Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Computational Statistics & Data Analysis 50 (2006) 1700–1734
COMPUTATIONAL STATISTICS & DATA ANALYSIS
www.elsevier.com/locate/csda

A comparison of algorithms for fitting the PARAFAC model

Giorgio Tomasi*, Rasmus Bro

Food Science, Royal Veterinary and Agricultural University, Rolighedsvej 30, DK-1958 Frederiksberg C, Denmark

Received 24 November 2004; accepted 26 November 2004
Available online 22 December 2004

Abstract

A multitude of algorithms have been developed to fit a trilinear PARAFAC model to a three-way array. Limits and advantages of some of the available methods (i.e. GRAM-DTLD, PARAFAC-ALS, ASD, SWATLD, PMF3 and dGN) are compared. The algorithms are explained in general terms together with two approaches to accelerate them: line search and compression. In order to compare the different methods, 720 sets of artificial data were generated with varying level and type of noise, collinearity of the factors and rank. Two PARAFAC models were fitted on each data set: the first having the correct number of factors F and the second with $F + 1$ components (the objective being to assess the sensitivity of the different approaches to the over-factoring problem, i.e. when the number of extracted components exceeds the rank of the array). The algorithms have also been tested on two real data sets of fluorescence measurements, again by extracting both the right and an exceeding number of factors. The evaluations are based on: number of iterations necessary to reach convergence, time consumption, quality of the solution and amount of resources required for the calculations (primarily memory).

Keywords: PARAFAC; ALS; SWATLD; PMF3; Levenberg–Marquadt

* Corresponding author. Tel.: +45 35 28 32 96; fax: +45 55 28 32 45.
E-mail addresses: gt@kvl.dk (G. Tomasi), rb@kvl.dk (R. Bro).

0167-9473/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2004.11.013

1. Introduction

The PARAFAC (PARallel FACtor analysis) model was introduced in 1970 by Harshman (1970) and simultaneously by Carroll and Chang (1970) under the name CANDECOMP. For a three-way data array $\underline{\mathbf{X}}$, the PARAFAC model is defined as

$$x_{ijk} = \sum_{f=1}^{F} a_{if} b_{jf} c_{kf} + r_{ijk}, \quad i = 1 \cdots I, \ j = 1 \cdots J, \ k = 1 \cdots K, \tag{1}$$

where x_{ijk} is the measured value, a_{if}, b_{jf}, and c_{kf} represent the parameters to estimate, r_{ijk} are the residuals and F is the number of factors extracted.

In general terms, fitting model (1) boils down to minimising (usually in a least squares sense) the fitting error made. This means finding the parameters $a_{11}, a_{12}, \ldots, c_{KF}$ that minimise the loss function:

$$L(a_{11}, a_{12}, \ldots, c_{KF}) = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} \left(x_{ijk} - \sum_{f=1}^{F} a_{if} b_{jf} c_{kf} \right)^2. \tag{2}$$

Several algorithms for solving such problems are described in the literature (Faber et al., 2003; Hayashi and Hayashi, 1982; Paatero, 1997; Tomasi and Bro, 2004). Some of those tested in this work (DTLD, ASD and SWATLD) do not minimise (2) and in fact their loss functions are not strictly well-defined (Faber et al., 2003).

While the PARAFAC model remains the same, the different methods of fitting it to a three-way array $\underline{\mathbf{X}}$ can be better explained introducing distinct notations. The parameters can be gathered in three loading matrices $\mathbf{A}$ (also referred to as scores matrix), $\mathbf{B}$ and $\mathbf{C}$ defined as

$$\mathbf{A} = \left\{ a_{if} \middle| \, i = 1 \cdots I, \, f = 1 \cdots F \right\} = [\, \mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_F \,], \tag{3a}$$

$$\mathbf{B} = \left\{ b_{jf} \middle| \, j = 1 \cdots J, \, f = 1 \cdots F \right\} = [\, \mathbf{b}_1 \quad \mathbf{b}_2 \quad \cdots \quad \mathbf{b}_F \,] \tag{3b}$$

and

$$\mathbf{C} = \left\{ c_{kf} \middle| \, k = 1 \cdots K, \, f = 1 \cdots F \right\} = [\, \mathbf{c}_1 \quad \mathbf{c}_2 \quad \cdots \quad \mathbf{c}_F \,], \tag{3c}$$

where $\mathbf{a}_f$, $\mathbf{b}_f$ and $\mathbf{c}_f$ denote the columns of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ respectively.

Using the matricised format for multi-way arrays (Bro, 1998) and with the introduction of the column-wise Khatri–Rao product $\odot$ (Rao and Mitra, 1971) (see Appendix C), the model can be written as

$$\mathbf{X}^{(I \times JK)} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})^{\mathrm{T}} + \mathbf{R}^{(I \times JK)}, \tag{4}$$

where "T" means the transpose and $(I \times JK)$ refers to the way the multi-way array is matricised (K slabs of size $I \times J$ are put one beside the other forming a matrix of dimension $I \times JK$).

Another possibility is to express the model in a slab-wise form as:

$$\mathbf{X}_{..k} = \mathbf{A}\mathbf{D}_k\mathbf{B}^{\mathrm{T}} + \mathbf{R}_{..k}, \tag{5}$$

where $\mathbf{D}_k$ is a diagonal matrix containing the k-th row of $\mathbf{C}$, $\mathbf{X}_{..k}$ is the $I \times J$ matrix representing the k-th frontal slab and $\mathbf{R}_{..k}$ is the corresponding matrix of the residuals. Eqs. (4) and (5) have to be trivially modified in case other matricisations (e.g. $J \times IK$) or slabs (e.g. the horizontal slab $\mathbf{X}_{i..}$ of size $K \times J$) are required (Bro, 1998).

The algorithms fitting the PARAFAC model can be classified in three main groups: alternating algorithms, which update only a subset of the parameters at each step; derivative-based methods, seeking an update for all the parameters simultaneously by successive approximations; and direct (non-iterative) procedures. To the first group belong PARAFAC-ALS (Alternating Least Squares (Harshman, 1970)), ASD (Alternating Slice-wise Diagonalisation (Jiang et al., 2000)) and SWATLD (Self Weighted Alternating TriLinear Decomposition (Chen et al., 2000)). The second set is represented by PMF3 (Positive Matrix Factorisation for 3-way arrays (Paatero, 1997)) and dGN (damped Gauss–Newton, also known as Levenberg–Marquadt (Paatero, 1997; Tomasi and Bro, 2004)). As for the third grouping of algorithms, the most known implementations are the Generalised Rank Annihilation Method (GRAM (Sanchez and Kowalski, 1986)) and the Direct TriLinear Decomposition method (DTLD), both based on a generalized eigenvalue problem. All these algorithms will only be described in general terms as more details are available in the original papers.

Fitting the PARAFAC model presents numerous problems. One aspect affecting speed of convergence and retrieval of the underlying solutions is the condition number of the loading matrices, which reflects both collinearity and relative magnitude of the factors (Hopke et al., 1998; Kiers, 1998). Compression, which is outlined in Section 2.5, has been reported to be beneficial in this respect and has the added advantage of reducing the computational expense (Bro and Andersson, 1998; Kiers, 1998). Another problem is the so-called two factor degeneracy (2FD), i.e. the presence in the solution of two factors that are almost perfectly collinear but have opposite signs and almost cancel out each others' contribution. The essential problem with 2FDs is that the degenerate factors can grow arbitrarily large while the loss function continuously (and very slowly) decreases. 2FDs may appear in the final solution as a consequence of e.g. a wrong estimation of the rank of the array, or some aspects specific to the data set at hand, (Kruskal et al., 1989; Paatero, 2000). A connection has been established between some of the possible causes of 2FDs in the final solution and the so-called swamps (Mitchell and Burdick, 1994; Rayens and Mitchell, 1997), which are sequences of iterations where the interim solutions contain features similar to 2FDs of increasing severity and the loss function decreases very slowly. When an (iterative) algorithm encounters a swamp, it either emerges from it (i.e. after an unpredictably large number of iterations, the loss function starts decreasing more rapidly again and the 2FD slowly disappears) or it reaches an earlier stop because one of the convergence criteria is suddenly met. Great attention has been given in the development of new methods capable of dealing effectively with swamps and more in general with the more frequent high collinearity case. Two examples are regularisation (Paatero, 1997; Rayens and Mitchell, 1997) and line search (Bro, 1998; Harshman, 1970) (described in Section 2.4).

2. Algorithms

2.1. DTLD

Ho et al. (1978, 1980, 1981) developed an algorithm called RAFA (Rank Annihilation Factor Analysis) for estimating the concentration of a chemical analyte in an unknown sample-matrix solely using the measurements of the unknown sample and of a pure standard. This property was coined the second-order advantage, as it is obtained by using the second-order or two-way structure of the individual sample measurements instead of vectorising the corresponding matrix. The second-order advantage is in essence equivalent to the uniqueness of the trilinear structure. The idea behind RAFA was based on reducing the rank of the calibration sample by subtracting the contribution from the analyte of interest. That is, if the signal from the analyte of interest is subtracted from the sample data, then the rank of this matrix will decrease by one as the contribution of the analyte of interest is one in case of ordinary bilinear data such as chromatographic or fluorescence data. This was intuitively appealing, but the method itself was somewhat unsatisfactory and slow. Later, Lorber (1985) showed that it was possible to automate this search for rank-reduction and extended the method. This new automated method was called Generalized Rank Annihilation Method (GRAM) and works with two samples. Using only two samples, there is a direct solution to the PARAFAC model based on solving a generalized eigenvalue problem. Despite the fact that this solution is not a least squares solution, it has been found to work well for data that are well approximated by a PARAFAC model. From a GRAM solution for any two samples, the loading matrices in the two variable modes are obtained (as well as the two scores for the sample mode). For more than two samples, the two loading matrices obtained from any two samples can be used for calculating the score-values for all samples in a simple regression step. This is the principle behind DTLD (Sanchez and Kowalski, 1990), whereby there are alternative strategies for selecting the two samples used in the GRAM step. Thus, DTLD is a direct method using a generalized eigenvalue problem of fixed known complexity. The required storage is represented in essence by the sole data array $\underline{\mathbf{X}}$, the three loading matrices and three matrices (two with the same size of the loading matrices and one having only two columns) used to compress the array and to select the two samples used for GRAM step.

2.2. Alternating Least Squares

Alternating least squares is a relatively old method (its principles were introduced in 1933 by Yates (1933)) and it is based on the idea of reducing the optimisation problem to smaller sub-problems that are solved iteratively. The parameters to be determined are separated in different groups and, by fixing all of the groups but one, a new loss function depending only on the set left free to vary is minimised. The solution of this (linear) least squares problem is known and relatively simple to calculate. The subsequent stages of the algorithm consist of applying the same principle on the other groups of parameters. The algorithm iterates, alternating from one set to the next, until the variation of the loss function or of the parameters is less than a predefined convergence criterion. Since all the steps are

optimisations in the least squares sense, the loss function is bound not to increase at any step and tends asymptotically to a minimum.

2.2.1. PARAFAC-ALS (ALS)

In the PARAFAC case (Carroll and Chang, 1970; Harshman, 1970) and having a three-way array $\underline{\mathbf{X}}$, there are three sets of parameters ($\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$) and the loss function (2), using (4), can be written as

$$L(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \left\| \mathbf{X}^{(I \times JK)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^{\mathrm{T}} \right\|_{\mathrm{F}}^{2}, \tag{6}$$

where $\|\bullet\|_{\mathrm{F}}$ denotes the Frobenius norm. While calculating all the three loading matrices that minimise L at the same time is a rather complicated non-linear problem (and it is treated as such in other algorithms), if initial approximations of $\mathbf{B}$ and $\mathbf{C}$ (respectively $\mathbf{B}^{(0)}$ and $\mathbf{C}^{(0)}$) are available, the interim optimal $\mathbf{A}$ can be easily calculated as

$$\mathbf{A}^{(1)} = \mathbf{X}^{(I \times JK)} \left(\left(\mathbf{C}^{(0)} \odot \mathbf{B}^{(0)} \right)^{+} \right)^{\mathrm{T}}, \tag{7a}$$

where '+' denotes the Moore–Penrose inverse. $\mathbf{B}^{(1)}$ and $\mathbf{C}^{(1)}$ are determined in a similar fashion. That is

$$\mathbf{B}^{(1)} = \mathbf{X}^{(J \times IK)} \left(\left(\mathbf{C}^{(0)} \odot \mathbf{B}^{(1)} \right)^{+} \right)^{\mathrm{T}} \tag{7b}$$

and

$$\mathbf{C}^{(1)} = \mathbf{X}^{(K \times IJ)} \left(\left(\mathbf{B}^{(1)} \odot \mathbf{A}^{(1)} \right)^{+} \right)^{\mathrm{T}}. \tag{7c}$$

The three steps in Eq. (7) are repeated until a predefined convergence criterion is met.

Eq. (7) shows the general approach of PARAFAC-ALS. In the actual implementation of the algorithm (Jiang et al., 1999), certain properties of the Khatri–Rao product and line search (Section 2.4) are employed in order to accelerate the calculations (Bro, 1998).

This algorithm has several advantages: it is easy to implement, guaranteed to converge and simple to extend to higher order arrays. The shortcomings are mainly in the occasional slowness of the convergence process in presence of swamps (Mitchell and Burdick, 1994; Paatero, 2000; Rayens and Mitchell, 1997) or high collinearity (Kiers, 1998). Furthermore, the loss function decreases almost linearly with the iterations while other methods can provide, at least in principle, superlinear or even quadratic convergence rate. It is also worth mentioning, even if it is not relevant for the purposes of this work, that several types of constraints can be imposed to the loading vectors in a relatively straightforward way (Bro, 1998).

The memory consumption for PARAFAC-ALS is limited; in essence: two arrays (one for $\underline{\mathbf{X}}$ and one for the model/residuals) of size $I \times J \times K$, a matrix of size $\max(IJ, JK, IK) \times F$ for the Khatri–Rao products and the three loading matrices.

2.2.2. SWATLD and ASD

Recently, several algorithms (ASD (Jiang et al., 2000), SWATLD (Chen et al., 2000) ATLD (Wu et al., 1998), PALS (Chen et al., 2001), ACOMAR (Li et al., 2000), ACOVER (Jiang et al., 1999)) based on ideas similar to PARAFAC-ALS have been proposed. A comparison between these methods and standard ALS already exists in the literature (Faber et al., 2003). Therefore, based also on some preliminary tests, only SWATLD and ASD have been chosen to represent this group of new algorithms.

As mentioned by its authors, SWATLD does not attempt to find the minimum of (6), instead it alternates between minimising three different (non-least squares) loss functions, one per each of the loading matrices. Such loss functions are structured in the following way (shown here with respect to **C**):

$$L_{\mathbf{C}}(\mathbf{C}) = \sum_{k=1}^{K} \left\| \left(\mathbf{A}^{+}\mathbf{X}_k - \mathbf{D}_k\mathbf{B}^{\mathrm{T}}\right)^{\mathrm{T}} \mathbf{D}_{\mathbf{B}}^{-1} \right\|_{\mathrm{F}}^{2} + \sum_{k=1}^{K} \left\| \left(\mathbf{X}_k\left(\mathbf{B}^{+}\right)^{\mathrm{T}} - \mathbf{A}\mathbf{D}_k\right) \mathbf{D}_{\mathbf{A}}^{-1} \right\|_{\mathrm{F}}^{2}, \quad (8)$$

where $\mathbf{D}_{\mathbf{A}}$ and $\mathbf{D}_{\mathbf{B}}$ are $F \times F$ diagonal matrices holding the norm of the loading vectors of respectively **A** and **B**, $\mathbf{X}_{..k}$ is the k-th frontal slab of $\underline{\mathbf{X}}$ and $\mathbf{D}_k$ is an $F \times F$ diagonal matrix with the elements of the k-th row of **C** on the diagonal. Thus, given **A** and **B**, a new estimate for **C** can be found as (cf. Appendix B):

$$\mathbf{C} = 0.5 \cdot \mathbf{X}^{(K\times IJ)} \left[\left(\mathbf{B}^{+\mathrm{T}} \odot \mathbf{A}\mathbf{D}_{\mathbf{A}}^{-2}\right) + \left(\mathbf{B}\mathbf{D}_{\mathbf{B}}^{-2} \odot \mathbf{A}^{+\mathrm{T}}\right) \right]. \quad (9)$$

The update evidently depends on the values of $\mathbf{D}_{\mathbf{A}}$ and $\mathbf{D}_{\mathbf{B}}$ and thus on the scaling convention applied to the interim loading matrices. Hence, the loading matrices were scaled so that $\mathbf{D}_{\mathbf{A}} = \mathbf{D}_{\mathbf{B}}$ (cf. Appendix A). Equivalent equations are developed mutatis mutandis for the updates of **A** and **B**. It is apparent that the solution is not found in the least squares sense as $0.5 \cdot \left[\left(\mathbf{B}^{+\mathrm{T}} \odot \mathbf{A}\mathbf{D}_{\mathbf{A}}^{-2}\right) + \left(\mathbf{B}\mathbf{D}_{\mathbf{B}}^{-2} \odot \mathbf{A}^{+\mathrm{T}}\right) \right] \neq (\mathbf{B} \odot \mathbf{A})^{+\mathrm{T}}$. In fact, SWATLD typically yields solutions that are less affected by deviations from exact trilinearity than the least squares ones (see the experimental part), which seems to bear some advantages in terms of convergence speed and resistance to over-factoring. However, since its optimisation criterion is not well-defined, this algorithm must be used with care.

One important remark about SWATLD regards the three matrices **A**, **B** and **C**, which must have full column rank in order for the algorithm to resolve uniquely the components of interest. In other words, the maximum number of components that can be extracted is upper-bounded by any of the dimensions of $\underline{\mathbf{X}}$. Such conditions for uniqueness are stricter than the theoretical ones (Sidiropoulos and Bro, 2000).

With regard to memory consumption, SWATLD is approximately as demanding as ALS.

ASD is based on the hypothesis of **A** and **B** having full column rank (hence also stricter than the theoretical conditions for uniqueness (Sidiropoulos and Bro, 2000)) and thus on the existence of two matrices **P** and **Q** for which it holds $\mathbf{P}^{\mathrm{T}}\mathbf{A} = \mathbf{I}_F$ and $\mathbf{B}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}_F$, where **I** denotes the identity matrix of appropriate size.

Eq. (5) can then be written as

$$\mathbf{P}^{\mathrm{T}}\mathbf{X}_{..k}\mathbf{Q} = \mathbf{D}_k + \mathbf{P}^{\mathrm{T}}\mathbf{R}_{..k}\mathbf{Q} \equiv \mathbf{D}_k + \tilde{\mathbf{R}}_{..k}. \quad (10)$$

Loss function (6) is minimised by ASD using an alternating algorithm of the form

$$L_{\mathrm{ASD}}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{P},\mathbf{Q}) = \sum_{k=1}^{K} \left\| \mathbf{P}^{\mathrm{T}}\mathbf{X}_{..k}\mathbf{Q} - \mathbf{D}_k \right\|_{\mathrm{F}}^2 + \lambda \left(\left\| \mathbf{P}^{\mathrm{T}}\mathbf{A} - \mathbf{I}_F \right\|_{\mathrm{F}}^2 + \left\| \mathbf{B}^{\mathrm{T}}\mathbf{Q} - \mathbf{I}_F \right\|_{\mathrm{F}}^2 \right), \tag{11}$$

where λ is a predefined constant (Faber et al., 2003; Jiang et al., 2000).

It is apparent that $\mathrm{L}_{\mathrm{ASD}}$ is a function of $\tilde{\mathbf{R}}_{..k}$, not of the residuals, and that it contains a penalty term that does not necessarily decrease to a negligible magnitude. Consequently, as it is the case for SWATLD, the final solution is not providing a least squares fit and must be assessed with care.

A final observation is that ASD employs a compression procedure based on singular value decomposition that allows a significant reduction in the number of operations per single iteration (Faber et al., 2003; Jiang et al., 2000). This compression step further reduces the memory consumption of the algorithm, which requires, beside the original array and the loading matrices, an array $\underline{\tilde{\mathbf{R}}}$ of size $F \times F \times K$, four matrices of size $F \times F$ and 2 matrices holding the bases for the compression and having size $I \times F$ and $J \times F$.

2.3. Gauss–Newton method

As mentioned in Section 2.2.1, minimising (6) is a relatively difficult non-linear least squares problem: several authors (Hayashi and Hayashi, 1982; Paatero, 1997; Tomasi and Bro, 2004) have proposed the use of algorithms usually employed for such optimisation problems. All these algorithms are based on the Gauss–Newton method (Björck, 1996). However, in order to deal with some of the peculiarities of the PARAFAC model it is necessary to introduce some modifications to the basic algorithm.

Defining the model values as

$$y_{ijk}(\mathbf{A},\mathbf{B},\mathbf{C}) = \sum_{f=1}^{F} a_{if} b_{jf} c_{kf}, \tag{12}$$

loss function (6) can be written as

$$L(\mathbf{A},\mathbf{B},\mathbf{C}) = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} \left(x_{ijk} - y_{ijk}(\mathbf{A},\mathbf{B},\mathbf{C}) \right)^2 = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} r_{ijk}^2(\mathbf{A},\mathbf{B},\mathbf{C}), \tag{13}$$

where $r_{ijk}(\mathbf{A},\mathbf{B},\mathbf{C})$ are the residuals. By stringing out $\underline{\mathbf{X}}$ and $\underline{\mathbf{Y}}$ in two vectors $\mathbf{x} = \mathrm{vec}\,\mathbf{X}^{(I\times JK)}$ and $\mathbf{y} = \mathrm{vec}\left(\mathbf{A}(\mathbf{C}\odot\mathbf{B})^{\mathrm{T}}\right)$ of length $M = IJK$ (the vec operator is defined as in Magnus and Neudecker (1999)) and considering a vector $\mathbf{p} = \mathrm{vec}\left[\mathbf{A}^{\mathrm{T}} \middle| \mathbf{B}^{\mathrm{T}} \middle| \mathbf{C}^{\mathrm{T}}\right]$ of length $N = (I + J + K)F$ with all the sought parameters, loss function (13) can be

expressed as

$$L(\mathbf{p}) = \sum_{m=1}^{M} (x_m - y_m(\mathbf{p}))^2 = \sum_{m=1}^{M} r_m^2(\mathbf{p}) = \mathbf{r}(\mathbf{p})^{\mathrm{T}}\mathbf{r}(\mathbf{p}), \tag{14}$$

where $m = IJ(k-1) + I(j-1) + i$ and $\mathbf{r} = [\, r_1 \quad \cdots \quad r_M \,]^{\mathrm{T}}$.

The Gauss–Newton method works under the assumption that the residuals in the neighbourhood of a point $\mathbf{p}^0$ can be approximated by a Taylor expansion truncated after the linear term:

$$\begin{aligned} r_m(\mathbf{p}) &= r_m\left(\mathbf{p}^0\right) + \sum_{n=1}^{N} \frac{\partial r_m}{\partial p_n}\left(p_n - p_n^0\right) + \mathrm{O}\left(\left\|\mathbf{p} - \mathbf{p}^0\right\|_2^2\right) \\ &\cong r_m\left(\mathbf{p}^0\right) - \sum_{n=1}^{N} \frac{\partial y_m}{\partial p_n}\left(p_n - p_n^0\right) \equiv \tilde{r}_m(\mathbf{p}), \quad m = 1, \ldots, M. \end{aligned} \tag{15}$$

Thus, by defining an $M \times N$ matrix $\mathbf{J}\left(\mathbf{p}^0\right)$, the Jacobian, having as elements j_{mn}

$$j_{mn} = \frac{\partial r_m\left(\mathbf{p}^0\right)}{\partial p_n} = -\frac{\partial y_m\left(\mathbf{p}^0\right)}{\partial p_n}$$

and if the linear approximation (15) suffices, (14) can be expressed as a function of $\Delta\mathbf{p} = \mathbf{p} - \mathbf{p}^0$:

$$\tilde{L}(\Delta\mathbf{p}) = \tilde{\mathbf{r}}(\Delta\mathbf{p})^{\mathrm{T}}\tilde{\mathbf{r}}(\Delta\mathbf{p}) = \left\|\mathbf{r}\left(\mathbf{p}^0\right) + \mathbf{J}\left(\mathbf{p}^0\right)\Delta\mathbf{p}\right\|_2^2. \tag{16}$$

A new approximation of the parameter-vector is then calculated as $\mathbf{p}^{(s+1)} = \mathbf{p}^s + \Delta\mathbf{p}^{(s)}$, where the update $\Delta\mathbf{p}^{(s)}$ for the s-th iteration is computed as a solution to the linear problem

$$\min_{\Delta\mathbf{p}} \left\|\mathbf{r}\left(\mathbf{p}^{(s)}\right) + \mathbf{J}\left(\mathbf{p}^{(s)}\right)\Delta\mathbf{p}^{(s)}\right\|_2^2. \tag{17}$$

This can be done by solving for $\Delta\mathbf{p}^{(s)}$ the system of normal equations:

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}\right)\Delta\mathbf{p}^{(s)} = -\mathbf{J}^{\mathrm{T}}\mathbf{r} \equiv \mathbf{g}, \tag{18}$$

where $\mathbf{g}$ is the gradient of $\tilde{L}(\Delta\mathbf{p})$. At this point, convergence is checked and if the loss function has not decreased in relative terms for less than a predefined convergence criterion or the number of iterations has not reached the maximum, the algorithm computes a new Jacobian and a new update $\Delta\mathbf{p}$ and keeps on iterating. Note that, as the minimum of a function is a stationary point, the algorithm should stop if $\|\mathbf{g}\|_\infty$ is smaller than a sufficiently small number (Madsen et al., 2004).

This iterative method for fitting the PARAFAC model (i.e. the Gauss–Newton algorithm) corresponds to the one proposed by Hayashi and Hayashi (1982).

$\tilde{\mathbf{H}} = \mathbf{J}^{\mathrm{T}}\mathbf{J}$ is in fact an approximation of the Hessian matrix $\mathbf{H}$, which holds the second derivatives of the loss function (Madsen et al., 2004). The smaller the residuals r_m are, the better is the approximation. This implies that if the elements of $\mathbf{r}$ are small enough, the update $\Delta\mathbf{p}^{(s)}$ is close to the one calculated with the Newton method (requiring $\mathbf{H}$), which guarantees, under the same conditions, quadratic convergence (Madsen et al., 2004). On the other hand, if the behaviour of L is far from linear (e.g. QS the number of modes of the array increases) or the elements of r_m are relatively large, the approximation is not appropriate and the Newton method should be preferred (Madsen et al., 2004). Line search procedures and trust region methods (Björck, 1996; Madsen et al., 2004) can be used to deal with this problem without requiring the full calculation of the Hessian, as outlined in the ensuing sections.

The very sparse structure of $\mathbf{J}$ in the PARAFAC case (Paatero, 1997) induces some observations regarding whether or not it is advantageous to exploit it in solving (17) without using the system of normal equations. By partitioning the Jacobian as $[\mathbf{J_A}|\,\mathbf{J_{BC}}]$ where $\mathbf{J_A}$ and $\mathbf{J_{BC}}$ refer to the parameters in $\mathbf{A}$ and in $\mathbf{B}$ and $\mathbf{C}$ respectively and by a simple row permutation, it is apparent that problem (17) has a block angular form (i.e. $\mathbf{J_A}$ is block diagonal, each block being $JK \times F$). A substantial reduction in the number of operations could be attained by using a QR algorithm specifically designed for this type of problems (Björck, 1996). Even greater savings may be achieved considering that the I blocks in $\mathbf{J_A}$ are identical. It is important to notice, though, that such savings are significant if one of the dimensions of $\underline{\mathbf{X}}$ is much larger than the others (Björck, 1996; Paatero, 1997). The fill relative to $\mathbf{J_{BC}}$ as the QR decomposition progresses, makes the use of sparse QR function impractical in most of the other cases: after $\mathbf{J_A}$ part has been treated, $\mathbf{J_{BC}}$ is full, sparse storage is consequently less effective and bears no advantage in terms of number of operations. For this reason, the artificial data sets employed in the experimental part are not designed to test this aspect and the implementation of a block angular QR algorithm specific for PARAFAC is left for future developments. Hence, problem (17) is solved by means of the system of normal equations (19) using the Matlab (www.themathworks.com) built-in (full) Cholesky decomposition and back-substitution. Since $\mathbf{J}$ is typically a very thin and tall matrix, this solution is particularly advantageous. Furthermore, if no weights are required, the Jacobian $\mathbf{J}$ need not be formed and both $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ and $\mathbf{J}^{\mathrm{T}}\mathbf{r}$ can be computed directly without recurring to sparse matrices (Liu and Sidiropoulos, 2001), whose creation and update are rather slow.

2.3.1. dGN

The algorithm as outlined in the previous section is not sufficient to yield a globally convergent algorithm for fitting the PARAFAC model, which is intrinsically indeterminate with respect to factor scaling (cf. Appendix A). This phenomenon leads to rank deficiency of the Jacobian, which has always at least $2F$ zero singular values, and consequently to singularity of $\tilde{\mathbf{H}}$, a problem that can be successfully handled by the damped Gauss Newton (dGN) algorithm devised by Levenberg and Marquadt (Björck, 1996; Levenberg, 1944; Marquadt, 1963). In dGN, the update $\Delta\mathbf{p}^{(s)}$ is calculated from the modified normal equations:

$$\left(\tilde{\mathbf{H}} + \lambda^{(s)}\mathbf{I}_N\right)\Delta\mathbf{p}^{(s)} = -\mathbf{g}, \tag{19}$$

which is equivalent to solving problem (17) under the constraint:

$$\left\|\Delta\mathbf{p}^{(s)}\right\|_2^2 \leqslant \delta\left(\lambda^{(s)}\right). \tag{20}$$

That is to say, the minimum is sought within a region (the trust region) of radius $\delta\left(\lambda^{(s)}\right)$ (Madsen et al., 2004). If the damping parameter $\lambda^{(s)}$ is large enough compared to the singular values of $\tilde{\mathbf{H}}$, the matrix $\left(\tilde{\mathbf{H}}+\lambda\mathbf{I}_{\mathrm{N}}\right)$ is non-singular and system (19) can be efficiently solved (Björck, 1996; Madsen et al., 2004).

The updating strategy for $\lambda^{(s)}$ is crucial for the efficiency of the algorithm. The scheme used in the implemented dGN algorithm is thoroughly described in Madsen et al. (2004) and is based on the ratio between the actual variation of the loss function (ΔL) and the linear decrease $\Delta\tilde{L}=\tilde{L}(\mathbf{0})-\tilde{L}(\Delta\mathbf{p})$. E.g., if this ratio is less than 0 (which, if $\Delta\mathbf{p}^{(s)}$ is a descent direction, implies $\Delta L<0$), the step is rejected and $\lambda^{(s)}$ is increased. The idea behind such updating scheme is that if the ratio is small, the linear approximation does not hold and thus the region must be shrunk. Vice versa, if the ratio is very large the linear model holds and the region can be enlarged. This strategy allows dealing effectively with non-linearities and, as can be seen from Eq. (20), the damping parameter can determine the shortening or the elongation of the step by acting on the trust region size. Thus, it produces results similar to those expected by a line search procedure, which consequently was not implemented in the dGN algorithm.

The intrinsic scaling indeterminacy poses another problem that may affect the numerical stability of the algorithm: if the conventional scaling used for the PARAFAC model is applied $\left(\text{i.e. } \|\mathbf{b}_f\|=\|\mathbf{c}_f\|=1\right)$, the values of the norms of the columns of $\mathbf{J}$ may differ by several orders of magnitude (typically $a_{if}b_{jf}\cong a_{if}c_{kf}\gg b_{jf}c_{kf}$) depending on the data. Such problem is reflected in the practical condition number of $\mathbf{J}$ (i.e. computed disregarding the last $2F$ zero singular values (Tomasi and Bro, 2004)) and thus in the accuracy of the solution of the system of normal equations and can be avoided by employing a different scaling convention (see Appendix A), whereby the different loading vectors for the same factor are given the same norm $\left(\text{i.e. } \|\mathbf{a}_f\|=\|\mathbf{b}_f\|=\|\mathbf{c}_f\|\right)$. The rescaling is performed at each step as it represents a small additional overhead compared to other steps of the algorithm.

With specific reference to memory consumption, dGN is the most expensive method together with PMF3 (cf. the next section). It requires, besides the original array $\underline{\mathbf{X}}$, three vectors of size $M\times 1$, $N\times 1$ and $N\times 1$ for respectively $\mathbf{r}$, $\mathbf{g}$ and $\mathbf{p}$, and two $NF\times NF$ matrices, one for $\tilde{\mathbf{H}}$ and one for the Cholesky factor of $\left(\tilde{\mathbf{H}}+\lambda\mathbf{I}_N\right)$. Both the matrices are almost full and are treated as such (Paatero, 1997).

2.3.2. PMF3

Paatero (1997) proposed several modifications to dGN in the PMF3 algorithm: the presence of a regularisation factor and a very specific non-linear update. Besides, PMF3 employs a line search procedure that is called whenever the algorithm diverges. In its original form, PMF3 also includes a weighted least squares loss function and possible non-negativity constraints on the parameters, but these are of no concern in this work. Since they can both be

safely removed from PMF3, they will be ignored in the brief description given here as well as in the simulations.

The loss function of PMF3 differs from (13) by the presence of a regularisation term:

$$L_{\mathrm{PMF3}}(\mathbf{p}) = \sum_{m=1}^{M} r_m^2(\mathbf{p}) + \gamma \sum_{n=1}^{N} \widehat{p}_n^{\,2} = \mathbf{r}(\mathbf{p})^{\mathrm{T}}\mathbf{r}(\mathbf{p}) + \gamma \widehat{\mathbf{p}}^{\mathrm{T}}\widehat{\mathbf{p}}\,, \tag{21}$$

where $\widehat{p}_n = \left(p_n - p_n^{\bullet}\right)$ and $p_n^{\bullet}$ is the target value (0 in this case) for the p_n parameter. γ is a coefficient that varies depending on the problem at hand. Eq. (21) leads to the following modified system normal equations (see Appendix B):

$$\tilde{\tilde{\mathbf{H}}}\Delta\mathbf{p}'^{(s)} \equiv \left(\tilde{\mathbf{H}} + \left(\lambda^{(s)} + \gamma^{(s)}\right)\mathbf{I}_N\right)\Delta\mathbf{p}'^{(s)} = -\mathbf{J}^{\mathrm{T}}\mathbf{r} + \gamma^{(s)}\widehat{\mathbf{p}}^{(s)}, \tag{22}$$

which is again solved for $\Delta\mathbf{p}'^{(s)}$ (the apex is to distinguish this update from the non-linear one described below).

As $\mathbf{p}^{\bullet} = \mathbf{0}$, the regularisation term is a scalar γ times the norm of the vector $\mathbf{p}$, and thus penalises high absolute values for the parameters. The goal of this procedure is to correct the loss function for the scaling indeterminacy as well as non-identifiability (Paatero, 1997). Paatero also suggests equating the influence of the different slabs in the penalty term by using different γ_s for each estimated parameter when there are essential scale differences between the slabs of array $\mathbf{X}$. The scaling convention illustrated in Section 2.3.1 attains a similar result in the sense that the three modes have exactly the same weight in the penalty, thus one γ was deemed sufficient.

The update strategy for γ followed the scheme suggested in Paatero (1997): γ is decreased by a factor γ_{update} when the loss function decreased (in relative terms) by less than a certain threshold for at least γ_{it} consecutive iterations. No indications are given in Paatero (1997) about the magnitude of γ (which is problem dependent) or the spacing of the thresholds. An initial value for γ that seemed reasonable and that heuristically appeared to be good for the analysed problems was 1. By similar means, five thresholds were set so that their base 10 logarithms were equally spaced between -1 and the $\log_{10}$ of five times the convergence criterion in terms of relative fit (e.g. -5.3 for a 10^{-6} criterion).

The non-linear update $\Delta\mathbf{p}''$ is computed by solving the system:

$$\tilde{\tilde{\mathbf{H}}}\Delta\mathbf{p}''^{(s)} = -\mathbf{J}\left(\mathbf{p}'\right)^{\mathrm{T}}\mathbf{r}\left(\mathbf{p}'\right) + \gamma^{(s)}\widehat{\mathbf{p}}^{(s)}, \tag{23}$$

where $\mathbf{p}' = \mathbf{p}^{(s)} + 0.5\Delta\mathbf{p}'^{(s)}$. The final update is chosen as the one between $\Delta\mathbf{p}'^{(s)}$ and $\Delta\mathbf{p}''^{(s)}$ that provides the largest reduction in the loss function. The main reason for this rather complicated procedure seems to be avoiding to compute the full Hessian (which may not be positive definite (Madsen et al., 2004)) and still obtain an update that may be a better approximation of the Newton direction, which again is more appropriate than the dGN step in case of e.g. large residuals (Madsen et al., 2004). Furthermore, the Cholesky factor of $\tilde{\tilde{\mathbf{H}}}$ need not be recomputed to solve (23), which allows for some savings in the number of operations, and $\mathbf{J}\left(\mathbf{p}'\right)^{\mathrm{T}}\mathbf{r}\left(\mathbf{p}'\right)$ can be efficiently determined without having to compute $\mathbf{J}$ (Liu and Sidiropoulos, 2001). However, no reference or reason are given for the 0.5 coefficient to obtain $\mathbf{p}'$.

The actual implementation of PMF3 employs a soft line search procedure (described in Section 2.4) for both the linear and the non-linear update whenever the algorithm diverges. In the original algorithm, should the line search procedure fail to retrieve a step-length yielding a reduction of L_{PMF3}, a special step optimising one parameter at a time would follow. The strategy followed in the implementation of PMF3 used for the tests is slightly different and if 10 iterations of the line search fail, the step is rejected and the damping parameter is increased using the standard dGN procedure.

The memory requirements of PMF3 are the same as for the dGN algorithm.

2.4. Line search

Line search procedures are applicable for all iterative methods with a well-defined loss function. The reasons and the implementation vary depending on the algorithm. In PARAFAC-ALS, line search is employed solely to speed up the algorithm. After the s-th iteration, the variation of the three loading matrices with respect to iteration $s-1$ is calculated $\left(\text{e.g. } \Delta\mathbf{A} = \mathbf{A}^{(s)} - \mathbf{A}^{(s-1)}\right)$ and is used to linearly predict the corresponding matrix d iterations ahead $\left(\text{e.g. } \mathbf{A}^{(s+d)} = \mathbf{A}^{(s)} + d\Delta\mathbf{A}\right)$, where d is determined empirically as a function of the number of iterations (Bro, 1998).

In the case of derivative based methods, this procedure copes with non-linearities of the loss function and with high residuals (Madsen et al., 2004). The basic idea is to consider the update $\Delta\mathbf{p}^{(s)}$ as a direction along which a step of length $\alpha^{(s)}$ is taken; the step length can be found as a solution to the univariate minimisation problem

$$\min_{\alpha} \tilde{L}(\mathbf{p} + \alpha\Delta\mathbf{p}) = \min_{\alpha} L'(\alpha). \tag{24}$$

The search for the solution minimiser α^*, which need not be determined with great accuracy (Madsen et al., 2004), is performed in the implemented PMF3 algorithm by a simple procedure interpolating a quadratic model on at least three values of $L'(\alpha)$ calculated at $\alpha = 0$, 0.5 and 1. If these three initial points bracket a minimum there are sufficient conditions for α^* to be estimated; otherwise new points are added until a minimum is bracketed.

2.5. Compression

The main problem with derivative based methods is that, even though they typically require fewer iterations than ALS, the number of operations per step and the memory requirements are significantly higher, up to the point of being inapplicable for larger arrays (Paatero, 1997). A possible solution is to reduce the dimensions of the array on which the Gauss–Newton method is used. The compression based on the Tucker3 model fulfills this task (Bro and Andersson, 1998; Kiers, 1998). Although it introduces additional complexity in the algorithm, this part eventually provides a large reduction in the number of operations. Furthermore, it needs only be done once, whereas several alternative PARAFAC models are usually fitted to find the most feasible one. Once the PARAFAC model has been calculated on the core extracted by the Tucker3 algorithm, the solution can be expanded to its original dimensions providing very good starting values for the PARAFAC-ALS standard algorithm, which is used only to refine it.

The compression of the array $\underline{\mathbf{X}}$ is based on the search for three truncated bases spanning the variation in each mode and the projection of $\underline{\mathbf{X}}$ on them. Calling these bases $\mathbf{U}$ $(I \times L)$, $\mathbf{V}$ $(J \times M)$ and $\mathbf{Z}$ $(K \times N)$ for the first, the second and the third mode respectively, the loadings of the PARAFAC model can be expressed as

$$\mathbf{A} = \mathbf{UP}, \tag{25a}$$

$$\mathbf{B} = \mathbf{VQ} \tag{25b}$$

and

$$\mathbf{C} = \mathbf{ZR}, \tag{25c}$$

where $\mathbf{P}$ is a $L \times F$ matrix, $\mathbf{Q}$ is a $M \times F$ and $\mathbf{R}$ is $N \times F$. The CANDELINC theorem (Carroll et al., 1980) states that, if a PARAFAC model is sought under the linear constraints expressed in (25a–c), it is only necessary to estimate $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$, which can be done by fitting a PARAFAC model to the array $\underline{\mathbf{G}}$ calculated as

$$\mathbf{G}^{(L\times MN)} = \mathbf{U}^{\mathrm{T}}\mathbf{X}^{(I\times JK)}(\mathbf{Z} \otimes \mathbf{V}). \tag{26}$$

Typically $\underline{\mathbf{G}}$ is much smaller than $\underline{\mathbf{X}}$ due to the redundancy of information in the latter. The three bases and $\underline{\mathbf{G}}$ are found by applying the Tucker3 model, which is both fast and efficient, to $\underline{\mathbf{X}}$. The $\underline{\mathbf{G}}$ array is the core produced by Tucker3 while the bases are the three loading matrices.

For this application, it is not necessary to calculate an exact Tucker3 model because the compressed array is only an approximation of the original one and the final model would need to be refined anyhow. Hence, several simplifications are utilised to minimise computational cost of this step: first of all the number of iterations for Tucker3 is limited both by fixing a relatively small maximum number of allowed iterations and by increasing the convergence criterion. Tucker3 is initialised by using orthogonal matrices of random numbers. In order to balance the lower fit deriving from the low number of iterations, the number of components extracted in each mode (that is the number of vectors forming the truncated bases) is slightly increased; it has been observed empirically that a reasonable choice is equal to the number of factors sought by PARAFAC plus 2 (of course upper limited by the dimensions of the array) (Andersson and Bro, 2000; Bro, 1998).

With respect to memory consumption, the expressions given in Sections 2.2.1 and 2.3 are applicable here, with the sole difference that the array upon which the algorithm is applied has dimensions $(F + 2) \times (F + 2) \times (F + 2)$.

3. Experimental part

Nine algorithms are studied in this section using both simulated and real data: DTLD-GRAM, PARAFAC-ALS, ASD, SWATLD, dGN, PMF3, PARAFAC-ALS with compression, dGN with compression and PMF3 with compression. For space reasons the single implementations will not be discussed in detail; in general, the MATLAB 6.5 (www.themathworks.com) guidelines for improving performances were followed (e.g. maximising the use of built-in functions). DTLD and PARAFAC-ALS are part of the N-Way

Table 1
Design factors and levels for the artificial data sets

Rank		3,5
Noise[a]	Homoscedastic	1,5,10
	Heteroscedastic	0,1,5
Congruence		0.5,0.9

[a]Percent of $\left\|\mathbf{X}^{(I\times JK)}\right\|_{\mathrm{F}}^{2}$ (i.e. % of total variation).

Table 2
Condition number for **A**, **B** and **C** in the artificial data sets

	Congruence		
		0.5	0.9
Rank	3	2	5.29
	5	2.44	6.78

toolbox (Andersson and Bro, 2000). ASD was implemented according to Faber et al. (2003). The other m-files are available for download at www.models.kvl.dk.

3.1. Simulated data

Simulated data sets have the advantage that important features can be controlled so that their effect on the different methods may be assessed. Hence, a Monte Carlo study has been carried out based on 720 arrays of dimension $20\times20\times20$ considering the following aspects: rank of the systematic part of the multi-way array (F), congruence of the true components (see Appendix C) and amount and type of noise (Table 1). For each combination of these conditions 20 replicates were generated to counterpoise minor statistical fluctuations. The final number of data sets amounted to 2 ranks $\times3$ levels of homoscedastic noise $\times3$ levels of heteroscedastic noise $\times2$ congruences $\times20$ replicates, which is again equal to 720.

As to provide the proper collinearity, the components were generated according to the following scheme, applied to all the required loading matrices (Kiers et al., 1999):

- The Cholesky factor $\mathbf{R}$ is calculated of an $F\times F$ (where F is the number of factors) matrix having ones on the diagonal and the desired values of congruence between the loading vectors as off-diagonal elements.
- A column-wise orthogonal matrix $\mathbf{U}$ of size $I\times F$ (where I here represents the dimension in the mode the loading matrix refers to) is generated.
- The loading matrix is then defined as $\mathbf{L}=\mathbf{UR}$.

One thing to notice is that for all the data sets having the same number of underlying components and equal congruence, the condition number of the loadings matrices is the same (see Table 2). In particular, according to Kiers (1998), for congruence 0.5 the level

of collinearity is relatively low while for congruence 0.9 the underlying factors are only mildly collinear. With the employed method, higher values for the condition number are only attainable by increasing the congruence among the factors to an extremely high value (viz. to yield a condition number of 11, the average for severely collinear set in Kiers (1998), the congruence has to be 0.9956) or by reducing the numerical relevance of one or more of the factors. The first option was ruled out because such high values of congruence are seldom found in practice, while the second option was avoided to limit the number of design factors in the simulation.

Homoscedastic and heteroscedastic noise were generated from random identically (and normally) distributed numbers and two distinct arrays were added to the noise-less array $\underline{\mathbf{X}}$: $\underline{\mathbf{R}}_{\mathrm{ho}}$ for the homoscedastic part and $\underline{\mathbf{R}}_{\mathrm{he}}$ for the heteroscedastic one. In order to make the latter proportional, each element of $\underline{\mathbf{R}}_{\mathrm{he}}$ was multiplied by the corresponding element in $\underline{\mathbf{X}}$. Both $\underline{\mathbf{R}}_{\mathrm{ho}}$ and $\underline{\mathbf{R}}_{\mathrm{he}}$ were also normalised to a Frobenius norm of $\left\|\mathbf{X}^{(I\times JK)}\right\|_{\mathrm{F}}$ and multiplied by two suitable scalars depending on the desired level of noise (see Appendix D).

Finally, for each experimental set up two models were fitted, extracting F or $F+1$ factors. The aim was to acquire information on the stability of the methods when the wrong number of components is extracted. A total of 1440 models were fitted using each algorithm.

3.2. Real data sets

The algorithms were tested on two real data sets both constituted by fluorescence measurements:

1. Twenty-two solutions containing different amounts of four compounds (DOPA, hydroquinone, tryptophan and phenylalanine) were analysed on a Perkin-Elmer LS50 B spectrofluorometer. The excitation wavelengths ranged between 245 and 305 nm, with a step of 5 nm while in the emission mode the interval was: 260–390 nm with a step of 1 nm (Baunsgaard et al., 2000). The Rayleigh scatter was removed by subtracting from each sample a model for the scatter.
   Three replicates from each sample were then generated by excluding every third wavelength in the emission mode, i.e replicate one the wavelengths 260, 263 nm, etc. were removed, in replicate two the wavelengths 261, 264 nm, etc. were kept out while in the third the left out wavelengths started at 262 nm again with 3 nm increments. Hence, the data set consisted of three different arrays having dimensions $22 \times 87 \times 13$, for the first two replicates, and $22 \times 88 \times 13$ for the last.
2. Fifteen solutions containing DOPA, hydroquinone, tyrosine and tryptophan in different amounts were examined using a Cary Eclipse spectrofluorometer. The excitation wavelengths ranged between 230 and 300 nm measured at intervals of 5 nm, the emission was measured at 282–412 nm with 2 nm steps and the scatter was removed by subtracting a blank from each sample. Six replicates of each solution were analysed leading to six different arrays of dimensions $15 \times 66 \times 15$.

In both data sets, the different replicates have been modelled separately to give different realisations of the same model. As for the simulated data, the effect of over-factoring has been

Table 3
Condition numbers for the loading matrices and congruences between factors

	Condition number			Congruence (min − max)		
	A	B	C	A	B	C
Data set I	2.83	5.96	20.44	0.31–0.49	0.002–0.88	0.15–0.94
Data set II	8.99	12.24	8.07	0.46–0.51	0.15–0.86	0.54–0.94

Table 4
Tested algorithms and relative parameters for the simulated data

Algorithm[a]	Convergence criterion	Value	Max *n*. iterations	Other parameters
DTLD	—	—	—	—
ALS[b], ALSc, SWA	Relative fit	10^{-6}	10000	—
ASD	Relative fit	10^{-6}	10000	$\lambda = 10^{-3}$
dGN, dGNc	Relative fit	10^{-6}	1000	$\lambda^{(0)} = \max\left(\text{diag}\left(\tilde{\mathbf{H}}^{(0)}\right)\right)$[c]
	Gradient	10^{-9}		
PMF3, PMF3c	Relative fit	10^{-6}	1000	$\lambda^{(0)} = \max\left(\text{diag}\left(\tilde{\mathbf{H}}^{(0)}\right)\right)$[c]
	Gradient	10^{-9}		$\gamma = 1, \gamma_{\text{update}} = 5, \gamma_{\text{it}} = 3$

[a]DTLD: DTLD-GRAM; ALS: PARAFAC-ALS; SWA: SWATLD; ALSc: PARAFAC-ALS with compression; dGNc: damped Gauss–Newton with compression; PMF3c: PFM3 with compression.
[b]Covers also the refining step for ALSc, dGNc and PMF3c.
[c]$\text{diag}(\mathbf{H}) = [\,h_{11} \quad h_{22} \quad \cdots \quad h_{NN}\,]^{\mathrm{T}}$ for an $N \times N$ matrix.

analysed by fitting both 4 (the expected rank, represented by the number of constituents) and 5 components.

Table 3 shows congruence and condition numbers of the estimated loading matrices when 4 components are extracted; as it can be seen, both data sets are mildly collinear according to Kiers (1998). In fact, data set II is more problematic then data set I if the evaluation is based on the Jacobian's practical condition number (Tomasi and Bro, 2004), hence likely giving rise to greater numerical problems.

3.3. Initialisation and conventions

Initial estimations for the parameters were necessary for all the algorithms but DTLD-GRAM and were obtained by running five standard PARAFAC-ALS iterations (the 'relative fit decrease' convergence criterion set to 10^{-4}) starting with ten different sets of matrices of uniformly distributed values and then choosing the best fitting set to continue the process with the main algorithm.

The settings for the different tuning parameters, namely convergence criteria, λs and γs, are shown in Tables 4and 5 for respectively the artificial and the real data. The different choices were due to the number of models that needed to be calculated. Hence, for the real data sets it was possible to make the convergence criteria stricter without making the

Table 5
Modified convergence criteria for the real data sets

	Convergence criterion		Iterations
ALS[a], SWA, ASD, ALSc	Relative fit	10^{-8}	20000
dGN, PMF3,	Relative fit	10^{-8}	1000
dGNc, PMF3c	Gradient	10^{-10}	

For dGN, PMF3 and ALs the criteria are applied also when the method is used on the compressed array.
[a]Covers also to the refining step for ALSc, dGNc and PMF3c.

calculation time intolerably long.

With respect to the scaling indeterminacy the components were scaled and given a sign according to the convention described in Appendix A: the loading vectors belonging to the same component were scaled so that their norm was the same and the sign of the loading vectors in **B** and **C** was chosen so that the sign of the sum of the elements for each factor was positive.

3.4. Criteria of interest

The initial assessment regarded the quality of the solution. In this respect, four different parameters have been considered: value of the loss function, occurrence of full recoveries (defined below) and of 2FDs and mean squared error (MSE) of the parameter estimates. For the real data sets, the concentration of the different constituents was available and the Root Mean Squared Error in Calibration (RMSEC—cf. Appendix D) relative to predicting the concentrations from the first mode loadings (scores) using linear regression was used as an additional evaluation parameter.

Due to the large number of models, a graphical comparison and control was not feasible. Therefore, the quality of the factor estimates was assessed numerically in terms of congruence between the known underlying factors and the extracted factors (Mitchell and Burdick, 2004; Tomasi and Bro, 2004). Specifically, one can consider a factor as completely recovered if there is one component in the estimated model having a congruence greater than a certain 'recovery threshold' (close to 1). Congruence tends to be quite an optimistic method and a limit lower than 0.99 for the single loading vector may not be good enough for automated procedures. Besides, the aim of this threshold was to provide a common comparison ground for the different algorithms and thus a value of 0.99, which ensures that the full recovery occurred, was preferred. For the single component the threshold was thus set to $0.99^3 \simeq 0.97$ (Tomasi and Bro, 2004).

Due to the permutation indeterminacy (Bro, 1998), the order of the factors in the three loading matrices is not determined beforehand, but comes as a consequence of the initial estimations provided to the algorithm. All the possible permutations of extracted factors (up to F) were compared with the correct loading matrices and the final choice (the 'winning permutation', represented by the permutation matrix **P**) was the one associated to the highest sum of the congruences for all the original factors (Mitchell and Burdick, 1994). Finally, full recovery occurs if all the F congruences in the latter permutation were greater than the

recovery threshold.

The occurrence of a 2FD is established when the congruence between two components within the same solution is less than or equal to −0.8.

The MSE can be calculated separately for each of the loading matrices and is an indication of the distance between the extracted factors and the actual ones. For **A**, MSE can be calculated as

$$\mathrm{MSE}\left(\mathbf{A}, \hat{\mathbf{A}}, \mathbf{P}, \mathbf{S_A}\right) = \frac{\left\|\mathbf{A} - \hat{\mathbf{A}}\mathbf{P}\mathbf{S_A}\right\|_{\mathrm{F}}^2}{IF}, \tag{27}$$

where **A** refers to the underlying factors and has been adequately scaled (see Appendix A), $\hat{\mathbf{A}}$ is the extracted $I \times F$ (or $I \times F+1$) loading matrix, and $\mathbf{S_A}$ is an $F \times F$ scaling (diagonal) matrix solving the problem (Riu and Bro, 2003):

$$\begin{aligned} &\min_{\mathbf{S_A},\mathbf{S_B},\mathbf{S_C}} L\left(\mathbf{A}, \mathbf{B}, \mathbf{C}, \hat{\mathbf{A}}, \hat{\mathbf{B}}, \hat{\mathbf{C}}, \mathbf{P}, \mathbf{S_A}, \mathbf{S_B}, \mathbf{S_C}\right) \\ &\quad = \left\|\mathbf{A} - \hat{\mathbf{A}}\mathbf{P}\mathbf{S_A}\right\|_{\mathrm{F}} + \left\|\mathbf{B} - \hat{\mathbf{B}}\mathbf{P}\mathbf{S_B}\right\|_{\mathrm{F}} + \left\|\mathbf{C} - \hat{\mathbf{C}}\mathbf{P}\mathbf{S_C}\right\|_{\mathrm{F}}, \\ &\text{subject to } \mathbf{S_A}\mathbf{S_B}\mathbf{S_C} = \mathbf{I}_F \end{aligned} \tag{28}$$

In case of over-factoring, **P** is represented by the first F columns of the $F+1 \times F+1$ winning permutation matrix (thus **P** is $F \times F$ when F factors are extracted and $F+1 \times F$, if a $F+1$ components model is fitted). It has been observed that, if the variation captured by each factor is equally spread over the three corresponding loading vectors, the MSEs for the different loading matrices are similar in magnitude. For simplicity the MSE is presented as an average among the MSEs of **A**, **B** and **C**.

The second main aspect considered in the tests was the computational efficiency, which is relatively complex to define. In general, one may say that the best method is the one that produces the solution using as few floating point operations (FLOPs) as possible. Nevertheless, with the advent of vector machines and the implementation of block algorithms, this parameter is inadequate to give a clear picture; other aspects such as memory traffic and data access are in fact equally important and very machine dependent (Golub and Van Loan, 1996). As mentioned, the algorithms were implemented as to yield (to the authors' knowledge) optimal performances in terms of time consumption under MatLab 6.5 and used only built-in functions. All the tests were performed on the same dedicated machine (mounting a 2.6 GHz Pentium IV processor, 512 MB of memory and running on WindowsXP) and the attention was concentrated on the time consumption and the number of iterations necessary to reach convergence.

3.5. Results and discussion

3.5.1. Simulated data

The first aspect to be considered was recovery capability (factor congruence). Overall, all the iterative algorithms managed to retrieve the correct solution (with a congruence threshold of 0.97) in more than 50% of the cases (Table 6). The best algorithm in this respect is SWATLD, with 63.6% of correct solutions, followed by damped Gauss–Newton

Table 6
Percentage of full recoveries (threshold 0.97) for the different algorithms on the simulated data sets

Rank	3				5				Global
Congruence	0.5		0.9		0.5		0.9		
Extr. Factors	3	4	3	4	5	6	5	6	
DTLD	89	68	8	7	23	13	0	0	26.1
ASD	100	79	19	12	99	87	4	1	50.1
SWA	100	100	46	40	99	99	12	12	63.6
ALS	100	59	51	36	99	70	12	11	54.7
dGN	100	82	52	41	99	86	13	11	60.5
PMF3	100	91	52	40	99	91	13	13	62.4
ALSc	100	64	51	39	99	76	12	12	56.8
dGNc	100	94	50	42	99	92	12	12	62.7
PMF3c	100	90	49	42	99	91	12	13	62.0

All the noise levels have are considered. The last column (Global) reports the percentage of full recoveries for each algorithm over the 1440 models of the Monte Carlo simulations.

with compression (62.7%) and PMF3 (62.4%). DTLD, the only direct method, attained the correct solution in only 26.1% of the cases and yielded the worst performance, followed by ASD with 50.1%. ALS falls in the lower part of the range with 54.7% full recoveries.

While DTLD did consistently worse than the other algorithms in all setups, the performances of ALS were equivalent to those of the derivative based algorithms and SWATLD when the rank was correctly estimated (Table 6). Conversely, ALS was heavily affected by an incorrect estimation of the rank and this sensitivity was reduced, but not eliminated, by compression. It can be seen from both Table 6 and Fig. 1 that the most robust method in this respect is SWATLD, whose performance is remarkably good: when the rank was overestimated, SWATLD performed equally or better than when rank was correctly set in 30 out of the 36 setups while for example ALS did worse in 21 setups in case of over-factoring. Also the derivative based methods seem considerably more robust than PARAFAC-ALS with respect to over-factoring. This is true also when the algorithms are applied together with compression, even though ALS' performance improves considerably.

The higher resistance of the derivative based methods to over-factoring is likely to be associated to the regularisation terms (both the damping parameter λ and the PMF3 regularisation term). A link has been established between rank of the array, rank of the model, occurrence of 2FDs (Krijnen, 1993; Kruskal et al., 1989; Paatero, 2000) and swamps, which are characterised by 2FDs and may lead to a premature stop of an algorithm. Regularisation may help in dealing with 2FDs (Paatero, 2000; Rayens and Mitchell, 1997) and thus it seems reasonable that it operates favourably when the rank is overestimated. 2FDs (with a congruence threshold of -0.8) could be observed considerably more often when the rank is overestimated (e.g. 55 cases against 4 for ALS for overestimated and correct rank and 125 against 15 for dGN). The sole exception in this sense was SWATLD, which did not yield any 2FD in any of the models, regardless of over-factoring. This may help explaining the efficiency of the latter method with respect to over-factoring, and that avoiding 2FDs during the early stages of the fitting procedure may significantly help retrieving the correct

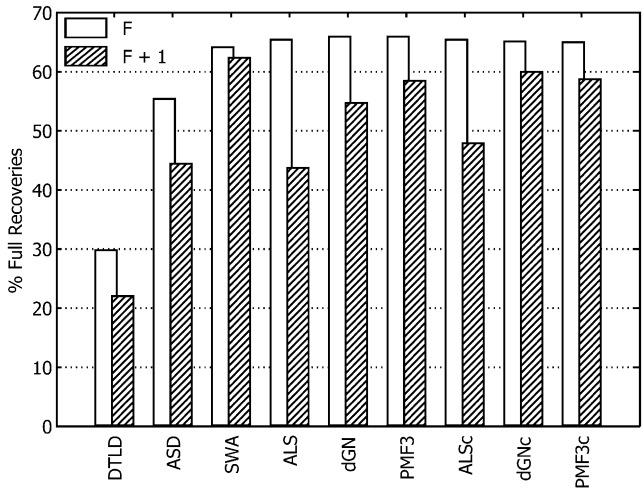

Fig. 1. Recovery (threshold 0.97) with correct number of factors (F) and in case of over-factoring ($F + 1$). All the conditions and replicates have been considered.

solution. The SWATLD solutions typically have a higher core consistency (Bro and Kiers, 2003) than the corresponding least squares solution (not shown), i.e. they seem to be biased towards describing only the strictly trilinear contributions and are likely less affected by possible interactions between factors that may emerge during convergence (e.g. the fact that part of the variation is associated to a factor is described by another). Nonetheless, it is difficult to establish what makes SWATLD so resistant to 2FDs and over-factoring. The fact that the algorithm uses three different loss functions and that its actual optimisation criterion is not known makes any conclusion in this direction rather inappropriate. Further analyses will be necessary to understand these reasons and will be the subject for future research. With respect to 2FDs, ALS yielded fewer in the final solution than any other method except SWATLD (4.1% of all models). The worst algorithm was ASD with 11.0% and this clearly contrasts with the claim of the original authors on the capability of this algorithm of avoiding 2FDs (Jiang et al., 2000). At the same time, PMF3 yields slightly fewer 2FDs than dGN (8.7% and 8.8% respectively—without compression) and compression increases their occurrence in all the methods to which it was applied (6.6% for ALSc, 9.7% for dGNc and 9.0% for PMF3c). The number of 2FDs for the least squares algorithms seems to follow the performances in terms of recovery capability, which suggests that the number of degeneracies is simply the consequence of the higher efficiency of some methods. Setting a smaller threshold to establish degeneracy (viz. -0.5) more or less equalises the number of 2FDs for ALS and the derivative based methods to approximately 20% of all models and seems to confirm that 2FDs become generally worse for dGN and PMF3. The small difference between dGN and PMF3 can be explained as an effect of regularisation.

The effect of the compression step on ALS, dGN and PMF3 is rather different. Particularly, the recovery capability of ALS and dGN visibly improves in case of over-factoring (plus 4.2% and 5.3% respectively when all ranks and degrees of collinearity are consid-

ered) remaining approximately the same (unchanged for ALS and −0.8% for dGN) when the rank is correctly set. On the contrary, PMF3 recovery capability remains largely unaltered (−1% and 0.3% respectively for correct tank and over-factoring). While the beneficial effects of compression may be explained in terms of removal of noise and spurious information, the different behaviour of PMF3 compared to dGN is not clearly understood yet. There are several differences between the two algorithms. In particular, dGN implements a more refined damping parameter's updating scheme (Madsen et al., 2004) than PMF3 (Paatero, 1997), whereas the second employs soft line search and a regularisation factor that operates in a similar fashion as the damping parameter, possibly supplementing to the shortcomings of its simple trust region implementation. In fact, preliminary additional results show that using the λ updating scheme of dGN for PMF3 yields results similar to those of dGN, with a reduction in the number of full recoveries for PMF3 and better results for PMF3c.

Collinearity between the underlying factors is the feature that most affects the number of full recoveries. For both ranks and when the number of components of the array is correctly set, a significant drop in the performance can be observed for all algorithms as collinearity increases (Table 6 and (Kiers, 1998)). It is particularly relevant that DTLD and ASD, which explicitly assume linear independence of loading vectors, are the most affected by collinearity. SWATLD, which works under similar basic assumptions, is affected to a much lesser extent and although visibly inferior to the least squares methods when the correct number of components is fit, it still performs slightly better than e.g. ALS in case of over-factoring. Again, the non-least squares criterion and the variety of ad-hoc solutions of this method make it difficult to explain this additional stability.

With respect to noise levels, the different algorithms (again with the exception of DTLD and ASD) are affected in a similar way (Fig. 2). An increase in the level of noise always leads to a worsening in the recovery capability when the rank is correctly set. This is partly due to lack of convergence to a meaningful solution, but also to the inclusion of some noise in the components that makes their congruence with the original factors smaller than the threshold (Kiers, 1998). Heteroscedastic noise has a greater effect than homoscedastic noise and considering the setups where the total amount of noise is the same (viz. $1-5/5-1$ and $5-5/10-0$), the recovery capability is worse when heteroscedastic noise is larger, dropping from 65% to 70% to approximately 50%. Conversely, an improvement could be observed for low or intermediate homoscedastic noise as heteroscedastic noise increased and the model rank exceeded the correct one. This was found true in the low collinearity case for all algorithms apart from DTLD. The improvement is consistent over the amount of heteroscedastic noise and is quite dramatic for ALS and ALSc, with an increase from 30% of full recoveries at 1% homoscedastic and no heteroscedastic noise to 90% (80% without compression) at 5% heteroscedastic noise and 95% (with and without compression) when both types of noise were fixed at 5%. In the high congruence case, this behaviour could be observed at 1% homoscedastic and heteroscedastic noise, in which case the extraction of an additional component brought to an improvement between 5% (for dGN and PMF3) and 12.5% (ALS) in the occurrence of full recoveries. A possible explanation is that part of the noise is retained in the additional component and that a certain degree of systematic behaviour in the noise facilitates its extraction. At the same time, the additional factor prevents noise from altering the loadings, which explains the increased number of full

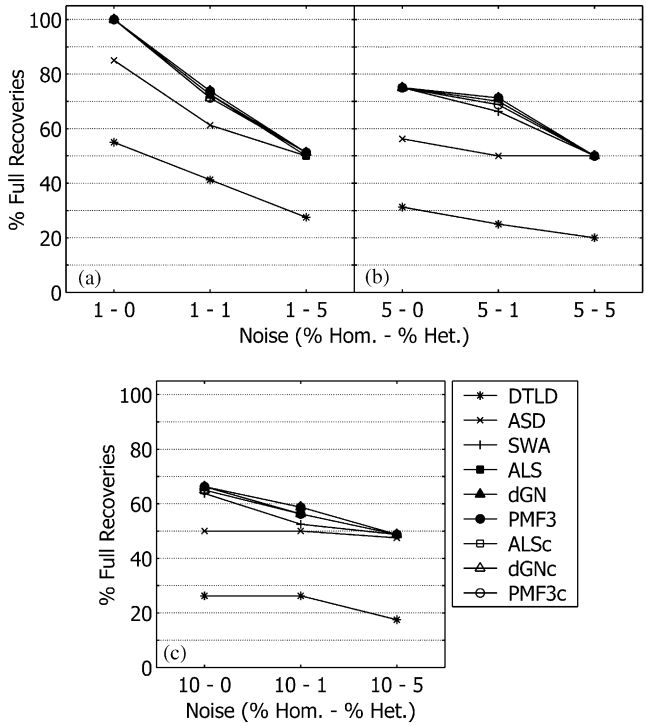

Fig. 2. Effect of noise on recovery capability (threshold 0.97) when the correct number of factors is extracted. All congruences and ranks are considered.

recoveries.

The second aspect on the quality of the solution to be analysed was how close the calculated solution was to the real underlying factors. Fig. 3 shows the median MSE for the case of correct model rank and low collinearity. The solution provided by the two non-least squares iterative algorithms (i.e. ASD and SWATLD) are noisier than the others. The use of the median penalises DTLD in the rank 5 case since this method does not yield the correct solution in at least 50% of cases. Nevertheless, it can still be seen that for rank 3 data sets characterised by low collinearity, when the performances in terms of recovery are sufficiently good, the MSE is still higher, albeit not by much, than for the least squares methods.

Similar conclusions can be drawn by studying the value of the loss function, or, more precisely, the occurrence of values of the loss function (6) that differ by less than 10^{-4} in relative terms from the minimum yielded by any of the algorithms (results not shown). Only in a negligible number of cases (0.4%, all of them in the low rank, low collinearity case with over-factoring) SWATLD reached the same minimum as the least squares methods. This is surprising considering the higher recovery capability of this algorithm, but confirms that this method does not find the least squares solution but rather one that is biased towards a higher core consistency. Analogous observations can be made for ASD, which never

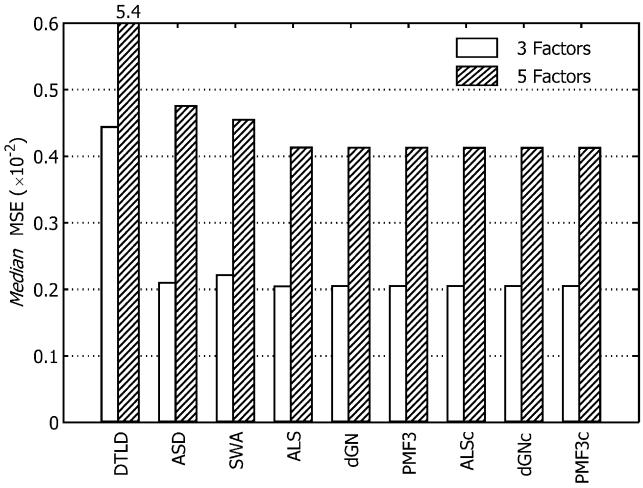

Fig. 3. Median MSE for the parameters estimates when the correct number of components is fitted. Only the low congruence case is depicted in the figure.

yields the lowest value of a least squares loss function. It is interesting to notice how the use of compression improves the results of all the algorithms in this respect. The most significant improvement is that of PARAFAC-ALS (plus 9.5% of attained minima), but also the two derivative methods (plus 8.5% and 6.0% respectively for dGN and PMF3) perform appreciably better.

The final assessment of the algorithms regarded their convergence rate and the computational costs expressed in terms of number of iterations and time consumption respectively. Although the number of iterations for the different methods varies significantly, particularly between alternating methods and derivative based ones (Paatero, 1997; Tomasi and Bro, 2004), it is still possible to assess the effect of the various design factors on convergence rate, the effect of compression on the various algorithms, and whether PMF3 (and correspondingly PMF3c) converges more rapidly than the standard dGN method (and dGNc).

All the considered design factors have an influence on the number of iterations in fitting, which increases with rank, congruence and over-factoring (Table 7), confirming previous results (Hopke et al., 1998; Kiers, 1998). With respect to noise, though, the behaviour is different and all the methods apart from SWATLD and PMF3 use fewer iterations when the rank is overestimated in at least 50% of the cases as heteroscedastic noise increase. The reason for this may be again that especially the heteroscedastic noise simplifies the fitting of an additional component holding the systematic part of the noise.

When using as initial values those found by fitting a model on the compressed space (Table 7), the number of PARAFAC-ALS full data iterations (FDIs (Kiers, 1998)) in 93–100% of the cases approximately 21.5% (median over all conditions). In particular, 2 FDIs were sufficient to refine the solution in 17.6% of the cases for PMF3c and in 19% for LMc, all but one for correctly estimated rank. The number of FDIs does not vary much depending on the algorithm used in fitting, and is identical to the minimum attained with any method

Table 7
Median number of iterations. All levels of noise are considered in each cell

Rank	3				5			
Congruence	0.5		0.9		0.5		0.9	
Extr. Factors	3	4	3	4	5	6	5	6
ASD	8	25	83	85	8	19	99	122
SWA	7	41	35	55	8	38	72	96
ALS	10	48	60	80	12	46	91	110
dGN	6	70	16	54	7	54	53	91
PMF3	7	63	17	51	7	55	47	85
ALSc	10 (2)	82 (8)	97 (5)	156 (16)	12 (2)	70 (10)	195 (28)	243 (34)
dGNc	7 (2)	75 (10)	17 (5)	61 (16)	7 (2)	67 (12)	54 (26)	87 (33)
PMF3c	8 (2)	51 (10)	18 (5)	50 (16)	8 (3)	47 (12)	47 (26)	78 (34)

The number in parenthesis for ALSc, dGNc and PMF3c indicates the number of PARAFAC-ALS iterations in the refinement step.

in 75.8% of the cases for PMF3c, 77.6% for dGNc and 79.7% for ALSc (not shown). This is particularly true for dGNc and PMF3c: the maximum difference in the number of FDIs for the refinement step is less than or equal to ten in 95.3% of the cases. With respect to the various design factors, the refinement step is affected by the same features as the PARAFAC-ALS algorithm, namely rank, congruence, noise and overfactoring, which determine an increase in the number of iterations necessary to attain convergence (results not shown).

The number of compressed data iterations (CDIs (Kiers, 1998)) for ALSc, dGNc and PMF3c depends on the varied conditions and is normally larger than the number of FDIs for the same algorithm (Table 7). However, whereas for ALSc the number of CDIs increases (or remains the same) in 89% of the cases compared to ALS, the same holds for the two derivative-based algorithms only when the rank is correctly set (84% and 81% for dGNc and PMF3c respectively). The opposite is true instead for PMF3, which requires fewer CDIs in case of over-factoring in more than one case out of two (i.e. 54.2%).

The fastest iterative methods were clearly ASD and SWATLD (Fig. 4). They required less time than any other iterative algorithm in 44.2% and 41.7% of all models respectively (not shown). ALS (13.6%) and dGN (0.5%) accounted for the remaining cases. As pointed out in Section 2.2.2, ASD employs compression by means of singular value decomposition, which explains the good performances of this method. On the other hand, as for the recovery capability, ASD is heavily affected by the collinearity of the underlying components and performs worse than SWATLD in more than one case out of two (Fig. 4) of the models with high congruence between the underlying factors. The 196 models (13.6%) for which ALS is the fastest method are more or less equally scattered between the various conditions, with a slight dominance for the high congruence cases (122 compared to 74).

In terms of time expense, ALSc was found slower than ALS 94.4% of the times (Table 8). The reason for this is that the cheaper iterations deriving from compression are counteracted for ALS by their increased number. On the contrary, the advantage of compression is apparent for the two derivative-based methods, which (without compression) are clearly the

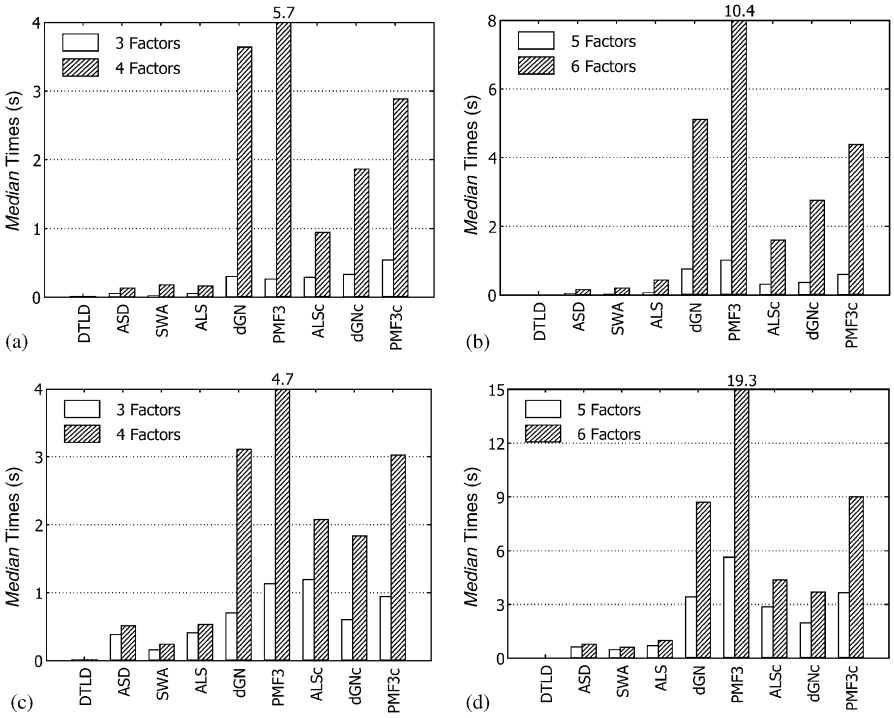

Fig. 4. Median time consumption. a,b) low congruence, c,d) high congruence case. All noise levels are considered.

most expensive ones (Fig. 4). Since compression makes the fitting step independent of the dimensions of $\underline{\mathbf{X}}$, the bad performances of ALSc compared to ALS are likely caused by the small dimensions of the $\underline{\mathbf{X}}$ array used in the tests (Bro and Andersson, 1998; Kiers, 1998). However, ALSc is also slower than dGNc for 55% of the models and is only faster than dGN, PMF3, and PMF3c (Table 8). The advantage of using dGNc is particularly evident for the high congruence case (Fig. 4). Nevertheless, compression heightens the dependence of time expense on the rank of the fitted model and, for larger values of model's rank, ALSc might be faster than the derivative based methods owing to the longer time required by the Cholesky decomposition.

Finally, with respect to PMF3 and dGN, the latter required as many or fewer iterations in 59.1% of the cases when employed without compression and 55.5% if PMF3c and dGNc are considered (not shown). The ratio becomes even less favourable to PMF3 when the model's rank is known (82.5% and 81.1% for PMF3 and PMF3c respectively), but is inverted in case of over-factoring (29.7% and 35.7%). However, since the PMF3 iterations are more expensive, this algorithm was found to be slower in 94.1% of the cases (Table 8) and the difference in time consumption between the two algorithms is, in relative terms less than 10% only for 1.9% of the models.

3.5.2. Real data set I

Table 8
Comparison of time consumption for the different algorithms

	ASD	SWA	ALS	dGN	PMF3	ALSc	dGNc	PMF3c
ASD	—	49.5	72.8	92.2	96.9	94.4	90.6	94.8
SWA	49.7	—	72.3	97.5	99.7	97.1	97.3	99.0
ALS	27.2	27.7	—	89.0	98.2	94.4	84.9	93.3
dGN	7.7	2.4	11.0	—	94.1	43.1	30.6	58.5
PMF3	3.1	0.3	1.8	5.9	—	14.4	5.3	18.2
ALSc	5.6	2.9	5.6	56.9	85.6	—	44.9	71.7
dGNc	9.4	2.7	15.0	69.2	94.7	55.0	—	98.2
PMF3c	5.2	1.0	6.7	41.5	81.7	28.3	1.8	—

The value reported in each cell is the occurrence (expressed as a percentage over all 1440 models) of strictly lower time consumption for the algorithm on the row compared to that on the column.

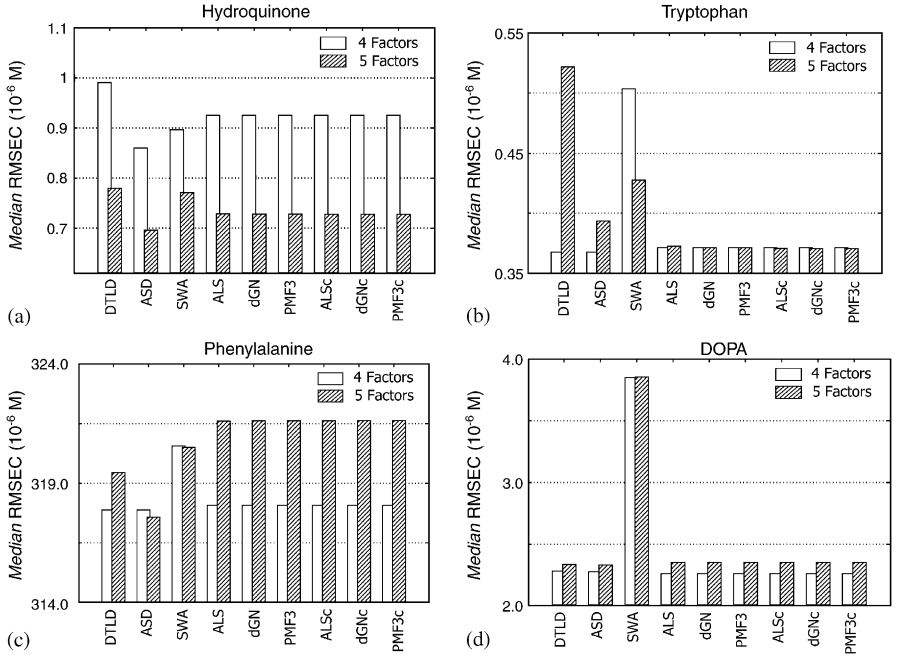

Fig. 5. Median of RMSEC over the three replicates for data set I.

Figs. 5 and 6 show the results from the analysis of data set I. All least-squares methods attain an almost identical solution for each of the replicates, both in terms of loss function and RMSEC. It is also apparent that the ALS algorithm is not as sensitive to over-factoring as the results on the simulated data may have suggested. The non-least squares algorithms (i.e. DTLD, SWATLD and ASD) retrieve factors that, although very close to the underlying ones (still full recovery occurs for all the methods), are visually identifiable as different (Fig.

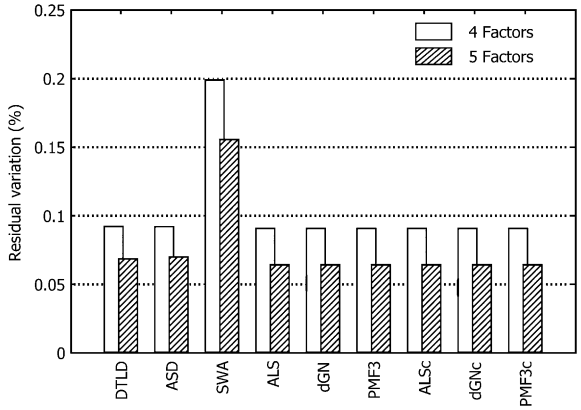

Fig. 6. Residual variation $\left(\text{as } \% \text{ of } \left\|\mathbf{X}^{(1\times JK)}\right\|_{\mathrm{F}}^{2}\right)$ for data set I.

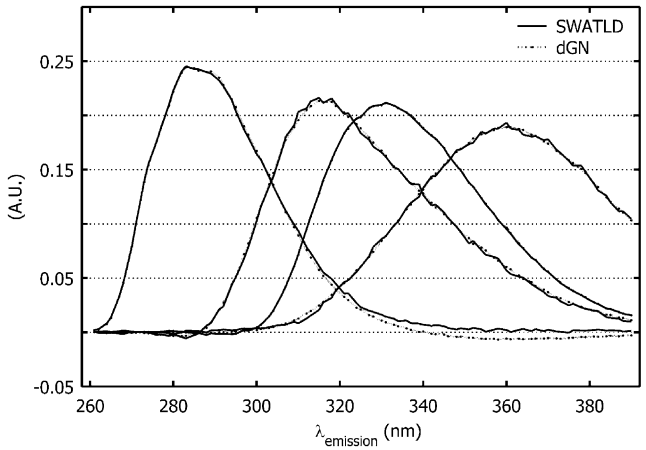

Fig. 7. Emission loadings for data set I (replicate 3) obtained with dGN and SWATLD. It can be noticed that the latter produces noisier profiles and that one of the loadings vectors present systematic differences at high emission wavelengths.

7) and result in different predictions and errors. In particular, SWATLD gives the worst performance for this data set, both in terms of the value of the loss function and of predicted concentrations. As for the simulated data, SWATLD yielded solutions with higher core consistency (mean over the replicates 99.48%) than the least squares ones (99.29%). The core consistency diagnostic may be regarded as an indication of the relevance of the 'model error', which includes the effect of small deviations from perfect trilinearity (Bro and Kiers, 2003). Such deviations are not present in simulated data sets that are not specifically designed to test this aspect, but are inherent to real data and are accounted for in the correct least

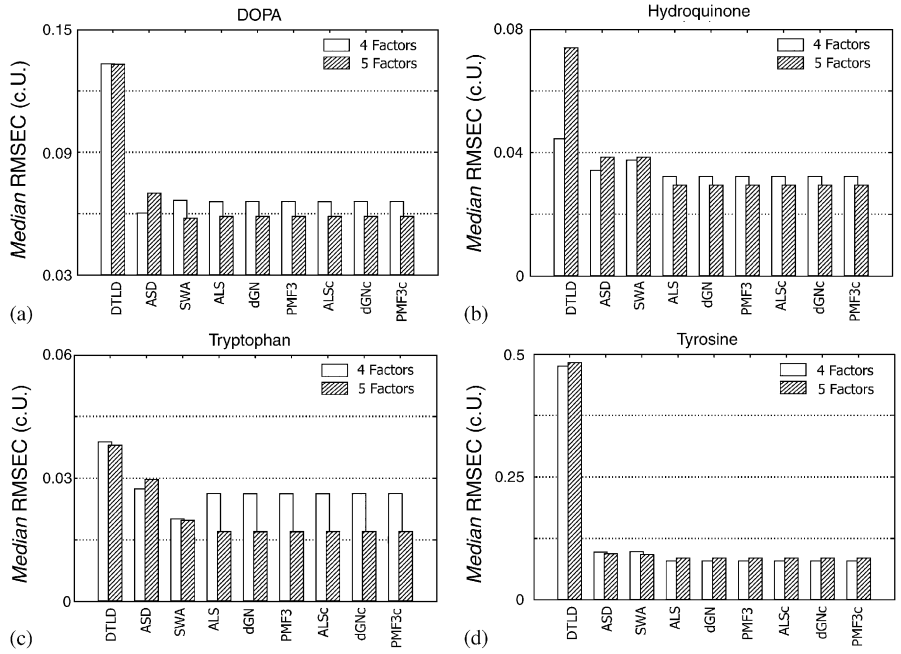

Fig. 8. Median of RMSEC over the six replicates for data set II.

squares solution (Bro and Kiers, 2003). The SWATLD algorithm seems to somewhat ignore such deviations, which results in different solutions and predictions. Similar arguments can be used for ASD, which attains the highest core consistency when the rank is over-estimated, although the differences in the loadings are considerably smaller. The fact that the smallest sum of squared residuals are obtained with the LS methods (both for correctly and over-estimated rank) makes evident once again that the actual optimisation criterion for DTLD, ASD and SWATLD in terms of PARAFAC loss function is not known or easily predictable.

3.5.3. Real data set II

The results for data set II (Figs. 8 and 9) do not differ much from those for data set I. All methods apart from DTLD managed to retrieve the correct solution and the residual variation for this algorithm is larger than for any other. The concentrations predicted using SWATLD are not as bad as for data set I, but this method yields once again a solution with larger residuals and better core consistency (mean over the replicates 99.1% compared to 98.9% for the least squares solutions).

If one excludes DTLD, the performance of ASD in predicting the four concentrations is the worst for three of the analytes. Note that the residual variation for these two methods increases with the number of extracted components. Over-factoring does not seem to hinder the retrieval of the correct components and the RMSEC remains largely unchanged. Yet, for the LS algorithms, it was necessary sometimes to restart the calculations from a new random estimation for the loading matrices (that is, more than 10 random starts were necessary)

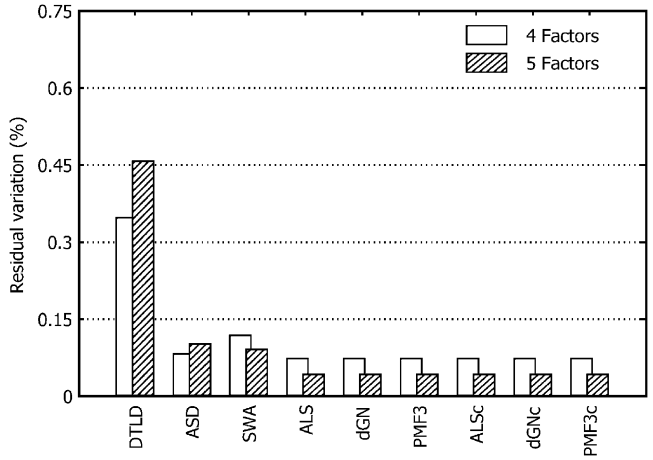

Fig. 9. Residual variation $\left(\text{as } \% \text{ of } \left\|\mathbf{X}^{(I\times JK)}\right\|_{\mathrm{F}}^{2}\right)$ for data set II.

as they tended to get stuck in degenerate solutions. This illustrates how these methods are sensitive to the initialisation values, apparently more than the non-least squares techniques. In any case, the correct solution has always been attained within a three attempts with further starts.

The high collinearity of some of the underlying factors did not impede the recovery of the correct components. As was the case for the simulated data sets, when the noise is small, the right components can be extracted also when their congruence is very high.

4. Conclusions

The tests on the nine algorithms showed quite clearly that the advantages of some of the recently proposed alternatives to PARAFAC-ALS were somewhat overestimated.

In particular, ASD appeared consistently inferior to the other iterative methods on both the simulated and the real data sets. Conversely, SWATLD showed a good capability of finding the real underlying components, albeit biased by the fact that it yields solutions with higher core consistency than the least squares one. The distortion induced in the loadings by the non-least squares criterion gives rise to conflicting performances of this algorithm on the real and the simulated data sets. However, the fact that SWATLD neither finds least squares solution nor explicitly optimises with respect to a well-characterised criterion (e.g. the core-consistency) should discourage the use of this algorithm alone for fitting a PARAFAC model and suggests that additional care should be used when evaluating the results it provides. On the other hand, its computational efficiency and resistance to a wrong estimation of the model's rank make this method an excellent candidate for the initialisation of least squares algorithms. The derivative-based methods showed better convergence properties than ALS, especially in case of highly collinear factors, but are too expensive (for time and memory

consumption) to be a feasible option in case of large data sets, such as those commonly encountered in spectroscopy and in most of the fields where the PARAFAC model is a possible choice. Consequently, their viability in these cases seems to require a preliminary compression step, which appears to have the advantage of improving the resistance to over-factoring. With respect to the latter point, dGNc with compression appears as more efficient than PMF3c (and PMF3) and ALS confirmed to be more sensitive to over-factoring than all the other methods apart from DTLD.

In conclusion, for difficult problems and when the quality of the solution is very important or the number of factors unknown, dGNc appears as the best choice, albeit more expensive than ALS. On the other hand, when the rank of the array is not known beforehand and one is interested in exploring the underlying trilinear phenomena linked to a given set of data, SWATLD can give some insight on the factors. For all other cases, the standard ALS represents a good trade-off between computational expense and quality of the solution.

An interesting possibility is to combine the different algorithms to exploit their best features and compensate their shortcomings. E.g. SWATLD could be used to provide the initial values for a PMF3/dGN algorithm with compression and a final refinement could be performed using the standard ALS algorithm.

Acknowledgements

The authors are grateful for support from STVF through the ODIN consortium. They would also like to thank N.M. Faber for providing some of the m-files.

Appendix A. Factor norm distribution over the loading vectors

Due the scaling indeterminacy each factor of a PARAFAC model with F components can be expressed as

$$\hat{\mathbf{X}} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})^{\mathrm{T}} = \sum_{f=1}^{F} r_f \mathbf{a}_f \left(t_f \mathbf{c}_f \odot s_f \mathbf{b}_f\right)^{\mathrm{T}} \equiv \sum_{f=1}^{F} \tilde{\mathbf{a}}_f \left(\tilde{\mathbf{c}}_f \odot \tilde{\mathbf{b}}_f\right)^{\mathrm{T}}, \tag{A.1}$$

where $\mathbf{a}_f$, $\mathbf{b}_f$ and $\mathbf{c}_f$ are the f-th columns of respectively $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, r_f, s_f and t_f are arbitrary non-zero scalars such that $r_f s_f t_f = 1$ and $\tilde{\mathbf{a}}_f = r_f \mathbf{a}_f$, $\tilde{\mathbf{b}}_f = s_f \mathbf{b}_f$ and, $\tilde{\mathbf{c}}_f = t_f \mathbf{c}_f$. To avoid this ambiguity, r_f, s_f and t_f are normally defined as

$$r_f = \|\mathbf{b}\|_2 \cdot \|\mathbf{c}\|_2, \tag{A.2a}$$

$$s_f = \|\mathbf{b}\|_2^{-1} \tag{A.2b}$$

and

$$t_f = \|\mathbf{c}\|_2^{-1}, \tag{A.2c}$$

or, in other words, such that $\left\|\tilde{\mathbf{b}}_f\right\|_2 = \left\|\tilde{\mathbf{c}}_f\right\|_2 = 1$. In order to equally weight the three loading matrices, a different choice was made for the implementation of dGN, PMF3 and SWATLD. By defining $q_f = \left(\left\|\mathbf{a}_f\right\|_2 \cdot \left\|\mathbf{b}_f\right\|_2 \cdot \left\|\mathbf{c}_f\right\|_2\right)^{1/3}$ and calculating r_f, s_f and t_f as $q_f\left\|\mathbf{a}_f\right\|_2^{-1}$, $q_f\left\|\mathbf{b}_f\right\|_2^{-1}$ and $q_f\left\|\mathbf{c}_f\right\|_2^{-1}$ respectively, it holds that

$$\begin{aligned} r_f s_f t_f &= q^3\left\|\mathbf{a}_f\right\|_2^{-1} \cdot \left\|\mathbf{b}_f\right\|_2^{-1} \cdot \left\|\mathbf{c}_f\right\|_2^{-1} \\ &= \left\|\mathbf{a}_f\right\|_2 \cdot \left\|\mathbf{b}_f\right\|_2 \cdot \left\|\mathbf{c}_f\right\|_2 \left\|\mathbf{a}_f\right\|_2^{-1} \cdot \left\|\mathbf{b}_f\right\|_2^{-1} \cdot \left\|\mathbf{c}_f\right\|_2^{-1} = 1 \end{aligned} \tag{A.3}$$

and that the norms of the three loading vectors $\tilde{\mathbf{a}}_f$,$\tilde{\mathbf{b}}_f$ and $\tilde{\mathbf{c}}_f$ are identical and equal to q_f.

It is worth noticing that no mention is made about the sign of the three scalars in Eqs. (A.2a–c). This problem is normally referred to as 'sign indeterminacy' and no specific convention is generally preferred. For the implemented algorithms, the choice has been to set the sign of each loading vector of $\mathbf{B}$ and $\mathbf{C}$ so that the sign of the sum of its elements was positive.

Appendix B. Improved updates for SWATLD

The loss function $L_C(\mathbf{C})$ for the SWATLD algorithm with respect to $\mathbf{C}$ can be split in:

$$\begin{aligned} L_1(\mathbf{C}) &= \sum_{k=1}^{K} \left\| \left(\mathbf{A}^{+}\mathbf{X}_{..k} - \mathbf{D}_k\mathbf{B}^{\mathrm{T}}\right)^{\mathrm{T}} \mathbf{D}_{\mathbf{B}}^{-1} \right\|_{\mathrm{F}}^2 \\ &= \sum_{j=1}^{J} \left\| \mathbf{D}_{\mathbf{B}}^{-1}\left(\mathbf{A}^{+}\mathbf{X}_{.j.} - \mathbf{D}_j\mathbf{C}^{\mathrm{T}}\right) \right\|_{\mathrm{F}}^2 \end{aligned} \tag{B.1}$$

and

$$\begin{aligned} L_2(\mathbf{C}) &= \sum_{k=1}^{K} \left\| \left(\mathbf{X}_{..k}\mathbf{B}^{+\mathrm{T}} - \mathbf{A}\mathbf{D}_k\right) \mathbf{D}_{\mathbf{A}}^{-1} \right\|_{\mathrm{F}}^2 \\ &= \sum_{i=1}^{I} \left\| \left(\mathbf{X}_{j..}^{\mathrm{T}}\mathbf{B}^{+\mathrm{T}} - \mathbf{C}\mathbf{D}_i\right) \mathbf{D}_{\mathbf{A}}^{-1} \right\|_{\mathrm{F}}^2, \end{aligned} \tag{B.2}$$

where $\mathbf{X}_{i..}$, $\mathbf{X}_{.j.}$, $\mathbf{D}_i$ and $\mathbf{D}_j$ are defined similarly to $\mathbf{X}_{..k}$ and $\mathbf{D}_k$. Using the relation $\|\mathbf{R}\|_{\mathrm{F}}^2 = \mathrm{tr}\left(\mathbf{R}^{\mathrm{T}}\mathbf{R}\right)$, where $\mathrm{tr}(\mathbf{R})$ denotes the trace of $\mathbf{R}$, the derivative of L_1 with respect to $\mathbf{C}$

is (Magnus and Neudecker, 1999):

$$\begin{aligned}\frac{\partial L_1(\mathbf{C})}{\partial \mathbf{C}} &= \frac{\partial}{\partial \mathbf{C}}\left[\sum_{j=1}^{J} \operatorname{tr}\left(\mathbf{X}_j^{\mathrm{T}}\mathbf{A}^{+\mathrm{T}}\mathbf{D}_{\mathbf{B}}^{-2}\mathbf{A}^{+}\mathbf{X}_j\right) + \sum_{j=1}^{J} \operatorname{tr}\left(\mathbf{C}\mathbf{D}_{\mathbf{B}}^{-2}\mathbf{D}_j^2\mathbf{C}^{\mathrm{T}}\right)\right.\\ &\qquad \left. -2\sum_{j=1}^{J} \operatorname{tr}\left(\mathbf{D}_{\mathbf{B}}^{-2}\mathbf{D}_j\mathbf{A}^{+}\mathbf{X}_j\mathbf{C}\right)\right]\\ &= \sum_{j=1}^{J}\left[0 + 2\mathbf{C}\mathbf{D}_{\mathbf{B}}^{-2}\mathbf{D}_j^2 - 2\left(\mathbf{D}_{\mathbf{B}}^{-2}\mathbf{D}_j\mathbf{A}^{+}\mathbf{X}_j\right)^{\mathrm{T}}\right]\\ &= 2\mathbf{C} - 2\left(\sum_{j=1}^{J}\left(\mathbf{D}_j\mathbf{A}^{+}\mathbf{X}_j\right)^{\mathrm{T}}\right)\mathbf{D}_{\mathbf{B}}^{-2}, \end{aligned} \tag{B.3}$$

where the last equality stems from $\sum_{j=1}^{J}\mathbf{D}^2 j = \mathbf{D}_{\mathbf{B}}^2$. Since $\sum_{j=1}^{J}\left(\mathbf{D}_j\mathbf{A}^{+}\mathbf{X}_j\right)^{\mathrm{T}} = \mathbf{X}^{(K\times IJ)}\left(\mathbf{B}\odot\mathbf{A}^{+\mathrm{T}}\right)$, Eq. (B.3) becomes

$$\frac{\partial L_1(\mathbf{C})}{\partial \mathbf{C}} = 2\mathbf{C} - 2\mathbf{X}^{(K\times IJ)}\left(\mathbf{B}\odot\mathbf{A}^{+\mathrm{T}}\right)\mathbf{D}_{\mathbf{B}}^{-2}. \tag{B.4}$$

Eq. (9) follows by applying the analogous derivation to $L_2(\mathbf{C})$ and by solving for $\mathbf{C}$ the equation

$$\frac{\partial L_{\mathbf{C}}(\mathbf{C})}{\partial \mathbf{C}} = \frac{\partial L_1(\mathbf{C})}{\partial \mathbf{C}} + \frac{\partial L_2(\mathbf{C})}{\partial \mathbf{C}} = 0, \tag{B.5}$$

which holds because one is looking for a stationary point.

Appendix C. Derivation of the system of normal equations for PMF3

The loss function for PMF3 can be expressed as

$$L_{\mathrm{PMF3}}(\mathbf{p}) = \mathbf{r}^{\mathrm{T}}\mathbf{r} + \gamma\left(\mathbf{p}-\mathbf{p}^{\bullet}\right)^{\mathrm{T}}\left(\mathbf{p}-\mathbf{p}^{\bullet}\right), \tag{C.1}$$

where $\mathbf{p}^{\bullet}$ is the target value ($\mathbf{0}$ in case of the regularisation: the penalty can be considered as some sort of equality soft constraint operating towards $\mathbf{p} = \mathbf{p}^{\bullet} = \mathbf{0}$).

A Taylor expansion truncated after the linear term yields:

$$\begin{aligned}\tilde{L}_{\mathrm{PMF3}}(\Delta\mathbf{p}) &= \left(\mathbf{r}^0 + \mathbf{J}\Delta\mathbf{p}\right)^{\mathrm{T}}\left(\mathbf{r}^0 + \mathbf{J}\Delta\mathbf{p}\right)\\ &\quad + \gamma\left(\mathbf{p}^0 + \Delta\mathbf{p} - \mathbf{p}^{\bullet}\right)^{\mathrm{T}}\left(\mathbf{p}^0 + \Delta\mathbf{p} - \mathbf{p}^{\bullet}\right), \end{aligned} \tag{C.2}$$

where $\mathbf{r}^0 = \mathbf{r}\left(\mathbf{p}^0\right)$. Expanding:

$$\begin{aligned}\tilde{L}_{\mathrm{PMF3}}(\Delta\mathbf{p}) &= \mathbf{r}^{0\mathrm{T}}\mathbf{r}^0 + 2\Delta\mathbf{p}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{r}^0 + \Delta\mathbf{p}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{J}\Delta\mathbf{p} + \gamma\left(\widehat{\mathbf{p}}^{0\mathrm{T}}\,\widehat{\mathbf{p}}^{0} + 2\Delta\mathbf{p}^{\mathrm{T}}\widehat{\mathbf{p}}^{0} + \Delta\mathbf{p}^{\mathrm{T}}\Delta\mathbf{p}\right) \\ &= \mathbf{r}^{0\mathrm{T}}\mathbf{r}^0 + \gamma\widehat{\mathbf{p}}^{0\mathrm{T}}\widehat{\mathbf{p}}^{0} + 2\Delta\mathbf{p}^{\mathrm{T}}\left(\mathbf{J}^{\mathrm{T}}\mathbf{r}^0 + \gamma\widehat{\mathbf{p}}^{0}\right) + \Delta\mathbf{p}^{\mathrm{T}}\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \gamma\mathbf{I}_N\right)\Delta\mathbf{p}, \end{aligned} \tag{C.3}$$

where $\widehat{\mathbf{p}}^{0} = \mathbf{p}^0 - \mathbf{p}^{\bullet}$. The first derivative is then:

$$\frac{\partial\tilde{L}_{\mathrm{PMF3}}(\Delta\mathbf{p})}{\partial\Delta\mathbf{p}} = 0 + 0 + 2\left(\mathbf{J}^{\mathrm{T}}\mathbf{r}^0 + \gamma\widehat{\mathbf{p}}^{0}\right) + 2\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \gamma\mathbf{I}_N\right)\Delta\mathbf{p} = 0, \tag{C.4}$$

where the last holds because one is looking for a stationary point. Hence

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \gamma\mathbf{I}_N\right)\Delta\mathbf{p} = -\left(\mathbf{J}^{\mathrm{T}}\mathbf{r}^0 + \gamma\widehat{\mathbf{p}}^{0}\right). \tag{C.5}$$

Finally, including the Levenberg–Marquadt modification one yields Eq. (22).

The regularisation term operates to reduce the step length (provided that the γ has a suitable value), but also has the effect of preventing exceedingly high parameters (i.e. one of the possible effects of scaling indeterminacy and 2FDs), which are penalised.

Appendix D. Helpful definitions and equations

1. Given two matrices $\mathbf{A}$ and $\mathbf{B}$ with the same number f of columns the Khatri–Rao product $\odot$ is defined as

$$\begin{aligned}\mathbf{A}\odot\mathbf{B} &= [\,\mathbf{a}_1\otimes\mathbf{b}_1 \quad \mathbf{a}_2\otimes\mathbf{b}_2 \quad \cdots \quad \mathbf{a}_F\otimes\mathbf{b}_F\,] \\ &= [\,\mathrm{vec}\,\mathbf{b}_1\mathbf{a}_1^{\mathrm{T}} \quad \cdots \quad \mathrm{vec}\,\mathbf{b}_F\mathbf{a}_F^{\mathrm{T}}\,], \end{aligned} \tag{D.1}$$

   where $\otimes$ is the Kronecker product (Magnus and Neudecker, 1999).
2. The congruence (also referred to as 'uncorrected correlation' or 'Tucker congruence coefficient') between the f-th factor $\mathbf{z}_f = \mathbf{a}_f\otimes\mathbf{b}_f\otimes\mathbf{c}_f$ and the g-th factor $\mathbf{z}_g = \mathbf{a}_g\otimes\mathbf{b}_g\otimes\mathbf{c}_g$, can be written as

$$\begin{aligned}\mathrm{cong}\left(\mathbf{z}_f,\mathbf{z}_g\right) = \cos\left(\mathbf{z}_f,\mathbf{z}_g\right) &= \frac{\mathbf{a}_f^{\mathrm{T}}\mathbf{a}_g}{\|\mathbf{a}_f\|_2\|\mathbf{a}_g\|_2}\cdot\frac{\mathbf{b}_f^{\mathrm{T}}\mathbf{b}_g}{\|\mathbf{b}_f\|_2\|\mathbf{b}_g\|_2} \\ &\quad\cdot\frac{\mathbf{c}_f^{\mathrm{T}}\mathbf{c}_g}{\|\mathbf{c}_f\|_2\|\mathbf{c}_g\|_2}. \end{aligned} \tag{D.2}$$

3. The relation between the scalar L_n multiplying an array $\underline{\mathbf{R}}$ and the level of noise Noise% (expressed as percentage of the total sum of squares) associated to it is the following:

$$\mathrm{Noise}\% = \frac{L_n^2\|\mathbf{R}\|_{\mathrm{F}}^2\cdot\|\mathbf{X}\|_{\mathrm{F}}^2}{\|\mathbf{X}\|_{\mathrm{F}}^2 + L_n^2\|\mathbf{R}\|_{\mathrm{F}}^2\cdot\|\mathbf{X}\|_{\mathrm{F}}^2}100 = \frac{L_n^2}{1+L_n^2}100, \tag{D.3}$$

where Noise% is the level, **R** is the matricised array containing the noise normalised to a Frobenius norm of 1 and **X** is the noiseless data.
L_n assumes values of 0.1005, 0.2294 and 0.3333 to attain respectively 1%, 5% and 10% noise.

4. Given the score matrix **A** (i.e. the first loading matrix) and assumed that the f-th component corresponds to the g-th constituent in the analysed data set (i.e. g-th column $\mathbf{y}_g$ of the concentration matrix **Y**), the root mean squared error in calibration (RMSEC) for the g-th constituent is defined as

$$\text{RMSEC} = I^{-0.5} \cdot \left\| \mathbf{y}_g - \hat{\mathbf{y}}_g \right\|_2, \tag{D.4}$$

where $\hat{\mathbf{y}}_g = [\mathbf{a}_f \quad \mathbf{1}][\mathbf{a}_f \quad \mathbf{1}]^{+}\mathbf{y}_g$ holds the predicted concentrations for the g-th constituent.

References

Andersson, C.A., Bro, R., 2000. The N-way Toolbox for MATLAB. Chemometrics Intell. Lab. Systems 52 (1), 1–4.

Baunsgaard, D., Munck, L., Norgaard, L., 2000. Analysis of the effect of crystal size and color distribution on fluorescence measurements of solid sugar using chemometrics. Appl. Spectrosc. 54 (11), 1684–1689.

Björck, Å., 1996. Numerical Methods for Least Squares Problems. First ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Bro, R., 1998. Multi-way analysis in the food industry. Models Algorithms and Applications, University of Amsterdam.

Bro, R., Andersson, C.A., 1998. Improving the speed of multiway algorithms part II. Compression. Chemometrics Intell. Lab. Systems 42 (1–2), 105–113.

Bro, R., Kiers, H.A.L., 2003. A new efficient method for determining the number of components in PARAFAC models. J. Chemometrics 17 (5), 274–286.

Carroll, J.D., Chang, J.J., 1970. Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart–Young decomposition. Psychometrika 35 (3), 283–319.

Carroll, J.D., Pruzansky, S., Kruskal, J.B., 1980. Candelinc—a general-approach to multidimensional-analysis of many-way arrays with linear constraints on parameters. Psychometrika 45 (1), 3–24.

Chen, Z.P., Wu, H.L., Jiang, J.H., Li, Y., Yu, R.Q., 2000. A novel trilinear decomposition algorithm for second-order linear calibration. Chemometrics Intell. Lab. Systems 52 (1), 75–86.

Chen, Z.P., Li, Y., Yu, R.Q., 2001. Pseudo alternating least squares algorithm for trilinear decomposition. J. Chemometrics 15 (3), 149–167.

Faber, N.M., Bro, R., Hopke, P.K., 2003. Recent developments in CANDECOMP/PARAFAC algorithms: a critical review. Chemometrics Intell. Lab. Systems 65 (1), 119–137.

Golub, G.H., Van Loan, C.F., 1996. Matrix Computations. The Johns Hopkins University Press, Baltimore, London. pp. 1–47.

Harshman, R.A., 1970. Foundations of the PARAFAC procedure: models and conditions for an 'explanatory' multi-modal factor analysis. UCLA working papers in phonetics, vol. 16, pp. 1–84.

Hayashi, C., Hayashi, F., 1982. A new algorithm to solve PARAFAC-model. Behaviormetrika 11, 49.

Ho, C.N., Christian, G.D., Davidson, E.R., 1978. Application of method of rank annihilation to quantitative-analyses of multicomponent fluorescence data from video fluorometer. Anal. Chem. 50 (8), 1108–1113.

Ho, C.N., Christian, G.D., Davidson, E.R., 1980. Application of the method of rank annihilation to fluorescent multicomponent mixtures of polynuclear aromatic-hydrocarbons. Anal. Chem. 52 (7), 1071–1079.

Ho, C.N., Christian, G.D., Davidson, E.R., 1981. Simultaneous multicomponent rank annihilation and applications to multicomponent fluorescent data acquired by the video fluorometer. Anal. Chem. 53 (1), 92–98.

Hopke, P.K., Paatero, P., Jia, H., Ross, R.T., Harshman, R.A., 1998. Three-way (PARAFAC) factor analysis: examination and comparison of alternative computational methods as applied to ill-conditioned data. Chemometrics Intell. Lab. Systems 43 (1–2), 25–42.

Jiang, J.H., Wu, H.L., Li, Y., Yu, R.Q., 1999. Alternating coupled vectors resolution (ACOVER) method for trilinear analysis of three-way data. J. Chemometrics 13 (6), 557–578.

Jiang, J.H., Wu, H.L., Li, Y., Yu, R.Q., 2000. Three-way data resolution by alternating slice-wise diagonalization (ASD) method. J. Chemometrics 14 (1), 15–36.

Kiers, H.A.L., 1998. A three-step algorithm for CANDECOMP/PARAFAC analysis of large data sets with multicollinearity. J. Chemometrics 12 (3), 155–171.

Kiers, H.A.L., Ten Berg, J.M.F., Bro, R., 1999. PARAFAC2—Part I. A direct fitting algorithm for the PARAFAC2 model. J. Chemometrics 13 (3–4), 275–294.

Krijnen, W.P., 1993. The Analysis of Three-Way Arrays by Constrained PARAFAC Methods. University of Groningen, The Netherlands.

Kruskal, J.B., Harshman, R.A., Lundy, M.E., 1989. How (MFA) data can cause degenerate PARAFAC solutions, among other relationships. In: Coppi, R., Bolasco, S. (Eds.), Multiway Data Analysis. Elsevier Science Publishers, Amsterdam, pp. 115–122.

Levenberg, K., 1944. A method for the solution of certain problems in least squares. Quart. Appl. Math. 2, 164–168.

Li, Y., Jiang, J.H., Wu, H.L., Chen, Z.P., Yu, R.Q., 2000. Alternating coupled matrices resolution method for three-way arrays analysis. Chemometrics Intell. Lab. Systems 52 (1), 33–43.

Liu, X.Q., Sidiropoulos, N.D., 2001. Cramer–Rao lower bounds for low-rank decomposition of multidimensional arrays. IEEE Trans. Signal Process. 49 (9), 2074–2086.

Lorber, A., 1985. Features of quantifying chemical-composition from two-dimensional data array by the rank annihilation factor-analysis method. Anal. Chem. 57 (12), 2395–2397.

Madsen, K., Nielsen, H.B., Tingleff, O., 2004. Methods for non-linear least squares problems. second ed. Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

Magnus, J.R., Neudecker, H., 1999. Matrix Differential Calculus With Applications in Statistics and Econometrics. revised ed. Wiley, New York.

Marquadt, D., 1963. An Algorithm for least-squares estimation of nonlinear parameters. SIAM J. Appl. Math. 11, 431–441.

MATLAB 6.5, The Mathworks, www.themathworks.com.

Mitchell, B.C., Burdick, D.S., 1994. Slowly converging parafac sequences—swamps and 2-Factor degeneracies. J. Chemometrics 8 (2), 155–168.

Paatero, P., 1997. A weighted non-negative least squares algorithm for three-way 'PARAFAC' factor analysis. Chemometrics Intell. Lab. Systems 38 (2), 223–242.

Paatero, P., 2000. Construction and analysis of degenerate PARAFAC models. J. Chemometrics 14 (3), 285–299.

Rao, C.R., Mitra, S., 1971. Generalized Inverse of Matrices and Its Applications. Wiley, New York.

Rayens, W.S., Mitchell, B.C., 1997. Two-factor degeneracies and a stabilization of PARAFAC. Chemometrics Intell. Lab. Systems 38 (2), 173–181.

Riu, J., Bro, R., 2003. Jack-knife technique for outlier detection and estimation of standard errors in PARAFAC models. Chemometrics Intell. Lab. Systems 65 (1), 35–49.

Sanchez, E., Kowalski, B.R., 1986. Generalized rank annihilation factor-analysis. Anal. Chem. 58 (2), 496–499.

Sanchez, E., Kowalski, B.R., 1990. Tensorial resolution: a direct trilinear decomposition. J. Chemometrics 4, 29–45.

Sidiropoulos, N.D., Bro, R., 2000. On the uniqueness of multilinear decomposition of N-way arrays. J. Chemometrics 14 (3), 229–239.

Tomasi, G., Bro, R., 2004. PARAFAC and missing values. Chemometrics Intell. Lab. Systems, in press.

Wu, H.L., Shibukawa, M., Oguma, K., 1998. An alternating trilinear decomposition algorithm with application to calibration of HPLC-DAD for simultaneous determination of overlapped chlorinated aromatic hydrocarbons. J. Chemometrics 12 (1), 1–26.

Yates, F., 1933. The analysis of replicated experiments when the field results are incomplete. Empire J. Exp. Agri. 1, 129.