JOURNAL OF CHEMOMETRICS
J. Chemometrics **12,** 155–171 (1998)

A THREE–STEP ALGORITHM FOR CANDECOMP/PARAFAC ANALYSIS OF LARGE DATA SETS WITH MULTICOLLINEARITY

HENK A. L. KIERS
Department of Psychology, University of Groningen, Grote Kruisstraat 2/1, NL-9712 TS Groningen, The Netherlands

SUMMARY

Fitting the CANDECOMP/PARAFAC model by the standard alternating least squares algorithm often requires very many iterations. One case in point is that of analysing data with mild to severe multicollinearity. If, in addition, the size of the data is large, the computation of one CANDECOMP/PARAFAC solution is very time-consuming. The present paper describes a three-step procedure which is much more efficient than the ordinary CANDECOMP/PARAFAC algorithm, by combining the idea of data compression with a form of regularization of the compressed data array. © 1998 John Wiley & Sons, Ltd.

KEY WORDS Three-way analysis; trilinear decomposition; CANDECOMP/PARAFAC; multicollinearity

1. INTRODUCTION

A popular model for the analysis of three-way data is the CANDECOMP or PARAFAC model (independently introduced by Carroll and Chang[1] and Harshman[2] respectively), henceforth denoted as the 'CP model'. The CP model can be described as follows. Let $\underline{X}$ denote a three-way array with elements x_{ijk}, $i = 1,\ldots,I$, $j = 1,\ldots,J$, $k = 1,\ldots,K$. Then the CP model is given by

$$x_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} + e_{ijk} \tag{1}$$

where a_{ir}, b_{jr} and c_{kr} are elements of the three component matrices **A**, **B** and **C** of orders $I \times R$, $J \times R$ and $K \times R$ respectively and e_{ijk} denotes the error term for observation x_{ijk}. Both Carroll and Chang[1] and Harshman[2] proposed to fit the CP model to a data array by minimizing the sum of squared error terms, using an alternating least squares (ALS) algorithm.

Having been introduced in psychometrics, the CP model turned out to be especially successful in chemometric applications.[3–5] This is because physical-chemical principles behind response mechanisms often correspond to the CP model: the main deviation from the model is caused by an often small amount of noise. Typically, in chemometric models, some of the underlying component matrices contain spectra or other smooth profiles, and often the underlying component matrices show at least mild degrees of multicollinearity. Furthermore, typically, the size of at least two modes in chemometric data is rather large (e.g. of the order of 100). The multicollinearity in the data tends to

Correspondence to: Henk A. L. Kiers, Department of Psychology, University of Groningen, Grote Kruisstraat 2/1, NL-9712 TS Groningen, The Netherlands.

CCC 0886–9383/98/030155–17 $17.50

Received May 1997
Accepted October 1997

cause the CP algorithm to require very many iterations (e.g. several tens of thousands), and the size of the data array causes each iteration to be rather expensive, making a single CP analysis practically unfeasible.

Several approaches have been suggested to remedy the problem of slow convergence. For instance, Mitchell and Burdick[6] realized that one cause of slow convergence of the CP algorithm was that the algorithm landed in a region of the solution space consisting of 'degenerate solutions'.[7] After very many iterations the algorithm was able to leave this 'swamp' area and then converge quickly, but while residing in such an area, the algorithm proceeds only very slowly. Mitchell and Burdick suggest avoiding such runs of an algorithm by stopping a run as soon as it enters a swamp area, as indicated by at least one 'triple cosine' tending to -1, and randomly starting a new run; here a triple cosine is a cosine between tensor products $\mathbf{a}_l \otimes \mathbf{b}_l \otimes \mathbf{c}_l$ and $\mathbf{a}_m \otimes \mathbf{b}_m \otimes \mathbf{c}_m$, $l \neq m$, with $\mathbf{a}_l$, $\mathbf{b}_l$ and $\mathbf{c}_l$ denoting the lth columns of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ respectively and $\otimes$ the (right) Kronecker product.[8] Assuming that at least one randomly started run will avoid the degenerate solution space, this is an effective way of avoiding 'swamping' due to degeneracy. However, slow convergence is not always accompanied by near degeneracies. In particular, in the case of high multicollinearity, convergence tends to be slow, whereas degeneracies are only rarely encountered.

A more general type of approach to handle slow convergence has already been used by Harshman.[2] He employed a relaxation technique of changing each variable individually as a function of its current short-range and long-range trends. A different procedure for accelerating slow convergence of the CP algorithm is the Gauss–Newton-type procedure proposed by Paatero,[9] coupled with penalty terms of gradually decreasing impact. A disadvantage of these techniques, however, is that a monotonical decrease in the loss function value is no longer guaranteed. As a consequence, the algorithm may in certain situations lead away from a relatively good solution, or it may, in principle, even oscillate between solutions. Another problem with such an approach is that it requires a certain amount of tinkering to choose relaxation parameters or penalty parameters, which is quite problematic for users with little feeling for the optimization problem at hand. A third problem is its large memory requirement, making the analysis of large data arrays unfeasible.

In the present paper a three-step approach is proposed based on three monotonically converging sequences involving CP analyses of differently compressed versions of the data and of the full data. This procedure can be carried out in a fully automated way, thus not requiring any tinkering by the user. The procedure is a combination of compression[10–13] and regularization of the data array. The first step consists of the CP analysis of the regularized compressed data array; in the second step the optimally compressed data array is analysed using the result from the first step as a start; and in the third step the full array is analysed with the solution from the second step as a start. To avoid local optima, the first step should be repeated from different random initial estimates (from now on denoted as 'initials'). Because the first step usually requires only a few iterations, this procedure is an efficient way (even when run from several starts) to obtain good initials. To improve the estimates, the second step uses optimal compression bases;[13] the analysis of the thus compressed array does require a considerable number of iterations, but due to the good start obtained in step one, this number tends to be considerably smaller than that needed for randomly started runs; moreover, because of the availability of good initials, there seems to be no need to use several such runs. Finally, the third step tends to need only very few iterations, which is important because each iteration in the final analysis is very costly (involving the full data array).

In Section 2 the rationale behind the regularizing compression step will be explained and details will be given of the computations involved in the two compression steps. In Section 3, results of a simulation study comparing several compression-based procedures will be given as well as results from analyses of data sets from the literature. In Section 4 it will be discussed how to handle possible degeneracies. The paper is concluded with a general discussion in Section 5.

2. REGULARIZED AND OPTIMAL COMPRESSION

When analysing practical data sets by a PARAFAC-like procedure, Kiers and Smilde[14] found that the algorithm required very many iterations, and results of different randomly started runs tended to differ only slightly in terms of loss function value but considerably in terms of obtained parameter estimates. Although this type of outcome often indicates that a wrong number of components has been used, this cannot explain the present problem, because we actually knew that we took the correct number of components. Here the problem can probably be attributed to the extreme flatness of the criterion function and to premature termination of the algorithm. Two of the three component matrices underlying these data were known to have considerable multicollinearity, so it seemed reasonable to attribute the slow convergence to this multicollinearity. In fact, to check this, data sets of the same size but with component matrices of considerably smaller condition numbers have been constructed, and indeed for such data the algorithm converged quickly. It was thus concluded that high condition numbers in underlying component matrices may well be the cause of slow convergence. Moreover, high condition numbers in component matrices are directly related to high condition numbers in the data written as supermatrices with rows referring to the entries of the mode at hand. Therefore a possibility to remedy the thus caused slow convergence might be to alleviate the multicollinearity in the data.

To see how we can alleviate the multicollinearity in the data, we first consider the error-free situation. Error-free data satisfying the CP model can be described equivalently as

$$\mathbf{X}_a = \mathbf{A}\mathbf{H}(\mathbf{C}^T \otimes \mathbf{B}^T) \tag{2a}$$

$$\mathbf{X}_b = \mathbf{B}\mathbf{H}(\mathbf{A}^T \otimes \mathbf{C}^T) \tag{2b}$$

$$\mathbf{X}_c = \mathbf{C}\mathbf{H}(\mathbf{B}^T \otimes \mathbf{A}^T) \tag{2c}$$

where $\mathbf{X}_a$ is the $I \times JK$ matrix with the frontal planes of $\underline{X}$ next to each other, $\mathbf{X}_b$ ($J \times IK$) and $\mathbf{X}_c$ ($K \times IJ$) are the matrices obtained analogously from $\underline{X}$ after permuting the modes of $\underline{X}$ such that one obtains a $J \times K \times I$ and a $K \times I \times J$ array respectively, and $\mathbf{H}$ is the $R \times R^2$ matrix with the frontal planes of the $R \times R \times R$ 'superidentity array' $\underline{I}$ with all elements equal to zero, except the elements with three equal indices, which all equal one. Now suppose that the matrix $\mathbf{A}$ has high multicollinearity whereas, for the sake of argument, $\mathbf{B}$ and $\mathbf{C}$ are orthonormal. Then $\mathbf{X}_a$ and $\mathbf{A}$ have the same condition number (as can be seen upon realizing that in this case $\mathbf{H}(\mathbf{C}^T \otimes \mathbf{B}^T)$ is row-wise orthonormal). Furthermore, from equation (2a) it follows that $\mathbf{X}_a$ is in the column space of $\mathbf{A}$. Hence, if $\mathbf{P}_a$ denotes a basis matrix for the column space of $\mathbf{A}$, we have

$$\mathbf{P}_a\mathbf{P}_a^+\mathbf{X}_a = \mathbf{P}_a\tilde{\mathbf{A}}\mathbf{H}(\mathbf{C}^T \otimes \mathbf{B}^T) \tag{3}$$

where $\mathbf{P}_a^+$ denotes the Moore–Penrose inverse of $\mathbf{P}_a$ and $\tilde{\mathbf{A}}$ denotes the $R \times R$ matrix such that $\mathbf{A} = \mathbf{P}_a\tilde{\mathbf{A}}$, hence $\tilde{\mathbf{A}} = \mathbf{P}_a{}^+\mathbf{A}$. Equation (3) gives an expression of the data and the model projected onto the basis matrix $\mathbf{P}_a$. This expression is at the basis of the postponed basis multiplication procedure,[11] which has been shown[12] to be equivalent to the analysis of a compressed data array (as had already been proposed by Carroll *et al*.[15] and Appellof and Davidson[10]). Thus studying the error-free case leads us naturally to *compression* of the data array. As shown by Kiers and Harshman,[12] CP analysis of the full data is equivalent to CP analysis of the compressed data in $\mathbf{P}_a\mathbf{P}_a^+\mathbf{X}_a$, provided that the basis matrix is column-wise orthonormal. If $\mathbf{P}_a$ spans both $\mathbf{X}_a$ and $\mathbf{A}$, then this equivalence also holds in the case of imperfect model fit, albeit only if the basis matrix is column-wise orthonormal, because then we have $\|\mathbf{P}_a\mathbf{P}_a^+\mathbf{X}_a - \mathbf{P}_a\tilde{\mathbf{A}}\mathbf{H}(\mathbf{C}^T \otimes \mathbf{B}^T)\|^2 = \|\mathbf{P}_a^+\mathbf{X}_a - \tilde{\mathbf{A}}\mathbf{H}(\mathbf{C}^T \otimes \mathbf{B}^T)\|^2$. However, in the present case, where we consider *perfect* fit, we have equivalence of the full model to a compressed model irrespective of possible (non-)orthonormality of $\mathbf{P}_a$, because it follows from (3) that

$$\mathbf{P}_a^{+}\mathbf{X}_a = \tilde{\mathbf{A}}\mathbf{H}(\mathbf{C}^{T}\otimes\mathbf{B}^{T}) \tag{4}$$

Thus, as long as we have perfect fit, we can even compress the data with respect to non-orthonormal basis matrices and still be able to write the compressed data as a CP model in terms of alternative (and smaller) parameter matrices. Obviously, similar compressions can be performed for the B– and C–mode, yielding

$$\mathbf{Y}_a \equiv \mathbf{P}_a^{+}\mathbf{X}_a(\mathbf{P}_c^{T+}\otimes\mathbf{P}_b^{T+}) = \tilde{\mathbf{A}}\mathbf{H}(\mathbf{C}^{T}\otimes\mathbf{B}^{T}) \tag{5}$$

where $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$ denote the $R \times R$ matrices needed to obtain $\mathbf{B}$ and $\mathbf{C}$ from $\mathbf{P}_b$ and $\mathbf{P}_c$ respectively.

As mentioned above, we can derive (5) from the perfect fitting case (2) for any basis matrices $\mathbf{P}_a$, $\mathbf{P}_b$ and $\mathbf{P}_c$. If we were to take column-wise orthonormal basis matrices, as in most compression procedures proposed so far,[10,12,13,15] the condition number of $\mathbf{X}_a$ would be the same as that of $\mathbf{Y}_a = \mathbf{P}_a^{+}\mathbf{X}_a(\mathbf{P}_c^{T+}\otimes\mathbf{P}_b^{T+})$, and the same would hold for the $J \times IK$ matrix $\mathbf{X}_b$ and its compressed version $\mathbf{Y}_b \equiv \mathbf{P}_b^{+}\mathbf{X}_b(\mathbf{P}_a^{T+}\otimes\mathbf{P}_c^{T+})$ and for the $K \times IJ$ matrix $\mathbf{X}_c$ and its compressed version $\mathbf{Y}_c \equiv \mathbf{P}_c^{+}\mathbf{X}_c(\mathbf{P}_b^{T+}\otimes\mathbf{P}_a^{T+})$, which can be obtained by permuting the indices of the arrays $\mathbf{X}_a$ and $\mathbf{Y}_a$. Thus it can be seen that the standard compression techniques compress the data in such a way that the underlying multicollinearity remains in the compressed array. Therefore the CP algorithm applied to this compressed array can be expected to converge slowly, just as it would when applied to the full data. However, if we take $\mathbf{P}_a$, $\mathbf{P}_b$ and $\mathbf{P}_c$ such that the compressed versions of $\mathbf{X}_a$, $\mathbf{X}_b$ and $\mathbf{X}_c$ become (close to) row-wise orthonormal (for which a procedure will be given below), then in fact (most of) the multicollinearity is *removed* from the array and *transferred* to the basis matrices. As a result, the compressed array is well conditioned in all directions, and the CP algorithm applied to this array will converge quickly. In the case of perfect fit the real $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$ can then be reobtained from the solutions for **A, B** and **C** upon premultiplying these by $\mathbf{P}_a$, $\mathbf{P}_b$, and $\mathbf{P}_c$ respectively.

In the case of *imperfect* fit the above no longer holds true, and even if $\mathbf{P}_a$, $\mathbf{P}_b$ and $\mathbf{P}_c$ were to give bases for both $\underline{\mathrm{X}}$ and the optimal **A, B** and **C** respectively, the least squares solution minimizing

$$\|\mathbf{X}_a - \mathbf{A}\mathbf{H}(\mathbf{C}^{T}\otimes\mathbf{B}^{T})\|^2 \tag{6}$$

would not equal that obtained from minimizing

$$\|\mathbf{P}_a^{+}\mathbf{X}_a(\mathbf{P}_c^{T+}\otimes\mathbf{P}_b^{T+}) - \tilde{\mathbf{A}}\mathbf{H}(\tilde{\mathbf{C}}^{T}\otimes\tilde{\mathbf{B}}^{T})\|^2 \tag{7}$$

in contrast with what is the case if the basis matrices are column-wise orthonormal. However, when the fit is reasonably close to perfect (i.e. when the noise level is low), one may expect that the solution from minimizing (7) premultiplied by the corresponding basis matrices so as to obtain solutions for **A, B** and **C** will be close to the **A, B** and **C** actually minimizing (6). Hence the solution from (7) could still be expected to give good estimates for minimizing (6).

So far, it has not been explained how we obtain basis matrices such that $\mathbf{P}_a^{+}\mathbf{X}_a(\mathbf{P}_c^{T+}\otimes\mathbf{P}_b^{T+})$ and the compressed versions of $\mathbf{X}_b$ and $\mathbf{X}_c$ are close to row-wise orthonormality. This can be done as follows. We start by finding a basis matrix $\mathbf{P}_a$ such that $\mathbf{P}_a^{+}\mathbf{X}_a$ is row-wise orthonormal. Such a matrix can be found in various ways. Here we obtain such a matrix from the singular value decomposition (SVD) of $\mathbf{X}_a$, given as $\mathbf{X}_a = \mathbf{U}\mathbf{D}\mathbf{V}^{T}$ with $\mathbf{U}^{T}\mathbf{U} = \mathbf{V}^{T}\mathbf{V} = \mathbf{I}$ and $\mathbf{D}$ diagonal, the sizes of these matrices depending on the rank of $\mathbf{X}_a$, which we assume to be at least R and usually much larger. Let $\mathbf{U}_R$, $\mathbf{D}_R$ and $\mathbf{V}_R$ denote truncated matrices with the first R left singular vectors, singular values and right singular vectors respectively. Then we take the basis matrix $\mathbf{P}_a$ as $\mathbf{P}_a = \mathbf{U}_R\mathbf{D}_R$, and, as a result, $\mathbf{P}_a^{+}\mathbf{X}_a$ equals $\mathbf{V}_R{}^{T}$, which indeed is row-wise orthonormal; note that the high condition number of $\mathbf{X}_a$ is transferred to $\mathbf{P}_a$, because this high condition number is reflected in the singular values in $\mathbf{D}_R$. Upon permuting the indices, we obtain the array $\mathbf{X}_b(\mathbf{P}_a^{T+}\otimes\mathbf{I})$. If multicollinearity recides not only in the underlying **A** but also in **B**, the matrix $\mathbf{X}_b(\mathbf{P}_a^{T+}\otimes\mathbf{I})$ will be multicollinear as well. To find a basis matrix that

captures this multicollinearity, we use the SVD $\mathbf{X}_b(\mathbf{P}_a^{T+}\otimes\mathbf{I}) = \tilde{\mathbf{U}}\tilde{\mathbf{D}}\tilde{\mathbf{V}}^T$ and take $\mathbf{P}_b$ as $\mathbf{P}_b = \tilde{\mathbf{U}}_R\tilde{\mathbf{D}}_R$, thus having $\mathbf{P}_b^{+}\mathbf{X}_b(\mathbf{P}_a^{T+}\otimes\mathbf{I}) = \tilde{\mathbf{V}}_R{}^T$, which is row-wise orthonormal. Then we again permute the array to obtain $\mathbf{X}_c(\mathbf{P}_b^{T+}\otimes\mathbf{P}_a^{T+})$ and find $\mathbf{P}_c = \bar{\mathbf{U}}_R\bar{\mathbf{D}}_R$, where $\bar{\mathbf{U}}_R$ and $\bar{\mathbf{D}}_R$ are obtained from the SVD $\mathbf{X}_c(\mathbf{P}_b^{T+}\otimes\mathbf{P}_a^{T+})=\bar{\mathbf{U}}\bar{\mathbf{D}}\bar{\mathbf{V}}^T$. Now $\mathbf{Y}_c = \mathbf{P}_c{}^{+}\mathbf{X}_c(\mathbf{P}_b^{T+}\otimes\mathbf{P}_a^{T+}) = \bar{\mathbf{V}}_R{}^T$, which is hence row-wise orthonormal. Upon permuting the array again, we find $\mathbf{Y}_a$ and $\mathbf{Y}_b$ respectively. These matrices will usually not be row-wise orthonormal, but their condition numbers tend to be much lower than those of $\mathbf{X}_a$ and $\mathbf{X}_b$, and, as a consequence of this compression procedure, we have found a compressed array with condition numbers for its associated three matrix expressions ($\mathbf{Y}_a$, $\mathbf{Y}_b$ and $\mathbf{Y}_c$) much smaller than the condition numbers of the matrix expressions ($\mathbf{X}_a$, $\mathbf{X}_b$ and $\mathbf{X}_c$) of the original array $\underline{\mathbf{X}}$.

To further decrease the condition numbers of the matrix expressions of the compressed array, we suggest repeating the above procedure on the current compressed array $\mathbf{Y}$. The resulting new basis matrices are incorporated in the first set of basis matrices (by simply postmultiplying the first basis matrices by the corresponding newly obtained ones). Practical experience suggests that repeating this cycle, say, ten times usually leads to condition numbers close to unity for all three matrices, and continuing the procedure even further appears to converge to a solution with all condition numbers tending to unity, although we cannot prove that this must always be the case. Because for our purpose it suffices to have condition numbers close to unity, we propose to use a limited number of such iterations; in all our analyses we used ten. Because this procedure turns an ill-conditioned array into a well-conditioned array, this procedure will be denoted as *regularizing compression* and the ensuing compressed array is called the *regularized compressed array*.

To see what the above procedure amounts to, we first consider the case of perfect data again. In that case we know that there is a three-way array, namely $\underline{\mathbf{I}}$, which is row-wise orthonormal in all three directions and which can be obtained by taking $\mathbf{P}_a = \mathbf{A}$, $\mathbf{P}_b = \mathbf{B}$ and $\mathbf{P}_c = \mathbf{C}$. If the basis matrices are indeed found as above, then the ensuing CP analysis of the compressed array is very simple. The CP analysis of $\underline{\mathbf{I}}$ (the matrix expression of which is $\mathbf{H}$) boils down to minimizing $\|\mathbf{H} - \tilde{\mathbf{A}}\mathbf{H}(\mathbf{C}^T\otimes\mathbf{B}^T)\|^2$, which has the trivially simple solution $\tilde{\mathbf{A}} = \tilde{\mathbf{B}} = \tilde{\mathbf{C}} = \mathbf{I}$. Clearly, premultiplying this solution by the corresponding basis matrices leads to the matrices **A, B** and **C** which actually fit the full data array perfectly. In practice the iterative procedure tends to converge to rotations (in all directions) of $\underline{\mathbf{I}}$, so the basis matrices will be rotations of **A, B** and **C**, possibly rescaled column-wise. The CP solution of the compressed array then consists of (rescaled) rotation matrices, and again premultiplying this solution by the corresponding basis matrices leads to the matrices **A, B** and **C**. It is now intuitively clear that arrays with close to perfect fit will turn into regularized compressed arrays close to rotations of the superidentity array, and the CP analysis of such an array will yield matrices $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$ which, when premultiplied by the corresponding basis matrices, will lead to solutions close to the ones actually minimizing the CP loss function. Of course, it remains to be seen what is 'close' to perfect fit. However, if one is not satisfied with the ensuing solution, one could update this by the ordinary CP algorithm started from this solution, which, assuming that the present solution is close to the best one, can be expected to converge quickly.

This procedure has initially been tested by Kiers and Smilde[14] in order to fit their PARAFAC-like model. It turned out that the procedure itself often led to insufficiently adequate solutions, so it needed finishing by a series of final runs with the CP algorithm applied to the full data. However, the number of iterations required for this analysis tended to be still quite high, although considerably lower than when starting from scratch. This led them to use an analysis of the 'ordinary compressed array' (based on SVDs of $\mathbf{X}_a$, $\mathbf{X}_b$ and $\mathbf{X}_c$) as an intermediate step. Experience with this analysis of the 'ordinary compressed array' was that it requires very many iterations *when started from scratch*, but, once converged, it provides a very good start for the CP analysis of the full data, requiring only a few such expensive iterations. Combining this with the experience that the analysis of regularized compressed data leads to a reasonably good start for the ordinary algorithm, they suggested using the analysis of

the regularized compressed array as a first step; as a second step, the ordinary compressed array (i.e. using column-wise orthonormal basis matrices) is analysed using the current solution as a start (after projecting this on the present basis matrices); the third and final step consists of an analysis of the full data array, started from the solution from the second step (obviously, after premultiplying the solution by the corresponding basis matrices). In the present paper we suggest a similar three-step approach for CP analysis.

However, we use the above iterative procedure for finding low condition numbers in all three directions, whereas Kiers and Smilde[14] only used compression in two directions. Furthermore, rather than 'ordinary compression', we used the 'optimal' compression proposed by Bro and Andersson,[13] which is based on a TUCKALS3[16] analysis of the data array and finds $\mathbf{P}_a$, $\mathbf{P}_b$ and $\mathbf{P}_c$ by minimizing $\|\mathbf{X}_a - \mathbf{P}_a\mathbf{Y}_a(\mathbf{P}_c^T \otimes \mathbf{P}_b^T)\|^2$ over column-wise orthonormal matrices $\mathbf{P}_a$, $\mathbf{P}_b$ and $\mathbf{P}_c$ and over $\mathbf{Y}_a$. Because these basis matrices are indeed column-wise orthonormal, this compression will not regularize the data. The compression, however, is optimal in the sense that it gives the orthonormal projection of $\underline{X}$ on R-dimensional subspaces that remains closest to the original $\underline{X}$. For a schematic description of the three-step approach, refer to Section 3.2 and Table 1.

We tested the above suggested three-step approach by comparing it with ordinary CP analysis and other compression-based procedures, applying these methods to 40 data sets with known underlying structure as well as to four data sets reported in the literature. This comparison is described in the next section.

3. COMPARISON OF SEVERAL PROCEDURES FOR CANDECOMP/PARAFAC

To test the three-step approach and compare it with CP analysis of the full data as well as with three variants based on compression, we performed a simulation study on the basis of 40 constructed data sets. In addition, we analysed four data sets from the literature so as to put the simulation study in proper perspective. The algorithms were programmed in MATLAB[17] v. 4.2. The analyses were carried out in a Windows 3.11 environment on a personal computer with a Pentium 100 MHz processor and 32 Mb RAM.

3.1. Construction of data for simulation study

For the simulation study, 40 data sets of order $20 \times 20 \times 20$ were constructed, with known (three-dimensional) underlying CP structure and various amounts of noise and multicollinearity, as follows. For each data set, random matrices $\mathbf{A}_0$, $\mathbf{B}_0$ and $\mathbf{C}_0$ of order 20×3 were constructed; the elements of these matrices were drawn from the uniform [0.5,1.5] distribution in the conditions with mild multicollinearity (20 data sets) and from the uniform [1,2] distribution in the conditions with severe multicollinearity (20 data sets). In the conditions with severe multicollinearity the condition numbers of $\mathbf{A}_0$, $\mathbf{B}_0$ and $\mathbf{C}_0$ ranged from nine to 18, with an average of about eleven; in the conditions with mild multicollinearity the condition numbers of $\mathbf{A}_0$, $\mathbf{B}_0$ and $\mathbf{C}_0$ ranged from six to 14, with an average of about seven. Random matrices, rather than fixed matrices, were used so as to cover a broad range of possible situations. The matrices were all forced to have non-negative elements so as to mimic situations where these modes pertain to non-negative physical entities such as frequencies in spectra or concentrations. To avoid complications in setting up the design, no provisions were made to ensure smooth profiles in the component matrices, which would be an important cause of multicollinearity; instead, multicollinearity was forced simply by the choice of possible values for the component matrices, upon noting that random values from relatively high-valued intervals tend to lead to relatively severe multicollinearity in the ensuing matrix of random numbers.

From the matrices $\mathbf{A}_0$, $\mathbf{B}_0$ and $\mathbf{C}_0$ the data array was constructed as

Table 1. Schematic representation of five methods for fitting CP model

	CP–Full	RegComp	OptComp	SVDComp	Three-Step
Five CP runs of	$\underline{X}$	$\underline{Y}_{reg}$	$\underline{Y}_{opt}$	$\underline{Y}_{svd}$	$\underline{Y}_{reg}$
Select best	$\mathbf{A}_c,\mathbf{B}_c,\mathbf{C}_c$	$\tilde{\mathbf{A}}_r,\tilde{\mathbf{B}}_r,\tilde{\mathbf{C}}_r$	$\tilde{\mathbf{A}}_o,\tilde{\mathbf{B}}_o,\tilde{\mathbf{C}}_o$	$\tilde{\mathbf{A}}_s,\tilde{\mathbf{B}}_s,\tilde{\mathbf{C}}_s$	$\tilde{\mathbf{A}}_r,\tilde{\mathbf{B}}_r,\tilde{\mathbf{C}}_r$
Obtain full-sized estimates	$\mathbf{A}_c$ $\mathbf{B}_c$ $\mathbf{C}_c$	$\mathbf{A}_r = \mathbf{R}_a\tilde{\mathbf{A}}_r$ $\mathbf{B}_r = \mathbf{R}_b\tilde{\mathbf{B}}_r$ $\mathbf{C}_r = \mathbf{R}_c\tilde{\mathbf{C}}_r$	$\mathbf{A}_o = \mathbf{O}_a\tilde{\mathbf{A}}_o$ $\mathbf{B}_o = \mathbf{O}_b\tilde{\mathbf{B}}_o$ $\mathbf{C}_o = \mathbf{O}_c\tilde{\mathbf{C}}_o$	$\mathbf{A}_s = \mathbf{S}_a\tilde{\mathbf{A}}_s$ $\mathbf{B}_s = \mathbf{S}_b\tilde{\mathbf{B}}_s$ $\mathbf{C}_s = \mathbf{S}_c\tilde{\mathbf{C}}_s$	$\mathbf{A}_r = \mathbf{R}_a\tilde{\mathbf{A}}_r$ $\mathbf{B}_r = \mathbf{R}_b\tilde{\mathbf{B}}_r$ $\mathbf{C}_r = \mathbf{R}_c\tilde{\mathbf{C}}_r$
Initials for intermediate analysis					$\tilde{\mathbf{A}}_o = \mathbf{O}_a{}^T\mathbf{A}_r$ $\tilde{\mathbf{B}}_o = \mathbf{O}_b{}^T\mathbf{B}_r$ $\tilde{\mathbf{C}}_o = \mathbf{O}_c{}^T\mathbf{C}_r$
Intermediate solution					$\tilde{\mathbf{A}}_i,\tilde{\mathbf{B}}_i,\tilde{\mathbf{C}}_i$
Obtain full-sized estimates					$\mathbf{A}_i = \mathbf{O}_a\tilde{\mathbf{A}}_i$ $\mathbf{B}_i = \mathbf{O}_b\tilde{\mathbf{B}}_i$ $\mathbf{C}_i = \mathbf{O}_c\tilde{\mathbf{C}}_i$
Solution of final CP run		$\mathbf{A}_{rf},\mathbf{B}_{rf},\mathbf{C}_{rf}$	$\mathbf{A}_{of},\mathbf{B}_{of},\mathbf{C}_{of}$	$\mathbf{A}_{sf},\mathbf{B}_{sf},\mathbf{C}_{sf}$	$\mathbf{A}_{if},\mathbf{B}_{if},\mathbf{C}_{if}$

Notation:
$\underline{X}$, full data array;
$\underline{Y}_{reg}$, regularized compressed array;
$\underline{Y}_{opt}$, optimally compressed array;
$\underline{Y}_{svd}$, SVD-based compressed array;
$\mathbf{R}_a$, $\mathbf{R}_b$, $\mathbf{R}_c$, regularized three-dimensional compression basis matrices;
$\mathbf{O}_a$, $\mathbf{O}_b$, $\mathbf{O}_c$, optimal three-dimensional compression basis matrices;
$\mathbf{S}_a$, $\mathbf{S}_b$, $\mathbf{S}_c$, SVD-based three-dimensional compression basis matrices.

$$\mathbf{X}_a = \mathbf{A}_0\mathbf{H}(\mathbf{C}_0{}^T \otimes \mathbf{B}_0{}^T) + \mathbf{N} \tag{8}$$

where $\mathbf{N}$ denotes the matrix expression of a three-way array with proportional noise values; specifically, following Mitchell and Burdick,[6] each noise value was computed by multiplying the corresponding data element by a value drawn from the normal distribution with standard deviation α(with α=0.025, 0.05, 0.10 or 0.25, depending on the noise condition at hand). Thus levels of 2.5%, 5%, 10% and 25% proportional noise were generated. For each combination of multicollinearity condition (mild or severe) and noise level, five data sets were generated, thus leading to the total of 40 data sets mentioned earlier.

3.2. Five Methods for Fitting CP model

The five methods used in the simulation study are summarized in Table 1. Each method starts with five randomly started CP runs to a particular (usually compressed) data array. From these, the best solution is selected and used to initialize the next CP run (except in the first method). Only in the three-step method are the results of this analysis again used to initialize a new run. Specifically, in the first method, CP–Full, the full data array is analysed by five runs and the best solution is taken as *the* solution. The second, third and fourth methods (RegComp, OptComp and SVDComp respectively) start by five CP runs applied to the regularized, optimally and SVD-based compressed data arrays respectively. The resulting solutions are premultiplied by the corresponding basis matrices to obtain full-sized estimates for **A, B** and **C**. These estimates in turn are used as initials for a final CP run applied to the full data set. The three-step method is a bit more complex. The first step of Three-Step is the same as that of RegComp. However, the full-sized estimates obtained from RegComp are now

used to compute initials for a CP run of the optimally compressed array; these initials are obtained by projecting the full-sized estimates on the optimal basis matrices (see 'initials for intermediate analysis' in Table 1). The second step of Three-Step consists of the analysis of the optimally compressed array, initialized as mentioned just above. From this intermediate analysis, full-sized estimates $\mathbf{A}_i$, $\mathbf{B}_i$ and $\mathbf{C}_i$ are obtained. These in turn are used as initials for the third step, consisting of a CP run applied to the full data set.

3.3. Criteria of Interest

All methods described above were applied to the 40 data sets constructed for this simulation study. One of the main criteria of interest in this study was how well the original component matrices were recovered by each of the methods. To assess this, rather than comparing each obtained component with each corresponding underlying component matrix, which would have to take into account permutational, scaling and sign indeterminacies in the solution, we compared solutions by computing cosines between tensor products $\mathbf{a}_l^0 \otimes \mathbf{b}_l^0 \otimes \mathbf{c}_l^0$, l=1,2,3, and $\hat{\mathbf{a}}_m \otimes \hat{\mathbf{b}}_m \otimes \hat{\mathbf{c}}_m$, m=1,2,3, where the superscript 'zero' pertains to columns from the underlying component matrices and the superimposed 'hat' pertains to columns of estimated component matrices. Given a set of component matrices {**A,B,C**}, other sets of component matrices that yield the same representation of the data are constituted of the same such tensor products, although possibly in a different order. Hence a useful overall measure of recovery is the sum of three cosines between tensor products from the underlying component matrices with those from the obtained component matrices, with the latter tensor products ordered such that they lead to the highest overall sum of cosines. These cosines, denoted as ϕ(because they are computed simply as Tucker's ϕ-coefficient of congruence[18]), are the same as the measures employed by Mitchell and Burdick,[6] called 'uncorrected correlation coefficients'. A drawback of this cosine measure is that it tends to yield high values even when differences between tensor products are appreciable, as long as all values are positive. Therefore even small differences should be taken seriously.

Although the main aim of all methods is to optimally recover the underlying components, this is operationalized in all methods by minimizing the loss function (6). Thus, in addition to how well the underlying components are recovered by the different methods, we compared the loss function values for the full data resulting from the five methods; in RegComp, OptComp, SVDComp and Three-Step these values are simply the function values upon convergence of the final CP run of the full data, whereas for CP–Full this is the lowest value resulting from the five randomly started runs. Furthermore, we compared these with the loss function values associated with the initials for these CP runs (which, if good, would obviate the need for the final CP runs).

In the present paper a method is proposed which, under certain circumstances, yields good estimates at little cost. Hence, besides the quality of the estimates, a main point of interest is computational efficiency. Rather than focusing on computation time, which is implementation-dependent, we report numbers of iterations required. Because the various compressed arrays are all of the same size ($3\times3\times3$) and, similarly, the full data sizes are always of size $20\times20\times20$, each iteration for a compressed array requires exactly the same amount of time and so does each iteration for the full data array. In our configuration this was approximately 0.018 s. for an iteration for the compressed array and 0.11 s for an iteration for the full data array. Differences between methods that are not captured in this way are differences in time for computation of the basis matrices, but, as will be reported, these are typically small compared with those of the iterative parts.

A third point of concern in the present simulation study is the sensitivity to finding suboptimal solutions ('suboptima'), by which we mean local optima and solutions where the algorithm has not converged yet. In a first series of runs, using our standard convergence criterion, we encountered

Table 2. Summed ϕ-values, averaged within each condition

Multicollinearity	Noise (%)	CP–Full $\mathbf{A}_c,\mathbf{B}_c,\mathbf{C}_c$	RegComp $\mathbf{A}_{rf},\mathbf{B}_{rf},\mathbf{C}_{rf}$	OptComp $\mathbf{A}_{of},\mathbf{B}_{of},\mathbf{C}_{of}$	SVDComp $\mathbf{A}_{sf},\mathbf{B}_{sf},\mathbf{C}_{sf}$	Three-Step $\mathbf{A}_{if},\mathbf{B}_{if},\mathbf{C}_{if}$
Severe	2·5	2·997	2·997	2·998	2·997	2·998
	5	2·988	2·988	2·988	2·988	2·988
	10	2·945	2·405	2·947	2·944	2·621
	25	1·056	1·040	0·994	0·983	0·992
Mild	2·5	2·999	2·999	2·999	2·999	2·999
	5	2·997	2·997	2·997	2·997	2·997
	10	2·982	2·982	2·981	2·981	2·981
	25	2·699	2·503	2·323	2·197	2·508
		$\mathbf{A}_c,\mathbf{B}_c,\mathbf{C}_c$	$\mathbf{A}_r,\mathbf{B}_r,\mathbf{C}_r$	$\mathbf{A}_o,\mathbf{B}_o,\mathbf{C}_o$	$\mathbf{A}_s,\mathbf{B}_s,\mathbf{C}_s$	$\mathbf{A}_i,\mathbf{B}_i,\mathbf{C}_i$
Severe	2·5	2·997	2·984	2·998	2·997	2·998
	5	2·988	2·941	2·988	2·984	2·988
	10	2·945	1·134	2·948	2·134	2·622
	25	1·056	0·961	0·990	1·062	0·990
Mild	2·5	2·999	2·997	2·999	2·999	2·999
	5	2·997	2·988	2·997	2·997	2·997
	10	2·982	2·747	2·980	2·816	2·980
	25	2·699	1·221	2·326	1·096	2·194

many suboptima, most of which probably pertained to premature terminations of the algorithm. Therefore we used a stricter convergence criterion in the actual analyses reported here: we considered a run to be converged if differences in consecutive function values (see (6)) differed by less than 10^{-6}% of the current function value. With this criterion, suboptima were encountered far less frequently, and those that were encountered often differed more from the best value than we would be willing to explain by premature termination. In the present study we counted the number of suboptima in each series of five randomly started runs as the number of runs that led to loss function values larger than 1.0001 times the lowest function value encountered (the alleged global minimum). The results of the analyses of the optimally compressed arrays are also compared with those from the second step of Three-Step, which analyse the same compressed arrays.

Finally, we checked for each (intermediate) solution whether it could be considered a degenerate solution, as indicated by dimensions with very low (close to -1) ϕ-coefficients between tensor products pertaining to the different dimensions of the solution.

3.4. Results of simulation study

The first results considered here pertain to the highest sums of ϕ-values for the triple tensor products from the obtained matrices (both intermediate and final solutions) and the underlying matrices, averaged over five replications in each condition. The results are given in Table 2. It can be seen that for the severe multicollinearity conditions with 2.5% and 5% noise and the mild multicollinearity conditions with 2.5%, 5% and 10% noise the quality of all five final solutions is virtually the same (and very good, considering that the maximum is three). For the conditions with more noise the results for all methods become worse. In fact, upon inspection of the results for the individual data sets, we saw that *all* methods failed to recover the underlying components (as indicated by $\phi < 2$) for

Table 3. Average numbers of FDIs (large print) and CDIs (small print)

Multicollinearity	Noise (%)	CP–Full	RegComp	OptComp	SVDComp	Three-Step
Severe	2·5	$5 \times 4272{\cdot}5$	1737·8	3·0	163·0	3·0
			$5 \times 6{\cdot}6$	$5 \times 9331{\cdot}4$	$5 \times 8898{\cdot}4$	$5 \times 6{\cdot}6 + 6045{\cdot}4$
	5	$5 \times 3322{\cdot}2$	1346·8	4·8	840·2	4·8
			$5 \times 10{\cdot}6$	$5 \times 6877{\cdot}2$	$5 \times 7597{\cdot}7$	$5 \times 10{\cdot}6 + 4605{\cdot}8$
	10	$5 \times 2616{\cdot}5$	2988·4	83·6	2294·4	155·4
			$5 \times 1802{\cdot}9$	$5 \times 6294{\cdot}5$	$5 \times 7986{\cdot}0$	$5 \times 1802{\cdot}9 + 6124{\cdot}0$
	25	$5 \times 857{\cdot}4$	54·0	59·2	377·8	48·6
			$5 \times 2843{\cdot}8$	$5 \times 3459{\cdot}4$	$5 \times 4668{\cdot}1$	$5 \times 2843{\cdot}8 + 1478{\cdot}0$
Mild	2·5	$5 \times 1317{\cdot}0$	448·4	2·0	3·8	2·0
			$5 \times 5{\cdot}7$	$5 \times 2204{\cdot}2$	$5 \times 2158{\cdot}5$	$5 \times 5{\cdot}7 + 1291{\cdot}8$
	5	$5 \times 1318{\cdot}0$	534·2	2·6	14·4	2·6
			$5 \times 5{\cdot}9$	$5 \times 2341{\cdot}3$	$5 \times 2355{\cdot}5$	$5 \times 5{\cdot}9 + 1524{\cdot}6$
	10	$5 \times 1221{\cdot}6$	709·0	49·8	418·2	50·6
			$5 \times 8{\cdot}1$	$5 \times 2025{\cdot}4$	$5 \times 2431{\cdot}4$	$5 \times 8{\cdot}1 + 1534{\cdot}6$
	25	$5 \times 886{\cdot}7$	590·4	253·8	788·4	257·0
			$5 \times 20{\cdot}3$	$5 \times 973{\cdot}8$	$5 \times 7734{\cdot}2$	$5 \times 20{\cdot}3 + 3641{\cdot}4$

all data in the 25% noise condition with severe multicollinearity and also for one data set in the 25% noise condition with mild multicollinearity. Because the ensuing components are of no use whatsoever, it makes little sense to compare relative differences in performance for these data sets. For the remaining 14 data sets (ten in the 10% noise conditions and four in the 25% noise condition with mild multicollinearity) we counted how many times each of the methods 'lagged behind', in that the obtained ϕ-value was more than 0.1% lower than the highest ϕ-value found for this data set. It turned out that CP–Full lagged behind five times, RegComp seven times, SVDComp three times and OptComp and Three-Step twice; thus no method always led to the highest ϕ-value, but OptComp and Three-Step were very good and notably better than CP–Full.

We also inspected the final loss function values for all methods. In the severe multicollinearity conditions with 2.5% and 5% noise and the mild multicollinearity conditions with 2.5%, 5% and 10% noise these values were, just as the ϕ-values, virtually equal. Again discarding the data sets for which all methods failed, for the remaining 14 data sets we found that CP–Full never lagged behind (i.e. its function value was never more than 0.01% higher than the lowest function value found), RegComp lagged behind twice and OptComp, SVDComp and Three-Step only once. Thus it can be seen that, as far as loss function values are concerned, considerable differences are not found very often.

As a first conclusion, we mention that in the case of low noise levels and/or mild multicollinearity all methods yield adequate results. At higher levels, for some data sets all analyses fail; for the data sets where this is not the case, OptComp and Three-Step give the best results.

Table 2 also gives the ϕ-values for the *intermediate* solutions of RegComp, OptComp, SVDComp and Three-Step. It can be seen that for RegComp and to some extent also for SVDComp these are considerably lower than those for the final solutions; for OptComp and Three-Step the differences are rather small, the only big difference being found for Three-Step in the 25% noise condition with mild multicollinearity. This indicates that the final step is certainly necessary in RegComp and SVDComp, whereas for OptComp and Three-Step this step seems necessary only rarely.

The second point of interest is the efficiency of each method, measured in terms of full-data iterations (FDIs) and compressed-data iterations (CDIs), where it should be kept in mind that the FDIs

are much more expensive than the CDIs. The average numbers of FDIs and CDIs for each condition and each method are reported in Table 3. Because of the implementation dependence of computation times, we will base our conclusions on the numbers of iterations as far as possible; we resort to computation times only for comparisons that are otherwise impossible, but handle these with caution.

It can be seen from Table 3 that CP–Full requires very many (expensive) FDIs, especially in the conditions with low noise percentages. In these conditions, OptComp and Three-Step need very few FDIs, indicating that, especially in these conditions, the use of optimal compression leads to good starting values for a full-data CP run. In this respect, OptComp clearly outperforms SVDComp in all conditions, thus sustaining the results reported by Bro and Andersson,[13] who introduced OptComp as an alternative to SVDComp. Furthermore, it can be seen that the RegComp solution does not provide good initials for a full-data CP run: the number of iterations required does (usually) decrease in comparison with that required in randomly started CP runs, but the decrease is only small compared with that encountered with OptComp and Three-Step. Clearly, the latter two almost obviate the need for (expensive) full-data CP runs, at least in the low-noise conditions, as was also deduced from the quality of the intermediate solutions from these methods in these conditions (see Table 2); in the higher-noise conditions, however, the full-data CP runs cannot always be left out.

In terms of CDIs, RegComp is by far the fastest method, at least in the low-noise conditions. However, except in the 25% noise conditions, this advantage of RegComp will not compensate for the relatively large amount of time involved in the FDI computations with this method. OptComp and SVDComp require very many CDIs, which does tend to take considerable amounts of time, even compared with the time needed for the FDIs. Finally, it can be seen that Three-Step requires far fewer CDIs than OptComp and SVDComp in the low-noise conditions. This is a consequence of the fact that the five first runs require only a few iterations, and the run for the analysis of the optimally compressed data is apparently so well started that it requires considerably fewer iterations than randomly started OptComp runs (except in the mild multicollinearity condition with 25% noise). It can be seen that even if only one randomly started OptComp run were used, Three-Step would still require fewer CDIs. This advantage of Three-Step over OptComp (and hence over all other methods) is particularly clear in the severe multicollinearity conditions with 2.5% and 5% noise and the mild multicollinearity conditions with 2.5%, 5% and 10% noise. Hence it can be concluded that for these conditions, Three-Step seems to be the preferred method: it is most efficient and the quality of its estimates is as good as that of the other methods.

In the severe multicollinearity condition with 10% noise, OptComp requires considerably fewer FDIs than all other methods. However, it turns out that the high number of FDIs required by Three-Step in this condition is caused by the analysis of a single data set; in the other four analyses, Three-Step required just as many or even fewer FDIs than OptComp did. In Section 4 it will be explained how this anomalous result can be avoided and hence that, also in this condition, Three-Step and OptComp are preferred as far as the number of FDIs is concerned, whereas Three-Step again requires fewest CDIs. Finally, in the 25% noise conditions, RegComp turned out to require relatively few FDIs (the fewest of all in the severe multicollinearity condition and ‘only’ about twice as many as OptComp and Three-Step in the mild multicollinearity condition). Considering that it uses very few CDIs, RegComp can be seen to be the most efficient method here. However, this efficiency is reached at the cost of quality: in six of these analyses the underlying components were not recovered (see above); for three of the four remaining cases the quality of the RegComp solution lagged behind that of the other methods. This leaves OptComp and Three-Step as the preferred techniques even here.

To get a proper idea of the *full* computation times for each method, we should also consider the computation of the basis matrices, and hence we have to rely on results which hold for our implementations only. It turned out that the basis matrices for regularized compression were computed in a split second (0.23 s on average), which is negligible. Computation times for the

optimal basis matrices ranged from 5 to 7 s for the 2.5%, 5% and 10% noise conditions (using less than ten iterations) and never took more than 15s (and 105 iterations). In comparison with times needed for CP–Full, this is negligible, considering that 1000 FDIs took approximately 110 s and usually several thousands were needed. Furthermore, in the comparison between RegComp and Three-Step the computation time for the optimal basis matrices is much smaller than the difference between total computation times for these methods, except in the 25% noise conditions. In the comparison between SVDComp and Three-Step this holds in all conditions. Finally, the comparison between OptComp and Three-Step is not affected by the computation time of the optimal basis matrices, because they both use these. It can be concluded that in our implementations, Three-Step was clearly most efficient in all conditions except the 25% noise conditions, where RegComp was more efficient. Thus it can be seen that on the basis of our implementations, taking the computation of the basis matrices into account does not change the overall conclusion on the relative efficiency of the methods.

The sensitivity to suboptima of all methods was assessed by comparing the solutions from the five randomly started runs in the first step of each method. It turned out that CP–Full led to suboptima 19 times, all in the 25% noise conditions; RegComp (on which is also based Three-Step) and OptComp respectively found six and one suboptima, all in the severe multicollinearity conditions with 10% and 25% noise; SVDComp led to suboptima 17 times, all in 10% and 25% noise conditions. These results suggest that using five random starts for RegComp and OptComp may not be necessary. For RegComp there is little reason to reduce this number, since these runs are usually extremely fast; for OptComp, using fewer starts would certainly make the method more efficient, but, as has been mentioned above, even if OptComp used only one run, Three-Step would still use fewer CDIs. Moreover, the results in Section 3.5 will indicate that OptComp may not always be so insensitive to suboptima as it was in the present simulation study. Furthermore, comparison of the randomly started OptComp runs with the OptComp run within Three-Step also indicates that suboptimal solutions are sometimes found for OptComp: in 36 cases the same solutions were found, in three cases the best of the five OptComp runs was better than the intermediate Three-Step run, and in one case the latter was better than all five randomly started runs; the last result indicates that OptComp led to suboptima in all five runs for this data set.

Finally, we inspected to what extent degenerate solutions occurred in the analyses. It turned out that a degenerate solution was encountered in only one of the 200 CP–Full runs; all other runs did not even tend to low triple cosine values. This indicates that slow convergence for these data has not been caused by swamping due to entering a region of near degeneracy (see also Section 4, where this indication has actually been verified). We also inspected the final results from the other methods. Degeneracies were only found in the 10% and 25% noise conditions with severe multicollinearity. In these conditions, RegComp, OptComp, SVDComp and Three-Step led to degeneracies in three, two, two and four cases respectively.

3.5 Analyses of four data sets from literature

The above results of the simulation study demonstrate the value of the three-step approach for CP analysis of large data sets. However, the results are limited to one data size and to constructed data. To see how well the method holds up in other situations, we reanalysed four data sets reported in the literature. The first data set, 'Fluorescence data', has been described by Bro.[19] This is a data set pertaining to five samples for which emission was measured at 201 emission wavelengths related to excitation at 61 excitation wavelengths. Because of the presence of three substances in the samples and because of theoretical properties of the measurement, these data can be approximated by a three-dimensional CP model. The second data set, 'Sugar data', is reported by Bro[20] and pertains to 268

Table 4. Numbers of FDIs (large print) and CDIs (small print) as well as time required in analyses of four data sets

	CP–Full	RegComp	OptComp	Three-Step
Fluorescence data ($5 \times 201 \times 61$; $R = 3$)				
Iterations	703	$26 + 66$	$759 + 2$	$112 + 2$
Time (s)	668	76	39	38
Sugar data ($268 \times 392 \times 7$; $R = 3$)				
Iterations	2968	$34 + 428$	$3147 + 63$	$539 + 64$
Time (s)	192653	16657	3467	3425
PP1 data ($10 \times 8 \times 6$; $R = 3$)				
Iterations	68576	$37 + 6042$	$83981 + 138$	$9540 + 138$
Time (s)	1634	143	1473	170
PP2 data ($10 \times 8 \times 6$; $R = 4$)				
Iterations	98188	$60 + 10671$	$96606 + 2$	$12672 + 2$
Time (s)	2524	274	1832	242

samples for which emission spectra (with 571 wavelengths, only 392 of which were used) were obtained at seven excitation wavelengths. The resulting $268 \times 392 \times 7$ data set was analysed using three dimensions. The third and fourth data sets are two ill-conditioned, synthetic data sets reported by Hopke *et al*.[21] These data sets were deliberately constructed to be 'difficult' test cases with a high degree of collinearity in the underlying component matrices. The first data set, 'PP1', was based on three components, the second, 'PP2', on four. These four data sets were analysed by CP–Full, RegComp, OptComp and Three-Step; SVDComp was left out from the analyses because it did not seem to have any particular merit over the other methods in the simulation study.

In comparing the analyses of the four data sets here, we focused on the efficiency of the methods. As for the simulated data, we expressed the efficiency in terms of numbers of iterations and we distinguished between full-data iterations and compressed-data iterations. The results are given in Table 4. To illustrate the effects of different data sizes on the ensuing overall computation times (including computation of the compression basis matrices), we also report the computation times with our implementations of the programs. However, we warn that these computation times depend considerably on the implementation and even on conditions of the computer environment (e.g. for full analyses of the largest data set the computer continuously read and wrote from virtual memory, which caused the computations to become very slow and the computation times per iteration very unstable). Therefore small differences in computation times should not be taken seriously and conclusions should foremostly be based on the numbers of iterations required, since these are not implementation-dependent.

From Table 4 we see that the conclusions from the simulation study are corroborated. For all four data sets the number of FDIs required was much smaller for RegComp, OptComp and Three-Step than for CP–Full; it should be noted that this even holds when in CP–Full only one run is used, which indicates that the analyses of compressed data lead to good initials for the full-data analyses. This turns out to be particularly true for OptComp and Three-Step, which both required considerably fewer FDIs than RegComp, which, in turn, used by far the fewest CDIs. The numbers of FDIs required by OptComp and Three-Step were virtually equal in all four analyses, but Three-Step required far fewer CDIs than did OptComp. It can thus be seen that regularization serves to reduce the number of CDIs and optimal compression serves to reduce the number of FDIs. The former is particularly useful in

cases of high multicollinearity (notably PP1 data and PP2 data), whereas the latter is particularly useful for very large data sets (Sugar data and, to some extent, Fluorescence data). Hence it is not surprising to find that RegComp and Three-Step (which both use regularized compression) are most efficient in the analyses of the smaller data sets with high multicollinearity, whereas Three-Step and OptComp (which both use optimal compression) are most efficient in the analyses of the large data sets. Overall, we can conclude that Three-Step is the only method that is relatively efficient both in the analysis of very large data sets and in the analysis of fairly small data sets with high multicollinearity.

We also compared the solutions in terms of final loss function values and component matrices. It turned out that in all analyses the differences between the final solutions were very small. We also checked the quality of the solutions obtained from RegComp, OptComp and Three-Step *before* the final full-data analyses were carried out. It turned out that in all four analyses the intermediate RegComp solution was inferior to the final solution, but the intermediate OptComp and Three-Step solutions were virtually equal to the final solutions in all four cases. This means that for these data sets the expensive FDIs could have been left out altogether and the total analysis time would be reduced to that of the (much) more efficient CDIs. Unfortunately, it does not seem wise to always leave out the final analysis step: intermediate solutions do sometimes differ markedly from the final solutions, as found in the simulation study.

Finally, we inspected the occurrence of suboptimal solutions. We found suboptimal solutions only in cases where the maximal number of iterations (20 000) was reached. This never happened in the RegComp runs, but did with the OptComp runs, although only in the analyses of the PP1 data (twice) and the PP2 data (three times). It also happened in the CP–Full runs, again only in the analyses of the PP1 data (twice) and the PP2 data (four times). We may conclude that suboptimal solutions are not found frequently, but taking some precautions (such as using five randomly started runs) does seem recommendable, especially for CP–Full and OptComp.

To summarize the above results, we can conclude that Three-Step was the only technique that worked well in all four widely differing cases. This conclusion corroborates the results of the simulation study.

4. DEALING WITH DEGENERACIES

The main aim of the presently proposed three-step method is to avoid slow convergence and still find good estimates for one's parameters. The three-step method does avoid slow convergence and finds good estimates in low-noise conditions, but in other conditions we sometimes get suboptimal solutions and slow convergence, notably in one 10% and several 25% noise conditions. To get a bit more insight into the problems with Three-Step in such analyses, for the (single) case where this happened in the 10% noise condition, we investigated the intermediate analyses in more detail. It turned out that for this data set the five RegComp runs took very many iterations and all led to degeneracies. The succeeding runs on the optimally compressed data and on the full data also led to degeneracies. Clearly, here all five RegComp runs entered a Mitchell–Burdick-type swamp, from which they did not emerge any more before the maximum number of iterations (20 000) was reached. These results suggest that Three-Step be combined with Mitchell and Burdick's[6] proposal to simply stop any CP runs that enter a swamp and restart them randomly. Thus it is suggested to use their procedure not only for CP analysis of the full data but also for intermediate CP analyses of compressed data. In this way the three-step approach can be used rather generally in practice.

The above discussion may provoke the question whether *intermediate* Mitchell–Burdick-type swamping may have caused slow convergence in *all* analyses. Therefore we checked the degeneracy indicator (lowest cosine between tensor products) every tenth iteration for all randomly started CP–Full runs as well as for all randomly started RegComp and OptComp runs. It turned out that of all 200

CP–Full runs, only one entered a swamp, from which it, moreover, did not emerge before the 20 000th iteration. This clearly demonstrates that for these data, slow convergence is mainly not caused by Mitchell–Burdick-type swamping. The RegComp runs and OptComp runs, on the other hand, did reveal intermediate and final degeneracies more often, but only in the severe multicollinearity conditions with 25% noise (OptComp) and 10% or 25% noise (RegComp). As far as OptComp is concerned, however, it can be concluded that the slow convergence of OptComp, which typically happened in the *low*-noise conditions, is *not* caused by Mitchell–Burdick-type swamping. For RegComp, more interestingly, it turned out that *all* cases where RegComp converged slowly were in fact cases where the algorithm entered a swamp (and in these analyses it was never able to pull itself out of it). If we leave out all these cases from our results, we find that the average number of iterations of the 'proper' RegComp runs in the severe multicollinearity conditions with 10% and 25% noise is 13.0 and 6.4 respectively, and the very large values of 1802.9 and 2843.8 in Table 3 should thus be replaced by these values. Clearly, these results suggest that much of the inefficiency as well as some of the inadequacy of parameter estimates by RegComp (and hence by Three-Step) in the higher-noise conditions could be remedied by avoiding degeneracies in the way suggested by Mitchell and Burdick.

The above suggested combination of Three-Step with swamp avoidance (implemented as randomly restarting the iterations as soon as the triple cosine fell below –0.95) was tested on all data sets where RegComp or OptComp found degenerate solutions in all five runs; these cases coincide with the cases where Three-Step found a degenerate solution. These data sets were based on severe multicollinearity and noise (one data set) or 25% noise (three data sets). For the 10% noise data set, RegComp and Three-Step originally failed to find good solutions, but CP–Full and Three-Step did (ϕ-values higher than 2.83), so for this data set it was hoped that by means of swamp avoidance, RegComp and Three-Step would give similarly good solutions; for the three 25% noise data sets, none of the methods found an adequate solution (all ϕ-values smaller than 1.3), but since non-degenerate solutions exist, we hoped that by swamp avoidance all methods would yield non-degenerate solutions. For the 10% noise data set, indeed, using swamp avoidance, RegComp and Three-Step did find good solutions (ϕ-values were 2.827 and 2.835 respectively), although it should be noted that this required many restarts and hence very many iterations (all RegComp runs reached the maximum of 20 000 iterations). For the first 25% noise data set, where RegComp and Three-Step originally found a degeneracy, RegComp still led to a degeneracy (although the prior runs now did lead to a non-degenerate initial), but Three-Step now found a non-degenerate solution. For the second 25% noise data set, where OptComp and Three-Step originally led to degeneracies, they no longer found degenerate solutions now. For the third 25% noise data set, where again OptComp and Three-Step originally led to degeneracies, OptComp no longer did so, but Three-Step still found a degenerate solution.

It can be concluded that swamp avoidance can be useful in avoiding degenerate solutions. In fact, Three-Step supplemented with swamp avoidance led to good solutions (with ϕ-values higher than 2.8) for all 34 data sets for which good solutions were found by at least one method, and with one exception, the associated ϕ-values were less than 0.1% lower than the highest. Also, for none of these 34 data sets did it lead to suboptimal function values. CP–Full is the only other method that likewise always found good solutions and no suboptimal function values for these 34 data sets. Thus it can be concluded that Three-Step with swamp avoidance turned out to be the best alternative to CP–Full. Moreover, for most of these data sets, Three-Step was the most efficient method. Therefore it can be recommended to always supplement Three-Step with the swamp avoidance procedure: in cases where the iterations do not even tend to a degeneracy, swamp avoidance does not make a difference and leaves Three-Step very efficient, whereas in cases where Three-Step would tend to a degeneracy, the swamp avoidance procedure will prevent Three-Step from finding such a suboptimal solution.

5. CONCLUSIONS AND DISCUSSION

The results from the simulation study and from the analyses of the data reported in the literature demonstrate that the three-step approach effectively combines the advantages of the regularizing compression procedure (fast convergence) and the optimal basis compression (accurate approximation), unless too much noise (25% in our simulations) is present. Combining each step with Mitchell and Burdick's[6] proposal for avoiding swamps appears a useful additional step to avoid suboptimal estimates which one might otherwise obtain. It should be noted that often the estimates yielded by analyses of optimally compressed data (either by OptComp or by Three-Step) are very good and obviate the need for a full-data analysis started from these estimates. However, because this cannot be guaranteed, it is recommended to always use such a final analysis, which, in cases where it would not have been needed, will require very few iterations.

In the present paper we compared Three-Step with some simpler compression-based variants and with CP–Full and saw that, overall, the Three-Step approach performed best. This, of course, leaves the question open whether alternative procedures could be devised that work even better or give good results in a broader range of applications, e.g. cases with higher noise levels. In fact, we experimented with some other approaches as well. For instance, rather than taking the dimensionality of the compression basis matrices equal to that of the CP solution, we tested procedures where the dimensionality of the basis matrix was two higher than the dimensionality of the CP solution.

However, it turned out that these procedures performed worse than their lower-dimensional counterparts. Also, we replaced the regularized compression procedure by a procedure which only regularizes but does not compress. This also made results worse. A third alternative approach tested was a three-step approach with the optimal compression in the second step replaced by SVD-based compression; in fact, this was how we implemented Three-Step to begin with. It turned out that this procedure worked reasonably well, but the use of optimal rather than SVD-based basis matrices considerably improved Three-Step.

A further limitation of the present simulation is that it only considered CP analyses with the correct dimensionality. In practice the dimensionality is not always known *a priori*. To see what happens in situations where the underlying dimensionality differs from the one used in the CP analysis, in an earlier stage of our simulation study we analysed the present three–dimensional data sets by models with dimensionalities of two (one too small) and four (one too large) respectively. It turned out that differences between methods in terms of efficiency were very similar to those obtained for analyses with the correct dimensionality, so we did not proceed with these studies in a later stage of our research.

A potential limitation of the relative usefulness of Three-Step is that it no longer seems to outperform RegComp and OptComp when the noise level becomes high. However, the results from the simulation study and the analyses of the data in the literature indicate that even in such cases little harm is done in using Three-Step, because it does not tend to become very inefficient: it is always *among* the most efficient methods, which does not hold for any other method tested here.

ACKNOWLEDGEMENTS

The author is obliged to Rasmus Bro and Jos ten Berge for helpful comments on an earlier version of this paper, and to Rasmus Bro also for providing the Fluorescence data and the Sugar data. Claus Andersson is gratefully acknowledged for obtaining the Fluorescence data, and I thank Phil Hopke and Penti Paatero for sending the two synthetic data sets prepared by Hong Jia and used in their collaborative research.

REFERENCES

1. J. D. Carroll and J.-J. Chang, *Psychometrika*, **35**, 283 (1970).
2. R. A. Harshman, *UCLA Working Papers Phonet*. **16**, 1 (1970).
3. A. K. Smilde, P. H. van der Graaf, D. A. Doornbos, T. Steerneman and A. Sleurink, *Anal. Chim. Acta*, **235**, 41 (1990).
4. S. Leurgans and R. T. Ross, *Statist. Sci.* **7**, 289 (1992).
5. R. Bro and H. Heimdal, *Chemometrics Intell. Lab. Syst.* **34**, 85 (1996).
6. B. C. Mitchell and D. S. Burdick, *J. Chemometrics*, **8**, 155 (1994).
7. J. B. Kruskal, R. A. Harshman and M. E. Lundy, in Multiway Data Analysis, ed. by R. Coppi and S. Bolasco, p. 115, Elsevier, Amsterdam 1989.
8. J. R. Schott, *Matrix Analysis for Statistics*, Wiley, New York (1997).
9. P. Paatero, *Chemometrics Intell. Lab. Syst.* **38**, 223 (1997).
10. C. J. Appellof and E. R. Davidson, *Anal. Chem.* **53**, 2053 (1981).
11. B. K. Alsberg and O. M. Kvalheim, *Chemometrics Intell. Lab. Syst.* **24**, 43 (1994).
12. H. A. L. Kiers and R. A. Harshman, *Chemometrics Intell. Lab. Syst.* **36**, 31 (1997).
13. R. Bro and C. A. Andersson, *Chemometrics Intell. Lab. Syst.* in press.
14. H. A. L. Kiers and A. K. Smilde, *J. Chemometrics*, **12**, 125 (1998).
15. J. D. Carroll, S. Pruzansky and J. B. Kruskal, *Psychometrika*, **45**, 3 (1980).
16. P. M. Kroonenberg and J. de Leeuw, *Psychometrika*, **45**, 69 (1980).
17. MATLAB, Mathworks Inc. Natick, MA (1992).
18. L. R. Tucker, *Personnel Research Section Report 984*, Department of the Army, Washington, DC (1951).
19. R. Bro, *Chemometrics Intell. Lab. Syst.* **38**, 149 (1997).
20. R. Bro, *Doctoral Dissertation*, University of Amsterdam (1998).
21. P. K. Hopke, P. Paatero, H. Jia, R. T. Ross and R. A. Harshman, *Chemometrics Intell. Lab. Syst.* in press.

