ELSEVIER

Chemometrics and Intelligent Laboratory Systems 24 (1994) 43–54

Chemometrics and intelligent laboratory systems

Speed improvement of multivariate algorithms by the method of postponed basis matrix multiplication Part II. Three-mode principal component analysis

Bjørn K. Alsberg *, Olav M. Kvalheim

Department of Chemistry, University of Bergen, Allégt, 41, 5007 Bergen, Norway

(Received 28 October 1993; accepted 14 February 1994)

Abstract

Compression is one way of making analysis of large data arrays faster. Compression is here defined as the case when a large array $\underline{\mathbf{X}}$ is replaced by a smaller *coefficient* array $\underline{\mathbf{C}}$. The coefficients are obtained by least squares fitting to some compression basis. This paper will deal with three-mode arrays. When performing, e.g., three-mode principal component analysis of $\underline{\mathbf{C}}$ the results are comparable but not equal to the results from analyzing $\underline{\mathbf{X}}$. In this paper we suggest a solution to this problem by rewriting the three-mode PCA algorithm which utilizes $\underline{\mathbf{C}}$ and the compression basis matrices. This has been accomplished by applying a method where speed improvement is achieved by postponing basis matrix calculations in key steps of the three-mode PCA algorithm.

1. Introduction

Compression of raw data from analytical instruments is useful for storage and faster computation of large arrays [1,2]. The approach advocated by us is to fit the original data to a suitable chosen basis set and use the *coefficients* obtained from the fitting instead of the original data. When using such coefficients in, e.g., a principal component analysis (PCA) scores and loadings are not directly comparable to the corresponding scores and loadings of the uncompressed array. One way to circumvent this problem is to premultiply the scores and loading matrices with the compression matrices used [3]. This multiplication, however, does not produce orthogonal scores and loading vectors. A simple orthogonalization/orthonormalization of the transformed matrices does not provide the correct result. In Part I of this article [4], a method was developed for PCA of two-mode arrays which solved the problem. This method is referred to as the method of postponed basis matrix multiplication (PBM) where the central idea is to postpone the multiplication of large basis matrices until after the converged PCA solution is obtained for all factors. The aim of this work is to show the PBM method can be applied to PCA of three-mode arrays. The method can be extended to N-mode PCA also.

A three-array $\underline{\mathbf{X}}$ is assumed to be modeled by three compression matrices $\{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$. After

* Corresponding author. e-mail: alsberg@kj.uib.no.

SSDI 0169-7439(94)00014-A

compression the original three-array can be represented by a much smaller three-array of coefficients $\underline{\mathbf{C}}$. $\underline{\tilde{\mathbf{X}}}$ is the *reconstructed* array which is constructed exclusively from $\underline{\mathbf{C}}$ and the basis matrices ($\{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3\}$). Of course it is not necessary to compress along all the modes. This is the case if some modes are much smaller than the other modes or uncompressible by the chosen basis. If the error $\underline{\mathbf{E}} = \underline{\mathbf{X}} - \underline{\tilde{\mathbf{X}}}$ is large and contains important components the compression will not be satisfactory and the coefficient representation does not reflect the essential structures in $\underline{\mathbf{X}}$. The scores and loadings from the analysis of $\underline{\mathbf{C}}$ have smaller dimensions since the coefficient array is much smaller than $\underline{\tilde{\mathbf{X}}}$ and $\underline{\mathbf{X}}$. As was mentioned above a premultiplication of the basis matrix associated with each mode can be used to increase the dimensions of the scores and loadings vectors obtained from analysis from $\underline{\mathbf{C}}$ to the dimensions of the scores and loadings vectors of $\underline{\tilde{\mathbf{X}}}$ or $\underline{\mathbf{X}}$. When these reconstructed scores and loadings vectors of $\underline{\mathbf{C}}$ are compared to the corresponding vectors of $\underline{\tilde{\mathbf{X}}}$, it is found that a distortion has been introduced. Sometimes this distortion is so small that it has no practical consequence. For other problems the distortions can be quite large. Simple orthogonalizations/orthonormalizations or other transformations have been unsuccessful in trying to nullify the distortions. The PBM method, however, solves the problem by rewriting the algorithm in question such that when the basis matrices are premultiplied with the PBM scores and loadings they are identical to the scores and loadings obtained from direct analysis of $\underline{\tilde{\mathbf{X}}}$.

A diagram notation [5] is used to represent the three-mode array equations and the reader is encouraged to study the Appendix for an explanation of the notation used.

2. Three-mode PCA

It is instructive to start an explanation of three-mode PCA by first looking at how singular value decomposition (SVD) is performed. In SVD for matrices (two-arrays) an $\mathbf{X}$ matrix can be decomposed as

$$\mathbf{X} = \mathbf{U}\mathbf{S}^{1/2}\mathbf{V}^{\mathrm{T}} \tag{1}$$

where $\mathbf{U}$ and $\mathbf{V}$ are columnwise orthonormal matrices and $\mathbf{S}^{1/2}$ is a diagonal matrix containing the square root of eigenvalues of the two covariance matrices $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. The two associated eigen equations are

$$\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{U} = \mathbf{U}\mathbf{S} \tag{2}$$

$$\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{V} = \mathbf{V}\mathbf{S} \tag{3}$$

In SVD for 3-arrays an $\underline{\mathbf{X}}$ array can be decomposed as shown in Fig. 1. This equation is the Tucker3 model [6,7]. The diagram equation in Fig. 1 can also be expressed in explicit summation notation:

$$x_{ijk} = \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} h_{ip}g_{jq}e_{kr}d_{pqr} \tag{4}$$

There are different methods for solving the Tucker3 model. A method similar to the approach used for SVD of two-arrays (see Eqs. (2) and (3)) is the Tucker Method I [6,7] which obtains estimates for the loading matrices $\mathbf{H}$, $\mathbf{G}$, $\mathbf{E}$ by extracting *all* eigenvectors corresponding to non-zero roots of the three covariance matrices with elements, see Fig. 2:

$$l_{ii'} = \sum_{j=1}^{J}\sum_{k=1}^{K} x_{ijk}x_{i'jk} \tag{5}$$

$$m_{jj'} = \sum_{i=1}^{I}\sum_{k=1}^{K} x_{ijk}x_{ij'k} \tag{6}$$

$$n_{kk'} = \sum_{i=1}^{I}\sum_{j=1}^{J} x_{ijk}x_{ijk'} \tag{7}$$

Using the estimated three loading matrices, Eq. (4) shows the core array $\underline{\mathbf{D}}$ can be expressed as

$$d_{pqr} = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} h_{ip}g_{jq}e_{kr}x_{ijk} \tag{8}$$

This equation is also shown in Fig. 3 using the diagram notation.

In practice an investigator is often interested in only the first largest eigenvectors of $\mathbf{H}$, $\mathbf{G}$, $\mathbf{E}$. Using the Tucker Method I, however, the estimators for d_{pqr} will no longer be least-squares ones.

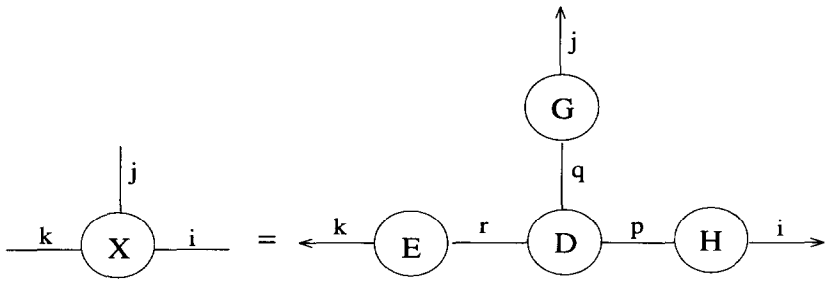

Fig. 1. Illustration of the Tucker3 model.

In order to achieve least squares estimates an alternating least squares (ALS) algorithm [6] can be used. The diagram formulation of this algorithm is presented in Fig. 4. The ALS algorithm is very computer demanding and the idea of using the compressed representation $\underline{\mathbf{C}}$ instead of $\underline{\mathbf{X}}$ to

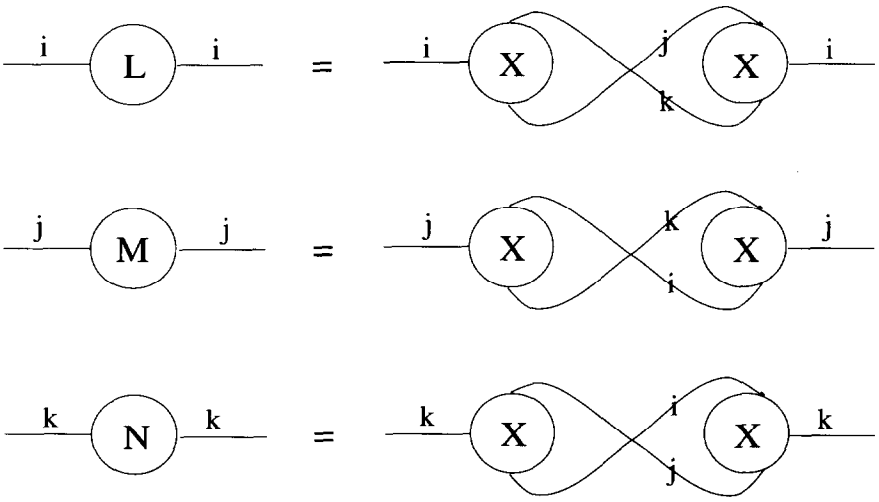

Fig. 2. Construction of the covariance like matrices **L**, **M**, and **N** used in the Tucker I algorithm. In the Tucker3 model the eigenvectors of these matrices are used as initial estimates before entering the alternating least squares iterations.

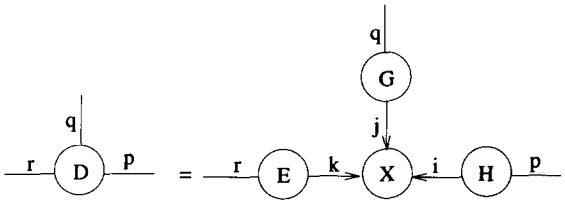

Fig. 3. How to find the core array using a Tucker Method I (here $p = i$, $q = j$, and $r = k$). For cases when $p < i$, $q < j$, and $r < k$, the solution is more complicated.

enable faster computations on three-mode arrays is promising [2,8]. When using $\underline{\mathbf{C}}$ the ALS algorithm returns the much smaller loading matrices $\mathbf{H}_c$, $\mathbf{G}_c$ and $\mathbf{E}_c$. The incorrect transformation back into the original domain by premultiplication of the corresponding compression matrices as mentioned in Part I of this article is formulated as

$$\mathbf{H}_b = \mathbf{B}_1\mathbf{H}_c \tag{9}$$

$$\mathbf{G}_b = \mathbf{B}_2\mathbf{G}_c \tag{10}$$

$$\mathbf{E}_b = \mathbf{B}_3\mathbf{E}_c \tag{11}$$

The PBM method will provide different $\mathbf{H}_c$, $\mathbf{G}_c$, and $\mathbf{E}_c$ which subjected to proper premultiplication of basis matrices are identical to the loading matrices obtained from direct analysis of $\underline{\tilde{\mathbf{X}}}$.

3. The PBM method on three-mode PCA

For an introduction to the PBM method please consult Part I of this article.

There are two different places in the ALS algorithm where postponing of basis matrix multiplication can be made:

1. The computation of matrices which are to be diagonalized. These steps are the array equations as described in detail for one of the three loadings matrices in Fig. 5. These steps are referred to as the 'ALS iteration steps'.

2. The eigenvalue decomposition algorithm which is used to obtain estimates of the loading matrices $\mathbf{H}_c$, $\mathbf{G}_c$, $\mathbf{E}_c$.

The following two subsections will discuss in detail how each of the two parts of the ALS

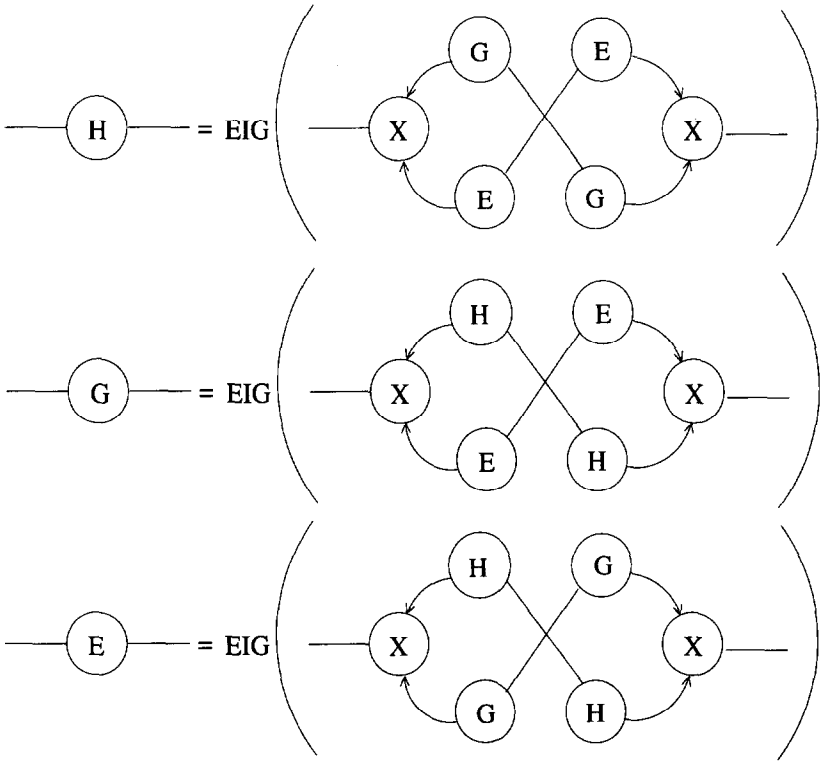

Fig. 4. Central part of the ALS algorithm. Here estimates of the loadings matrices for each mode are generated. The figure shows one iteration.

algorithm is changed according to the PBM method.

3.1. PBM eigenvalue decomposition

In Fig. 4 the word 'EIG' is used to indicate that the resulting matrix of the diagram expression inside the parentheses is diagonalized (EIG returns the A largest eigenvectors). Using the PBM method it is easy to see that the basis matrix associated with the mode is redundant and unnecessary for the computation of eigenvalues and eigenvectors.

Let $\mathbf{K}$ be a symmetric matrix which is to be subjected to an eigenvalue decomposition and assumed to be generated from

$$\mathbf{K} = \mathbf{R}^T\mathbf{R} \tag{12}$$

where

$$\mathbf{R} = \mathbf{W}\mathbf{B}^T \tag{13}$$

i.e., a result from a compression model ($\mathbf{B}^T$ is the compression basis matrix). An alternative formulation of $\mathbf{K}$ is

$$\mathbf{K} = \mathbf{B}\mathbf{W}^T\mathbf{W}\mathbf{B}^T \tag{14}$$

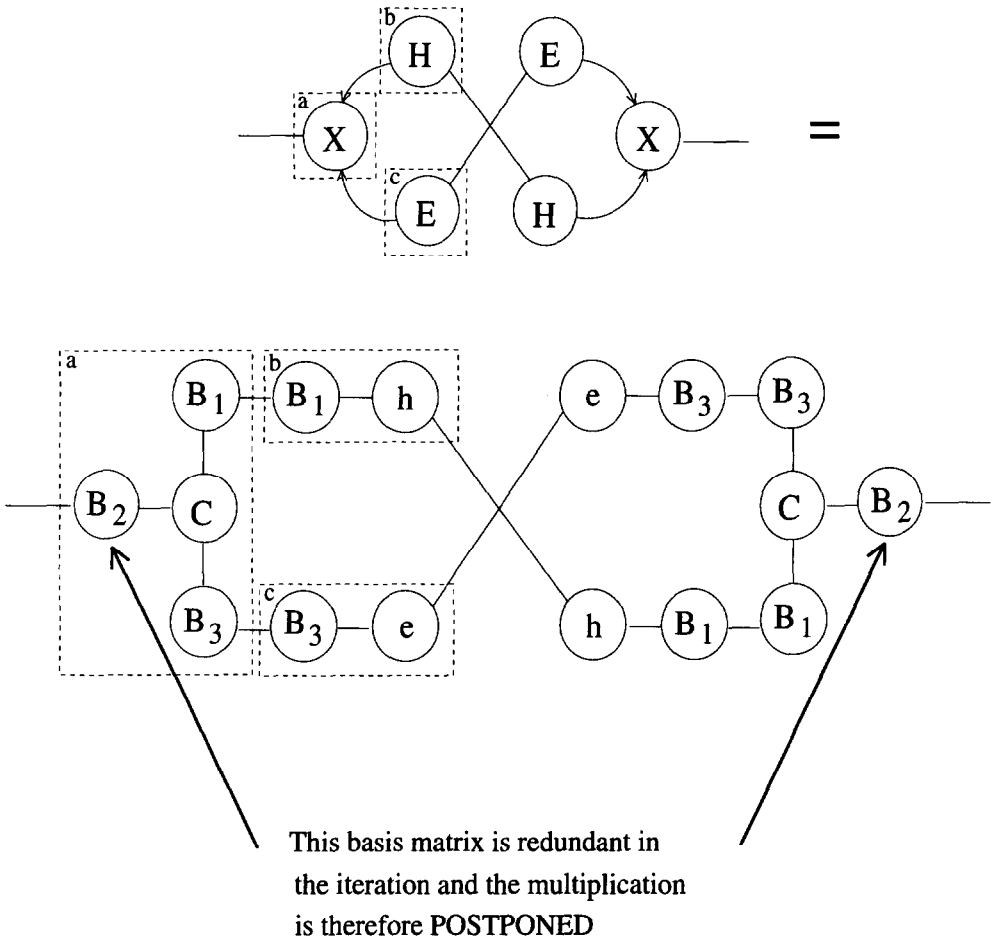

Fig. 5. The central part of the PBM method is illustrated for one part of the ALS iteration steps using diagram notation. When the model equations are written out it is possible to discover that redundant multiplication of large basis matrices is present. The multiplication of these large basis matrices is postponed to after the iteration has converged.

The algorithm chosen for the eigenvalue decomposition of real symmetric matrices is a power method approach [9]. For each component the following is iterated:

$$\boldsymbol{v}_0 = \mathbf{K}\boldsymbol{u}_0 \tag{15}$$

$$\boldsymbol{u}_1 = \frac{\boldsymbol{v}_0}{\left(\boldsymbol{v}_0^{\mathrm{T}}\boldsymbol{v}_0\right)^{1/2}} \tag{16}$$

$$\boldsymbol{u}_0 = \boldsymbol{u}_1 \tag{17}$$

After each such step the difference between earlier iteration steps is measured and the iteration will halt when this difference is small enough. The following updating of the symmetric $\mathbf{K}$ matrix is performed after termination of the iteration:

$$\lambda = \boldsymbol{u}_0^{\mathrm{T}}\mathbf{K}\boldsymbol{u}_0 \tag{18}$$

$$\mathbf{K} = \mathbf{K} - \lambda\boldsymbol{u}_0\boldsymbol{u}_0^{\mathrm{T}} \tag{19}$$

where λ is the eigenvalue.

Two new vectors $\boldsymbol{a}$ and $\boldsymbol{f}$ are introduced which have the same dimensions as a row or a column of $\mathbf{W}$ and the following relations are assumed to be true:

$$\boldsymbol{u} = \mathbf{B}\boldsymbol{a} \tag{20}$$

$$\boldsymbol{v} = \mathbf{B}\boldsymbol{f} \tag{21}$$

Using this information it is possible to reformulate the eigenvalue decomposition iteration:

$$\mathbf{B}\boldsymbol{f}_0 = \mathbf{B}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{B}^{\mathrm{T}}\mathbf{B}\boldsymbol{a}_0 \tag{22}$$

$$\mathbf{B}\boldsymbol{a}_1 = \frac{\mathbf{B}\boldsymbol{f}_0}{\left(\boldsymbol{f}_0^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{B}\boldsymbol{f}_0\right)^{1/2}} \tag{23}$$

$$\mathbf{B}\boldsymbol{a}_0 = \mathbf{B}\boldsymbol{a}_1 \tag{24}$$

Setting $\boldsymbol{\Gamma} = \mathbf{B}^{\mathrm{T}}\mathbf{B}$ and observing that the premultiplication by $\mathbf{B}$ is redundant in the iteration, the PBM iteration will look like:

$$\boldsymbol{f}_0 = \mathbf{W}^{\mathrm{T}}\mathbf{W}\boldsymbol{\Gamma}\boldsymbol{a}_0 \tag{25}$$

$$\boldsymbol{a}_1 = \frac{\boldsymbol{f}_0}{\left(\boldsymbol{f}_0^{\mathrm{T}}\boldsymbol{\Gamma}\boldsymbol{f}_0\right)^{1/2}} \tag{26}$$

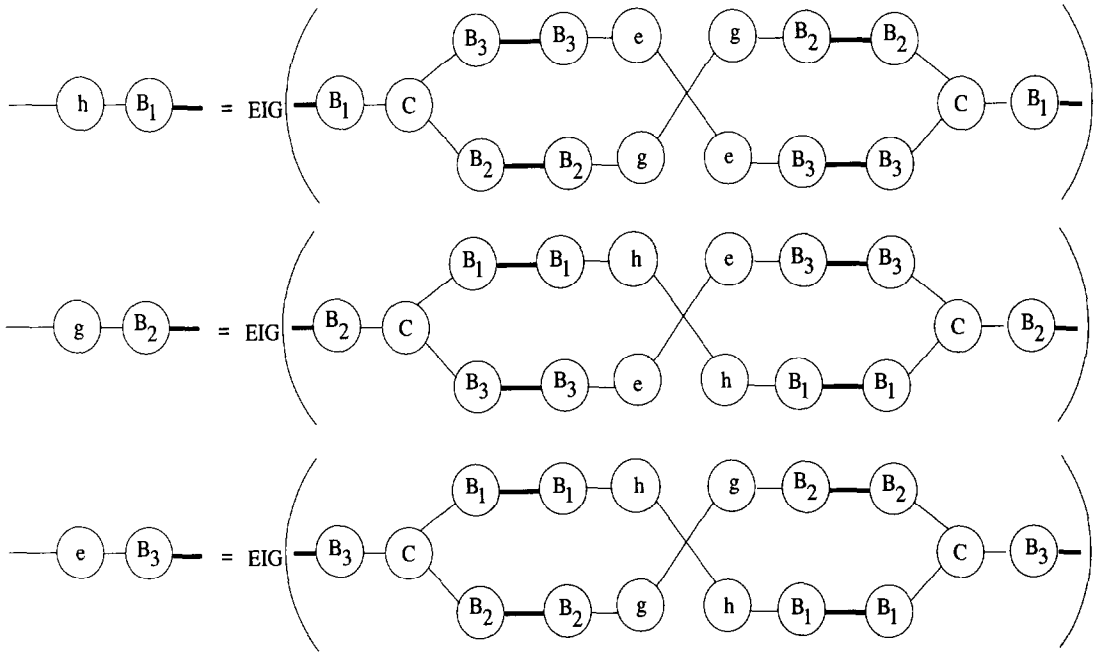

Fig. 6. The iteration steps in the ALS algorithm written in terms of the compression model.

$$\boldsymbol{a}_0 = \boldsymbol{a}_1 \tag{27}$$

The updating steps now become:

$$\lambda = \boldsymbol{a}_0^{\mathrm{T}} \boldsymbol{\Gamma} \mathbf{W}^{\mathrm{T}} \mathbf{W} \boldsymbol{\Gamma} \boldsymbol{a}_0 \tag{28}$$

$$\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathbf{W}^{\mathrm{T}}\mathbf{W} - \lambda \boldsymbol{a}_0 \boldsymbol{a}_0^{\mathrm{T}} \tag{29}$$

If the compression is perfect, the eigenvalue λ from the PBM method is identical to the original eigenvalue of the uncompressed representation.

3.2. PBM applied to the ALS iteration steps

In Fig. 5 the matrix to be diagonalized (i.e., the matrix inserted into the 'EIG' routine) in order to obtain a current estimate of the **G** loading matrix is shown. The computations for the other loadings are analogous. The upper part of Fig. 5 can be said to be a three-mode analogue to a covariance matrix (similar to the **L**, **M**, **N** matrices mentioned above). One way to understand the PBM method is to write out the compression model for each of the arrays in the equation. It is assumed that a loading matrix for a particular mode can be written as linear combinations of the basis matrix for that mode, i.e.,

$$\mathbf{H}_{\mathrm{h}} = \mathbf{B}_1 \mathbf{h} \tag{30}$$

$$\mathbf{G}_{\mathrm{h}} = \mathbf{B}_2 \mathbf{g} \tag{31}$$

$$\mathbf{E}_{\mathrm{h}} = \mathbf{B}_3 \mathbf{e} \tag{32}$$

where {**h**, **g**, **e**} are matrices *not* vectors. The lower part of Fig. 5 shows the upper part of the figure written out in terms of the compression models. Rectangles with dotted lines are drawn around important parts together with a small label symbol in the upper left of the rectangles to indicate which rectangles are comparable. The rectangle labeled 'a' shown in the upper part of the figure encircles the uncompressed $\underline{\mathbf{X}}$ array while in the lower part this array is written in terms of its compression model. The rectangles 'b' and 'c' contain the expressions for the loading matrices **H** and **E**, respectively. Since the basis matrix does not change from one iteration to another it is redundant in the expressions of the ALS algorithm. The redundant basis matrix is indicated with arrows in Fig. 5. Fig. 6 shows the rewriting of the equations for all the loading matrices. Thick lines are used to designate large modes (see Appendix for a short introduction to the diagram notation). By removing the redundant basis matrices on each side of the equation sign and using the PBM version of the eigenvalue decomposition algorithm described above (designated 'PBM-EIG' in the figure) the new algorithm depicted in Fig. 7 is obtained. No thick lines are 'exposed' or 'free' and thus there are no large modes in the calculation. Of course it must again be stressed that the Grammian matrices $\boldsymbol{\Gamma}_i = \mathbf{B}_i^{\mathrm{T}}\mathbf{B}_i$, $i \in [1,2,3]$ should be much smaller than the dual Grammian matrices $\mathbf{B}_i\mathbf{B}_i^{\mathrm{T}}$, $i \in [1,2,3]$. The results from this algorithm (the PBM loading matrices **h**, **g**, **e**) are similar to the PBM scores and loadings in standard (two-mode) principal component analysis as described in Part I of this article. They are not orthogonal, i.e., $\mathbf{h}^{\mathrm{T}}\mathbf{h} \neq \mathbf{I}$, $\mathbf{g}^{\mathrm{T}}\mathbf{g} \neq \mathbf{I}$ and $\mathbf{e}^{\mathrm{T}}\mathbf{e} \neq \mathbf{I}$. If premultiplication of the corresponding compression basis matrices is performed we get that $\mathbf{H}_{\mathrm{h}}^{\mathrm{T}}\mathbf{H}_{\mathrm{h}} = \mathbf{h}^{\mathrm{T}}\boldsymbol{\Gamma}_1\mathbf{h} = \mathbf{I}$, $\mathbf{G}_{\mathrm{h}}^{\mathrm{T}}\mathbf{G}_{\mathrm{h}} = \mathbf{g}^{\mathrm{T}}\boldsymbol{\Gamma}_2\mathbf{g} = \mathbf{I}$ and $\mathbf{E}_{\mathrm{h}}^{\mathrm{T}}\mathbf{E}_{\mathrm{h}} = \mathbf{e}^{\mathrm{T}}\boldsymbol{\Gamma}_3\mathbf{e} = \mathbf{I}$.

4. FLOPS estimations

All FLOPS equations were generated on the basis of the *flops* command in MATLAB. Part I of this article discusses in more detail how FLOPS can be calculated for different types of matrix operations. The FLOPS equations below are constructed on the assumption that the basis matrices are *sparse* [10], which significantly reduces the number of FLOPS required. Sparsity means that a matrix contains a large number of zero elements. See Part I of this article for more details.

The expression for the estimate of the approximate number of FLOPS required for the three-mode PCA is

$$\begin{aligned}\hat{F}_1 = I\{&2A(3NMK + NMA + NKA \\ &+ MKA + N^2A + M^2A + K^2A) \\ &+ A\left[q_a(2N^2 + 8N + 1) + 5N^2 + 3N\right] \\ &+ A\left[q_a(2M^2 + 8M + 1) + 5M^2 + 3M\right] \\ &+ A\left[q_a(2K^2 + 8K + 1) + 5K^2 + 3K\right]\}.\end{aligned} \tag{33}$$

where q_a is the number of iterations per eigenvector in the diagonalization routine. For the estimations performed here it is always assumed that the number of iterations is the same for each factor. This is of course not true, but is introduced to simplify the analysis of the algorithmic performance. I is the number of ALS iteration steps, A is the maximum number of factors and N,M,K are the dimensions of the $\underline{\mathbf{X}}$ array. Before defining the FLOPS formula for the PBM algorithm it is convenient to define the formula for estimated number of FLOPS required for the PBM–EIG routine described above:

$$
\begin{aligned}
&J(n, d_1, g_1, A) \\
&\quad = 2n^2Nd_1^2 + 2n^3g_1 \\
&\quad + A\left[q_a\left(2n^2 + 2n^2g_1 + 3n + 1\right)\right. \\
&\quad \left. + 2n^2g_1 + 5n^2 + 3n + 2n^3g_1\right] \qquad (34)
\end{aligned}
$$

The approximate number of FLOPS for the PBM algorithm is

$$
\begin{aligned}
\hat{\mathcal{F}}_2 &= 2m^2Ag_2 + 2n^2Ag_2 + I\left(6knmA + 2kA^2n\right. \\
&\quad + 2k^2A^2 + 2k^2Ag_3 + 2mA^2k + 2m^2A^2 \\
&\quad \left. + 2m^2Ag_2 + 2nA^2m + 2n^2A^2 + 2n^2Ag_1\right) \\
&\quad + I\left[J(n, d_1, g_1, A) + J(m, d_2, g_2, A)\right. \\
&\quad \left. + J(k, d_3, g_3, A)\right] \qquad (35)
\end{aligned}
$$

where n, m, k are the dimensions of the $\underline{\mathbf{C}}$ array; d_1, d_2, d_3 are the *densities* of the basis matrices $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_3$. The density is the number of non-zero elements in a matrix divided by the total number of elements; g_1, g_2, g_3 are the densities of the Grammian matrices $\mathbf{B}_1^T\mathbf{B}_1$, $\mathbf{B}_2^T\mathbf{B}_2$, $\mathbf{B}_3^T\mathbf{B}_3$.

In order to analyze the two FLOPS formulas some simplifications are introduced: $N = K = M$ and $n = k = m$ and $n = N/r$; r is a parameter for

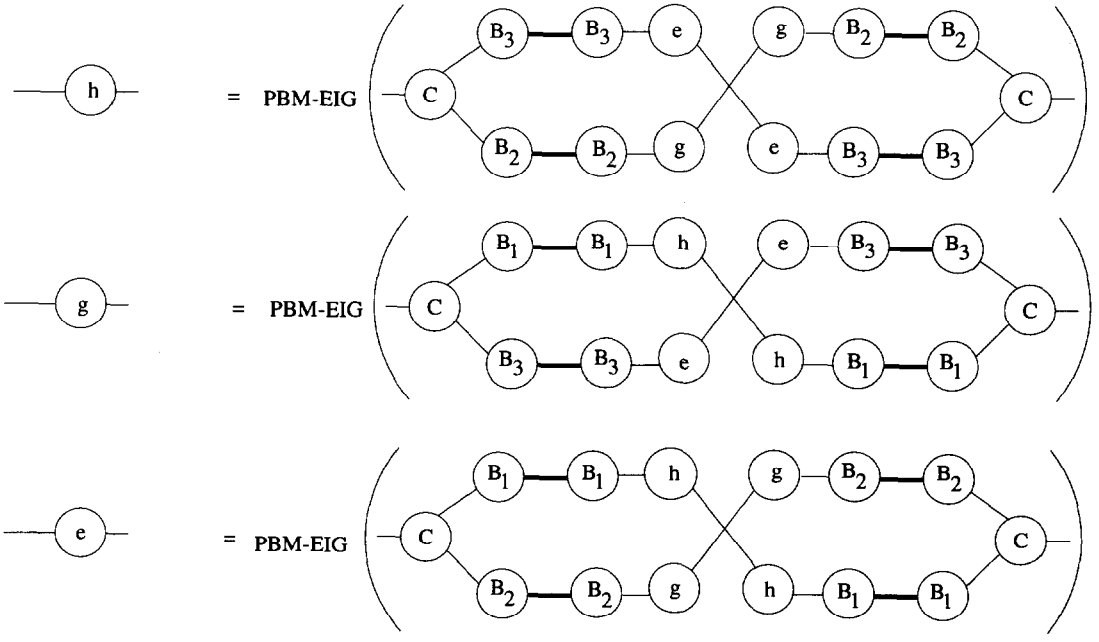

Fig. 7. Same as Fig. 6 except that the redundant basis matrix multiplications have been postponed to after the iteration has converged. Note that now we must use the PBM–EIG routine and not the usual eigenvalue algorithm for decomposition of each of the covariance like matrices.

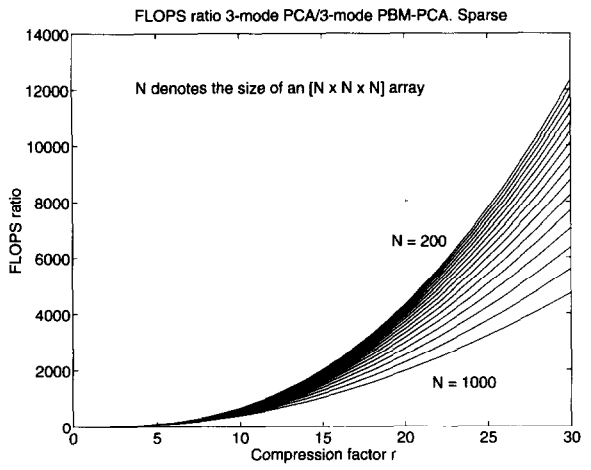

Fig. 8. This figure uses the FLOPS formulas which are discussed in the text and adequately estimates the FLOPS usage of the three-mode PBM–PCA algorithm. Here it is assumed that sparse technology can be used on the basis matrices. A density of 0.1 for every basis matrix is assumed and 0.2 for the corresponding Grammian matrices.

the compression. In a general case this is a vector where each element is a parameter for the different modes. For this particular case the same compression parameter has been selected for each mode. It is desirable that the observed FLOPS ratio

$$f = \frac{F_1}{\mathcal{F}_2} > 1 \tag{36}$$

should be as large as possible. Fig. 8 shows the result of a calculation based on the formulas for $\hat{F}_1$ and $\hat{\mathcal{F}}_2$ assuming the simplifications mentioned above. The following ranges were selected: $N \in [200, 1000]$, $r \in [2, 30]$, five factors are assumed to be extracted and $I = 5$, $q_a = 15$. The densities of the basis matrices were chosen to be 0.1 and the densities of their Grammians to be 0.2. This example corresponds to the one presented in Part I of this article. The highest values of f is obtained when r is large and N is small. If, e.g., $r = 20$, $N = 600$, i.e., from a $[600 \times 600 \times 600]$ array to a $[30 \times 30 \times 30]$ array the PBM algorithm will run approximately 3579 times faster.

5. Results

5.1. Data set 1

An artificial three-mode data set was used for these experiments. A large ratio between the compressed and original representation was constructed. Each basis matrix was given the same dimensions: $\mathrm{Dim}(\mathbf{B}_i) = [84 \times 7]$, $i \in \{1, 2, 3\}$, i.e., each mode in the original representation was more than ten times the size of the coefficient array $\underline{\mathbf{C}}$ with dimensions $\mathrm{Dim}(\underline{\mathbf{C}}) = [7 \times 7 \times 7]$. The number of factors extracted was three and number of ALS iterations was set to five. The number of iterations per factor was set to 15. The number of FLOPS consumed for the uncompressed representation was $F_1 = 118\,803\,325$, and $\mathcal{F}_2 = 486\,025$ FLOPS for the PBM algorithm. The FLOPS ratio is ca. 244. The theoretical estimates of the FLOPS consumption are $\hat{F}_1 = 118\,787\,625$ and $\hat{\mathcal{F}}_2 = 591\,050$ which gives a FLOPS ratio of ca. 200. In this case the formulas underestimate the effect of the PBM approach.

A much larger example could have been chosen but the standard three-mode PCA program could not handle arrays of size larger than $[90 \times 90 \times 90]$.

The calculations were performed on a HP 9000/730 with 64 Mbyte RAM and 1.3 Gbyte hard disk. All programs were written in MATLAB (version 4.0).

5.2. Data set 2

The data matrix was a three-dimensional electron density distribution of a molecule [8] with dimensions $\mathrm{Dim}(\underline{\mathbf{X}}) = [71 \times 38 \times 44]$. This data set was compressed using B-splines to a three mode array with dimensions $\mathrm{Dim}(\underline{\mathbf{C}}) = [30 \times 13 \times 17]$. The compression parameters for the different modes are approximately $\boldsymbol{r} = [2.4, 2.9, 2.6]$. In this data set it was unfortunately not possible to compress the original $\underline{\mathbf{X}}$ further without losing significant information. The number of FLOPS consumed for the original data set was $F_1 = 16\,829\,025$ and for the PBM algorithm $\mathcal{F}_2 = 2\,146\,461$ which corresponds to a FLOPS ratio of

ca. 8. In order to compare the different results the number of iterations per factor was set to 15. The estimated FLOPS were $\hat{F}_1 = 16\,820\,415$ and $\hat{\mathcal{F}}_2 = 2\,074\,699$ which gives a FLOPS ratio of ca. 8 also. The results from the PBM algorithm were converted to the original domain and compared with the results of the three-mode PCA of the original data set, see Fig. 9. The results for this data set are very satisfactory. In addition to a relatively modest compression the three basis matrices were not sparse enough. The densities of the three basis matrices were approximately 0.21, 0.20, 0.12 and this is illustrated in Fig. 10.

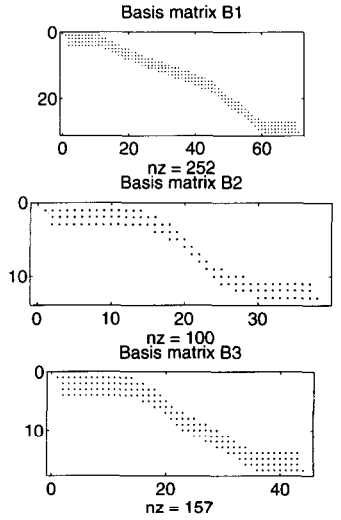

Fig. 10. The densities of the basis matrices used in data set 2.

6. Conclusion

The PBM method applied to three-mode PCA is efficient if the compression ratio is high and/or the basis matrices are sparse. The PBM should also be effective for N-mode PCA in general.

Appendix. Notation

A.1. Name of data objects

Data objects which are extensions of matrices have different names. Some names used are: tensors; multilinear forms; multidimensional arrays; N-arrays; N-mode arrays; N-order arrays; N-way arrays.

In this article we refer to such objects as arrays, N-arrays or N-mode arrays.

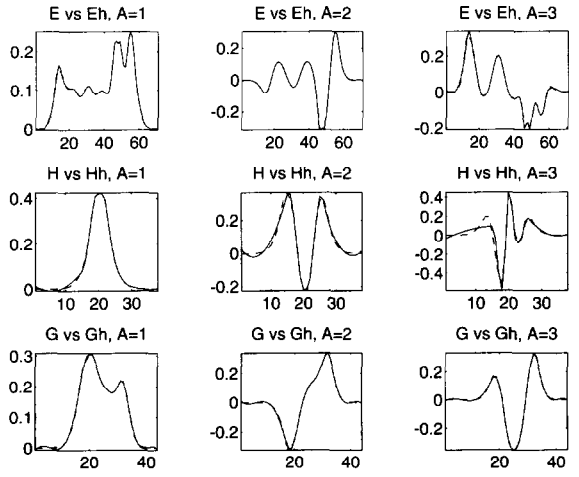

Fig. 9. Reconstruction of loadings for three factors of data set 2. It is in excellent agreement with the loadings from analysis of the uncompressed array. Dotted lines are the loadings from analysis of the uncompressed array.

A.2. The diagram notation

The diagram notation for use in chemometrics is fully described in Ref. [5] and only a short summary is given here. The diagram notation is a graphical visualization of the index topology in explicit summation notation. The diagrams have the appearances of *graphs* and use of graph terminology is therefore appropriate [11]. A diagram contains *nodes* and *edges*. An edge can be attached to one or several nodes. An edge connected to one node only is called *unconnected*. An edge connected to more than one node is called *connected*. It is possible to enable connection between more than two nodes by introducing a special *sum nexus* symbol, but this will not be presented in this article. For more details see Ref. [5]. A node with N unconnected edges is the diagram representation of a single N-mode array not connected to any other. A connected edge signifies summation of *one* index. For the continuous case a connected edge signifies an *integra-*

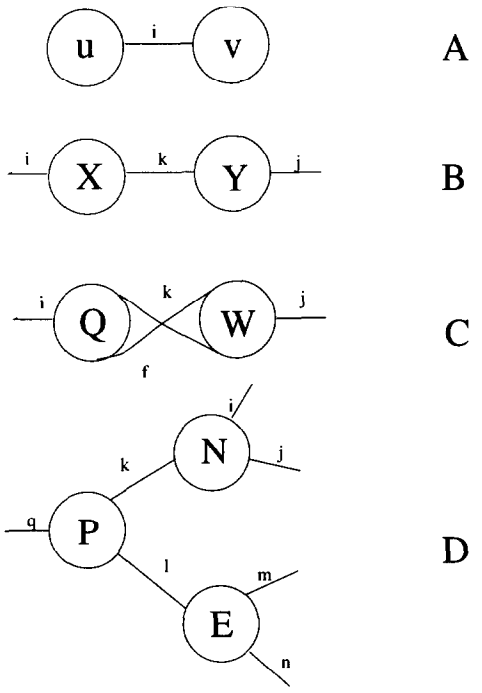

Fig. 11. Presentation of examples from matrix algebra in diagram notation. (A) Scalar product; (B) product of two matrices; (C) and (D) equations involving products of three-mode arrays.

tion. The total number of unconnected edges of an expression is the number of modes (or the *mode number*) of the result. If, e.g., two 3-arrays combine by summing one common index the mode number of the result is $3 + 3 - 2 = 4$. In the center of the node the name of the array is placed. In the vicinity of the edges the correct index names are sometimes written in order to clarify. The index names are written anti-clockwise from the first index. Sometimes it is necessary to indicate which edge signifies the *first* index. For this we use a small bar perpendicular to the edge and this mark is called the *first index pointer* or for short the *fip*.

Mirror reflections of the diagrams are not allowed since this will change the unique placement of the different indices. If reflections were allowed it would not be possible to discriminate between, e.g., the four-mode array x_{ijkl} and the four-mode array x_{ijkl} (assuming a fip has been used to indicate that i is the first index). The consequence of this is that, e.g., when two identical nodes which are connected through more than one edge, a crossing of edges will occur (as in Fig. 2). The crossings themselves have no

Fig. 12. For orthonormal matrices a shorthand notation is introduced which enables a faster manipulation of diagrams. Two arrows which meet head to head will result in an identity matrix. An identity matrix is drawn as a line.

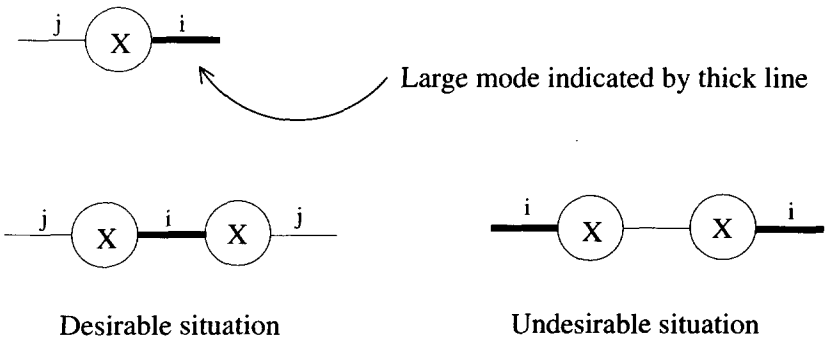

Fig. 13. For a SVD of a matrix of size [10 × 10 000] it is desirable to avoid the large mode. The large mode has been signified by a thick line.

meaning and are just a result of the topological constraints imposed on the notation.

Fig. 11 shows a few examples of array diagram equations. The corresponding summation formulas for the diagram equations presented are:

(A) $\sum_i u_i v_i$. This is the standard inner product.

(B) $\sum_k x_{ik} y_{kj}$. This is standard matrix product.

(C) $\sum_k \sum_f q_{ifk} w_{kfj}$. An array product between two three-mode arrays. The result is a matrix because the number of unconnected edges is two.

(D) $\sum_k \sum_l p_{qlk} n_{ikj} e_{lnm}$. Here the result is a five-mode array since five free indices are seen.

Instead of using index names and fips for indicating the different modes of the arrays a shorthand notation has been constructed. There are especially two cases where a shorthand notation has been found useful: (i) matrices with orthonormal column vectors; (ii) modes of large size.

If $\mathbf{Q}$ is a matrix with orthonormal column vectors we have that $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix. If $\mathbf{Q}$ is not square we have that $\mathbf{Q}\mathbf{Q}^T \neq \mathbf{I}$. Thus we need to discriminate between the two different modes of the matrix. *Arrows* have been chosen to distinguish between the modes. When two arrows meet head to head we have the case $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$. Fig. 12 illustrates the idea (here we see $\mathbf{E}^T\mathbf{E} = \mathbf{I}$). The identity matrix is for convenience drawn as a connected or as an unconnected edge with no matrix element attached.

In some problems it is necessary to avoid that large modes become unconnected edges. A simple example from SVD illustrates the main idea. If the dimension of $\mathbf{X}$ is $[10 \times 10\,000]$ and the object is to find the eigenvalues the fastest method is to calculate the eigenvalues of the covariance matrix $\mathbf{X}\mathbf{X}^T$ which has a dimension of $[10 \times 10]$. The rank of $\mathbf{X}$ cannot be larger than 10 which means that eigenvalue decomposition of $\mathbf{X}^T\mathbf{X}$ (dimension $[10\,000 \times 10\,000]$) would be a waste of resources; see Fig. 13 for illustration.

References

[1] I. Pelczer and S. Szalma, Multidimensional NMR and data processing, *Chemical Reviews*, 91 (1991) 1507–1524.

[2] B.K. Alsberg and O.M. Kvalheim, Compression of *n*th-order data arrays by B-splines. Part 1. Theory, *Journal of Chemometrics*, 7 (1993) 61–73.

[3] B.K. Alsberg, E. Nodland and O.M. Kvalheim, Compression of *n*th-order data arrays by B-splines. Part 2. Application to second-order FT-IR spectra, *Journal of Chemometrics*, 8 (1994) 127–145.

[4] B.K. Alsberg and O.M. Kvalheim. Speed improvement of multivariate algorithms by the method of postponed basis matrix multiplication. Part I. Principal component analysis, *Chemometrics and Intelligent Laboratory Systems*, 24 (1994) 31–42.

[5] B.K. Alsberg, A diagram notation for *N*-mode array equations, *Psychometrika*, (1993) submitted.

[6] P.M. Kroonenberg, *Three Mode Principal Component Analysis*, DSWO Press, Leiden, 1983.

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

[8] B.K. Alsberg and O.M. Kvalheim, Compression of three-mode data arrays by B-splines prior to three-mode principal component analysis (PCA), *Chemometrics and Intelligent Laboratory Systems*, 23 (1994) 29–38.

[9] G.H. Golub and C.F. van Loan, *Matrix Computations* (John Hopkins Series in the Mathematical Sciences, No. 3), The John Hopkins University Press, Baltimore, MD, 2nd edn., 1989.

[10] S. Pissanetsky, *Sparse Matrix Technology*, Academic Press, London, 1984.

[11] J.R. Wilson, *Introduction to graph theory*, Longman Scientific & Technical, Essex, 3rd edn., 1985.