JOURNAL OF CHEMOMETRICS
J. Chemometrics 2004; **18**: 22–36
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cem.841

Bootstrap confidence intervals for three-way methods

Henk A. L. Kiers*
Heymans Institute, University of Groningen, Grote Kruisstraat 2/1, NL-9712 TS Groningen, The Netherlands

Received 7 August 2003; Revised 16 December 2003; Accepted 24 December 2003

Results from exploratory three-way analysis techniques such as CANDECOMP/PARAFAC and Tucker3 analysis are usually presented without giving insight into uncertainties due to sampling. Here a bootstrap procedure is proposed that produces percentile intervals for all output parameters. Special adjustments are offered for handling the non-uniqueness of the solutions. The percentile intervals indicate the instability of the sample solutions. By means of a simulation study it is demonstrated that the percentile intervals can fairly well be interpreted as confidence intervals for the output parameters. Copyright © 2004 John Wiley & Sons, Ltd.

KEYWORDS: confidence intervals; bootstrap; resampling; three-way analysis; CANDECOMP/PARAFAC; Tucker3

1. INTRODUCTION

For the analysis of three-way data sets (e.g. data with scores of a number of environmental stations on a number of variables under a number of conditions, or spectral profiles of a number of samples for a number of emission wavelengths and a number of excitation wavelengths), various exploratory three-way methods are available. The two most common methods for the analysis of three-way data are CANDECOMP/PARAFAC [1,2] and Tucker3 analysis [3,4]. Both methods summarize the data by components for all three modes, and for the entities pertaining to each mode they yield component loadings; in the case of Tucker3 analysis, in addition, a so-called core array is given which relates the components for all three modes to each other.

If we denote our $I \times J \times K$ three-way data array by $\underline{\mathbf{X}}$, then the two methods can be described as fitting the model

$$x_{ijk} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} a_{ip} b_{jq} c_{kr} g_{pqr} + e_{ijk} \qquad (1)$$

where a_{ip}, b_{jq} and c_{kr} denote elements of the component matrices **A** (for mode A), **B** (for mode B) and **C** (for mode C) of orders $I \times P$, $J \times Q$ and $K \times R$ respectively, g_{pqr} denotes the element (p,q,r) of the $P \times Q \times R$ core array $\underline{\mathbf{G}}$, e_{ijk} denotes the error term for element x_{ijk}, and P, Q and R denote the numbers of components for the three respective modes. The difference between CANDECOMP/PARAFAC and Tucker3 analysis is that in CANDECOMP/PARAFAC the core is actually set equal to a superidentity array (i.e. with $g_{pqr} = 1$ if $p = q = r$, $g_{pqr} = 0$ otherwise). As a consequence, in the case of CANDECOMP/PARAFAC we have the same number of components for all modes, and Equation (1) actually reduces to

$$x_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} = e_{ijk} \qquad (2)$$

Clearly, when these models are fitted to data, we end up with component matrices **A**, **B** and **C** and, in the case of Tucker3 analysis, we also get a three-way core array as outcome of the analysis.

Often the analyses are applied to data pertaining to a sample from a larger population, and usually the results for the sample are assumed to be, at least to some extent, generalizable to the population from which the sample was drawn. In the practice of three-way analysis the generalizability issue is usually dealt with by means of either cross-validation or split-half comparisons (see e.g. Reference [5]). However, neither procedure gives concrete estimates of the uncertainties due to sampling fluctuations of all parameters in our solutions.

There are techniques that do give uncertainty estimates for three-way analysis solutions, but most of these are based on analyzing derived rather than original data. For instance, Bentler and Lee [6], and McArdle and Cattell [7] proposed techniques for maximum likelihood fitting of the covariance matrices associated with the Tucker3 and the CANDECOMP/PARAFAC model respectively. These approaches 'automatically' yield standard errors for all parameters that are estimated, provided that appropriate distributional assumptions are made and the models are completely identified. However, for the direct analysis (i.e. where the data rather than their covariances are analyzed), very few attempts have been made so far to provide estimates of the uncertainties of all parameters in three-way analysis solutions. In fact, in the case of direct analysis of the data, no theoretically derived

*Correspondence to: H. A. L. Kiers, Heymans Institute, University of Groningen, Grote Kruisstraat 2/1, NL-9712 TS Groningen, The Netherlands.
E-mail: h.a.l.kiers@ppsw.rug.nl

standard errors seem to be available, and the best way seems to be to resort to resampling techniques such as the bootstrap or the jackknife [8]. In the literature, as far as is known to the author, the bootstrap has been used in three-way methods only for determining the instability of fit measures of a Tucker3 solution [9] over five bootstrap samples, but no attempt was made to offer standard errors or confidence intervals. An approach that does yield standard errors has been proposed by Riu and Bro [10], who offer a jackknife procedure for estimating standard errors for CANDECOMP/PARAFAC solutions. The jackknife can be seen as an approximation to the bootstrap (see Reference [8], pp. 145–146); in general, the bootstrap can be expected to be the more efficient of the two procedures (see Reference [8], p. 146). Therefore here we will focus on the bootstrap rather than the jackknife. The use of the jackknife by Riu and Bro [10], however, also offered further possibilities, e.g. outlier detection, which are less easy to combine with the bootstrap.

The main purpose of the present paper is to describe how to obtain uncertainty estimates, in the form of confidence intervals, for all parameters resulting from a three-way analysis (CANDECOMP/PARAFAC or Tucker3), by means of the bootstrap. The application of the bootstrap itself is relatively straightforward, as described in Section 2. The more challenging aspect is how to deal with the underidentification in the CANDECOMP/PARAFAC model and especially in the Tucker3 model. One solution would be to simply fully identify the model, but, as will be seen, this leads to procedures that may give undesirably wide confidence intervals. As will be shown in Section 3, this problem can be solved by effectively exploiting the transformational freedom in the solutions. In Section 4 the procedure is illustrated for the Tucker3 analysis of an example data set. In Section 5 the quality of the ensuing confidence intervals is evaluated by means of an extensive simulation study. Finally, in Section 6 the computational feasibility of the computer-intensive bootstrap method is dealt with and a simple and effective way of speeding up the procedure is suggested and evaluated empirically.

2. THE BOOTSTRAP FOR DETERMINING CONFIDENCE INTERVALS

The bootstrap was introduced by Efron [11] as a method for estimating standard errors for arbitrary statistics computed in a sample drawn from a population. He also proposed the bootstrap percentile interval, which is often used as a (somewhat crude) estimate of a confidence interval. Improvements of the bootstrap percentile interval have been proposed, most notably the BC_A method (see Reference [8], Chap. 14), but we will ignore this procedure here, because its generalization to situations with rotational freedom does not seem straightforward.

The basic idea of the bootstrap is to mimic the sampling process that generated our actual data sample, as follows (see Figure 1 for a schematic overview). We suppose here (and in the sequel) that the entities in mode A (e.g. the different mixtures for which we have data) are considered a random sample from a population of such entities (i.e. the population of all conceivable mixtures). Then with the bootstrap procedure we investigate what could happen if we consider our sample as a population and if we randomly (re)sample from this 'pseudo population'. In fact, we consider the distribution of score profiles in our actual sample as an optimal estimate of the distribution of such profiles in our population and we consider what could happen upon frequently randomly sampling from this population. In practice, if our three-way data set of order $I \times J \times K$ is denoted by $\underline{\mathbf{X}}$ and the score profiles are denoted by $\mathbf{X}_i$, which is a matrix of order $J \times K$, then we randomly draw (with replacement) I matrices $\mathbf{X}_i$ from the set of matrices $\{\mathbf{X}_1, \ldots, \mathbf{X}_I\}$. This creates one bootstrap sample (in which some matrices occur repeatedly and others not at all) which is then reorganized into a three-way array. This bootstrap sample then represents one case of 'what could happen if we randomly sample from our pseudo population'. To get an impression of 'what else' could happen or, more generally, of what can happen under sampling fluctuation, we need many such bootstrap samples. Hence this procedure is to be repeated, for instance, 500 times. As a result we will get 500 bootstrap samples, associated with which are 500 three-way arrays. Now to each three-way array we apply a three-way analysis method in *exactly the same way* as we applied it originally to our sample (hence including the preprocessing procedure we use), and we compute the statistics we are interested in. These statistics can, for instance, be the fit or the misfit of a particular model, the loading of an entity on a particular component, or the correlation between components. Let the statistics of interest be collected in a single vector $\boldsymbol{\theta}$. The vector of outcomes for our original sample is denoted as $\boldsymbol{\theta}^s$, whereas those for the bootstrap samples are denoted as $\boldsymbol{\theta}^b$, $b = 1, \ldots, B$, where B denotes the number of bootstrap samples drawn (e.g. $B = 500$). Now the variation in the B bootstrap sample outcome vectors indicates how and how much the outcome vectors vary if we randomly resample from our pseudo population, and this is used as an estimate of how much real samples from our real population can be expected to vary if we sample repeatedly from our actual population. Specifically, in the bootstrap procedure the variation across the outcome vectors $\boldsymbol{\theta}^b$, $b = 1, \ldots, B$, is considered indicative of the variation of vectors $\boldsymbol{\theta}^s$ if we repeatedly sample from the population.

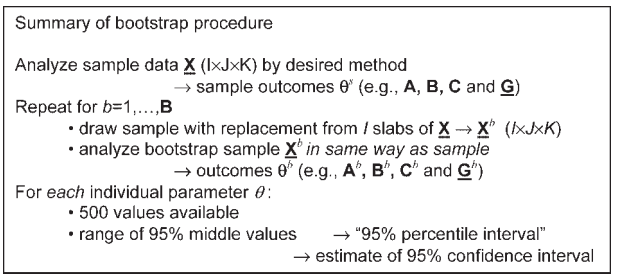

Figure 1. Summary of the bootstrap procedure.

A simple way to describe the variation across the bootstrap sample outcomes is to give, for each parameter separately, a percentile interval (e.g. a 95% percentile interval) which describes the range in which we find the middle 95% values of the parameter at hand. For instance, if we look at the model fit of a particular model, which we denote by f, then we have B bootstrap values f^b, $b = 1, \ldots, B$; we sort these

and determine the values f_{low} and f_{up} between which 95% of the f values lie. In the case of $B = 800$ we would search the values f_{low} and f_{up} between which lie $0.95 \times 800 = 760$ values; hence we must also have 20 values smaller than f_{low} and 20 values higher than f_{up}. Then we find f_{low} as the average between the 20th and the 21st sorted f value and, likewise, we find f_{up} as the average between the 780th and the 781st sorted f value. In other cases, for instance when $B = 500$, we can use actual values of f^b to determine f_{low} and f_{up} (which then are the 13th and the 488th sorted f value respectively).

The interval $[f_{low}, f_{up}]$ gives the 95% bootstrap percentile interval for the fit value f^s; likewise, percentile intervals can be determined for all other parameter estimates resulting from our three-way analysis. Such intervals tell us what we could expect to happen if we repeatedly sample from our pseudo population: we would expect to find a value in this interval in 95% of the cases. Thus they are useful indicators of the stability of our solution. However, we can make a much stronger use of them if we consider them as 95% confidence intervals for the population values of our parameter. Indeed, if our statistic is more or less symmetrically distributed across the bootstrap samples, then the halfwidth of the interval (say h) indicates the maximal distance of the pseudo population parameter to the bootstrap sample in 95% of the cases; if this distance is a good estimate for what happens in the actual population, then the halfwidth of the interval also indicates (with 95% certainty) the maximal distance of the actual population parameter to the obtained sample parameter. Thus, if we know with 95% certainty that the population parameter (ϕ) is no more than h separated from the sample value f^s, the interval $[f^s - h, f^s + h]$ is a good 95% confidence interval for the population parameter ϕ. In fact, percentile intervals give proper confidence intervals whenever a monotonic transformation of the parameter exists such that its likewise transformed sample statistic has a normal distribution (see Reference [8], p. 173). This assumption is somewhat reminiscent of the use of the Fisher z-transformation to obtain a confidence interval for a correlation coefficient. There the idea is that the Fisher z-transformation ensures that the sample correlation is distributed normally (under certain population distribution assumptions). Here we also need that such a 'normalizing' transformation exists, but in this case we do not have to know which, because, if such a transformation exists, the bootstrap percentile intervals are correct confidence intervals, irrespective of the shape of the transformation. Whether or not our assumption holds in practice is hard to verify. Therefore in a simulation study in Section 5 we will test the quality of our percentile intervals when considered as estimates of confidence intervals.

It has been described above how we can determine a bootstrap percentile interval for any parameter in the vector $\boldsymbol{\theta}$. This was based on computing the associated value in each bootstrap sample. This implicitly requires that such a value is uniquely determined. For the fit (here denoted as f) indeed that is the case: the fit of a model is uniquely determined. However, this does not hold for most other outcome parameters resulting from CANDECOMP/PARAFAC or Tucker3 analysis. One way to deal with this problem is to simply identify the parameters completely, but this will require us to even identify the order of the components and the sign of the components, which, in practice, can be difficult. In the next section, therefore, a different procedure will be offered to deal with transformational non-uniqueness.

3. EXPLOITING TRANSFORMATIONAL FREEDOM TO GET BETTER BOOTSTRAP CONFIDENCE INTERVALS

In Section 2 it has been explained how the bootstrap can be used to produce estimates of confidence intervals for any parameter resulting from a three-way method. However, it has also been mentioned that this procedure requires the output parameters to be uniquely identified. Here it will first be discussed to what extent three-way solutions are identified and how, if so desired, we can full identify them. Next it will be described how the bootstrap can be adjusted to handle non-identified solutions.

3.1. Identification of three-way solutions

One of the key features of the CANDECOMP/PARAFAC model is that it is 'essentially' uniquely identified [1,2,12]. By this it is meant that the component matrices **A**, **B** and **C** resulting from a CANDECOMP/PARAFAC analysis are, under mild assumptions, unique up to a joint permutation of the columns of the three matrices and up to scaling of the columns of the three matrices. The latter scaling should be such that, if column $\mathbf{a}_r$ is multiplied by α and column $\mathbf{b}_r$ is multiplied by β, then column $\mathbf{c}_r$ should be multiplied by $(\alpha\beta)^{-1}$. The freedom of permutation and scaling is a consequence of the fact that such scalings and permutations leave the model estimates $\sum_r a_{ir}b_{jr}c_{kr}$ unaffected. A somewhat common procedure to further identify the CANDECOMP/PARAFAC solution is to scale components such that the component matrices for two modes (e.g. the first two modes) have unit column sum of squares, or such that the component matrices for all modes have equal column sum of squares.

The order of the components can be fixed by ordering them in descending importance, i.e. such that the column sums of squares decrease. However, even then the solution is not completely identified, because an arbitrary multiplication by -1 of, for instance, $\mathbf{a}_r$ compensated by multiplication by -1 of $\mathbf{b}_r$ does not affect the model estimates, or the sum of squares, or the descending order of the component sum of squares. Hence, for a full identification, one would need to also identify the sign of the component matrices. This can be done in various ways which are, however, all rather arbitrary (e.g. ensure that the column sums in the component matrices **A** and **B** are positive).

The Tucker3 model is also not uniquely identified. In fact, here the lack of identification is much more dramatic. As already shown by Tucker [3], the model estimates are not affected by arbitrarily multiplying each of the component matrices by a non-singular square matrix, provided that the core is multiplied appropriately by the inverse of these transformations. Specifically, in matrix notation we can write model (1) as

$$\mathbf{X} = \mathbf{AG}(\mathbf{C} \otimes \mathbf{B})' + \mathbf{E} \qquad (3)$$

where **X**, **G** and **E** denote the A-mode matricized versions of $\underline{\mathbf{X}}$, $\underline{\mathbf{G}}$ and $\underline{\mathbf{E}}$ [13] respectively and $\otimes$ denotes the Kronecker product. Now multiplication of the component matrices **A**, **B** and **C** by **S**, **T** and **U** respectively does not affect the model estimates if the core is replaced by $\mathbf{S}^{-1}\mathbf{G}(\mathbf{U}^{-1} \otimes \mathbf{T}^{-1})'$. This follows from the fact that

$$\begin{aligned} &\mathbf{AS}\ \mathbf{S}^{-1}\mathbf{G}(\mathbf{U}^{-1} \otimes \mathbf{T}^{-1})'(\mathbf{CU} \otimes \mathbf{BT})' \\ &= \mathbf{AG}(\mathbf{U}^{-1} \otimes \mathbf{T}^{-1})'(\mathbf{U} \otimes \mathbf{T})'(\mathbf{C} \otimes \mathbf{B})' \\ &= \mathbf{AG}(\mathbf{C} \otimes \mathbf{B})' \end{aligned} \quad (4)$$

Clearly, the Tucker3 model has a great deal of transformational freedom. In fact, the only identified aspect of the solution is the set of subspaces spanned by the component matrices. In practice, however, one wishes to interpret the individual elements of the component matrices, or at least their relative sizes, and for practical purposes the subspaces have no useful meaning.

It is possible to further identify the Tucker3 solution. A first commonly used step is to require the component matrices to be column-wise orthonormal, which reduces the transformational non-uniqueness to *rotational* non-uniqueness. A requirement to further identify the solution is to rotate the component matrices to what we call here the 'principal axes' orientation of the Tucker3 solution (equivalent to what Kroonenberg [14], pp. 152–153, called the 'basic output' from the analysis). This solution has the property that the component matrices then contain eigenvectors of particular matrices, e.g. **A** contains eigenvectors of $\mathbf{X}(\mathbf{C} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{B})'\mathbf{X}'$; **B** and **C** contain eigenvectors of similarly defined matrices. The components are ordered naturally in the order of the eigenvalues associated with such eigenvectors. In this way the components are ordered in terms of importance. Thus the principal axes solution identifies the rotation of all component matrices, as well as their permutation. What remains is to identify the sign of the columns of component matrices, which can now be done as in CANDECOMP/PARAFAC.

The principal axes solution has nice theoretical properties but is usually not easy to interpret. Often some kind of interpretation enhancing rotation is used, e.g. simple structure rotation of the core and component matrices [15]. By means of simple structure rotation one can identify the Tucker3 solution up to permutation and scaling, and we thus end up in the same situation as with CANDECOMP/PARAFAC. To further identify the solution, we can use the same steps for identification as with CANDECOMP/PARAFAC.

It has been shown above how the CANDECOMP/PARAFAC solution and the Tucker3 solution can be identified completely. If we use exactly the same identification procedure for all bootstrap solutions then we can compare bootstrap solutions, and sensibly compute percentile intervals, and use these as estimates of confidence intervals for our parameters. However, in doing so, we imply that in our actual data analysis we consider as our solution only the one that we get from the exact same identification procedure. As a consequence, if we have two samples from the same population and we analyze both in exactly the same way, and it so happens that the solutions are almost identical but have a different ordering of the columns, then we would not recognize this near identity of the solutions. To make the example a bit more concrete, suppose the first sample gives a CANDECOMP/PARAFAC solution with component matrices $\mathbf{A} = (\mathbf{a}_1\ \mathbf{a}_2)$, $\mathbf{B} = (\mathbf{b}_1\ \mathbf{b}_2)$ and $\mathbf{C} = (\mathbf{c}_1\ \mathbf{c}_2)$, where the components are ordered in descending sum of squares; let us further assume that vectors $\mathbf{a}_1$ and $\mathbf{a}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$, and $\mathbf{c}_1$ and $\mathbf{c}_2$ differ considerably, leading to entirely different interpretations of the components. Now suppose the scaling ensures that $\|\mathbf{a}_1\|^2 = \|\mathbf{b}_1\|^2 = \|\mathbf{a}_2\|^2 = \|\mathbf{b}_2\|^2 = 1$ and $\|\mathbf{c}_1\|^2$ is slightly higher than $\|\mathbf{c}_2\|^2$. Then the second sample could give a solution $\mathbf{A} = (\mathbf{a}_2 + \boldsymbol{\varepsilon}_1\ \mathbf{a}_1 + \boldsymbol{\varepsilon}_2)$, $\mathbf{B} = (\mathbf{b}_2 + \boldsymbol{\varepsilon}_3\ \mathbf{b}_1 + \boldsymbol{\varepsilon}_4)$ and $\mathbf{C} = (\mathbf{c}_2 + \boldsymbol{\varepsilon}_5\ \mathbf{c}_1 + \boldsymbol{\varepsilon}_6)$, where $\boldsymbol{\varepsilon}_1, \ldots, \boldsymbol{\varepsilon}_6$ denote vectors of appropriate order and with relatively small elements; the order of the components has changed because, apparently, $\|\mathbf{c}_2 + \boldsymbol{\varepsilon}_5\|^2$ is slightly higher than $\|\mathbf{c}_1 + \boldsymbol{\varepsilon}_6\|^2$. Now comparing these two solutions, as far as interpretation is concerned, we conclude that we have roughly the same solutions: one dimension with component vectors roughly equal to $\{\mathbf{a}_1, \mathbf{b}_1, \mathbf{c}_1\}$ and one with component vectors roughly equal to $\{\mathbf{a}_2, \mathbf{b}_2, \mathbf{c}_2\}$. However, if we compare the first components with the first and the second components with the second in the two fully identified solutions, we would here conclude that we are dealing with two entirely different solutions. Thus, if for such data, where the first and second components are roughly equally important, we use fully identified solutions with the bootstrap procedure, we would get roughly half of the bootstrap solutions resembling $\mathbf{A} = (\mathbf{a}_1\ \mathbf{a}_2)$, $\mathbf{B} = (\mathbf{b}_1\ \mathbf{b}_2)$ and $\mathbf{C} = (\mathbf{c}_1\ \mathbf{c}_2)$, whereas the others would more resemble the 'reverse' solution $\mathbf{A} = (\mathbf{a}_2\ \mathbf{a}_1)$, $\mathbf{B} = (\mathbf{b}_2\ \mathbf{b}_1)$ and $\mathbf{C} = (\mathbf{c}_2\ \mathbf{c}_1)$. In such cases, all bootstraps would actually pertain to very similar solutions, and a proper conclusion would be that the sample solution is very stable across bootstrap sampling. However, if we compute percentile intervals, we would get very broad intervals, because for each component value the bootstrap solutions do differ considerably (resembling either $\mathbf{A} = (\mathbf{a}_1\ \mathbf{a}_2)$, $\mathbf{B} = (\mathbf{b}_1\ \mathbf{b}_2)$, $\mathbf{C} = (\mathbf{c}_1\ \mathbf{c}_2)$, or $\mathbf{A} = (\mathbf{a}_2\ \mathbf{a}_1)$, $\mathbf{B} = (\mathbf{b}_2\ \mathbf{b}_1)$, $\mathbf{C} = (\mathbf{c}_2\ \mathbf{c}_1)$), and we would then conclude that the solution is very unstable owing to sampling fluctuation. However, in this case the sampling fluctuation is only caused by the, for interpretational purposes uninteresting, identification of the solution.

A similar situation may arise if we identify the sign of components by requiring column sums to be positive; then, if some such column sums are close to zero, the identification may lead to differences across bootstrap samples that are merely caused by the identification and not by differences that pertain to really different solutions. Therefore, for interpretational purposes, we should not 'overidentify' our solution, but rather use the transformational freedom in the three-way solutions in such a way that the bootstrap procedure makes appropriate comparisons of bootstrap solutions.

3.2. Using freedom of scaling and permutation

In Section 3.1 it has been described how overidentification of a solution can lead to percentile intervals that are much too broad. In fact, it was described that bootstrap solutions that are similar *up to* a permutation and scaling should indeed be considered as similar also when we compute percentile intervals. This can be attained when we use the freedom in

scaling and permutation of all bootstrap solutions such that they become as similar as possible. This idea has also been used by Riu and Bro [10] in comparing their CANDECOMP/PARAFAC jackknife solutions. One way of doing so would be to design a procedure that finds those scalings and permutations that optimize the average similarity of all bootstrap samples. A straightforward way of doing this is by using the actual sample solution as a reference solution. That is, by transforming all bootstrap solutions such that they optimally resemble the sample solution, it will be ascertained that they also resemble each other well (see also Reference [10]). In principle, it is possible to develop a procedure for optimizing the average similarity of all bootstrap samples, but this would increase the necessary computational effort and would also make comparison with the actual sample solution less straightforward, so this route is not followed here. Specifically, it is here suggested to use the following procedure.

Let **A**, **B** and **C** denote the CANDECOMP/PARAFAC solution for our actual sample data and let $\mathbf{A}^b$, $\mathbf{B}^b$ and $\mathbf{C}^b$ denote a bootstrap solution. The scale of the columns is identified by scaling the columns of the B- and C-mode component matrices to unit sum of squares. The A-mode is the sampling mode, so the bootstrap sample is a sample with replacement from the original set of A-mode entities, and hence the matrices **A** and $\mathbf{A}^b$ are not directly comparable. We are interested in bootstrap percentile intervals for the elements of the B- and C-mode component matrices, so the bootstrap solutions $\mathbf{B}^b$ and $\mathbf{C}^b$ must be made optimally comparable to **B** and **C** respectively. Then the components in $\mathbf{B}^b$ and $\mathbf{C}^b$ are first jointly permuted such that their columns become optimally similar to those of **B** and **C** in the sense that

$$\sum_{r=1}^{R} |\varphi(\mathbf{b}_r, \mathbf{b}_r^b)|\,|\varphi(\mathbf{c}_r, \mathbf{c}_r^b)| \tag{5}$$

is maximal over all possible joint permutations of the columns of $\mathbf{B}^b$ and $\mathbf{C}^b$; here $\varphi(\mathbf{x}, \mathbf{y})$ denotes Tucker's [16] congruence coefficient (defined as $\mathbf{x}'\mathbf{y}/\|\mathbf{x}\|\|\mathbf{y}\|$). The best permutation is found by simply trying all permutations. Note that in (5) the absolute values of φ are taken so as to ensure that the sign of the components does not affect the choice of the optimal permutation. Next the columns of $\mathbf{B}^b$ and $\mathbf{C}^b$ are reflected (multiplied by -1) if necessary to ensure that $\varphi(\mathbf{b}_r, \mathbf{b}_r^b)$ and $\varphi(\mathbf{c}_r, \mathbf{c}_r^b)$, $r = 1, \ldots, R$, all become non-negative. Note that, when these permutations and reflections are compensated for in $\mathbf{A}^b$, then the fit of the bootstrap solution is not affected. Thus we have used the remaining transformational freedom in our CANDECOM/PARAFAC solutions to make the bootstrap solutions optimally similar to the actual sample solution. Note that, in practice, we are not interested in $\mathbf{A}^b$, so we do not actually carry out such compensating transformations. Also note that the above procedure is not the only procedure to make bootstrap CANDECOM/PARAFAC solutions optimally similar to the actual sample solution. In fact, Riu and Bro [10] use a different procedure, also employing the freedom in scale of the components, but now using it to scale the components jointly such that they optimally resemble their sample counterparts. In our procedure the freedom in scale was 'identified away' by scaling the matrices $\mathbf{B}^b$ and $\mathbf{C}^b$ to unit column sums of squares. It can be expected, however, that this procedure will give results very similar to the one where the freedom in scale is also used.

In Section 3.1 it was mentioned that the Tucker3 analysis can also be identified up to permutation and scaling, by using for instance the principal axes solution or a simple structure rotation of the solution. In the case of Tucker3 analysis we have an actual sample solution consisting of **A**, **B**, **C** and $\underline{\mathbf{G}}$, and bootstrap solutions $\mathbf{A}^b$, $\mathbf{B}^b$, $\mathbf{C}^b$ and $\mathbf{G}^b$. Now we want to make $\mathbf{B}^b$, $\mathbf{C}^b$ and $\mathbf{G}^b$ optimally similar to **B**, **C** and $\underline{\mathbf{G}}$ respectively, in some sense, by again using permutations and scalings. Usually the component matrices are scaled to have unit sums of squares, so it remains to find permutations and reflections of the bootstrap solutions to make $\mathbf{B}^b$, $\mathbf{C}^b$ and $\mathbf{G}^b$ optimally similar to **B**, **C** and $\underline{\mathbf{G}}$. Since our primary interest is usually in the component matrices, it is chosen here to first find optimal permutations and reflections of the columns of $\mathbf{B}^b$ and $\mathbf{C}^b$. Since any transformation can be compensated in the core, we no longer need joint permutations and we can separately find the permutation of the columns of $\mathbf{B}^b$ that maximizes $\sum_q |\varphi(\mathbf{b}_q, \mathbf{b}_q^b)|$, and the reflections of these columns that make $\varphi(\mathbf{b}_q, \mathbf{b}_q^b)$, $q = 1, \ldots, Q$, non-negative, and next do analogously for the columns of $\mathbf{C}^b$. These permutations and reflections must then be appropriately compensated in the core $\underline{\mathbf{G}}^b$. Having thus made $\mathbf{B}^b$ and $\mathbf{C}^b$ optimally similar to **B** and **C** and compensated for these transformations in $\underline{\mathbf{G}}^b$, we now use the remaining freedom (i.e. permutation and reflection of columns of $\mathbf{A}^b$ and hence rows of $\mathbf{G}^b$) to make the current version of $\underline{\mathbf{G}}^b$ optimally similar to $\underline{\mathbf{G}}$. For this purpose we first permute the rows of $\mathbf{G}^b$ such that $\sum_p |\varphi(\mathbf{g}_p, \mathbf{g}_p^b)|$ is maximal, where $\mathbf{g}_p$ and $\mathbf{g}_p^b$ denote the pth transposed row of **G** and $\mathbf{G}^b$ respectively. Next we reflect the rows of $\mathbf{G}^b$ as far as necessary to make $\varphi(\mathbf{g}_p, \mathbf{g}_p^b)$, $p = 1, \ldots, P$, non-negative. These permutations and reflections should be compensated in $\mathbf{A}^b$, but, since we are not interested in $\mathbf{A}^b$, we do not actually carry out such compensating transformations. It should again be noted that the above procedure is not the only procedure to make such bootstrap Tucker3 solutions optimally similar to the actual sample solution. In fact, a criterion could have been devised in which similarities of $\mathbf{B}^b$ to **B**, $\mathbf{C}^b$ to **C**, and $\underline{\mathbf{G}}^b$ to $\underline{\mathbf{G}}$ were combined. However, the optimization of such a criterion does not seem straightforward. Moreover, it can be expected that such a procedure will not give very different results, especially not when, after permutation and reflection, the bootstrap solutions differ very little, hence when the optimal permutations and reflections are very clearly defined.

The above procedures have employed the remaining freedom in CANDECOMP/PARAFAC solutions and in Tucker3 solutions in which the rotations of the component matrices have been identified. In the next subsection, procedures are described that exploit the full rotational and transformational freedom of the Tucker3 solution.

3.3. Using freedom of transformation (only for Tucker3 analysis)

In Section 3.2 it has been described how Tucker3 solutions can be first partially identified by using the principal axes solution or a simple structure rotation, and the remaining freedom can be used to make the bootstrap solutions

optimally similar to their actual sample counterpart. However, it is possible that the principal axes or simple structure identification is not as 'strong' as one might hope. That is, it is possible that bootstrap solutions differ considerably even when rotations exist that make them almost equal. For example, it is possible that we get bootstrap solutions $\mathbf{B}^1$, $\mathbf{C}^1$ and $\underline{\mathbf{G}}^1$, and $\mathbf{B}^2$, $\mathbf{C}^2$ and $\underline{\mathbf{G}}^2$ for which $\mathbf{G}^1(\mathbf{B}^1 \otimes \mathbf{C}^1)$ and $\mathbf{G}^2(\mathbf{B}^2 \otimes \mathbf{C}^2)$ are very similar indeed but yet $\mathbf{B}^1$ and $\mathbf{B}^2$, and $\underline{\mathbf{G}}^1$ and $\underline{\mathbf{G}}^2$ are not at all similar. This is possible in the case of the principal axes solution in the following way. Remember that in the principal axes solution the columns of each component matrix are eigenvectors of a particular matrix (see Section 3.1), associated with the eigenvalues of this matrix; moreover, the eigenvectors are given in the order of the eigenvalues with which they are associated, and the eigenvalues are ordered from high to low. Now, if two of the columns of $\mathbf{B}$ are associated with almost equal eigenvalues, then the ordering of the eigenvectors will not be very stable over perturbations of the data: a small perturbation of the data will not affect the eigenvectors themselves very much, but it may reverse the order when the size of the associated eigenvalues happens to reverse in order. Hence, if bootstrap samples are drawn, for some bootstrap samples the columns of $\mathbf{B}$ will be similar to those in the original matrix $\mathbf{B}$, but for others the columns of $\mathbf{B}$ have reversed order. This reversed order will also affect the core matrices (because these depend on $\mathbf{B}$), and this will ensure that, when $\mathbf{B}^1$ and $\mathbf{B}^2$ become quite different only owing to such near equality of eigenvalues, $\mathbf{G}^1(\mathbf{B}^1 \otimes \mathbf{C}^1)$ and $\mathbf{G}^2(\mathbf{B}^2 \otimes \mathbf{C}^2)$ are still very similar. In the case of a simple structure rotated solution a similar thing may happen when two rather different simple structure rotations lead to roughly the same value of the simplicity criterion optimized by the simple structure rotation procedure. Then in some bootstraps the one orientation will be found, in others the other.

We have seen above that bootstrap Tucker3 solutions may differ considerably even when the model parts $\mathbf{G}^1(\mathbf{B}^1 \otimes \mathbf{C}^1)$ and $\mathbf{G}^2(\mathbf{B}^2 \otimes \mathbf{C}^2)$ are very similar. If one very strictly adheres to the interpretation entailed by the principal axes solution or the particular simple structure solution that was obtained, then such differences are important differences across bootstraps. However, more often one will consider such bootstrap solutions as in fact very similar. To handle this situation, it is appropriate to consider as similar all bootstrap solutions that are similar after an optimal transformation towards each other or to a reference solution. This idea has repeatedly been used in bootstrap or jackknife procedures for two-way analysis techniques [17–21]. The two-way techniques cannot as such be used in the three-way situation. Therefore, here, two new procedures for transformations to make three-way bootstrap solutions optimally similar to the sample solution will be proposed. The first uses 'only' rotational freedom (leaving intact the orthonormality of the component matrices); the other uses the full transformational freedom in the Tucker3 model.

3.3.1. *Exploiting rotational freedom*

Let a Tucker3 solution be given by $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\underline{\mathbf{G}}$, and bootstrap solutions be indicated by $\mathbf{A}^b$, $\mathbf{B}^b$, $\mathbf{C}^b$ and $\underline{\mathbf{G}}^b$. As in the usual solutions the component matrices are column-wise orthonormal. Now we want to transform $\mathbf{B}^b$, $\mathbf{C}^b$ and $\underline{\mathbf{G}}^b$ such that they become optimally similar to $\mathbf{B}$, $\mathbf{C}$ and $\underline{\mathbf{G}}$ respectively and remain column-wise orthonormal. Thus we search for orthonormal rotation matrices $\mathbf{S}$, $\mathbf{T}$ and $\mathbf{U}$ such that $\mathbf{B}^b\mathbf{T}$, $\mathbf{C}^b\mathbf{U}$ and $\mathbf{S}^{-1}\mathbf{G}(\mathbf{U}^{-1} \otimes \mathbf{T}^{-1})' = \mathbf{S}'\mathbf{G}(\mathbf{U} \otimes \mathbf{T})$ become optimally similar to $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{G}$ respectively. As in the case of finding optimal permutations and reflections for Tucker3 solutions, we primarily focus on the similarity of the bootstrap component matrices to their actual sample counterparts, and only after that on the similarity of the bootstrap core to the actual sample core. Specifically, we first find the (orthonormal) rotation matrices $\mathbf{T}$ and $\mathbf{U}$ that minimize

$$f_1(\mathbf{T}) = ||\mathbf{B}^b\mathbf{T} - \mathbf{B}||^2 \tag{6}$$

and

$$f_2(\mathbf{U}) = ||\mathbf{C}^b\mathbf{U} - \mathbf{C}||^2 \tag{7}$$

The optimal $\mathbf{T}$ and $\mathbf{U}$ are obtained via the singular value decompositions $\mathbf{B}'\mathbf{B}^b = \mathbf{P}_\mathbf{T}\mathbf{D}_\mathbf{T}\mathbf{Q}_\mathbf{T}'$ and $\mathbf{C}'\mathbf{C}^b = \mathbf{P}_\mathbf{U}\mathbf{D}_\mathbf{U}\mathbf{Q}_\mathbf{U}'$ as $\mathbf{T} = \mathbf{Q}_\mathbf{T}\mathbf{P}_\mathbf{T}'$ and $\mathbf{U} = \mathbf{Q}_\mathbf{U}\mathbf{P}_\mathbf{U}'$; see Reference [22] or Reference [23] for a formulation in terms of eigenvectors. Thus $\mathbf{B}^b$ will be replaced by $\mathbf{B}^b\mathbf{T}$, and $\mathbf{C}^b$ by $\mathbf{C}^b\mathbf{U}$. These rotations do not affect the fit if they are compensated for in the core. This is done by replacing the core $\mathbf{G}^b$ by $\mathbf{G}^b(\mathbf{U} \otimes \mathbf{T})$. Next the rotational freedom of $\mathbf{A}^b$ is used to make the resulting $\mathbf{G}^b$ optimally similar to the actual sample core matrix $\mathbf{G}$. This is done by minimizing

$$f_3(\mathbf{S}) = ||\mathbf{S}'\mathbf{G}^b - \mathbf{G}||^2 \tag{8}$$

over orthonormal matrices $\mathbf{S}$. The optimal $\mathbf{S}$ is obtained via the singular value decomposition $\mathbf{G}\mathbf{G}^{b\prime} = \mathbf{P}_\mathbf{S}\mathbf{D}_\mathbf{S}\mathbf{Q}_\mathbf{S}'$ as $\mathbf{S} = \mathbf{Q}_\mathbf{S}\mathbf{P}_\mathbf{S}'$. Hence $\mathbf{G}^b$ is replaced by $\mathbf{S}'\mathbf{G}^b$ and, to ensure that the fit is not affected, $\mathbf{A}^b$ is replaced by $\mathbf{A}^b\mathbf{S}$, although the latter does not have to be carried out in practice, since we do not use $\mathbf{A}^b$.

This concludes the description of our procedure for exploiting rotational freedom to make bootstrap Tucker3 solutions (in a particular sense) optimally similar to the actual sample solution. The ensuing rotated bootstrap solutions are then used for computing percentile intervals (and hence estimated confidence intervals) for all parameters in the sample. It should again be noted that the above procedure is not the only procedure to make such bootstrap Tucker3 solutions optimally similar to the actual sample solution. In fact, a criterion could have been devised in which similarities of $\mathbf{B}^b$ to $\mathbf{B}$, $\mathbf{C}^b$ to $\mathbf{C}$, and $\underline{\mathbf{G}}^b$ to $\underline{\mathbf{G}}$ were combined, but the complexities of such an approach have been avoided here, because again little gain is expected by such a more complex procedure.

3.3.2. *Exploiting full transformational freedom*

Although using column-wise orthonormal component matrices is rather common in the practice of Tucker3 analysis, there is no intrinsic reason to do so. As we have seen in (4), the Tucker3 model is unaffected by a joint non-singular transformation of the component matrices and the core. This transformational freedom can be exploited for making bootstrap solutions optimally comparable, by a simple adjustment of the procedure sketched in Section 3.3.1, as follows.

We want to transform $\mathbf{B}^b$, $\mathbf{C}^b$ and $\underline{\mathbf{G}}^b$ such that they become optimally similar to $\mathbf{B}$, $\mathbf{C}$ and $\underline{\mathbf{G}}$ respectively; thus we search

for non-singular matrices **S**, **T** and **U** such that $\mathbf{B}^b\mathbf{T}$, $\mathbf{C}^b\mathbf{U}$ and $\mathbf{S}^{-1}\mathbf{G}(\mathbf{U}^{-1}\otimes\mathbf{T}^{-1})'$ become optimally similar to **B**, **C** and **G** respectively. As in Section 3.3.1, we primarily focus on the similarity of the bootstrap component matrices to their actual sample counterparts, hence we first find the non-singular matrices **T** and **U** that minimize (6) and (7) over arbitrary matrices **T** and **U**, which will in practice turn out to be non-singular always. The optimal **T** and **U** are then obtained by ordinary regression as

$$\mathbf{T} = (\mathbf{B}^{b\prime}\mathbf{B}^b)^{-1}\mathbf{B}^{b\prime}\mathbf{B} \tag{9}$$

and

$$\mathbf{U} = (\mathbf{C}^{b\prime}\mathbf{C}^b)^{-1}\mathbf{C}^{b\prime}\mathbf{C} \tag{10}$$

Thus $\mathbf{B}^b$ will be replaced by $\mathbf{B}^b\mathbf{T}$, and $\mathbf{C}^b$ by $\mathbf{C}^b\mathbf{U}$, and these transformations are compensated for in the core by replacing the core $\mathbf{G}^b$ by $\mathbf{G}^b(\mathbf{U}^{-1}\otimes\mathbf{T}^{-1})'$. Next the rotational freedom of $\mathbf{A}^b$ is used to make the resulting $\mathbf{G}^b$ optimally similar to the actual sample core matrix **G** by minimizing $\tilde{\mathbf{S}}||\mathbf{G}^b - \mathbf{G}||^2$ over arbitrary matrices $\tilde{\mathbf{S}}$. The solution for $\tilde{\mathbf{S}}$ (again obtained by ordinary regression) is given by

$$\tilde{\mathbf{S}} = \mathbf{G}\mathbf{G}^{b\prime}(\mathbf{G}^b\mathbf{G}^{b\prime})^{-1} \tag{11}$$

Hence $\mathbf{G}^b$ is replaced by $\tilde{\mathbf{S}}\mathbf{G}^b$ and, to ensure that the fit is not affected, $\mathbf{A}^b$ is replaced by $\mathbf{A}^b\tilde{\mathbf{S}}^{-1}$, although the latter again need not be carried out in practice, since we do not use $\mathbf{A}^b$.

The present procedure using full transformational freedom has the advantage over the procedure using only rotational freedom that it uses all freedom available in the bootstrap solutions. However, in practice it may sometimes turn out to be problematic, because it is possible that the transformation matrices become nearly singular, and then the inverses of such matrices become highly unstable. Therefore, particularly in this case, a procedure that would simultaneously optimize similarities of $\mathbf{B}^b$ to **B**, $\mathbf{C}^b$ to **C**, and $\underline{\mathbf{G}}^b$ to $\underline{\mathbf{G}}$ might turn out to give different and sometimes better results. However, in this case the problem of maximizing such a criterion seems most difficult. Again such difficulties are avoided here. Instead it is tested how well these procedures work, and if they seem to work well in practice, there may be little need to improve upon the methods.

3.4. Interpreting percentile intervals as estimates of confidence intervals

Four procedures have been described above for computing bootstrap percentile intervals for all parameters resulting from a CANDECOMP/PARAFAC or Tucker3 solution. These percentile intervals are considered estimates of confidence intervals. This means that, if we have a 95% percentile interval, we would like to conclude from this that with 95% certainty it contains the true population parameter. If we work with fully identified solutions, then it is clear what the actual population parameters refer to. However, in the present section we have used different forms of transformational freedom, and we should obviously take this into account when interpreting our percentile intervals as estimates of confidence intervals. For the three situations the interpretations are as follows.

If we only allow for permutations and reflections of the components, then we expect that, in 95% of all possible samples from our population, after optimal permutations and reflections of the population components towards the sample components, the population parameters will fall in the confidence intervals we set up. It may come as a surprise to find in our statement that the *population* components are to be transformed optimally towards the sample components. The reason is that the sample solution is the only solution that we actually have, and hence we are forced to take this solution as our reference solution. Our confidence intervals will necessarily be built around this sample solution. Therefore, if we want to describe to what extent the unknown population solution agrees with a confidence interval, we have to imagine that the population solution will be transformed towards the sample solution (and not the other way around).

If we allow for rotation of the component matrices and the core, then we expect that, in 95% of all possible samples from our population, after optimal rotation of the population component matrices and core towards their sample counterparts, the population parameters will fall in the confidence intervals we set up. Likewise, if we allow for non-singular transformations of the component matrices and the core, then we expect that, in 95% of all possible samples from our population, after optimal non-singular transformations of the population component matrices and the core towards their sample counterparts, the population parameters will fall in the confidence intervals we set up.

4. AN APPLICATION: BOOTSTRAP CONFIDENCE INTERVALS FOR RESULTS FROM A TUCKER3 ANALYSIS

To give an impression of the kind of confidence intervals that one can get with a three-way analysis, bootstrap confidence intervals were computed for the results from a Tucker3 analysis reported by Kiers [24]. The data set was described by Nomikos and MacGregor [25]. It consists of (simulated) measurements on 52 batches (A-mode) with respect to nine variables (B-mode) at 200 consecutive time points with 5 min intervals (C-mode). The first 50 batches are more or less 'normally behaving' batches, whereas the 51st and 52nd are known to behave abnormally. The nine variables, as described by Nomikos and MacGregor [25] are: (1) flow rates of styrene, (2) flow rates of butadiene, (3) temperature of the feeds, (4) temperature of the reactor, (5) temperature of the cooling water, (6) temperature of the reactor jacket, (7) density of the latex in the reactor, (8) total conversion and (9) instantaneous rate of energy release.

Just as in Reference [24], the data were preprocessed by centering across the 52 batches and by normalizing each of the variables (i.e. across batches *and* time points) such that for each variable the sum of squares was unity. The thus preprocessed data were analyzed by Tucker3 analysis using two components for each mode, which was considered the most fruitful choice by Kiers [24]. The Tucker3 solution gave a fit percentage of 22.0%. To identify the solution, here it was chosen to first rotate the matrices **A** and **B** to simple structure by means of varimax; the matrix **C** was not rotated to simple structure, because this is usually inappropriate for components referring to a time mode. Instead the remaining rotational freedom was used to rotate **C** such that the core was as simple as possible according to the varimax criterion.

Table I. Variable component matrix from Tucker3 analysis of Nomikos and MacGregor [25] data, with 95% percentile intervals in brackets

Flow rates of styrene	−0.01 [−0.09, 0.06]	−0.00 [−0.10, 0.10]
Flow rates of butadiene	−0.10 [−0.16, 0.12]	0.12 [0.01, 0.48]
Temperature of feeds	0.00 [−0.03, 0.02]	−0.00 [−0.05, 0.03]
Temperature of reactor	−0.03 [−0.14, −0.01]	−0.04 [−0.29, −0.02]
Temperature of cooling water	0.03 [−0.02, 0.05]	**0.58** [0.48, 0.62]
Temperature of reactor jacket	0.03 [−0.05, 0.07]	**0.62** [0.47, 0.69]
Density of latex in reactor	**0.78** [0.69, 0.82]	0.06 [−0.06, 0.13]
Total conversion	**0.62** [0.56, 0.70]	−0.08 [−0.19, 0.03]
Rate of energy release	0.04 [−0.03, 0.15]	−**0.50** [−0.60, −0.27]

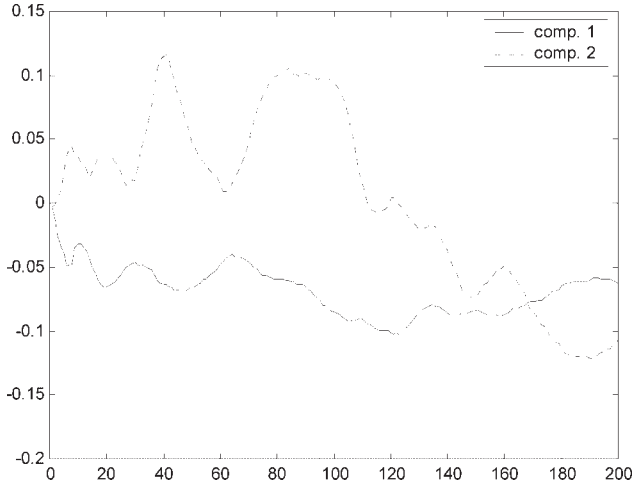

Figure 2. Plot of time components from Tucker3 analysis of Nomikos and McGregor [25] data.

The resulting solutions for the variable component matrix **B** and the time component matrix **C** are given in Table I and Figure 2 respectively. The core array is given in Table II. It can be seen that the variable component matrix has a clear simple structure (as emphasized by the loadings higher than 0.50 in absolute sense being set in bold-face). The first component is mainly related to the density of the latex and the total conversion, while the second is mainly related to some of the temperatures and (inversely) to the rate of energy release. The time components differ mainly in that the first gives a rather stable overall trend, while the second is more fluctuating. The core array has various high elements; an interpretation of these is not given here, since this would take us too far from our main goal now, which is illustrating the use of bootstrap percentile intervals.

To obtain bootstrap percentile intervals, the procedure described in Section 3.3.1 was used. Here we consider the batches as a random sample from a population, and therefore 500 bootstrap samples were drawn from the sample of 52 batches. For each of these bootstrap samples the same preprocessing and the same Tucker3 analysis were carried out as for the original sample. Next the bootstrap sample solutions were optimally rotated towards the original sample solution by the procedure described in Section 3.3.1. Finally, 95% percentile intervals were obtained for each element of the variable component matrix **B**, the time component matrix **C** and the core array. The resulting 95% percentile intervals for the elements of **B** and the core are given in brackets in Tables I and II. It can be seen that the loadings that play a key role in the interpretation of the B-mode components usually have fairly small percentile intervals (compared with their sizes) and hence seem rather accurate estimates of the population loadings; the main exception to this is the second loading of rate of energy release. On the other hand, some of the small loadings have broad percentile intervals, in particular the second loading of flow rates of butadiene, which was only 0.12 in the sample but has a percentile interval running from 0.01 to 0.48. If these percentile intervals are to be interpreted as confidence intervals, it can be concluded that the first component seems a rather accurate estimate of the first population component, while the second is a little less accurate, especially as far as the contributions of flow rates of butadiene and rate of energy release are concerned.

As to the core array, most percentile intervals tend to be rather wide, hence most core values are not very stable; the clearest exception is the highest value in the top left, which is by far the highest, even if we consider the lower bound of its percentile interval. Furthermore, it can be seen that most percentile intervals indicate that the sign of the core value can be taken seriously: in only one case (the core value 6.5) does the interval start negatively and end positively.

For the time component matrix (**C**) the lower and upper bounds of the 95% percentile intervals are drawn in the left- and right-hand plots in Figure 3, with the original component values plotted in between them. Thus we get confidence bands for the time component values. Clearly the bands for the second component are much wider than those for the first. Moreover, for the second component the bands are remarkably wide in the first time period and gradually converge as time proceeds.

Table II. Core array from Tucker3 analysis of Nomikos and MacGregor [25] data, with 95% percentile intervals in brackets

	Time component 1		Time component 2	
	Var. comp. 1	Var. comp. 2	Var. comp. 1	Var. comp. 2
Batch component 1	104.6 [84.7, 115.0]	−31.4 [−53.9, −10.4]	6.5 [−8.4, 23.3]	20.3 [4.5, 43.7]
Batch component 2	−21.7 [−45.6, −6.9]	61.6 [21.3, 84.8]	−49.8 [−73.7, −8.2]	38.7 [27.5, 66.9]

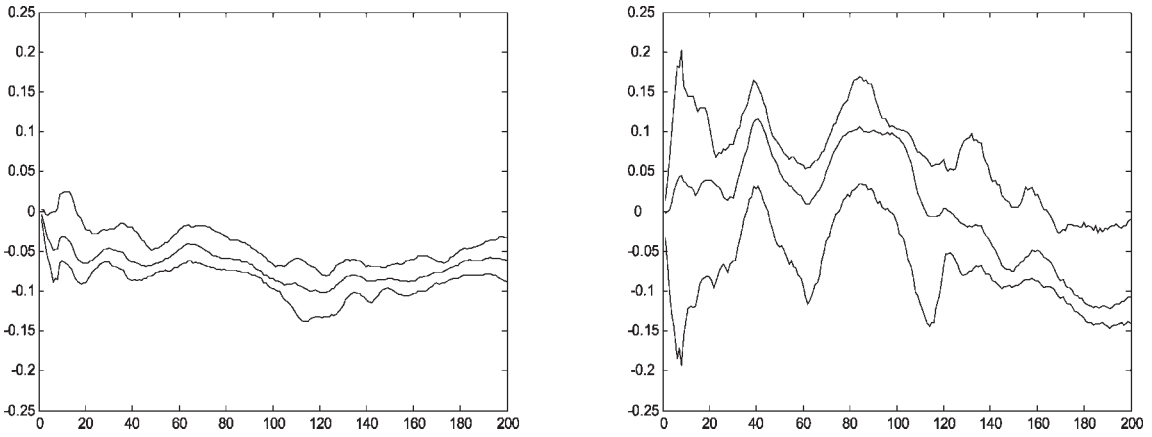

Figure 3. Plots of 95% percentile intervals for time components 1 (left) and 2 (right) from Tucker3 analysis of Nomikos and McGregor [25] data.

The present analyses were not meant to give an in depth analysis of the data set considered here, but to demonstrate how bootstrap percentile intervals can be obtained and what they may look like. The main question, however, still is how well these percentile intervals approximate confidence intervals. This has been investigated by means of the simulation study reported in the next section.

5. A SIMULATION STUDY FOR EVALUATING THE QUALITY OF BOOTSTRAP CONFIDENCE INTERVALS

5.1. Main study

5.1.1. Design

Several bootstrap procedures have been proposed above for obtaining percentile intervals for the parameters resulting from a CANDECOMP/PARAFAC or Tucker3 solution, and in Section 3.4 it has been described how these are to be interpreted as estimated confidence intervals of population parameters. That is, if we have a 95% percentile interval, we would like to conclude from this that with 95% certainty it contains the true population parameter (after proper transformation of the population solution). To test whether the percentile intervals work as such, a simulation study was set up in which a number of population data sets were created and analyzed by CANDECOMP/PARAFAC or Tucker3 analysis, samples were drawn from this population, the sample data were analyzed by CANDECOMP/PARAFAC or Tucker3 analysis, a bootstrap confidence interval was computed for each element of the solution of each sample, and each time it was checked for each population parameter whether it fell in the interval or not. If this held for approximately 95% of the cases that we checked, then the *coverage* of our interval would be approximately correct.

The first step in the simulation study was to create population data and a population solution for these data. This was done as follows. First, for each population a $(10\,000 \times P)$ matrix $\mathbf{A}$ with elements drawn randomly from the standard normal distribution was created, and fixed matrices $\mathbf{B}$ $(J \times Q)$, $\mathbf{C}$ $(K \times R)$ and, in the case of Tucker3, also a core array $\mathbf{G}$ $(P \times QR)$ were constructed in various different ways to cover a diversity of possible data situations. Specifically, for Tucker3 the following six choices for the sizes of these matrices were used.

1. $J=4,\ K=6,\ P=2,\ Q=2,\ R=2.$
2. $J=8,\ K=20,\ P=2,\ Q=2,\ R=2.$
3. $J=4,\ K=6,\ P=3,\ Q=3,\ R=3.$
4. $J=8,\ K=20,\ P=3,\ Q=3,\ R=3.$
5. $J=4,\ K=6,\ P=4,\ Q=3,\ R=2.$
6. $J=8,\ K=20,\ P=4,\ Q=3,\ R=2.$

The component matrices $\mathbf{B}$ and $\mathbf{C}$ were usually taken such that they had a reasonably simple structure. An example is

$$\mathbf{C} = \begin{pmatrix} 1 & 0.8 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0.8 & 1 \end{pmatrix}'$$

In one case, however, as indicated later, these matrices were taken to be rather smooth, e.g.

$$\mathbf{C} = \begin{pmatrix} 1 & 0.8 & 0.6 & 0.4 & 0.2 & 0 \\ 0 & 0.2 & 0.4 & 0.6 & 0.8 & 1 \end{pmatrix}'$$

The $2 \times 2 \times 2$ core was taken equal to

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \end{pmatrix}$$

the $3 \times 3 \times 3$ core was

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

and the $4 \times 3 \times 2$ core was

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

In the case of CANDECOMP/PARAFAC, only choices $1, \ldots, 4$ for the sizes of the matrices were possible; choices 5 and 6 are impossible, because in CANDECOMP/PARAFAC the number of components must be the same for all modes. For CANDECOMP/PARAFAC the component matrices were always chosen rather smooth. To be able to use one general set-up including a core matrix, for CANDECOMP/PARAFAC we used as matrix $\mathbf{G}$ the matricized version of the superidentity core array.

As described above, Tucker3 population data were often based on component matrices with simple structure, whereas CANDECOMP/PARAFAC data were only based on rather smooth component matrices. The reason for this is that CANDECOMP/PARAFAC applications often involve several smooth modes (e.g. spectra), whereas in Tucker3 applications finding simple components is often a useful objective; this is not to say, however, that the reverse situations cannot be encountered. To get some comparative information, for one particular method, data were generated based on both smooth and simple components, as explained below.

From the component matrices and core array obtained as described above, population data were computed as

$$\mathbf{X} = \mathbf{AG}(\mathbf{C} \otimes \mathbf{B})' + \varepsilon\mathbf{N} \tag{12}$$

where $\mathbf{N}$ is a $10\,000 \times JK$ matrix with elements drawn randomly from a standard normal distribution, which are next multiplied by a scalar such that $\|\mathbf{AG}(\mathbf{C} \otimes \mathbf{B})'\| = \|\mathbf{N}\|$; the scalar ε denotes the error level parameter, which was chosen equal to 0.3, 0.6 or 1.0. The data in (12) represent the population data.

The thus constructed population data were next analyzed by Tucker3 or CANDECOMP/PARAFAC. To mimic what is done in practice, the data were preprocessed before analysis. A common procedure for preprocessing in practice is to center across the mode representing the observation units, mode A, and to normalize within the mode representing the variables (which here is arbitrarily chosen to be mode B). Therefore here we always center across mode A and normalize within mode B.

To the thus preprocessed population data, CANDECOMP/PARAFAC or Tucker3 analysis was applied to obtain the population parameters for **B**, **C** and, in the case of Tucker3, **G**. These were denoted as $\mathbf{B}^{pop}$, $\mathbf{C}^{pop}$ and $\mathbf{G}^{pop}$. To keep computations feasible, in both cases only a single, rationally started, run of the common alternating least squares algorithm was used, with as convergence criterion that the difference between consecutive least squares loss function values became less than $10^{-4}\%$; in the case of CANDECOMP/PARAFAC the maximal number of iterations was set at 10 000.

From each of the above defined populations, random samples were drawn of different sizes ($I = 20$, 50 or 100), and on these sample data, exactly the same procedure was applied as to the population data (i.e. the same preprocessing step and the same algorithm, with the same settings). The solutions were denoted as $\mathbf{B}^{s}$, $\mathbf{C}^{s}$ and $\mathbf{G}^{s}$. Next, 95% percentile intervals were constructed using each of the procedures described in Section 3 (again using the same preprocessing steps and the same algorithms), in each case using $B = 500$ bootstrap samples. For the resulting 95% percentile intervals their relative widths (i.e. the width divided by the mean absolute value of the matrix at hand) were recorded and, most importantly, it was recorded whether, after optimally transforming the population solution towards the sample, the confidence interval contained the population parameter. Specifically, the population solution was transformed to the sample solution in the very same way as the bootstrap solutions were transformed towards the sample solution; the only difference was that, to take into account that the normalization of **A** (as used in Tucker3) has different effects in the population than in the sample and will hence affect the size of the population core elements, the population core elements were adjusted for this size difference by multiplying them by $(I/10\,000)^{1/2}$. The coverage and the relative interval width were assessed for each element of each matrix, and for each matrix the average interval width and the percentage of covered population parameters were recorded. In addition, for each bootstrap analysis the computation time was recorded.

It has been described above that population data were constructed based on six (for Tucker 3) or four (for CANDECOMP/PARAFAC) size conditions. In each condition, 10 different population matrices were constructed for each of the three noise levels, so for a Tucker3 simulation study, $6 \times 3 \times 10 = 180$ different population matrices were created, whereas for a CANDECOMP/PARAFAC simulation study, $4 \times 3 \times 10 = 120$ different population matrices were created. From each of these population matrices, 10 sample matrices were drawn of size $I = 20$, 10 of size $I = 50$ and, only in the case of Tucker3, 10 of size $I = 100$. (For CANDECOMP/PARAFAC, bootstrap analyses of this size took very long, and hence have not been considered in the main study here, but they are considered separately later in Section 5.2.) Thus, in total, 5400 Tucker3 sample data sets and 2400 CANDECOMP/PARAFAC sample data sets were obtained. Each of these was next analyzed by the appropriate method, followed by 500 bootstrap sample analyses. For each different bootstrap procedure to be studied, a separate simulation study was carried out.

For CANDECOMP/PARAFAC, only one simulation study was carried out, employing the bootstrap procedure described in Section 3.2. For Tucker3 the following procedures were studied.

1. Tucker3 with principal axes solution, bootstraps optimally permuted and reflected.
2. Tucker3 with simple structure solution, bootstraps optimally permuted and reflected.
3. Tucker3 with simple structure solution, bootstraps optimally rotated.
4. Tucker3 with simple structure solution, bootstraps optimally transformed.

Procedures 1 and 2 have been described in Section 3.2, procedure 3 in Section 3.3.1 and procedure 4 in Section 3.3.2. In the case of simple structure solutions, every time the solution was chosen where **B** and **C** are optimally simple in the varimax [26] sense, and after having compensated the core for the varimax rotations of **B** and **C**, also the rows of the core are rotated by varimax. Procedures 1–4 were all tested on data constructed on the basis of simple component matrices. In addition, procedure 1 was tested on data constructed on the basis of rather smooth component matrices.

5.1.2. *Results*

5.1.2.1. Coverage. The main criterion of interest is the coverage percentage. This should be close to 95% for each individual parameter. To verify this for each individual parameter separately, we should use only samples from

the same population, because individual parameters are not comparable across populations (e.g. there is no relation between the first loading of the first variable on the first B-mode component across populations). However, from each population we have only 10 samples of the same size and hence only 10 comparable percentile intervals. Thus we would have to check whether the 10 percentile intervals contain the true parameter in 95% of the cases, which clearly cannot even approximately be the case (at best it is 90% or 100%). Thus with 10 samples we cannot verify accurately whether the procedure covers the population parameter in roughly 95% of the cases. Taking more samples, if this were practically still feasible, could be a solution, but then it should be realized that even as many as 100 samples still leads to a rather inaccurate assessment: if the true proportion were 0.95, then an estimate of the standard error of a sample proportion would be

$$\sqrt{\frac{0.95 \times 0.05}{100}} = 0.022$$

hence a 95% confidence interval based on this standard error would still give a margin of ± 0.05 to this proportion, or 5% when using percentages. Given that the simulations already take rather long, this does not seem a viable way to go. Moreover, *a priori*, there does not seem to be much reason to expect that percentile intervals for different parameters in the same component matrix will have different coverage properties. Therefore, rather than assessing coverage percentages for each individual parameter, average coverage percentages for parameter matrices were studied. That is, for each parameter matrix the percentage of covered population parameters was recorded, and these percentages were used as the criterion variable. Indeed, also these percentages should, ideally, be equal to 95% on average.

For each cell in the design ($6 \times 3 \times 3$ for Tucker3 and $4 \times 3 \times 2$ for CANDECOMP/PARAFAC), 100 coverage percentages for **B**, **C** and (only for Tucker3) **G** were available (i.e. for 10 samples from each of 10 populations) for six different procedures. Table III lists the overall average coverage percentages for each procedure for each parameter matrix. It is clear from Table III that the average coverage percentages for CANDECOMP/PARAFAC are close to the desired 95%. In fact, this may seem almost more than we could hope for, because the interpretation of percentile intervals as confidence intervals depends on unverifiable assumptions (see Section 2). However, the coverage properties could still depend strongly on various conditions, and only happen to be close to 95% when averaged across those conditions. Therefore it is important to qualify the results in Table III. Detailed insight could be gained by inspecting all 54 cell means underlying each of the 17 reported average percentages. However, analyses of variance revealed that main effects or interaction effects in the design rarely accounted for more than 5% (or even 1%) of the overall variance in the coverage percentages. Thus, in the great majority of cases, differences across conditions were small compared with those across replications within cells. If considerable differences across conditions were found (accounting for, say, more than 5%), these were found across the size conditions. Specifically, for the first, second and third Tucker3 procedures (principal axes permuted/reflected, simple structure permuted/reflected and simple structure optimally rotated) the difference across size conditions accounted for 5.2%, 10.6% and 11.3% respectively of the overall variance of the B-mode coverage percentages. When inspecting average coverage percentages for each of the size conditions (i.e. averaged across the other conditions), we found that these range from 85.8% to 93.9% for the first procedure and from 87.0% to 95.0% for both the second and the third procedure. Clearly, this gives a somewhat less favorable picture. Indeed, some conditions lead to a rather large amount of undercoverage of 10% on average. These conditions are not always the same.

The above analyses indicate that coverage percentages do differ across conditions and seem to warrant a more detailed study. Rather than presenting averages for all cells separately, we report the ranges of average cell percentages and the associated standard errors of the upper and lower bounds of these ranges in Table IV. Thus Table IV gives an indication of 'the worst that may happen' and also indicates how reliable such average percentages are. From Table IV it can be seen that the worst cases sometimes are quite far off from the ideal 95%. Not reported in Table IV is that, except for one case, all lowest percentages were obtained in cases with the smallest sample size ($I = 20$); however, for other sample sizes also, percentages that were considerably too low were found. The most problematic cases were found with the Tucker3 bootstrap procedure based on permutation and reflection only of the principal axes solution: percentages of only 72.1% and 75.8% can hardly be seen as approximating the ideal of 95%. The standard errors for these cases were quite high, leading to 95% error margins around the observed percentages of roughly $\pm 4\%$ and even $\pm 6\%$ respectively. However, even with such error margins it is clear that these values are considerably too low.

The best results are clearly obtained for the CANDECOMP/PARAFAC bootstrap procedure, which did not miss the ideal 95% by more than 3.5%. The other Tucker3 bootstrap

Table III. Average coverage percentages of various procedures for each of the parameter matrices

Procedure	**B**	**C**	**G**
Bootstrap for CANDECOMP/PARAFAC	95	94	—
Tucker3 principal axes, bootstraps permuted/reflected	91	85	87
Tucker3 simple structure, bootstraps permuted/reflected	92	94	93
Tucker3 simple structure, bootstraps optimally rotated	92	94	93
Tucker3 simple structure, bootstraps optimally transformed	95	94	95
Tucker3 principal axes, bootstraps permuted/reflected, non-simple **B** and **C** used in construction	95	94	94

procedures yielded results that deviated roughly 10% at most from the ideal 95%. The standard errors were of the order of 1% or 2% (implying 95% error margins of $\pm 2\%$ to $\pm 4\%$). Thus we see that the methods tend to suffer from undercoverage (which was actually observed quite frequently) of up to 10%. Overcoverage was less strong and observed less frequently.

From these results we can conclude that the 95% percentile intervals are fairly good approximations to 95% confidence intervals in most cases, although some improvement remains to be desired. The exception to this was the Tucker3 bootstrap procedure based on the principal axes solution. It should be noted, however, that these results were found for the case where the population data were based on **B** and **C** matrices with simple structure; for data based on smooth **B** and **C** matrices (last row in Tables III and IV), the resulting percentile intervals gave considerably better approximations to confidence intervals. The former data sets are the very cases for which difficulties with the principal axes solution could be expected. In fact, the permutation and reflection should eliminate this problem, but possibly the problem is so strong that it still determines the coverage of the intervals.

5.1.2.2. Interval width. The first four Tucker3 bootstrap procedures can be expected to give increasingly small confidence intervals. This is because the first procedure does not use the simple structure of the data, the next one only uses permutation and reflection to increase similarity of bootstrap samples, the third uses more freedom (full rotation), whereas the fourth uses most freedom (arbitrary transformations). To verify whether, for our simulated data, indeed the procedures give increasingly small percentile intervals, we compare the average interval widths in Table V. Note that here we did not consider the procedure of using Tucker3 principal axes followed by permutation and reflection applied to smooth data, because these results are based on a different kind of data, so that differences in interval widths of this procedure compared with the others may to some extent be caused by this different data generation process. It can be seen from Table V that, as expected, indeed the interval widths decrease on going down the table.

5.1.2.3. Computation time. Finally, some remarks are to be made about the computation time. Each single simulation study took a lot of time, but this is to a large extent caused by the fact that we needed many replications to get a decent estimate of the coverage percentages. However, a single bootstrap analysis requiring 500 three-way analyses did not turn out to take very long, especially not for Tucker3. To give some idea about the computation times of all procedures, in Table VI the average computation times for the *largest data sizes* in the design are reported. Analyses were

Table IV. Ranges of average coverage percentages within cells for various procedures for each of the parameter matrices; in parentheses, standard errors of lower and upper bounds of ranges

Procedure	**B**	**C**	**G**
Bootstrap for CANDECOMP/PARAFAC	91.5–98.2 (1.4, 0.3)	91.8–97.4 (0.4, 0.5)	—
Tucker3 principal axes, bootstraps permuted/reflected	81.2–96.8 (1.8, 0.7)	72.2–92.1 (2.0, 1.9)	75.8–92.8 (3.2, 1.6)
Tucker3 simple structure, bootstraps permuted/reflected	84.3–95.8 (0.8, 0.5)	91.8–96.5 (0.5, 0.6)	88.8–96.8 (1.6, 0.4)
Tucker3 simple structure, bootstraps optimally rotated	84.2–95.8 (0.8, 0.5)	92.0–95.7 (0.5, 0.7)	86.0–95.0 (1.9, 0.9)
Tucker3 simple structure, bootstraps optimally transformed	91.1–98.1 (1.9, 0.5)	92.1–95.6 (0.5, 0.7)	89.4–96.8 (1.5, 0.6)
Tucker3 principal axes, bootstraps permuted/reflected, non-simple **B** and **C** used in construction	90.7–98.1 (1.2, 0.5)	86.4–97.2 (2.8, 0.5)	85.4–98.3 (2.0, 0.3)

Table V. Mean (normalized) interval widths

Method	**B**	**C**	**G**
Tucker3 principal axes, bootstraps permuted/reflected	0.85	1.45	2.14
Tucker3 simple structure, bootstraps permuted/reflected	0.19	0.34	0.87
Tucker3 simple structure, bootstraps optimally rotated	0.19	0.34	0.83
Tucker3 simple structure, bootstraps optimally transformed	0.19	0.33	0.27

Table VI. Average computation times of 500 bootstrap analyses for largest data sizes

Method	Largest size	Mean computation time (s)
Bootstrap for CANDECOMP/PARAFAC	$I=50, J=8, K=20, R=3$	312.8
Tucker3 principal axes, bootstraps permuted/reflected	$I = 100, J = 8, K = 20, P \times Q \times R = 4 \times 3 \times 2$ and $P \times Q \times R = 3 \times 3 \times 3$	35.7
Tucker3 simple structure, bootstraps permuted/reflected		30.5
Tucker3 simple structure, bootstraps optimally rotated		20.8
Tucker3 simple structure, bootstraps optimally transformed		20.6
Tucker3 principal axes, bootstraps permuted/reflected, non-simple **B** and **C** used in construction		39.3

carried out on a 2.5 GHz Pentium 4 computer with 768 MB RAM; all analyses were run using MATLAB [27]. It can be seen that the computation times for the Tucker3 bootstrap analyses are by no means prohibitive: even for the medium sized data of order $100 \times 8 \times 20$, using quite a few components, the average computation time of a full bootstrap analysis (i.e. including the analyses of 500 bootstrap samples) takes roughly 0.5 min. For CANDECOMP/PARAFAC the situation is less positive: for data of size $50 \times 8 \times 20$, using three components required on average about 5 min. This is obviously not unfeasible, but some speed-up would be welcome here. This is studied in Section 6.

5.2. Some detailed studies

5.2.1. Some simulations for CANDECOMP/PARAFAC with $I = 100$

As mentioned in Section 5.1.1, using samples of size $I = 100$ CANDECOMP/PARAFAC in the full simulation study would make the simulation study extremely time-consuming. To yet get some insight into results for $I = 100$, simulations have been carried out for the first three size conditions, with noise level fixed at the middle level (0.6), thus extending the original $4 \times 2 \times 3$ design with three extra cells.

The coverage percentages were 96.1%, 93.1% and 95.3% for **B** and 94.5%, 94.6% and 95.1% for **C**. Upon comparing these values with the ranges reported in Table IV, it is clear that these results are similar to those for other sample sizes.

The computation times can best be evaluated when compared with the corresponding conditions for the smaller sample size $I = 50$. For the first condition ($J = 4$, $K = 6$, $R = 2$) the average computation time for $I = 50$ was 12 s and for $I = 100$ now is 44 s; the second condition ($J = 8$, $K = 20$, $R = 2$) gave 161 s for $I = 50$ and now 596 s for $I = 100$; the third condition ($J = 4$, $K = 6$, $R = 3$) gave 41 s for $I = 50$ and now 95 s for $I = 100$. Clearly, computation time has increased by a factor of 2–4. With 10 min in the worst case studied here, the bootstrap procedure is still feasible, although clearly some speed improvement would be desirable.

5.2.2. Some simulations with 100 rather than 10 samples

In Section 5.1.2.1 it was remarked that the reliability of the coverage percentages may not be as high as desired. Indeed, standard errors for the extreme cases were (in Table IV) seen to be sometimes as high as 2% or 3%. By taking 100 samples rather than 10 samples, we can expect these standard errors to decrease roughly by a factor $\sqrt{10}$, so they would no longer exceed 1%. To see whether these more reliable analyses would lead to similar or rather different results, we ran simulations for three cells in the designs for both the CANDECOMP/PARAFAC simulations and the Tucker3 simulations and compared coverage results. These are given in Table VII. It is clear that the differences are small and do not essentially alter the picture given before.

6. IMPROVING COMPUTATIONAL FEASIBILITY

From the results on computation times reported in Table VI, it can be seen that the computational problems are most severe for CANDECOMP/PARAFAC. In fact, the computational procedure used in the simulation study was relatively simple, using only one rational start. Still, some improvement can be obtained by starting the algorithms for the bootstrap runs using as a start the solution from the actual sample. Thus it can be expected that convergence will be attained more quickly. However, using this sample based start might also detract from the coverage, because it may yield bootstrap solutions that remain too close to the sample solution. To test the latter, a second simulation study was carried out in which two procedures were compared. The first 'optimal' procedure uses the rational and four randomly started runs to find the optimal solution for the sample and

Table VII. Average coverage percentages of various procedures for each of the parameter matrices (using 100 samples), in three particular cells of the design (values based on 10 samples in parentheses)

Procedure	Cell	**B**	**C**	**G**
Bootstrap for CANDECOMP/PARAFAC	Cond. 1, $I = 20$, $\varepsilon = 0.6$	93.8 (91.5)	92.3 (91.8)	—
	Cond. 2, $I = 20$, $\varepsilon = 1$	94.1 (94.4)	92.9 (92.8)	—
	Cond. 3, $I = 50$, $\varepsilon = 1$	96.5 (97.3)	95.7 (97.4)	—
Tucker3 principal axes, bootstraps permuted/reflected	Cond. 1, $I = 50$, $\varepsilon = 0.6$	93.7 (94.1)	85.1 (89.5)	92.7 (75.8)
	Cond. 3, $I = 20$, $\varepsilon = 1$	93.7 (94.1)	85.8 (85.1)	90.8 (90.3)
	Cond. 6, $I = 50$, $\varepsilon = 1$	82.0 (84.0)	84.1 (87.0)	87.2 (88.5)
Tucker3 simple structure, bootstraps permuted/reflected	Cond. 1, $I = 50$, $\varepsilon = 0.6$	93.6 (95.1)	93.8 (94.3)	93.6 (91.8)
	Cond. 3, $I = 20$, $\varepsilon = 1$	85.1 (84.7)	97.1 (96.5)	96.6 (96.8)
	Cond. 6, $I = 50$, $\varepsilon = 1$	94.4 (94.6)	93.8 (93.5)	93.4 (92.8)
Tucker3 simple structure, bootstraps optimally rotated	Cond. 1, $I = 50$, $\varepsilon = 0.6$	93.6 (95.3)	94.0 (94.0)	93.2 (91.5)
	Cond. 3, $I = 20$, $\varepsilon = 1$	84.1 (83.8)	96.1 (95.7)	93.9 (94.2)
	Cond. 6, $I = 50$, $\varepsilon = 1$	94.4 (94.7)	93.8 (93.4)	93.1 (92.2)
Tucker3 simple structure, bootstraps optimally transformed	Cond. 1, $I = 50$, $\varepsilon = 0.6$	93.7 (95.1)	93.9 (94.0)	93.7 (94.1)
	Cond. 3, $I = 20$, $\varepsilon = 1$	97.7 (97.0)	95.5 (95.3)	95.7 (96.3)
	Cond. 6, $I = 50$, $\varepsilon = 1$	94.4 (94.5)	93.8 (93.5)	95.4 (95.5)
Tucker3 principal axes, bootstraps permuted/reflected, non-simple **B** and **C** used in construction	Cond. 1, $I = 50$, $\varepsilon = 0.6$	93.8 (93.5)	93.9 (92.7)	89.8 (89.8)
	Cond. 3, $I = 20$, $\varepsilon = 1$	96.8 (97.8)	94.4 (94.2)	98.0 (97.6)
	Cond. 6, $I = 50$, $\varepsilon = 1$	95.1 (94.3)	94.6 (93.8)	95.8 (95.1)

Table VIII. Comparison of three procedures for CANDECOMP/PARAFAC bootstrap analysis (in all cases $J = 4$, $K = 6$ and $R = 3$)

Bootstrap procedure	Mean computation time (s)	Mean coverage **B** (%)	Mean coverage **C** (%)
Optimal procedure. Sample and bootstrap analyses using 5 runs	146.5	95.5	95.1
Fast procedure. Sample analysis using 5 runs, bootstrap analyses using sample solution as start	22.8	95.3	94.7
Original procedure. Sample and bootstrap analysis using 1 rationally started run	35.9	95.4	95.5

does the same to find the optimal solution for each bootstrap. The second 'fast' procedure uses only the rationally started run to find the optimal solution for the sample and uses the sample solution as a start in the (single run of the) algorithm used for each bootstrap analysis. The data construction was as in the earlier CANDECOMP/PARAFAC simulations, but as size condition only $J = 4$, $K = 6$, $R = 3$ was considered. The results are reported in Table VIII. For comparison, in Table VIII also the results with our original procedure, used in the simulation study reported in Section 5, are reported for the same conditions. From Table VIII it can be seen that the fast procedure is indeed considerably faster than the optimal procedure (by roughly a factor of 6) and also quite a bit faster than the original procedure (by roughly a factor of 1.6). The speed improvement is not really surprising. The main question, however, is whether the fast procedure still gives adequate coverage percentages. It can be seen that this is indeed the case: both coverage percentages deviate by only 0.3% from the ideal 95%. More importantly, the coverage percentages are not farther off from 95% than those by the original method (indeed, they are actually better than those from the original procedure) or those by the optimal method. It can hence be concluded on the basis of this simulation study that the fast procedure works well and is indeed faster than the original procedure and much faster than the optimal procedure. The optimal procedure seems to be needlessly time-consuming: the use of several runs of the algorithm does not appear to improve the coverage percentages.

Obviously, a similar speed improvement could be implemented for Tucker3. However, the algorithm usually converges fast, and little gain can be expected from using a better start of the Tucker3 algorithm. Moreover, in the simulations reported in Section 5, the Tucker3 bootstrap procedure did not appear to be overly time-consuming.

7. DISCUSSION

In the present paper a procedure has been described for determining bootstrap percentile intervals for all parameters resulting from a Tucker3 or CANDECOMP/PARAFAC three-way analysis. These can be used as such, i.e. intervals indicating the stability of solutions across resampling from the same data, and hence give an important primary indication of their reliability. However, it was found that the 95% percentile intervals also turn out to be fairly good approximations to 95% confidence intervals in most cases, in the sense that they show coverage percentages that are not too far from the desired 95%. Thus the present bootstrap approach offers a procedure that yields intervals that, in most cases, can at least tentatively be used as confidence intervals as well. Some improvement of these intervals, however, still remains to be desired.

The approximate confidence intervals given here pertain to each individual output parameter. However, obviously, the output parameters are not independent from each other. For instance, already the unit column sums of squares constraints on the component matrices ensure that elements within columns of such matrices depend on each other. Moreover, the optimality of a solution does not depend on each parameter individually, but on the complete configuration of all output parameters. Thus one may expect that, if a percentile interval for a particular element of **B**, say, does not contain the population parameter value, then it is relatively likely that percentile intervals for other elements of **B** do not cover their population counterparts either. Such dependence even holds for elements from different matrices: consider that an 'extreme' solution for **B** is found (such that the associated percentile intervals miss most of the population parameters), then this will most likely also affect the solution for the core $\underline{\mathbf{G}}$ (and hence lead to misfitting percentile intervals for many elements of $\underline{\mathbf{G}}$). If there were only two or three parameters, this dependence problem could be dealt with by constructing percentile regions (or volumes) and using these as approximate confidence regions or volumes, and these could be displayed graphically. However, in the case of the present kind of three-way models we usually deal with very many parameters and hence we would require construction (which is feasible) and display (which is unfeasible) of high-dimensional percentile volumes. Clearly, further research is needed to deal with the dependence of output parameters. For now it suffices to remark that the confidence intervals are each taken as if they 'were on their own', and in interpreting the confidence intervals, their dependence should not be overlooked, in particular when they are to be used to make probability statements on sets of parameters jointly.

The present bootstrap procedure is meant for situations where the entities from one mode can be viewed as a random sample from a population. In practice the entities are not always sampled randomly. In experimental situations it may happen that the A-mode entities (e.g. the mixtures) are deliberately chosen to cover a particular range of values, or they may even be chosen by a particular design. In such cases the A-mode entities are clearly not to be considered a random sample. In other situations the sampling mode may refer to a series of consecutive time points. Even when the starting point of such a series is chosen randomly, the consecutive points are clearly dependent, and again we do not have an ordinary random sample. In such cases the

bootstrap does not work when applied in the way described in the present paper. A suitable alternative in such cases would be a procedure based on constructing bootstrap samples on the basis of residuals from the sample solution (see e.g. Reference [8], p. 113). The basic idea here is that the residuals resulting from a particular analysis of the sample data are considered themselves as a random sample, from which we construct bootstrap samples by drawing random samples from the profiles of residuals, with replacement. These profiles of resampled residuals are then added to the modeled part of the data to construct the data for a bootstrap sample. From then on the procedure continues as with the ordinary bootstrap procedure, thus analyzing all bootstrap samples by the same method and constructing percentile intervals for all output parameters. A disadvantage of this approach is that it depends on the validity of the model chosen, and hence it does not make sense when the residuals stem from a model that is far from correct for the data. For the type of dependent samples described above, however, there does not seem to be a viable alternative.

The bootstrap method is sometimes called a computer intensive method. When we apply it to three-way analysis, indeed, this intensity becomes apparent, especially when using CANDECOMP/PARAFAC. Computation times for moderately sized problems are non-negligible, though not prohibitive. Some speed improvement was obtained and further speed improvement may be possible. All in all, however, it can be concluded that the bootstrap now is a viable procedure for estimating confidence intervals for the results from exploratory three-way methods.

Acknowledgements

The author is obliged to two anonymous reviewers for their useful suggestions and to John MacGregor and Dora Kourti (McMaster, Canada) for making available the data used in Section 4.

REFERENCES

1. Carroll JD, Chang J-J. Analysis of individual differences in multidimensional scaling via an *N*-way generalization of 'Eckart–Young' decomposition. *Psychometrika* 1970; **35**: 283–319.
2. Harshman RA. Foundations of the PARAFAC procedure: models and conditions for an 'explanatory' multi-mode factor analysis. *UCLA Working Papers Phonet*. 1970; **16**: 1–84.
3. Tucker LR. Some mathematical notes on three-mode factor analysis. *Psychometrika* 1966; **31**: 279–311.
4. Kroonenberg PM, De Leeuw J. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika* 1980; **45**: 69–97.
5. Harshman RA, De Sarbo WS. An application of PARAFAC to a small sample problem, demonstrating preprocessing, orthogonality constraints, and split-half diagnostic techniques. In *Research Methods for Multimode Data Analysis*, Law HG, Snyder CW, Hattie JA, McDonald RP (eds). Praeger: New York, 1984; 602–642.
6. Bentler PM, Lee S-Y. Statistical aspects of a three-mode factor analysis model. *Psychometrika* 1978; **43**: 343–352.
7. McArdle JJ, Cattell RB. Structural equation models of factorial invariance in parallel proportional profiles and oblique confactor problems. *Multivar. Behav. Res.* 1994; **29**: 63–113.
8. Efron B, Tibshirani RJ. *An Introduction to the Bootstrap*. Chapman and Hall: New York, 1993.
9. Kroonenberg PM, Snyder Jr CW. Individual differences in assimilation resistance and affective responses in problem solving. *Multivar. Behav. Res.* 1989; **24**: 257–284.
10. Riu J, Bro R. Jack-knife technique for outlier detection and estimation of standard errors in PARAFAC models. *Chemometrics Intell. Lab. Syst.* 2003; **65**: 35–49.
11. Efron B. Bootstrap methods: another look at the jackknife. *Ann. Statist.* 1979; **7**: 1–26.
12. Harshman RA. Determination and proof of minimum uniqueness conditions for PARAFAC1. *UCLA Working Papers Phonet.* 1972; **22**: 111–117.
13. Kiers HAL. Towards a standardized notation and terminology in multiway analysis. *J. Chemometrics* 2000; **14**: 105–122.
14. Kroonenberg PM. *Three Mode Principal Component Analysis: Theory and Applications*. DSWO Press: Leiden, 1983.
15. Kiers HAL. Joint orthomax rotation of the core and component matrices resulting from three-mode principal components analysis. *J. Classifi.* 1998; **15**: 245–263.
16. Tucker LR. A method for synthesis of factor analysis studies. *Personnel Research Section Report No. 984*. Department of the Army, Washington, DC, 1951.
17. Meulman JJ, Heiser WJ. The display of bootstrap solutions in multidimensional scaling. *Technical Report*. University of Leiden, 1983.
18. Krzanowski WJ. Cross-validation in principal component analysis. *Biometrics* 1987; **43**: 575–584.
19. Markus MT. *Bootstrap Confidence Regions in Nonlinear Multivariate Analysis*. DSWO Press: Leiden, 1994.
20. Milan L, Whittaker J. Application of the parametric bootstrap to models that incorporate a singular value decomposition. *Appl. Statist.* 1995; **44**: 31–49.
21. Groenen PJF, Commandeur JJF, Meulman JJ. Distance analysis of large data sets of categorical variables using object weights. *Br. J. Math. Statist. Psychol.* 1998; **51**: 217–232.
22. Cliff N. Orthogonal rotation to congruence. *Psychometrika* 1966; **31**: 33–42.
23. Green BF. The orthogonal approximation of an oblique structure in factor analysis. *Psychometrika* 1952; **17**: 429–440.
24. Kiers HAL. Some procedures for displaying results from three-way methods. *J. Chemometrics* 2000; **14**: 151–170.
25. Nomikos P, MacGregor JF. Monitoring batch processes using multiway principal component analysis. *AIChE J.* 1994; **40**: 1361–1375.
26. Kaiser HF. The varimax criterion for analytic rotation in factor analysis. *Psychometrika* 1958; **23**: 187–200.
27. MATLAB Inc. *MATLAB*. MATLAB Inc.: Natick, MA, 1994.

