of degree 4.
- $R, R(R)$ where we optimize only w.r.t. to R_c.

In this article, the exploration of alternative ELS models is initiated by implementing (R, R, R), which is the simplest one that is "fully ELS." In this case, the error Υ_{ELS} is a polynomial of degree 6 in R (we omit the iteration index (it) to simplify the notation):

$$\begin{aligned}\Upsilon_{ELS}(R) &= \sum_{ijk}\left(X_{ijk} - \sum_{f=1}^{F}(A_{if} + RG_{a,if})(B_{jf} + RG_{b,jf})(C_{kf} + RG_{c,kf})\right)^2 \\ &= \sum_{d=0}^{6} p_d R^d, \end{aligned} \tag{6}$$

where p_d, $d = 0, \ldots, 6$ are functions of observed values stored array $\mathbf{X}$ and coefficients

of loading matrices of iterations $(it-1)$ and $(it-2)$; the expression of p_d are given in section A.2. To find the optimal R it suffices to determine the roots of polynomial $\Upsilon'_{ELS}(R)$, which provides five possible values of R. We feed those values into expression (6) and keep the one that gives the smallest error Υ_{ELS}.

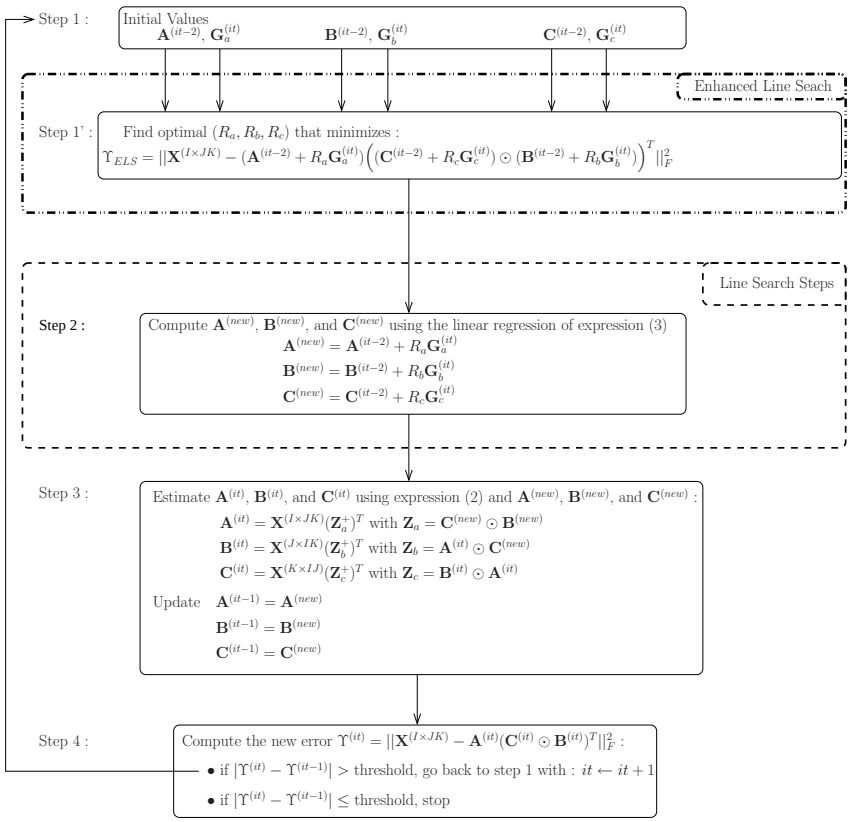

FIG. 3. *Steps of the ALS algorithm with ELS.*

To obtain some insight into whether the extrapolation is likely to be advantageous in the short-range sense (cf. question (i) posed at the end of this section), we can estimate the relative computation required by a single ELS iteration compared to LS. To do this, we compute the complexity of ALS and compare it with the complexity of optimization (R, R, R), for example. At each ALS iteration the following steps are performed:

1. Compute the optimal relaxation factor R by minimizing expression (5). To do so, take the derivative of (5) w.r.t. R, and root the obtained polynomial of degree 5 in one unknown.

2. Compute the new loading factors as in (3) and compute the corresponding error Υ_{new} given by expression (4).

3. Use $\mathbf{A}^{(new)}$, $\mathbf{B}^{(new)}$, and $\mathbf{C}^{(new)}$ as starting values for the PARAFAC iteration instead of $\mathbf{A}^{(it-1)}$, $\mathbf{B}^{(it-1)}$, and $\mathbf{C}^{(it-1)}$, and estimate the first loading factor $\widehat{\mathbf{A}}$ as shown in (2).

4. Perform step 3. To estimate each of the remaining loading factors $\widehat{\mathbf{B}}$ and $\widehat{\mathbf{C}}$, by using matrices $\widehat{\mathbf{A}}$ and $\mathbf{C}^{(new)}$ to estimate $\widehat{\mathbf{B}}$, and matrices $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{B}}$ to estimate $\widehat{\mathbf{C}}$.

According to the details given in section A.1, one (ALS+ELS) iteration corresponds to about $(F + 7F^2)(IJ + JK + IK) + 3IJKF + 11F^3 + 2F^2(I + J + K) + (8F + 10)IJK$ multiplications. For instance, this equals 2601 multiplications, when $I = 2$, $J = 3$, $K = 3$, and $F = 3$. Without ELS, ALS requires $(F + 7F^2)(IJ + JK + IK) + 3IJKF + 11F^3 + 2F^2(I + J + K)$, which means 1989 multiplications in the same conditions. On the other hand, ELS makes the number of iterations decrease from 7100 to 250 iterations, hence allowing a decrease of the overall complexity from 14121900 to 650250 multiplications.

It is worth noting that $\Upsilon^{(new)}$ is always smaller than $\Upsilon^{(it-1)}$ when we use optimal values for R_a, R_b, and R_c as is the case for the first three optimizations. However, when we use a fixed relaxation factor as in LS, $\Upsilon^{(new)}$ can exceed $\Upsilon^{(it-1)}$, which means that the acceleration may fail.

This can explain the fact that, in theory, a single iteration of ELS should always improve fit as much as and almost always more than LS at any given point in the solution space. The questions then become: (i) Does the fit improvement turn out to be more beneficial than the cost of added computation is detrimental? (ii) Does the method find a significantly better path to the solution? Question (ii) is particularly important in cases where progress becomes very slow because of local characteristics of the hypersurface along which the fitting procedure is moving. Question (ii) is much subtler than (i). For example, one can easily imagine that a good long-range method might trade off some locally slower steps for a much better path a bit further along (a shortcut just over the horizon). This sort of question can only be answered well through application of a method to simulated and/or real data.

5. Computer results. To compare the performance of ELS to LS, a standard PARAFAC program was minimally modified to change the line search to ELS and to record time and fit information on each iteration. The test datasets were three-way and four-way synthetic data arrays, constructed according to the PARAFAC model to have specific kinds of factor structure and levels of random error. Then, in each test, the two algorithms were given identical problems; that is, they were given the same synthetic datasets, with the same analysis options, and started from the same random starting positions. This allowed us to compare the progress of the two methods step-by-step as they proceeded from a given starting point toward the best least-squares solution.

To obtain a general picture of the relative performance of ELS and LS, we considered a wide range of datasets[1]. A fully systematic exploration has not yet been completed, and even the partial results obtained so far require much more space than is available for this article, so we present here summary conclusions for each of our main test conditions, and give a few illustrative examples.

In most experimental conditions (i.e., most of the data structure types tested), the iteration count for ELS was substantially lower than that for LS, and in no condition was it (reliably) higher. This is consistent with the theoretical expectations described earlier. On the other hand, ELS *execution times* were longer than LS by

[1]We took as our "standard" the PARAFAC function contained in Andersson and Bro's [1] N-way Toolbox for MATLAB (which may be found at http://www.models.kvl.dk/source/nwaytoolbox/index.asp). The same MATLAB code was used except that the ELS extrapolation code replaced the LS extrapolation code in the function. In both versions of the PARAFAC function, a few lines were inserted that obtained time information on each iteration and saved it along with the current value of fit computed by the program at that iteration. Release 7 of MATLAB was used in all experiments.

more than we expected—generally the ratio of ELS to LS execution time per iteration was always the same for any given problem, but was larger than we would predict from our approximate estimates of complexity computation; despite streamlining and vectorization of the code, the ELS to LS ratio was still somewhat larger than our complexity estimates by roughly a factor of three. This might be due to inefficiencies in our ELS algorithm, or perhaps some particularly efficient features in the LS code, or as yet unanticipated considerations. Since reduction in the execution time is the goal of the proposed method, we feel that it is important to communicate both the time and iteration reductions we have observed, even though we are not yet sure of how to interpret the time information (or even whether or not it is somehow artifactual). We will describe a theoretical adjustment that brings times in line with computational complexity, and this provides one way of dealing with the current uncertainty in our timing results.

Complexity differences. In applications where the rank F is small in the sense of inequality (7) given in the appendix, the (R, R, R) version of ELS will significantly increase the complexity per iteration, and hence the CPU time required per iteration over that of LS. Whether or not ELS is attractive thus depends in part on the problem size and dimensionality.

Datasets sizes and numbers of factors to be extracted from them vary from one discipline to the next. In chemometrics and signal processing, typical problems might involve data arrays of the order of $60 \times 60 \times 20$ and perhaps three or four factors to be extracted. In such cases, the computational complexity of ELS is approximately three times that of LS, at least as estimated by the formulae in the appendix. For these problems, use of ELS would be attractive only for classes of problems in which it is clearly superior to LS in ability to traverse the curvature of the solution space. To provide a benefit beyond the use of LS, the ELS method needs to reduce the number of iterations required by LS.

It turns out that a single "bottleneck"—one of the most common and simple kinds of convergence slowdown—appears to have the required properties. We have also found other classes, such as triple bottleneck, where ELS will actually increase substantially the time needed to find the solution.

The data variation that turned out to have the most important impact on the relative performance of the two methods was the factor correlation structure both within and across modes. When no modes had collinear factors and all factors were of roughly equal size, there was no convergence difficulty for either method. For example, midsized datasets (e.g., $45 \times 40 \times 35$) often satisfied the convergence criterion—usually a change in root mean square error (RMS) of 10^{-8} between successive iterations—in 15–75 iterations. ELS usually took fewer iterations to converge, but the iterations were slower. In other words, with our MATLAB implementation, ELS often took on the order of 15%-25% *more time* than LS to reach the convergence criterion, even if figures given by computational complexity calculations are more optimistic.

5.1. Dealing with bottlenecks. We have considered several different situations where convergence of ALS PARAFAC algorithms become slow, but have focused mainly on the one that is the simplest and (outside of the social sciences) the most common: simple factor collinearity. This has several different versions; some or all factors can be collinear in one or several factor loading matrices that define the variation structure of an array. We only briefly look at the other important case—the more complicated kind of convergence difficulties caused by "degenerate PARAFAC solutions."

When one of the factor matrices in the optimal solution has two or more collinear columns, resolving them can seriously slow down the overall progress of ALS estimation of the factors, even though the solution may eventually be well defined. Harshman terms this situation a "bottleneck" [12]. When such collinearity is present in two or

three modes of the array (i.e., two or three of the "latent" factor loading matrices), then one has a structure with a "double" or "triple" bottleneck. We created synthetic data involving single, double, and triple bottleneck structure to test the performance of ELS vs. LS in these conditions. The (R, R, R) version of ELS that we used for these tests behaved quite differently in single vs. multiple bottleneck situations—at least for three-way arrays.

5.2. Single bottleneck situations: When factors in one mode are collinear. In our tests, ELS always outperformed LS when only one mode had factor loading vectors that were almost collinear, providing the analysis reached a global optimum in which all factors were approximately recovered. The time and iteration values observed in one such run are shown in Figure 4.

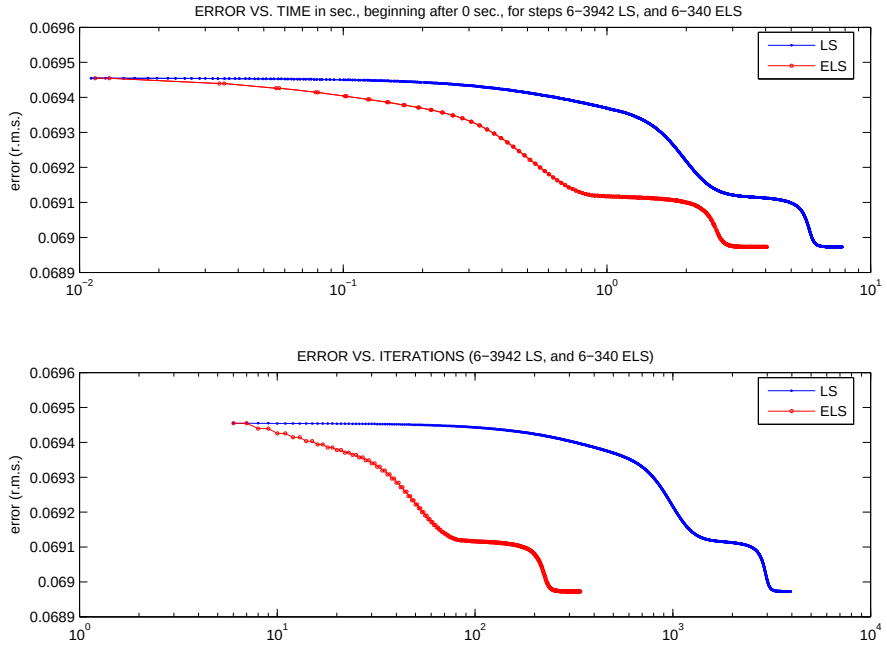

FIG. 4. *Performance of ALS with both LS or (R, R, R)-ELS accelerations, on a $12 \times 11 \times 10$ array of rank 5, in the presence of a single bottleneck (3 factors out of 5 are almost collinear in one of the 3 modes).*

The example shown is for an array with 5 factors and the lowest level of random noise used in the tests (0.1%). Three of the factors were almost collinear (separated by 10 degrees in the Mode C factor space) while the other two were roughly orthogonal to the other factors. The shape of the curves is quite similar. This suggests that the two algorithms are following "similar" or somewhat parallel paths through the solution space, and are encountering a similar sequence of more and less difficult regions in the solution space, but they are progressing at different rates because ELS tends to make bigger improvements in fit. As shown here, ELS often reduced the total number of iterations by approximately an order of magnitude, but because its iterations took somewhat longer (at least with our MATLAB implementation), the time reduction was between half and two-thirds of the size of the reduction in iterations. The fit and time curves in Figure 4 are based on a relatively small dataset ($12 \times 11 \times 10$ with five factors). In that case, index (7) is 1.37, which shows that an ELS iteration is more complex than LS.

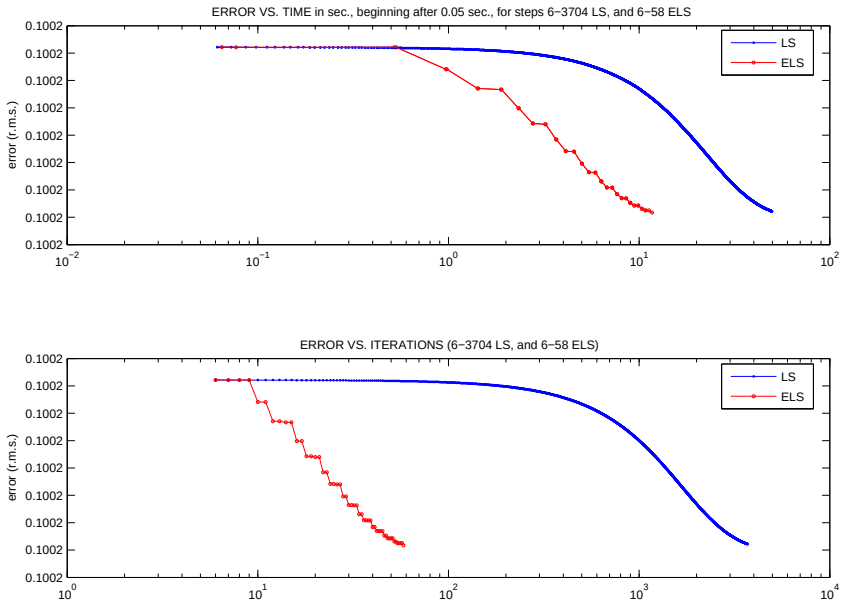

FIG. 5. *Performance of ALS with both LS or* (R, R, R)*-ELS accelerations, on a* $80 \times 60 \times 20$ *array of rank* 5*, in the presence of a single bottleneck.*

With the larger dataset considered next, index (7) is even smaller, and the relative computational complexity of ELS is at least three times that of LS per iteration. Our MATLAB implementation takes closer to 9 times as much time, for reasons that we are not yet able to fully explain. ELS is, however, still attractive in this low-rank large-dimension case, despite the relatively high complexity per iteration.

Note the clear two-step pattern in Figure 5, both in the fit drop and in the associated time needed at each successive iteration (i.e., two-iteration values are close to one another, and then there is a larger interval, and two are close to one another again). The pattern is present in both curves, but is much more obvious in the ELS curve because of the rapid drop in fit at every other step. This "paired-step" pattern is due to the way LS is currently implemented in the PARAFAC function for the N-way toolbox. The program collects two sets of loadings, from two successive iterations, and then extrapolates based on those two, and collects two more, etc. Thus the extrapolation occurs on every other iteration. Because ELS was incorporated into the exact same loops that governed LS, it too is applied on every other iteration.

Our tests indicate that LS is considerably more effective when the extrapolation is performed on every iteration, as in the original extrapolation used in Harshman 1970 [9]. However, we concentrate in this article on a direct comparison of paired-step LS with paired-step ELS (we have also begun some comparisons between the two methods when both are performed at every iteration, and our preliminary results suggest that in this case the performance difference between them is smaller).

For this dataset, as with all other "single bottleneck" cases we have tested, paired-step ELS clearly outperforms paired-step LS—so long as they find the global optimum. However, when the path taken by an analysis traps it in a local optimum, or when some of the highly collinear factors are too poorly resolved due to error in the data, the behavior of LS and/or ELS changes and the time advantage offered by ELS is reduced or eliminated. ELS continues to require fewer iterations, but the difference between the counts becomes small enough that ELS does not reduce the overall time

(at least with our MATLAB implementation).

5.3. Multiple bottlenecks and degenerate solutions. Multiple bottlenecks and degenerate solutions appear when experimental requirements or practical limitations in data collection make collinearity of some factors unavoidable; this most commonly applies to only one mode of the data array. The previous results are for single-mode bottlenecks which leads us to the tentative conclusion that for such cases ELS would seem an attractive estimation method. There are situations, though, in which some subset of the factors will be collinear in two modes, or even in all three modes of a three-way array. Our experiments therefore simulated these (less common) kinds of data as well. We found that the advantage of ELS does not generally extend to these situations. The curvature of the path to the optimum gets more complicated and apparently makes ELS "shortcut" methods less successful. This is another demonstration of the subtlety of the considerations involved in nonlinear extrapolation of PARAFAC solutions.

Neither double- nor triple-bottleneck situations benefited much from the simplest (R, R, R) version ELS. In general, the *time* required by our MATLAB implementation to reach the converged solution was increased. However, our test cases often were hard to fit, so we had to take care to distinguish global optimum cases from local optimum ones. To obtain global optima with adequate frequency in the double-bottleneck case, the angle between factors had to be increased to moderate values (25 degrees in the case of Figure 6), making the collinearity in individual factor spaces less extreme but the combined effects of the collinearity in the two or three modes was still fairly severe. Unfortunately, even in clear cases of having reached the global optimum, where recovery of all factors was close to perfect (when the noise level was set at 0.001), the ELS method usually took longer (in CPU time) to converge than simple LS. Figure 6 shows the results of one such triple-bottleneck case.

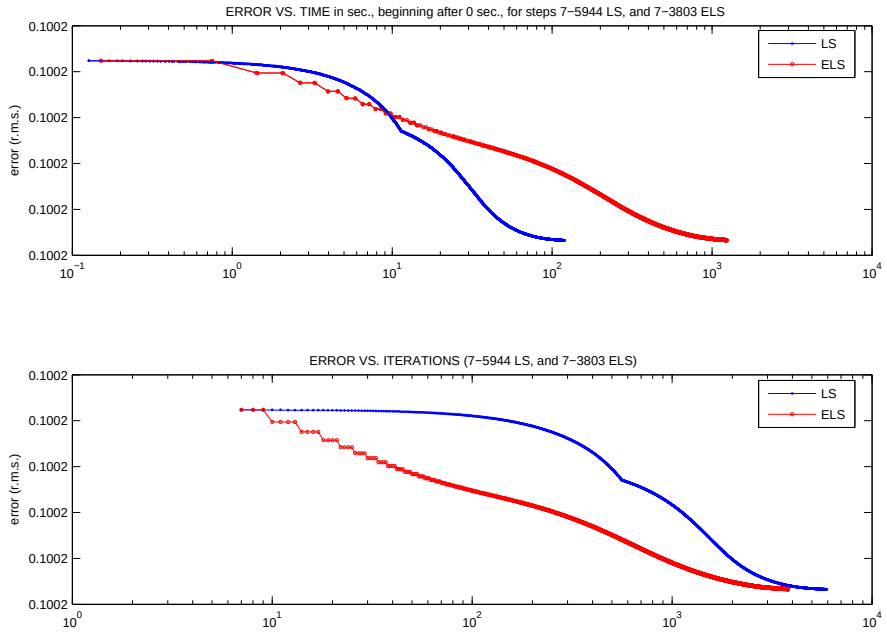

FIG. 6. *Performance of ALS with both LS or* (R, R, R)*-ELS accelerations, on a* $80 \times 60 \times 20$ *array of rank* 4*, in the presence of a triple bottleneck.*

5.4. Degenerate and quasi-degenerate solutions. There is another important situation in which serious convergence difficulties arise. This happens when the factors are not particularly collinear on average, but the angle between them varies greatly across levels of the array (i.e., differs across values of the third array index). This kind of variation in factor structure is not consistent with the PARAFAC model. However, PARAFAC can fit part of the "axis wobble" or "Tucker Variation" [14] by reweighting axes after the space has been sheared [11]. Thus, when too much axis "wobble" is present, the **A**, **B**, and **C** factor spaces become inversely sheared to better fit it, creating a "degenerate solution," which involves strong collinearities and seriously impedes convergence. Degenerate solutions are usually dealt with by imposing constraints. However, speedup methods like ELS could be useful if they accelerated progress through "swamps" or deep into a swamp to find a final solution. In our initial tests of the (R, R, R) method, ELS has not been helpful in dealing with swamps, but it is not unreasonable to conjecture that versions more sophisticated than (R, R, R), as suggested in section 4, might do better in these situations.

5.5. A four-way example with two collinear and two noncollinear modes. From the experiments reported so far, it is unclear whether the relative lack of ELS success when applied to factor structures with double and triple bottlenecks is because of too many bottlenecks or too little "wiggle room." That is, we cannot distinguish the cases that have multiple modes with bottlenecks from those that do not have multiple modes without bottlenecks. In the present section, we report some earlier studies on the impact of ELS on ALS in four-way arrays. In these arrays, two modes have almost collinear factors and two do not. If multiple bottlenecks is what creates the convergence problem, then ELS should also encounter difficulties in these datasets. If lack of "wiggle room" is what creates the convergence problem, then ELS should be better at dealing with double bottlenecks.

The experiments we are about to describe are simpler in two important ways: (a) the tests did not measure or record *execution time* information; (b) the datasets were constructed as error-free arrays, that is, without adding any random noise. The first limitation makes the interpretation difficult, but the dramatic reduction in iteration counts does appear impressive when compared to the relatively modest differences in iterations in, for example, Figure 6. The second limitation can be minimized by qualifying our interpretation. The difference is nontrivial because when collinearities are combined with error, it can complicate the algorithm's task of resolving the highly similar factor profiles. However, the results are still informative if considered as demonstrating certain theoretical/mathematical properties of four-way (R, R, R)-ELS. They might also be interpreted as simulations of real world cases where the error is sufficiently small to make the behavior of the algorithms roughly equivalent to those found in these error-free cases.

We consider the four-way PARAFAC model:

$$X_{ijkl} = \sum_{f=1}^{F} A_{if} B_{jf} C_{kf} D_{lf},$$

where

$$\mathbf{A} = \begin{pmatrix} 1 & \cos(\theta) & 0 & \sin(\theta) \\ 0 & \sin(\theta) & 1 & \cos(\theta) \end{pmatrix},$$

$$\mathbf{B} = \begin{pmatrix} 3 & \cos(\theta) & 0 & \sin(\theta) \\ 0 & \sin(\theta) & 1 & \cos(\theta) \\ 0 & \sin(\theta) & 0 & \sin(\theta) \end{pmatrix},$$

and $\mathbf{C}$ and $\mathbf{D}$ are randomly generated matrices of size 3×4. The collinearity is controlled through variable θ. We take $\theta = \pi/60$ in Figures 7 and 8. The first and second columns of each of the matrices $\mathbf{A}$ and $\mathbf{B}$ are almost collinear as θ is very close to zero ($\theta \simeq 0.052$). The same thing holds for the third and fourth columns of $\mathbf{A}$ and $\mathbf{B}$.

This example demonstrates one of the cases where results of [25] and [21] cannot be applied, since there are more columns than rows in the loading matrices and so they do not have full column rank.

We notice from Figure 7 that ELS reduces the number of iterations needed to meet the criterion for approximate convergence from more than 10000 to about 2000! We report in Figure 8 the median of the loss function for one dataset, over 100 independent trials (with 100 different random initial values). We notice that even though ALS + LS reaches the error 10^{-4} very quickly, it is then trapped for many iterations ("trapped in the bottleneck"). In contrast, ALS + ELS escapes comparatively quickly from the bottleneck (after 1000 iterations) and converges to smaller values of the error 10^{-12}, while ALS and ALS+LS remain in the plateaux 10^{-4} and 10^{-5}, respectively.

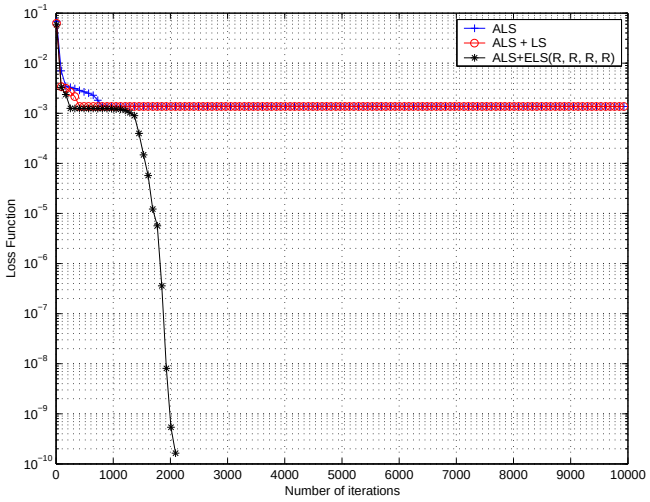

FIG. 7. *Loss function* Υ *as a function of the number of iterations for ALS with LS, and ELS with optimization* (R, R, R, R) *for* $\theta = \pi/60$.

Figure 9 appears to confirm this hypothesis by showing more frequent changes in parameter sets. It gives the variation of the coefficients of matrix $\hat{\mathbf{A}}$ as a function of the number of iterations. During progress through a bottleneck, the variation of $\hat{\mathbf{A}}$ coefficients is very small and this increases when we get out of the bottleneck. The same thing is evident in matrices $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$.

5.6. ELS applied to blind channel identification of an UDM. This second four-way example demonstrates an application of ELS to blind identification of an under-determined mixture (UDM). Specifically, we use ELS to accelerate ALESCAF, the algorithm proposed in [7] for blind channel identification based on the characteristic function in an UDM.

Using the notation defined in [7], ALESCAF leads to a four-way PARAFAC model:

$$\mathbf{T}^{(P \times KP^2)} = \mathbf{A}(\mathbf{D} \odot \mathbf{A} \odot \mathbf{A})^T.$$

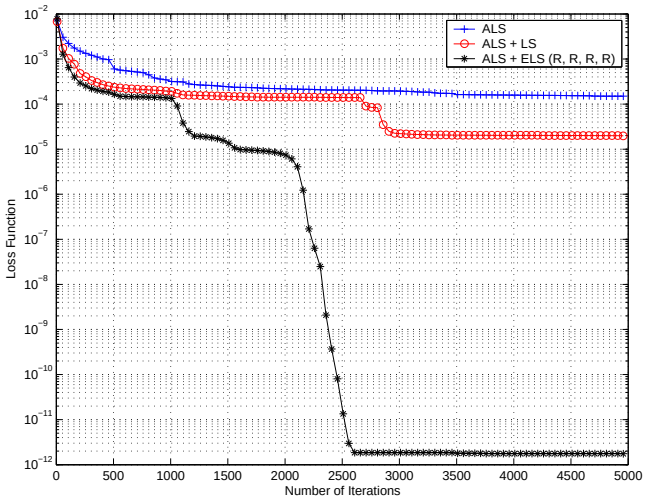

FIG. 8. *Loss function* Υ *as a function of the number of iterations for ALS with LS, and ELS with optimization* (R, R, R, R) *for* $\theta = \pi/60$*, median value over* 100 *independent trials.*

The array $\mathbf{T}$ contains the third derivatives of the joint characteristic function of the observations computed at K points of the grid Ω. Matrix $\mathbf{D}$ is obtained from the independence property of the sources and its entries are defined as

$$D_{kn} = \psi_n^{(3)} \left(\sum_q A_{qn} u_q[k] \right),$$

where $1 \leq k \leq K$ and $1 \leq n \leq N$. $\mathbf{A}$ is the channel matrix of size 2×3 to be identified.

As in our earlier tests, we use the MATLAB ALS implementation of PARAFAC made available by Andersson and Bro (http://www.models.kvl.dk) and described in [1]. Also as before, we create our ELS version by replacing their LS procedure by our ELS procedure, this time the (R, R, R, R) version. The three sources are BPSK, and we generate an "infinite block" of data by taking all of the 2^3 possible combinations of $\{-1, 1\}$, and we take 10000 as the maximum number of iterations. As in the previous four-way example, noise is not taken into account.

In Figure 10 we report the gap between estimated and actual mixing matrix using ELS and compare it with ALS with LS and nonaccelerated ALS. Figure 11 gives the error as a function of the number of iterations. The figure shows that ELS is very useful for reducing the number of iterations needed in three-bottleneck versions of four-way arrays. The number of iterations decreases from 5000 when using ALS with LS, to 500 when using optimization (R, R, R, R) of ELS. On the one hand, these results seem to be too dramatic to be "canceled out" by increases in iteration time, but on the other hand, the LS and ELS parameter changes (Figure 11) make us somewhat more cautious, since these seem more similar.

Overall, these four-way results encourage us to hope that when there is at least one mode that is free of collinearities, this type of ELS might be generally helpful.

A few cautionary points should be noted: When making comparisons of the methods, it might be wise to underemphasize the dramatic ELS drops of the error (or objective function) when the value falls below something like 10^{-3} or 10^{-4}, since these would be impossible with most real data containing measurement error or other disturbances of the data values. It is also not known how the behavior in the graphs

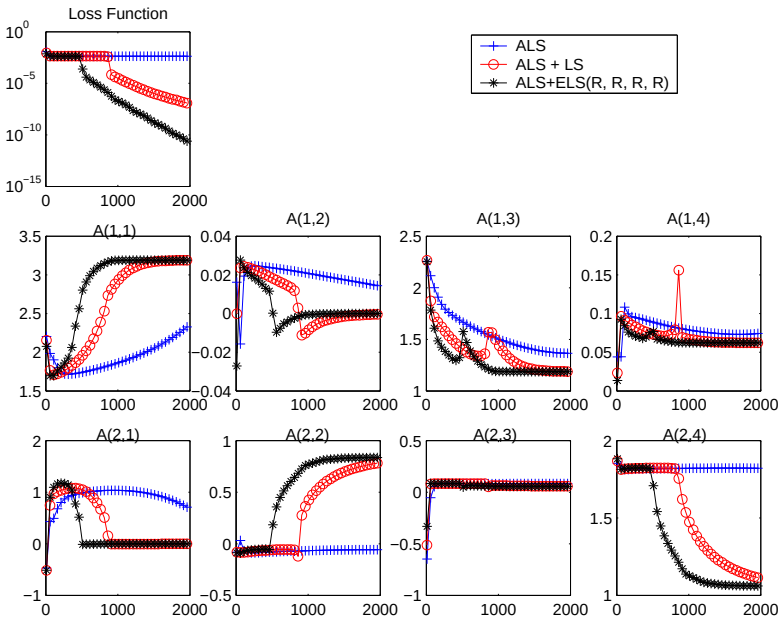

FIG. 9. $\hat{\mathbf{A}}$ *coefficients as a function of the number of iterations.*

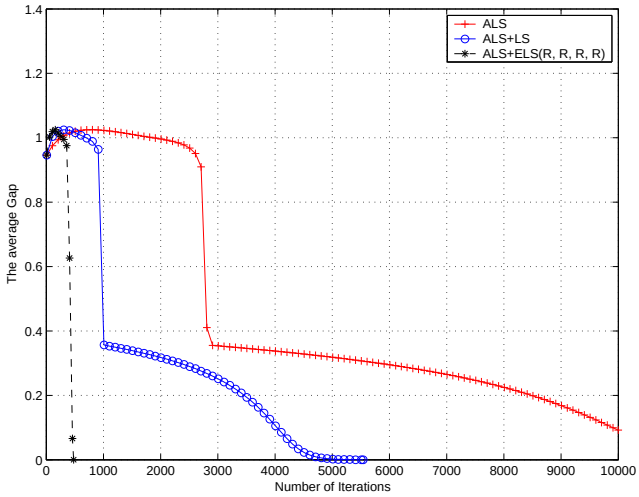

FIG. 10. *Gap between estimated and original channel matrix as a function of the number of iterations using ALS, ALS with LS, and ALS with ELS.*

associated with our four-way examples might be modified even at higher levels by the presence of random error; the three-way experiments suggest that some differences can be expected.

6. Concluding remarks. ELS is a novel technique aiming at accelerating convergence of the ALS algorithm when used to fit the PARAFAC model. Our simulations indicated that ELS could be a very attractive way to deal with "single bottleneck" situations—three-way arrays that have factor collinearities in one of the modes. As

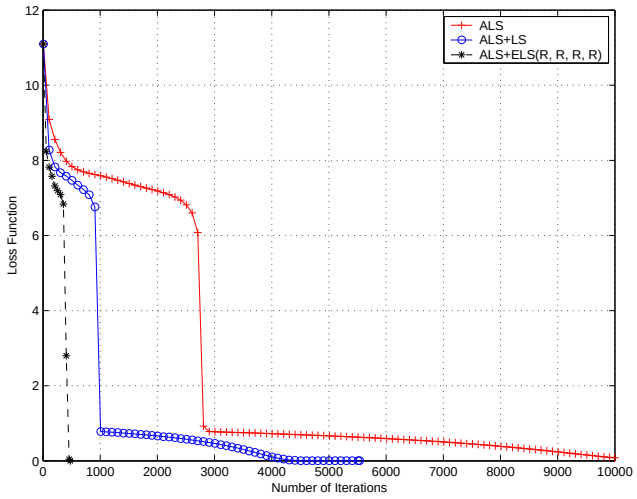

FIG. 11. *Loss function* Υ *as a function of the number of iterations using ALS, ALS with LS, and ALS with ELS.*

shown in Figures 4 and 5, ELS often decreased the number of iterations by an order of magnitude and the time to convergence by at least half to two-thirds of an order of magnitude (except when trapped in a local optimum). This was more than enough to counterbalance the longer iteration times due to the higher computational complexity.

On the other hand, in double- and triple-bottleneck three-way factor structures (i.e., where at most only one mode is free of collinearities), the (R, R, R) version of ELS that was tested here did not offer an advantage, but instead appeared to lengthen the overall time to convergence. However, we remain open to the possibility that our MATLAB time information is somehow unrepresentative or at least does not reflect what might be possible.

Two applications involving four-way arrays (with high rank) lacked information on execution time but some comparison based on computational complexity ratios could be a basis for tentative predictions. The encouraging reductions in iterations seen here open up the possibility that even when an array has three modes with collinearities, this is not necessarily a problem for ELS—if there is also one mode without factor collinearities. This possibility should be explored further.

This article presents some initial exploration of a nonlinear approach to ALS extrapolation, but the investigation is obviously a work in progress. Our theoretical understanding of multistep properties of ELS is still quite incomplete. We have identified some classes of problems where it works better than LS and others in which it may not. The dramatic improvements that can be obtained by ELS in "single bottleneck" situations is well demonstrated. And even if the (R, R, R) implementation of ELS reaches its limits and does not perform significantly better than LS in the absence of two noncollinear modes, or in the presence of convergence swamps arising from "degenerate PARAFAC solutions," there may be other implementations of ELS, such as that called $R(R, R)$ in section 4, which could be more successful in these circumstances. These issues remain to be studied.

Appendix A.

A.1. ELS steps and complexity. During one iteration of the ALS algorithm, the following operations are performed (we list the operations for the estimation of

$\widehat{\mathbf{A}}$, but similar operations are required for the two other loading factors):

1. Compute the Khatri–Rao product to obtain matrix $\mathbf{Z}^a$. This costs FJK multiplications.
2. Compute $\mathbf{Z}_a^+$ by reduced SVD of $\mathbf{Z}_a$, which requires $7JKF^2 + \frac{11}{3}F^3$ multiplications.
3. Estimate the factor loading $\widehat{\mathbf{A}}$ as shown in expression (2), which requires $IJKF + IF^2 + IF$ multiplications (if we assume $F \leq JK$).

As a consequence, the whole ALS iteration for the 3 modes requires an order of $(F + 7F^2)(IJ + JK + IK) + 3IJKF + 11F^3 + 2F^2(I + J + K)$multiplications.

Now let us evaluate the additional computational complexity involved by ELS. In order to do this, note that the ELS criterion can be rewritten as

$$\Upsilon_{ELS} = \sum_{ijk} \left[Y_{ijk} + R\, D_{ijk} - R^2\, E_{ijk} + R^3\, F_{ijk} \right]^2 .$$

The explicit calculation of arrays $\mathbf{Y}$, $\mathbf{D}$, $\mathbf{E}$, and $\mathbf{F}$ requires an order of $8IJKF$ multiplications. Next, the calculation of the coefficients of the degree-6 polynomial in R requires $10IJK$ multiplications. The computation of stationary points and the selection of the absolute minimum yield a negligible complexity since of order $O(5^3)$. As a conclusion, the additional complexity generated by ELS is thus of order $(8F+10)IJK$ multiplications when we choose the optimization with respect to a single factor R, that is, (R, R, R) for a three-way PARAFAC model. This is not negligible and can be considered to be small only for large enough F and small enough dimensions. More precisely, if

$$F \left(\frac{1}{I} + \frac{1}{J} + \frac{1}{K} \right) \gg 1, \tag{7}$$

then the additional complexity required by ELS may be considered to be negligible over that of LS. For instance, this is the case of generic arrays of size $(I, J, K) = (10, 10, 10)$, which have a rank $F = 36$.

More generally, it is interesting to evaluate the computational complexity for N-way arrays of size $I_1 \times I_2 \times \ldots \times I_N$, when all of the dimensions are of the same order $O(I)$. For ALS we get $\frac{11}{3}NF^3 + 2F^2NI + NI^NF + 7F^2NI^{N-1} + FNI^{N-1}$. On the other hand, it can be shown that ELS requires $[2^N F + O(N^2)]I^N + O((2N-1)^3)$ additional multiplications.

A.2. Expression of p_d. We define $q_{f,d}$ for $d = 0, \ldots, 6$ as

$$\begin{aligned}
q_{f,0} &= A_{if} B_{jf} C_{kf}, \\
q_{f,1} &= A_{if} B_{jf} G_{c,kf} + A_{if} G_{b,jf} C_{kf} + G_{a,if} B_{jf} C_{kf}, \\
q_{f,2} &= A_{if} G_{b,jf} G_{c,kf} + G_{a,if} B_{jf} G_{c,kf} + G_{a,if} G_{b,jf} C_{kf}, \\
q_{f,3} &= G_{a,if} G_{b,jf} G_{c,kf}.
\end{aligned}$$

$q_{f,d}$ depends on i, j, k but we omit the indices for simplicity. Then polynomial coefficients p_d are given by

$$\begin{aligned}
p_0 &= \textstyle\sum_{ijk} \big(X_{ijk} - \sum_f q_{f,0}\big)^2, \\
p_1 &= -2\textstyle\sum_{ijk} \big(X_{ijk} - \sum_f q_{f,0}\big)\big(\sum_f q_{f,1}\big), \\
p_2 &= \textstyle\sum_{ijk} \big(\sum_f q_{f,1}\big)^2 - 2\big(X_{ijk} - \sum_f q_{f,0}\big)\big(\sum_f q_{f,2}\big), \\
p_3 &= 2\textstyle\sum_{ijk} \big(\sum_f q_{f,1}\big)\big(\sum_f q_{f,2}\big) - \big(X_{ijk} - \sum_f q_{f,0}\big)\big(\sum_f q_{f,3}\big), \\
p_4 &= \textstyle\sum_{ijk} \big(\sum_f q_{f,2}\big)^2 + 2\big(\sum_f q_{f,1}\big)\big(\sum_f q_{f,3}\big), \\
p_5 &= 2\textstyle\sum_{ijk} \big(\sum_f q_{f,2}\big)\big(\sum_f q_{f,3}\big), \\
p_6 &= \textstyle\sum_{ijk} \big(\sum_f q_{f,3}\big)^2.
\end{aligned}$$

REFERENCES

[1] C. A. Andersson and R. Bro, *The n-way toolbox for MATLAB*, Chemom. Intell. Lab. Syst., 52 (2000), pp. 1–4.

[2] R. Bro, *Multi-way Analysis in the Food Industry: Models, Algorithms, and Applications*, Ph.D. thesis, University of Amsterdam, Amsterdam, The Netherlands, 1998.

[3] J. D. Carroll and J. J. Chang, *Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart–Young decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[4] R. B. Cattell, *Parallel proportional profiles and other principles for determining the choice of factors by rotation*, Psychometrika, 9 (1944), pp. 267–283.

[5] R. B. Cattell and A. K. S. Cattell, *Factor rotation for proportional profiles: Analytical solution and an example*, British Journal of Statistical Psychology, 8 (1955), pp. 83–92.

[6] P. Comon, *Blind channel identification and extraction of more sources than sensors*, in Proceedings of the SPIE Conference, San Diego, 1998, pp. 2–13. Republished in IEEE Trans. Signal Process, 52 (2004), pp. 11–22.

[7] P. Comon and M. Rajih, *Blind identification of under-determined mixtures based on the characteristic function*, in ICASSP'05, vol. IV, Philadelphia, 2005, pp. 1005–1008.

[8] A. Franc, *Etude Algébrique des Multitableaux: Apports de l'algèbre Tensorielle*, Ph.D. thesis, University of Montpellier II, Montpellier, France, 1992.

[9] R. A. Harshman, *Foundations of the Parafac procedure: Models and conditions for an explanatory multimodal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.

[10] R. A. Harshman, *Determination and proof of minimum uniqueness conditions for PARAFAC*1, UCLA Working Papers in Phonetics, 22 (1972), pp. 111–117.

[11] R. A. Harshman, *The Problem and Nature of Degenerate Solutions or Decomposition of* 3*-way Arrays*, 2004. Discussion presented at the American Institute of Mathematics Tensor Decompsition Workshop, Palo Alto, CA. Slides available online at http://publish.uwo.ca/~harshman.

[12] R. A. Harshman, *A Note on Several Different Kinds of PARAFAC Degeneracy and Other Ill-Conditioning*, 2007, unpublished.

[13] R. A. Harshman and S. Hong, *"Stretch" versus "slice" methods for representing three-way structure via matrix notation*, J. Chemom., 16 (2002), pp. 198–205.

[14] R. A. Harshman and M. E. Lundy, *The PARAFAC model for three-way factor analysis and multidimensional scaling*, in Research Methods for Multimode Data Analysis, H. G. Law, C. W. Snyder, J. Hattie, and R. P. McDonald, eds., Praeger, New York, 1984, pp. 122–215, also available online at http://publish.uwo.ca/~harshman/lawch5.pdf.

[15] R. A. Harshman and M. E. Lundy, *Conditions governing full and partial Parafac uniqueness when two loading matrices have full column rank*, submitted, 2007.

[16] J. B. Kruskal, *Three-way arrays: Rank and uniqueness of trilinear decompositions with applications to arithmetic complexity and statistics*, Linear Algebra Appl., 18 (1977), pp. 95–138.

[17] J. B. Kruskal, *Rank, decomposition, and uniqueness for* 3*-way and n-way arrays*, in Multiway Data Analysis, R. Coppi and S. Bolasco, eds., Elsevier Science, North-Holland, 1989, pp. 7–18.

[18] L. De Lathauwer, *Signal Processing Based on Multilinear Algebra*, Ph.D. thesis, K. U. Leuven, E. E. Department -ESAT, Belgium, 1997.

[19] L. De Lathauwer, *A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 642–666.

[20] L. De Lathauwer, B. De Moor, and J. Vandewalle, *Computation of the canonical decomposition by means of simultaneous generalized Schur decomposition*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 295–327.
[21] S. E. Leurgans, R. T. Ross, and R. B. Abel, *A decomposition for three-way arrays*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 1064–1083.
[22] P. Paatero, *The multilinear engine—A table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model*, J. Comput. Graph. Statist., 8 (1999), pp. 854–888.
[23] W. S. Rayens and B. C. Mitchell, *Two-factor degeneracies and a stabilization of PARAFAC*, Chemom. Intell. Lab. Syst., 38 (1997), pp. 173–181.
[24] R. T. Ross and S. Leurgans, *Component resolution using multilinear models*, Methods Enzymol., 246 (1995), pp. 679–700.
[25] E. Sanchez and B. R. Kowalski, *Tensorial resolution: A direct trilinear decomposition*, J. Chemom., 4 (1990), pp. 29–45.
[26] N. D. Sidiropoulos and R. Bro, *On the uniqueness of multilinear decomposition of n-way arrays*, J. Chemom., 14 (2000), pp. 229–239.
[27] A. Smilde, R. Bro, and P. Geladi, *Multi-way Analysis with Applications in the Chemical Sciences*, Wiley, Chichester, UK, 2004.
[28] G. Tomasi, *Practical and Computational Aspects in Chemometric Data Analysis*, Ph.D. thesis, Royal Veterinary and Agricultural University, Frederiksberg, Denmark, 2006.