PSYCHOMETRIKA—VOL. 57, NO. 3, 415–422
SEPTEMBER 1992

AN EFFICIENT ALGORITHM FOR TUCKALS3 ON DATA WITH LARGE NUMBERS OF OBSERVATION UNITS

HENK A. L. KIERS

UNIVERSITY OF GRONINGEN

PIETER M. KROONENBERG

LEIDEN UNIVERSITY

JOS M. F. TEN BERGE

UNIVERSITY OF GRONINGEN

A modification of the TUCKALS3 algorithm is proposed that handles three-way arrays of order I × J × K for any I. When I is much larger than JK, the modified algorithm needs less work space to store the data during the iterative part of the algorithm than does the original algorithm. Because of this and the additional feature that execution speed is higher, the modified algorithm is highly suitable for use on personal computers.

Key words: TUCKALS3, alternating least squares, multitrait-multimethod matrices.

Since Tucker (1966) fully described three-mode factor (and principal component) analysis, further research in the area has proceeded along two different lines. One, starting with Bloxom (1968), concentrated on further developing stochastic confirmatory common factor analysis versions of Tucker's model, particularly for multitrait-multimethod type covariance matrices. Estimation procedures for such three-way models have been described by Bentler, Lee, and co-workers (Bentler & Lee, 1978, 1979; Bentler, Poon, & Lee, 1988; Lee & Fong, 1983). In addition, McDonald (1984), Browne and associates (e.g., Browne, 1984, for an overview), Verhees (1989), and Verhees and Wansbeek (1990) have done work in this area. The second line, starting with Kroonenberg and de Leeuw (1980; Kroonenberg, 1983; Kroonenberg, ten Berge, Brouwer, & Kiers, 1989; Murakami, 1983; ten Berge, de Leeuw, & Kroonenberg, 1987), concentrated on further developing nonstochastic, exploratory principal component aspects of Tucker's model. In terms of algorithms, the exploratory line is a further development of Tucker's Method I (based on product-moment matrices of the frontal, lateral, and horizontal slices of the observed three-way array), whereas the stochastic line elaborates on Tucker's Method III (based on multitrait-multimethod matrices). In the present paper an exploratory version of Tucker's Method III is presented as a modification of the TUCKALS3 algorithm of Kroonenberg and de Leeuw (1980). The approach used here is similar to earlier work by Murakami (1983) on the TUCKALS2 algorithm.

The motivation for the present development of the Modified TUCKALS3 algorithm is similar to Tucker's motivation for developing his Methods II and III (Tucker, 1966, pp. 298ff.), that is, how to handle three-way data sets with (very) large numbers of individuals within storage and execution time limitations. When Tucker's original

This research has been made possible by a fellowship from the Royal Netherlands Academy of Arts and Sciences to the first author.

Requests for reprints should be sent to Henk A. L. Kiers, Department of Psychology, Grote Kruisstraat 2/1, 9712 TS Groningen, THE NETHERLANDS.

0033-3123/92/0900-91322$00.75/0

paper was published, these limitations occurred because computers were not that large and advanced; now, the same problems return in spite of the huge advances in computer technology, because of the general introduction of desktop micro-computers. In addition, the present approach enables the analysis of (published) multitrait-multimethod type covariance matrices in an exploratory fashion without making distributional assumptions.

Kroonenberg and de Leeuw (1980) have described an algorithm for least squares fitting of the Tucker3 model for three-mode principal component analysis (Tucker, 1966). Let X_k, $k = 1, \ldots,$ K denote the k-th frontal slice of the three-way array, X_k having the order I × J. Their procedure minimizes the loss function

$$f(A, B, C, G_1, \ldots, G_R) = \sum_{k=1}^{K} \left\| X_k - A \sum_{r=1}^{R} c_{kr} G_r B' \right\|^2, \qquad (1)$$

where A (I × P), B (J × Q), and C (K × R) are the component matrices for the first, second, and third mode, respectively, and $G_1, \ldots, G_R$ are the R frontal slices of the "core matrix" of order P × Q × R with the weights for the combinations of components of the three modes. Note that the formulation for the model part, $\hat{X}_k = A \sum_{r=1}^{R} c_{kr} G_r B'$, $k = 1, \ldots,$ K, is equivalent to the more prevalent formulation $\hat{X} = AG(C' \otimes B')$, where $\hat{X} = (\hat{X}_1 | \cdots | \hat{X}_K)$ and $G = (G_1 | \cdots | G_R)$, and ⊗ is the Kronecker product.

Kroonenberg and de Leeuw (1980) proposed to minimize this function by a procedure that alternatingly minimizes f over A while B and C are held fixed, over B with A and C fixed, and over C with A and B fixed. Kroonenberg, ten Berge, Brouwer, and Kiers (1989) have accelerated this algorithm by replacing the so-called Bauer-Rutishauser (BR) procedure (in which one step of an iterative routine for computing a subset of eigenvectors of a matrix is performed) by a Gram-Schmidt (GS) procedure. However, computation times dropped only slightly, probably because other computations during each iteration step use far more floating point operations (flops) than the BR or GS procedures. This effect is especially marked for large matrices. Therefore, in the present paper, a further modification of the TUCKALS3 algorithm is proposed that attempts to reduce the number of flops considerably for data sets in which the size (number of entities) of one of the modes is far larger than the product of the sizes of the other two. Because I usually refers to the observation units, and typically, I = max (I, J, K), we focus on cases with I ≫ JK.

To modify the Kroonenberg and de Leeuw (1980) algorithm, abbreviated as "the K & L algorithm", it is convenient to describe it in an unusual but strictly equivalent way. How this algorithm can be improved by a GS-step instead of a BR-step will be discussed next (see, Kroonenberg et al., 1989); finally, our further improvement of the K&L algorithm will be taken up.

The Modified TUCKALS3 Algorithm

Instead of describing the K&L algorithm with updating A, B, and C in turn in just three steps, the algorithm will be described in six steps: updating the core; updating A; updating the core; updating B; updating the core; updating C. Because these (unraveled) steps do not directly follow from Kroonenberg and de Leeuw (1980), they are rederived here as follows (also, see Weesie & Van Houwelingen, 1983).

First, it is useful to note that A, B, and C can be constrained to be columnwise

orthonormal without loss of generality (Kroonenberg & de Leeuw, 1980), and hence, f can be expanded as

$$f(A, B, C, G_1, \ldots, G_R) = \sum_{k=1}^{K} \text{tr } X_k'X_k - 2 \text{ tr } \sum_{k=1}^{K} \sum_{r=1}^{R} c_{kr} A'X_k BG_r' + \text{tr } \sum_{r=1}^{R} G_r G_r'. \quad (2)$$

The first, third, and fifth steps of the unraveled K&L algorithm involve minimizing f over $G_1, \ldots, G_R$ with A, B, and C fixed. To find the $G_1, \ldots, G_R$ for which f is minimal it is useful to rewrite (2) as

$$f(A, B, C, G_1, \ldots, G_R) = c_G - \sum_{r=1}^{R} \left\| \sum_{k=1}^{K} c_{kr} A'X_k B - G_r \right\|^2, \quad (3)$$

where c_G is a constant independent of $G_1, \ldots, G_R$. From (3) it follows immediately that the minimum of f over G_r is given by

$$G_r = \sum_{k=1}^{K} c_{kr} A'X_k B, \quad (4)$$

for $r = 1, \ldots, R$.

The minimum of f over A subject to $A'A = I_P$, with the other parameters fixed is attained by

$$A = P(P'P)^{-1/2}, \quad (5a)$$

with

$$P = \sum_{k=1}^{K} \sum_{r=1}^{R} c_{kr} X_k BG_r', \quad (5b)$$

as follows from Cliff (1966, p. 36), applied to the second term in (2). Similarly, the minimum of f over B subject to $B'B = I_Q$ is attained by

$$B = Q(Q'Q)^{-1/2}, \quad (6a)$$

with

$$Q = \sum_{k=1}^{K} \sum_{r=1}^{R} c_{kr} X_k' AG_r. \quad (6b)$$

To find the C that minimizes f over C subject to $C'C = I_R$, the second term in (2) should be rewritten as $\text{tr } \sum_k \sum_r c_{kr} A'X_k BG_r' = \sum_k \sum_r c_{kr} (\text{tr } A'X_k BG_r') = \text{tr } CR'$, in which R is defined such that

$$r_{kr} = \text{tr } A'X_k BG_r'. \quad (7a)$$

The C that minimizes f subject to $C'C = I_R$ is given by

$$C = R(R'R)^{-1/2}. \tag{7b}$$

The unraveled K&L algorithm can now be described as the algorithm that monotonically decreases (or "not increases") f by alternatingly updating G according to (4), A according to (5), G according to (4), B according to (6), G according to (4), and C according to (7), and then repeating the whole cycle until the function value stabilizes. Kroonenberg et al.'s (1989) modification comes down to replacing the relatively expensive computation of $P(P'P)^{-1/2}$, $Q(Q'Q)^{-1/2}$, and $R(R'R)^{-1/2}$ (the BR steps) by GS procedures applied to P, Q, and R, respectively. The thus modified algorithm minimizes f just as the original K&L algorithm does, with intermediate function values precisely the same, but, in cases of large matrices may still be slow.

When one of the modes, say I, is large, the K&L algorithm spends a large amount of time on computations involving $X_1, \ldots, X_K$, and A, which are of orders I × J and I × P, respectively. Here, a modification of the K&L algorithm is proposed in which computations with matrices where one of the orders is I are circumvented. This modification is based on the fact that, in all steps of the algorithm, X_k and A always occur in matrix-products $A'X_k$, or $X_k'X_l$ (implicitly in $P'P$), $k, l = 1, \ldots, K$, which are much smaller than X_k and A itself (when I is large). The crucial step in our modification of the K&L algorithm is that instead of expressing the updates for B, C, and $G_1, \ldots, G_R$, (6), (7), and (4), in A and X_k, these can be expressed directly in terms of $S_k \equiv X_k'A$, $k = 1, \ldots, K$. Hence, for computing these updates we need not have X_k and A in memory, as long as we have stored $S_1, \ldots, S_K$.

It remains to modify the updating procedure for A. Because A is not stored any longer and its role is taken over by S_k, $k = 1, \ldots, K$, we should update S_k (instead of A), thus updating A *implicitly*. The update for A is given by $A := \mathrm{GS}(P)$, where P is $P = \sum_l \sum_r c_{lr} X_l B G_r'$. Hence, the update for S_k is given by

$$S_k = X_k'A = X_k'\mathrm{GS}(P),$$

which can be elaborated as follows. GS(P) can be written as PT for some P × P transformation matrix T. Hence,

$$S_k = X_k'PT = \sum_l \sum_r c_{lr} X_k'X_l B G_r' T. \tag{8a}$$

The update (8a) for S_k involves X_k only in the cross-product terms $X_k'X_l$, $k, l = 1, \ldots, K$, and in the GS-transformation T as will be shown below. It can be verified that the GS-transformation can be found from the Cholesky decomposition of the inner product-moment of the matrix to which it is to be applied (see Lawson & Hanson, 1974, p. 125). Explicitly, let the Cholesky decomposition of

$$P'P = \sum_k \sum_l \sum_r \sum_{r'} c_{kr} c_{lr'} G_r B' X_k' X_l B G_{r'}', \tag{8b}$$

be written as $P'P = U'U$, where U is an upper triangular matrix. Then, the transformation matrix T used in the Gram-Schmidt orthonormalization of P can be taken as $T = U^{-1}$. The latter can be verified as follows. First, $T'P'PT = I$, showing that PT is columnwise orthonormal; second, the Gram-Schmidt transformation matrix is upper triangular, and if P has full column rank (which can always be assumed to hold), it is unique. Because T is upper triangular and PT is columnwise orthonormal, PT is the Gram-Schmidt orthonormalized version of P. It can be concluded that computation of T via the Cholesky decomposition of $P'P$, again involves X_k only in the cross-product

terms $X_k'X_l$. As a consequence, S_k can be updated without using the large matrices X_k, $k = 1, \ldots, K$.

It has been shown that for updating $G_1, \ldots, G_R$, B, C, and $S_1, \ldots, S_K$, only $X_k'X_l$ and S_k are needed (instead of the large matrices X_k and A). Apart from these updating steps one needs to evaluate the function during each iteration. It is readily verified that the evaluation of f only involves $X_k'X_k$ and S_k, and does not require the explicit computation of A either. Moreover, after updating $G_1, \ldots, G_R$, using $G_r = \sum_k c_{kr} A'X_kB$, the function f can be simplified as

$$f = \sum_{k=1}^{K} \text{tr } X_k'X_k - \sum_{r=1}^{R} \text{tr } G_rG_r'. \tag{9}$$

The Modified TUCKALS3 algorithm needs to store the matrices S_k (J × P), $P'P$ (P × P), and $V_{kl} = X_k'X_l$ (J × J), $k, l = 1, \ldots, K$ (i.e., KJP + P^2 + K^2J^2 reals), where the (most efficient version of the) original algorithm needs to store X_k (I × J) and A (I × P) (i.e., KIJ + IP reals), apart from the matrices B, C, and $G_1, \ldots, G_R$. The main merit of the present modification of the TUCKALS3 algorithm (denoted as the "Modified algorithm" in the sequel) is that both storage and computations involving matrices of row- or column-order I are circumvented. As a result, the Modified algorithm requires less RAM space than the (most efficient form of the) original algorithm whenever $I > JK + P^2(JK + P)^{-1}$, and in particular when $I \gg JK$. At the cost of a slightly more involved implementation, a further gain of just under 50% for large matrices can be achieved by only storing V_{kl} for $k = 1, \ldots, K$ and $l \leq k$.

The above modifications of the K&L algorithm yield the following Modified algorithm.

Step 1. Read cross-product matrices V_{kl}, $k, l = 1, \ldots, K$, or read (raw) data matrices such that cross-product matrices are built while reading the original data.

Step 2. Initialize S_k, $k = 1, \ldots, K$, B, C, compute G_r as $G_r = \sum_k c_{kr} S_k'B$, $r = 1, \ldots, R$, and evaluate f according to (9).

Step 3. Iteratively perform the following steps, until convergence:

- 3a. Compute $Q = \sum_k \sum_r c_{kr} S_k G_r$, and update B by $B = GS(Q)$,
- 3b. Update G_r, $r = 1, \ldots, R$, by $G_r = \sum_k c_{kr} S_k'B$,
- 3c. Compute $[R]_{kr} = \text{tr } S_k'BG_r'$, $k = 1, \ldots, K$, $r = 1, \ldots, R$, and update C by $C = GS(R)$,
- 3d. Update G_r, $r = 1, \ldots, R$, as in 3b,
- 3e. Update S_k, $k = 1, \ldots, K$, according to (8),
- 3f. Update G_r, $r = 1, \ldots, R$, as in 3b,
- 3g. Evaluate f according to (9).

If $f^{old} - f^{new} < \varepsilon$, where ε is some arbitrary small value, go to Step 4; else, repeat Step 3.

Step 4. If the original data are still available, compute A according to $A = GS(P)$ (optional).

Step 2 (of initializing S_k) requires some extra attention. The algorithm is based on the assumption that S_k should be of a form that can be expressed as $S_k = X_k'A$ for some A for which $A'A = I_p$. However, if this would not be the case at the start of the algorithm, the procedure for updating S_k automatically will take care of this condition, and from the first iteration onward, the condition is satisfied. Hence, one may initialize S_k at will.

Although S_k, A, and B can be initialized arbitrarily, it is recommended to use rational starts for these parameter sets. Kroonenberg and de Leeuw (1980, p. 76) propose to start their TUCKALS3 algorithm with the parameters resulting from Tucker's Method I (Tucker, 1966, pp. 297–298). Thus, for A they use the first P eigenvectors of $\sum_k X_k X_k'$, for B the first Q eigenvectors of $\sum_k X_k' X_k$, and for C the first R eigenvectors of the matrix Y with elements $y_{kl} = \text{tr } X_k' X_l$. If I is large compared to JK, the matrices $V_{kl} = X_k' X_l$ are used instead of the original matrices X_k. Clearly, the starting values for B and C can still be found using the matrices V_{kl}. However, for A we need the following alternative procedure, based on Tucker's (1966, pp. 299–301) Method III.

Let $X \equiv (X_1 | \cdots | X_K)$, and let A contain the rational starting values obtained from the first P (unit normalized) eigenvectors of $XX' = \sum_k X_k X_k'$. Then, $XX'A = A\Lambda$ for the diagonal matrix Λ with the first P eigenvalues of XX', and hence, $X'X(X'A) = (X'A)\Lambda$ with $(X'A)'(X'A) = \Lambda$. As a result, $W = X'A = (S_1' | \cdots | S_k')'$ contains the first p eigenvectors of $X'X$, normalized to sums of squares equal to the associated eigenvalues. It follows that starting values for $S_1, \ldots, S_K$ (associated with the rational starting values for A) can be obtained by taking the first P eigenvectors of $X'X$ (the JK $\times$ JK matrix with blocks V_{kl}), normalizing these to sums of squares equal to the associated eigenvalues, and partitioning the resulting matrix into $S_1, \ldots, S_K$.

The main modification of the original TUCKALS3 algorithm into the Modified TUCKALS3 algorithm is the use of matrices S_k. Apart from their auxiliary function, these matrices may also be useful interpretative tools. When the columns of the slices $X_1, \ldots, X_K$ are centered and normalized to unit length, the matrices $S_k = X_k' A$, $k = 1, \ldots, K$, contain correlations between the mode A components and the variables for each slice.

Computation Times

The gain in execution time with the Modified TUCKALS3 algorithm will be demonstrated with a small Monte Carlo study. Four three-way arrays of random numbers from a uniform distribution were generated, of order 6 $\times$ 5 $\times$ 5, 25 $\times$ 5 $\times$ 5, 50 $\times$ 5 $\times$ 5, and 100 $\times$ 5 $\times$ 5, respectively. For each data set, two different combinations of numbers of components for the three ways were used: 2 $\times$ 2 $\times$ 2 and 6 $\times$ 3 $\times$ 3. The most efficient version of the original algorithm (which still deals with matrices of row- or column-orders I) was compared with the Modified algorithm. The computation times per iteration are reported in Table 1. All comparisons were carried out with Fortran77 programs compiled with the MS-Fortran 4.1 compiler on a Commodore PC40-III with a 80287 mathematical coprocessor.

As the largest order I is not involved in the main computations of the Modified algorithm, the execution speed per iteration is unaffected by the increase of I (small differences in computation time per iteration may occur due to inaccuracies of time measuring). The time to compute the covariance matrix increases with increasing I, and does contribute to the overall computation time, but this is minor compared to the rest of the computations. The Modified algorithm becomes faster than the standard one at approximately I = 26 for the 2 $\times$ 2 $\times$ 2-solution, and approximately I = 80 for the 6 $\times$ 3 $\times$ 3-solution. As execution times are machine dependent, explicit recommendations cannot be made as to which algorithm to use when execution speed is the only consideration. It is clear, however, that for I $\gg$ JK, the modified algorithm will always be faster.

TABLE 1

Comparison of Average Execution Speeds Per Iteration of the TUCKALS3 and Modified TUCKALS3 Algorithms (in seconds)

Size Data Set	Number of Components	Average Execution times per iteration: TUCKALS3	Average Execution times per iteration: Modified TUCKALS3
$6 \times 5 \times 5$	$2 \times 2 \times 2$	0.27	0.66
$25 \times 5 \times 5$	$2 \times 2 \times 2$	0.65	0.67
$50 \times 5 \times 5$	$2 \times 2 \times 2$	1.17	0.67
$100 \times 5 \times 5$	$2 \times 2 \times 2$	2.19	0.67
$6 \times 5 \times 5$	$6 \times 3 \times 3$	0.61	4.45
$25 \times 5 \times 5$	$6 \times 3 \times 3$	1.59	4.47
$50 \times 5 \times 5$	$6 \times 3 \times 3$	2.89	4.47
$100 \times 5 \times 5$	$6 \times 3 \times 3$	5.50	4.47

Discussion

The above modification of the TUCKALS3 algorithm shows that the computations of the solution for B, C, and $G_1, \ldots, G_R$ do not involve the X_k matrices, but only the cross-product matrices V_{jk}, $j, k = 1, \ldots, K$. As a consequence, different data sets with the same cross-product matrices yield the same solutions for B, C, and $G_1, \ldots, G_R$. This result parallels the situation of principal components analysis, where the loadings depend on the correlation matrix only. A similar result was found for the three-way method PARAFAC (Kiers & Krijnen, 1991).

The information in three-way data sets is often reported only in terms of covariances or correlations between the variables. A prevalent case is that of multitrait-multimethod (MTMM) matrices. The original TUCKALS3 method cannot deal with such matrices because it requires the original three-way data array on which the MTMM matrix is based. The Modified algorithm developed in the present paper does not require the original three-way data as input for TUCKALS3, and thus provides us with an algorithm to apply TUCKALS3 to MTMM matrices.

As has been mentioned above, Murakami (1983) has developed a similar procedure

for handling large data in TUCKALS2. Apart from the fact that he used TUCKALS2 instead of TUCKALS3, another major difference is that Murakami's algorithm is still based on eigendecompositions in each iteration, whereas our procedure uses the much cheaper Gram-Schmidt procedures. It should be noted that a modification highly similar (but simpler) to that of the TUCKALS3 algorithm can be constructed to speed up the TUCKALS2 algorithm.

References

Bentler, P. M., & Lee, S.-Y. (1978). Statistical aspects of a three-mode factor analysis model. *Psychometrika, 43*, 343–352.

Bentler, P. M., & Lee, S.-Y. (1979). A statistical development of three-mode factor analysis. *British Journal of Mathematical and Statistical Psychology, 32*, 87–104.

Bentler, P. M., Poon, W.-Y., & Lee, S.-Y. (1988). Generalized multimode latent variable models: Implementation by standard programs. *Computational Statistics and Data Analysis, 6*, 107–118.

Bloxom, B. (1968). A note on invariance in three-mode factor analysis. *Psychometrika, 33*, 347–350.

Browne, M. W. (1984). The decomposition of multitrait-multimethod matrices. *British Journal of Mathematical and Statistical Psychology, 37*, 1–21.

Cliff, N. (1966). Orthogonal rotation to congruence. *Psychometrika, 31*, 33–42.

Kiers, H. A. L., & Krijnen, W. P. (1991). An efficient algorithm for PARAFAC of three-way data with large numbers of observation units. *Psychometrika, 56*, 147–152.

Kroonenberg, P. M. (1983). *Three-mode principal component analysis*. Leiden: DSWO Press.

Kroonenberg, P. M., & de Leeuw, J. (1980). Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika, 45*, 69–97.

Kroonenberg, P. M., ten Berge, J. M. F., Brouwer, P., & Kiers, H.A.L. (1989). Gram-Schmidt versus Bauer-Rutishauser in alternating least-squares algorithms for three-mode principal component analysis. *Computational Statistics Quarterly, 5*, 81–87.

Lawson, C. L., & Hanson, R. J. (1974). *Solving least squares problems*. Englewood Cliffs, NJ: Prentice-Hall.

Lee, S.-Y., & Fong, W.-K. (1983). A scale invariant model for three-mode factor analysis. *British Journal of Mathematical and Statistical Psychology, 36*, 217–223.

McDonald, R. P. (1984). The invariant factors model for multimode data. In H. G. Law, C. W. Snyder, J. A. Hattie, & R. P. McDonald (Eds.), *Research methods for multimode data analysis* (pp. 285–307). New York: Praeger.

Murakami, T. (1983). Quasi three-mode principal component analysis—A method for assessing the factor change. *Behaviormetrika, 14*, 27–48.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika, 31*, 279–311.

ten Berge, J. M. F., de Leeuw, J., & Kroonenberg, P. M. (1987). Some additional results on principal components analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika, 52*, 183–191.

Verhees, J. (1989). *Econometric analysis of multidimensional models*. Unpublished doctoral dissertation, University of Groningen.

Verhees, J., & Wansbeek, T. J. (1990). A multimode direct product model for covariance structure analysis. *British Journal of Mathematical and Statistical Psychology, 43*, 231–240.

Weesie, H. M., & Van Houwelingen, J. C. (1983). *GEPCAM users' manual: Generalized principal components analysis with missing values*. Unpublished manuscript, University of Utrecht, Institute of Mathematics.

Manuscript received 3/26/91
Final version received 11/26/91