ELSEVIER

Chemometrics and Intelligent Laboratory Systems 23 (1994) 29–38

Chemometrics and intelligent laboratory systems

Compression of three-mode data arrays by B-splines prior to three-mode principal component analysis

Bjørn K. Alsberg [*], Olav M. Kvalheim

Department of Chemistry, University of Bergen, Allegt. 41, N-5007 Bergen, Norway

(Received 6 July 1993; accepted 14 January 1994)

Abstract

Three-mode PCA is very computer demanding. It requires a large amount of storage space and many floating point operations (FLOPS). By using three-mode B-spline compression of three-mode data arrays, the original data array can be replaced by a smaller coefficient array. Three-mode principal component analysis (PCA) is then performed on the much smaller coefficient array instead of on the original array. For the compression approach to be efficient the three-mode data array is assumed to be well approximated by smooth functions. The smoothness affects the dimensions of the coefficient array. It is always possible to approximate the data to any precision but the reward in reduced computation time and storage is lost when the dimensions of the coefficient array approach the dimensions of the original array.

1. Introduction

N-mode arrays represent extensions of two-mode arrays, i.e., matrices. There are several other names for these objects: tensors, *N*-way arrays, *N*-arrays, and multilinear forms. We will here refer to such data objects as *N-mode arrays* or *N-arrays.*

Several analytical instruments produce data sets of the *N*-array type. Such arrays can be very useful for obtaining, e.g., information about constituents in solutions [1,2] or atom assignment of crosspeaks in multidimensional NMR [3]. Unfortunately, the *N*-arrays present serious storage and computational problems. The number of data elements increases rapidly and the need for some reduction and improvement in the data handling procedure is necessary. In previous papers [4–6] we have suggested that compression by B-splines or by means of another suitable basis may be an efficient way of partially solving the increased data size problem.

There are two main types of compressions: lossless and lossy [7]. Lossless compression restores the data perfectly but does not attain large compression ratios as in lossy compression. The B-spline method used here is a lossy compres-

[*] Corresponding author. E-mail: alsberg@kj.uib.no.

sion. What is an acceptable error in the reconstruction compared to the original array must be decided by the investigator and must be considered to be problem dependent.

In this article we focus on three-mode principal component analysis (3MPCA) which is computer demanding [8] for arrays of size larger than $[50 \times 50 \times 50]$. In such cases powerful workstations are necessary. If data from an instrument should be analyzed directly, arrays of, e.g., dimensions $[1000 \times 500 \times 500]$ are realistic. Examples of instruments giving rise to such data sets are, e.g., three-dimensional NMR spectra or two-dimensional IR versus time/temperature. Inserting such an array without any compression or pretreatment into a standard 3MPCA program would require a very powerful computer. Fortunately, the spectra from analytical instruments often contain some smoothness which enables the use of compression methodology. The compression part of course, must not be too computer demanding in itself.

Before reading the next section the reader is encouraged to read the appendix which explains the different notations used in this article.

2. Three-mode PCA

In singular value decomposition (SVD) for 2-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 column-wise 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:

$$\mathbf{X} = \mathbf{G}\mathbf{D}(\mathbf{H}^{\mathrm{T}} \otimes \mathbf{E}^{\mathrm{T}}) \tag{4}$$

where $\mathbf{X}$ and $\mathbf{D}$ are unfolded representations of the 3-array (see Appendix) and $\mathbf{G}$, $\mathbf{H}$ and $\mathbf{E}$ are column-wise orthonormal loading matrices for each mode. Eq. 4 is the Tucker3 model [8,9]. It should be stressed that the 3-arrays (represented as matrices) in Eq. 4 must be unfolded in the same way. $\mathbf{D}$ is the core matrix (or core array). Eq. 4 written in explicit summation is:

$$x_{ijk} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{ip} h_{jq} e_{kr} d_{pqr} \tag{5}$$

There are different methods for solving the Tucker3 model. A method similar to the approach used for SVD of 2-arrays (see Eqs. 2 and 3) is the Tucker Method I [8,9] which obtains estimates for the loading matrices $\mathbf{E}$, $\mathbf{H}$ and $\mathbf{G}$ by extracting all eigenvectors corresponding to nonzero roots of the three covariance matrices with elements:

$$l_{ii'} = \sum_{j=1}^{J} \sum_{k=1}^{K} x_{ijk} x_{i'jk} \tag{6}$$

$$m_{jj'} = \sum_{i=1}^{I} \sum_{k=1}^{K} x_{ijk} x_{ij'k} \tag{7}$$

$$n_{kk'} = \sum_{i=1}^{I} \sum_{j=1}^{J} x_{ijk} x_{ijk'} \tag{8}$$

Using the estimated three loading matrices, Eq. 5 shows the core array $\underline{\mathbf{D}}$ can be expressed as:

$$d_{pqr} = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{K} g_{ip} h_{jq} e_{kr} x_{ijk} \tag{9}$$

This equation is also shown in Fig. 1 using the diagram notation.

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

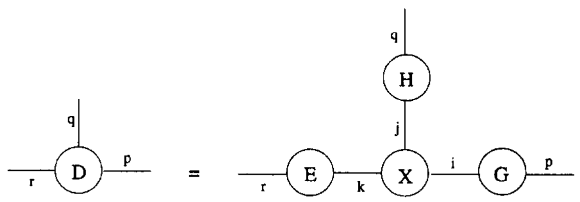

Fig. 1. A diagram equation which shows how to obtain the core array given the loading matrices **G**, **H**, **E** and the original data array **X**.

an alternating least squares (ALS) algorithm [8] can be used. The ALS approach is a common method for solving the Tucker3 model. The algorithm is initiated by first computing estimates of the loading matrices $\mathbf{G}_0$, $\mathbf{E}_0$ and $\mathbf{H}_0$ by the Tucker Method I. $\mathbf{G}_0$, $\mathbf{E}_0$ and $\mathbf{H}_0$ are subsequently inserted into the following iteration steps:

```
For $i = 1$ to iterations
  begin
    $\mathbf{A} = \mathbf{X}_1(\mathbf{H}_{i-1} \otimes \mathbf{E}_{i-1})$    (10)
    $\mathbf{G}_i = eig(\mathbf{A}\mathbf{A}^T)$    (11)
    $\mathbf{A} = \mathbf{X}_2(\mathbf{E}_{i-1} \otimes \mathbf{G}_i)$    (12)
    $\mathbf{H}_i = eig(\mathbf{A}\mathbf{A}^T)$    (13)
    $\mathbf{A} = \mathbf{X}_3(\mathbf{G}_i \otimes \mathbf{H}_i)$    (14)
    $\mathbf{E}_i = eig(\mathbf{A}\mathbf{A}^T)$    (15)
  end
```

Matrix **A** temporarily stores the results and is overwritten when new values are assigned to its matrix elements. The function *eig* returns the first largest eigenvectors of $\mathbf{A}\mathbf{A}^T$ (this is a reduced decomposition). This algorithm is also formulated in the diagram notation in Fig. 2. $\mathbf{X}_1$, $\mathbf{X}_2$ and $\mathbf{X}_3$ are unfolded representations of the original 3-array with dimensions $Dim(\mathbf{X}_1) = [N \times (MK)]$, $Dim(\mathbf{X}_2) = [M \times (NK)]$, $Dim(\mathbf{X}_3) = [K \times (NM)]$. The number of iterations can be determined by minimizing the error of fit of the model to the observed data. This approach, however, is not used in the present paper. The reason for this is to ensure total control with respect to the number of iterations when investigating the FLOPS usage of the ALS algorithm applied to different, but comparable, data representations.

3. FLOPS estimations

The 3MPCA program is written in MATLAB [10] which is very powerful for matrix computations. In order to obtain a measure of the efficiency of the ALS algorithm on uncompressed and compressed representations the *flops* command in MATLAB was employed. The FLOPS equation and observed performance of the ALS algorithm presented in this paper are based on the results from this command. In Table 1 a few examples are given where the FLOPS equations for simple matrix expressions are presented. For each matrix expression in an algorithm the FLOPS formula is found and the total FLOPS consumption is obtained by summing over the different contributions. Only the most important parts of an algorithm are investigated. All WHILE loops

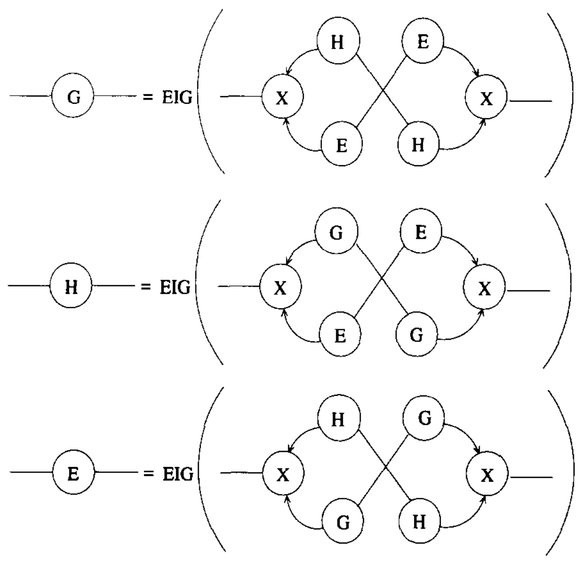

Fig. 2. Illustration of the ALS algorithm using diagram formulation.

Table 1
This table shows the number of FLOPS required for some example matrix operations

Matrix operation	FLOPS required	Dimensions of matrices
$\mathbf{XY}$	$2nmk$	$Dim(\mathbf{X}) = [n \times m]$, $Dim(\mathbf{Y}) = [m \times k]$
$\mathbf{X}y$	$2nm$	$Dim(\mathbf{X}) = [n \times m]$, $Dim(y) = [m \times 1]$
y^Ty	$2m$	$Dim(y) = [m \times 1]$
$\mathbf{X}+\mathbf{X}$	nm	$Dim(\mathbf{X}) = [n \times m]$
$\mathbf{X} = \mathbf{X} - tp^T$	$3nm$	$Dim(\mathbf{X}) = [n \times m]$, $Dim(t) = [n \times 1]$ $Dim(p) = [m \times 1]$,
$(\mathbf{XY})\mathbf{Z}$	$2nk(m+r)$	$Dim(\mathbf{X}) = [n \times m]$, $Dim(\mathbf{Y}) = [m \times k]$ $Dim(\mathbf{Z}) = [k \times r]$

are changed to FOR loops to determine the FLOPS consumption for such steps.

The detailed description of how a FLOPS formula is constructed will not be presented.

Using the same approach to the ALS algorithm the following equation is obtained:

$$F_{\text{tot}} = I\big[2A(3NMK + NMA + NKA + MKA + N^2A + M^2A + K^2A) + N^3 + M^3 + K^3\big] \tag{16}$$

where the last entries $N^3 + M^3 + K^3$ stem from the eigenvalue decomposition which is approximately a third degree increase in FLOPS. Here it is assumed that the core array is symmetric, i.e., its dimensions are $[A \times A \times A]$. This will also be the case in the examples presented below. I is the number of iterations. Note that this formula slightly underestimates the number of FLOPS required but is close to the true FLOPS count.

4. B-splines

B-splines [11,12] can be used to fit almost any type of function. Significant compression, however, is obtained if the data at hand have some smoothness. A one-dimensional B-spline is a linear combination of basis functions b_j:

$$f(x) = \sum_{j=1}^{u-\alpha-1} c_j b_j(x) \tag{17}$$

where $f(x)$ is the function to be approximated, c_j the B-spline coefficients, u is the number of knot points and α the local polynomial degree. The basis functions $b_j(x)$ (which for discrete representations are located in a matrix called **B**) are defined by the knot vector $\boldsymbol{h}$. Knots are points located along the independent axis which define the shape and location of the B-spline basis functions. The basis can be constructed by a recursive formula [12,13] using the information in the vector $\boldsymbol{h}$. The knot vector $\boldsymbol{h}$ is therefore stored instead of the much larger B-spline basis matrix **B**.

For compression of N-mode arrays N B-spline basis matrices (or N $\boldsymbol{h}$ vectors) are needed.

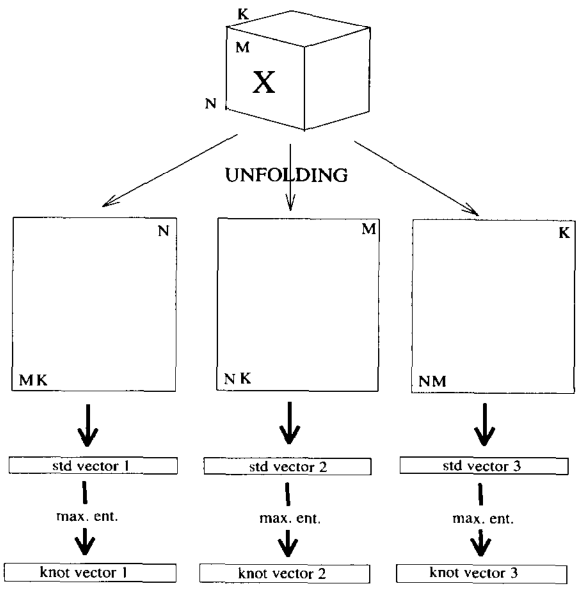

Fig. 3. Illustration of the steps for finding the knot vectors for the three different modes. For each unfolding we find a representative vector (RV) which is said to represent the current mode. In this case the RV is the standard deviation vector (indicated as 'std. vector' in the figure). The maximum entropy method as described in ref. [4] is applied to the standard deviation vector and a knot vector is obtained.

The steps for obtaining the knot vectors for each mode are:

(i) The original data array is unfolded to a matrix. One mode of this matrix corresponds to the current mode of the original data array for which the knot vector is to be found.

(ii) A representative vector (RV) is obtained for the current mode. An RV can be, e.g., the mean or the standard deviation vector of the unfolded matrix along the current mode. In this paper the standard deviation vector is used as the RV.

(iii) The so-called maximum entropy method [4,14] is applied to the RV and the interval vector from this procedure is used as the knot vector for the current mode.

See Fig. 3 for a graphical illustration of the method.

Given three knot vectors named $\boldsymbol{h}_n$, $\boldsymbol{h}_m$, and $\boldsymbol{h}_k$ three corresponding basis matrices $\mathbf{B}_n$, $\mathbf{B}_m$ and $\mathbf{B}_k$ can be generated (by using a recursive formula [12,13]). The three different modes were

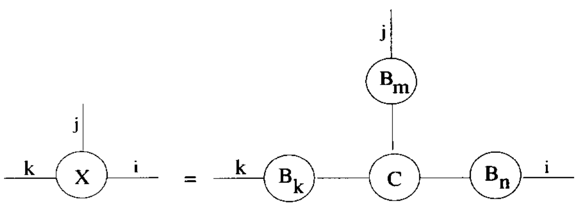

Fig. 4. Diagram of the three mode B-spline model. Illustration of how the three knot vectors for three-mode B-spline compression divide the data array into segments.

thus labeled n, m and k. If the three-mode data matrix $\underline{\mathbf{X}}$ has dimensions $[N \times M \times K]$ then the dimensions of the basis matrices are $[N \times n_c]$ for $\mathbf{B}_n$, $[M \times m_c]$ for $\mathbf{B}_m$ and $[K \times k_c]$ for $\mathbf{B}_k$. Let $\{\alpha_n, \alpha_m, \alpha_k\}$ be the local degree of polynomial for the spline and $Dim(\boldsymbol{h}_i)$ be the number of elements in a knot vector, then we have that $n_c = Dim(\boldsymbol{h}_n) - \alpha_n - 1$, $m_c = Dim(\boldsymbol{h}_m) - \alpha_m - 1$ and $k_c = Dim(\boldsymbol{h}_k) - \alpha_k - 1$. It is desirable to have small values of n_c, m_c and k_c without too much error. The coefficient array $\underline{\mathbf{C}}$ has thus dimensions $[n_c \times m_c \times k_c]$. The model assumption for the three-mode B-spline model is shown in Fig. 4 which has the same index topology as the Tucker3 model. The equation for the generation of the core array $\underline{\mathbf{C}}$ is shown in Fig. 5.

4.1. Compression of a subset of modes

As mentioned earlier an efficient compression ratio is achieved if the data array can be well represented by smooth basis functions. It is a problem when some of the modes are not smooth. B-spline compression may still be of benefit if applied to the smooth modes only. In spectral problems it is realistic to encounter, e.g., three-mode arrays where only two of the modes are smooth. The third mode may also be much smaller than the smooth modes.

In refs. [15,16] Kiers et al. presented two efficient algorithms for the PARAFAC and TUCKALS3 algorithms for cases when the size of one of the modes is much larger than the other two. These algorithms do not cover the case discussed in this article when all modes are large, but it is possible to imagine that two large modes are compressed and the third is handled by the algorithms developed by Kiers et al.

5. Experiments

5.1. Data set

The data used for this example are the three-dimensional electron density distribution of the inhibitory neurotransmitter γ-aminobutyric acid (GABA). This distribution was calculated for the molecule using the AM1 quantum mechanical model [17]. The electron density surface was computed on a grid of size $[71 \times 38 \times 44]$.

5.2. Results

The $\boldsymbol{h}_n$ knot vector was generated by using the maximum-entropy method [4,14] on the standard deviation vector obtained from the unfolded matrix of size $[(38 \cdot 44) \times 71]$. Correspondingly the $\boldsymbol{h}_m$ vector was generated from the unfolded matrix of size $[(71 \cdot 44) \times 38]$ and $\boldsymbol{h}_k$ from the unfolded matrix of size $[(71 \cdot 38) \times 44]$. The MATLAB B-spline toolbox [18] was used to produce the basis matrices. The sizes of the knot vectors after assuming third degree polynomial B-splines for the n and k mode and second degree polyno-

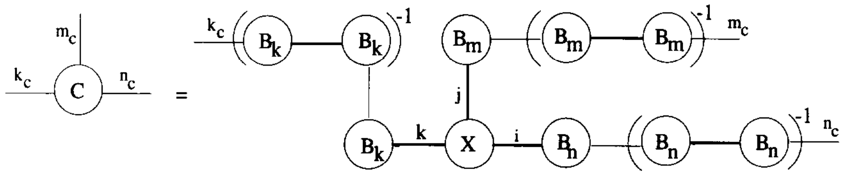

Fig. 5. Diagram of how to find the coefficient core matrix in three-mode B-spline compression. This diagram is obtained by solving for $\underline{\mathbf{C}}$ in Fig. 4.

Table 2
Dimensions of matrices involved in the experiment

Matrix	Dimensions
G	[71×3]
H	[38×3]
E	[44×3]
$\mathbf{G}_c$	[30×3]
$\mathbf{H}_c$	[13×3]
$\mathbf{E}_c$	[17×3]

mial for the m mode were [1 × 34] for $\boldsymbol{h}_n$, [1 × 16] for $\boldsymbol{h}_m$ and [1 × 21] for $\boldsymbol{h}_k$. The basis matrix $\mathbf{B}_n$ had size [71 × 30], $\mathbf{B}_m$ had size [38 × 13] and $\mathbf{B}_k$ had size [44 × 17]. The resulting $\underline{\mathbf{C}}$ 3-array from the B-spline compression had consequently dimensions [30 × 13 × 17] which is a compression ratio of ca.

$$\frac{71 \cdot 38 \cdot 44}{30 \cdot 13 \cdot 17} \approx 17.9$$

The 3MPCA program was used on both the original representation $\underline{\mathbf{X}}$ and the compressed representation $\underline{\mathbf{C}}$.

Two sets of loading matrices were produced: {**G**, **H**, **E**} for the original uncompressed representation and {$\mathbf{G}_c$, $\mathbf{H}_c$, $\mathbf{E}_c$} for the compressed representation. Three factors were extracted for both representations. The sizes of the different loading matrices are shown in Table 2.

The total number of FLOPS consumed for the original data array was $F = 75677508$. The corresponding number of FLOPS using the compressed representation was $F_c = 4799174$. The ra-

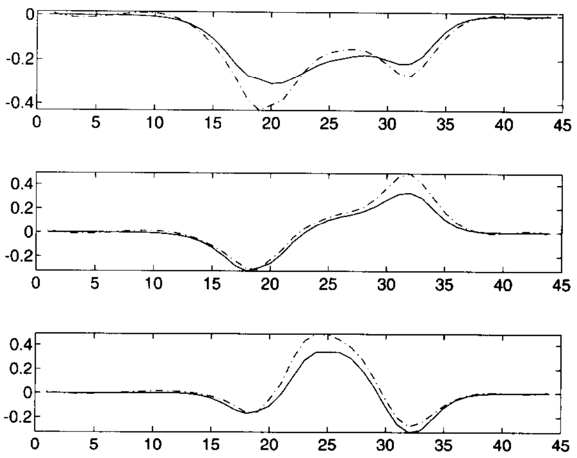
Fig. 7. Comparison between estimated and original representation of the **E** matrix. The dotted line indicates the estimated **E** matrix. The upper figure is the first factor, the middle figure is the second factor and the bottom figure is the third factor.

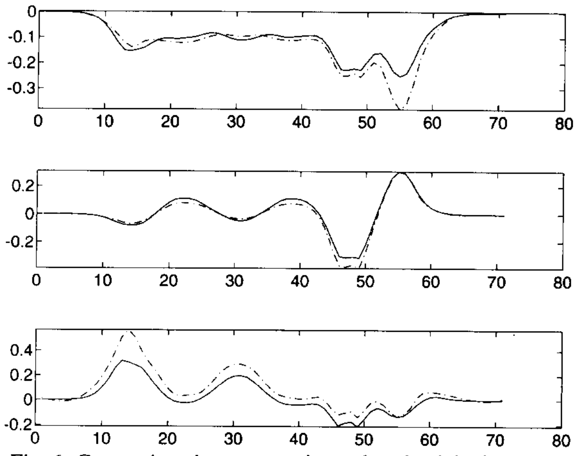
Fig. 6. Comparison between estimated and original representation of the **G** matrix. The dotted line indicates the estimated **G** matrix. The upper figure is the first factor, the middle figure is the second factor and the bottom figure is the third factor.

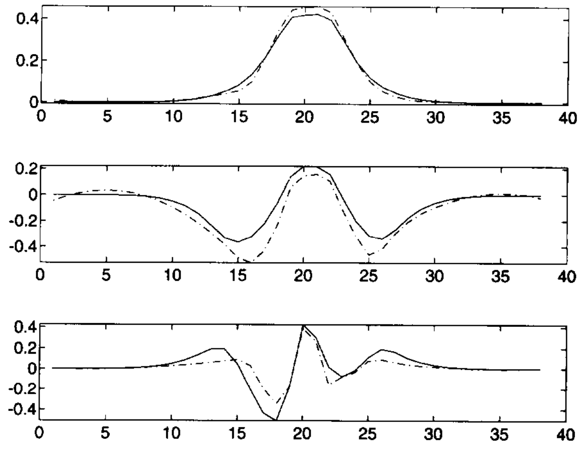
Fig. 8. Comparison between estimated and original representation of the **H** matrix. The dotted line indicates the estimated **H** matrix. The upper figure is the first factor, the middle figure is the second factor and the bottom figure is the third factor.

tio $F/F_c \approx 15.8$ is in the same range as the compression ratio. Using the derived formula 16 for the number of FLOPS consumed by the ALS algorithm the estimated ratio was approximately 14.7 which is in close agreement with the observed values. A comparison between the original loading matrices **G**, **H** and **E** and the estimated loading matrices $\hat{\mathbf{G}}$, $\hat{\mathbf{H}}$ and $\hat{\mathbf{E}}$ from the compressed representation was made. The following equations were used:

$$\hat{\mathbf{G}} = \mathbf{B}_n \mathbf{G}_c \tag{18}$$

$$\hat{\mathbf{H}} = \mathbf{B}_m \mathbf{H}_c \tag{19}$$

$$\hat{\mathbf{E}} = \mathbf{B}_k \mathbf{E}_c \tag{20}$$

It must be stressed that these estimates are not strictly correct, i.e., they do not provide true estimates which should be column-wise orthonormal. Orthogonalization of the loading matrix does not produce the correct result. At the present this is the only way of obtaining such estimates without rewriting 3MPCA to compensate for distortion effects in backestimates. It is possible to construct a 3MPCA program which compensates for these effects. A detailed presentation of a solution to the problem is presented in two forthcoming papers [19,20].

The comparison between $\hat{\mathbf{G}}$ and **G** can be seen in Fig. 6, between $\hat{\mathbf{E}}$ and **E** in Fig. 7 and between $\hat{\mathbf{H}}$ and **H** in Fig. 8. The results for this data set must be characterized as satisfactory.

6. Discussion

B-splines are not necessarily the optimal choice of basis functions for compression. B-splines can be viewed as a subset of the much broader class of wavelets [21] which has been used in several problems in physics, chemistry and image compression. The problem with compression as used in 3MPCA is that the estimates of later factors increasingly deviate from the true factors of the uncompressed array. The compression stage *does* remove information so there is a trade-off between faster computations and accuracy. In addition it has been stressed that the data must be smooth which is not always the case. For the case of non-smooth N-arrays when all the modes are large, no solution to the increased computational problem has yet been found. The use of coefficients instead of the original data has similarities to the sparse matrix technology [22]. Both methods utilize the special structure in data arrays to achieve compression and faster computation. Sparse algorithms can be constructed based on the fact that the matrix contains a large amount of zeros. These algorithms give the exact answer and therefore no errors are introduced because of the compressed representation. B-splines could have been fitted to the data such that it was perfect, but for real world data no compression or improvement in speed or storage would have been obtained. Still the development of more efficient higher mode algorithms is very important and should be used in combination with new ways of compressing/representing the array structure.

Acknowledgements

The authors wish to thank Dr. René Henrion at Humboldt University in Berlin for discussing the ALS algorithm and compression. Dr. Knut Børve at the University of Bergen, is thanked for providing the electron density distribution data of GABA. BKA is indebted to The Royal Norwegian Council for Scientific and Industrial Research (NTNF) for financial support.

Appendix A. Definitions and terminology

A.1. Names of data objects

The data objects discussed in the article are referred to by many different names. Some of the most common ones are tensor, N-order array, higher order array/matrix, N-way array, N-array, N-mode array or N-dimensional array. Thus the names mode, order, way and dimension are used to designate the different indices in the array. In this article we have chosen to use the word 'mode' when discussing the different indices in the arrays. A matrix has two indices and thus two modes. A 3-array has three indices and thus three modes and so on.

A.2. Typographical notes

All scalars are written as lowercase letters, e.g., k. All vectors are written as bold italic lowercase letters, e.g., $\boldsymbol{u}$. All matrices are written as bold uppercase letters, e.g. $\mathbf{P}$. We use underlined, boldface uppercase letters to indicate an array with three modes, e.g. $\underline{\mathbf{X}}$. Transpose is indicated by a superscript T.

A.3. Definition of unfolding

The word 'unfolding' is defined to be the reorganization of an N-mode array into a vector or a matrix. A more detailed description of unfolding can be found in refs. [23,24].

Appendix B. Notations for N-mode equations

B.1. The Kronecker notation

The $\otimes$ symbol signifies the Kronecker product [23]. The right-oriented Kronecker product of two matrices $\mathbf{A}$ and $\mathbf{B}$ of dimensions $[n \times m]$ and $[k \times j]$ is

$$\mathbf{M} = \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1m}\mathbf{B} \\ \vdots & & \vdots \\ a_{n1}\mathbf{B} & \dots & a_{nm}\mathbf{B} \end{bmatrix} \qquad (21)$$

where $\mathbf{M}$ has dimensions $[nk \times mj]$.

If a 3-array $\underline{\mathbf{X}}$ has dimensions $[I \times J \times K]$ it can be rearranged into three essential matrices: $[K \times IJ]$, $[I \times KJ]$ and $[J \times KI]$ (or $[K \times JI]$, $[I \times JK]$ and $[J \times IK]$). There are in general $N!$ possible unfoldings or permutation of indices. The Kronecker notation thus represents all N-arrays as ordinary matrices by unfolding.

B.2. The diagram notation

The diagram notation is fully described in ref. [25] and only a short summary will be 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 [26]. A diagram contains nodes and edges. Nodes signify array elements (e.g., x_{ijk}) and one edge can signify either summation over one index or an index which is not involved in summation. 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 is not presented in this article. For more details see ref. [25]. A node with N unconnected edges will be the diagram representation of a single array element not connected to any other. A connected edge signifies summation of one index. When two arrays share a common index a connected edge is drawn between them. The total number of unconnected edges of the 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 will be $3 + 3 - 2 = 4$. In the center of the node the name of the array is placed. In the vicinity of the edges the correct index names can be 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 a small bar perpendicular to the edge is used; this mark is called the *first index pointer* or just the *fip*.

Fig. 9 shows a few examples of array diagram equations. The corresponding summation formulas for the diagram equations presented are:

$$\sum_i u_i v_i \qquad \text{(A)}$$

This is the standard inner product.

$$\sum_k x_{ik} y_{kj} \qquad \text{(B)}$$

This is a standard matrix product.

$$\sum_k \sum_f q_{ifk} w_{kfj} \qquad \text{(C)}$$

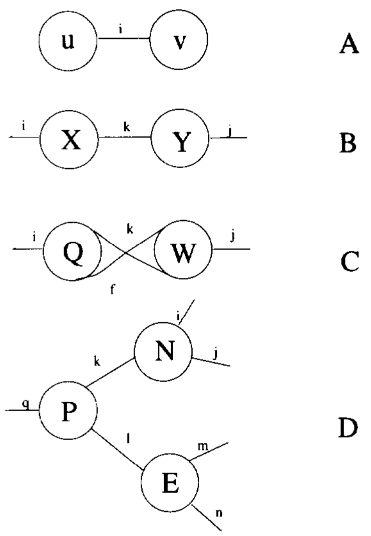

Fig. 9. Examples of diagrams. (A) is an inner product of two vectors. (B) is a matrix product. (C) and (D) are examples of products between three-mode arrays.

An array product between two three-mode arrays. The result is a matrix because the number of free or unconnected indices is two.

$$\sum_k \sum_l p_{qkl} n_{kij} e_{lnm} \qquad \text{(D)}$$

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 **W** is a matrix with orthonormal column vectors we have that $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ where **I** is the identity matrix. If **W** is not square we have that $\mathbf{W}\mathbf{W}^T \neq \mathbf{I}$. Thus we need to discriminate between the two different modes of the matrix. Arrows

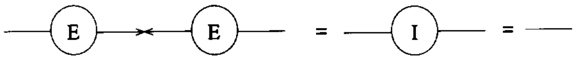

Fig. 10. A diagram with arrows is used to mark the two different indices on orthogonal matrices. Here we have chosen the convention to let two arrows heading versus each other signify an identity.

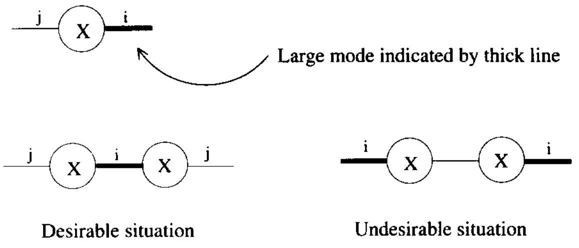

Fig. 11. For a SVD of a matrix of size [10×10000] it is desirable to avoid the large mode. The large mode is indicated by a thick line.

have been chosen to distinguish between the modes. When two arrows meet head to head we have the case $\mathbf{W}^T\mathbf{W} = \mathbf{I}$. Fig. 10 illustrates the idea. 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 **X** is [10 × 10000] 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 × 10]. The rank of **X** cannot be larger than 10 which means that eigenvalue decomposition of $\mathbf{X}^T\mathbf{X}$ (dimension [10000 × 10000]) would be a waste of resources; see Fig. 11 for illustration.

References

[1] O.M. Kvalheim and Y. Liang, Heuristic evolving latent projections: resolving two-way multicomponent data. 1. Selectivity, latent-projective graph, datascope, local rank, and unique resolution, *Analytical Chemistry*, 64 (1992) 937-946.

[2] Y. Liang, O.M. Kvalheim, H.R. Keller, D.L. Massart, P. Kiechle and F. Erni, Heuristic evolving latent projections: resolving two-way multicomponent data. 2. Detection and resolution of minor constituents, *Analytical Chemistry*, 64 (1992) 947-953.

[3] I. Pelczer and S. Szalma, Multidimensional NMR and data processing, *Chemical Review*, 91 (1991) 1507-1524.

[4] 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.

[5] 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 FTIR spectra, *Journal of Chemometrics*, 8 (1994) 127-146.

[6] B.K. Alsberg, Representation of spectra by continuous functions, *Journal of Chemometrics*, 7 (1993) 177–193.

[7] J.A. Storer, *Data Compression. Methods and Theory*, Computer Science Press, Rockville, MD, 1988.

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

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

[10] *MATLAB Reference Guide*, The MathWorks Inc., Natick, MA, 1992.

[11] C. de Boor, *A Practical Guide to Splines. Applied Mathematics Sciences*, Springer, New York, 1978.

[12] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide*, Academic Press, San Diego, CA, 2nd edn., 1990.

[13] W. Cheney and D. Kincaid, *Numerical Mathematics and Computing*, Brooks/Cole, Belmont, CA, 1980.

[14] T.V. Karstang and R.J. Eastgate, Multivariate calibration of an X-ray diffractometer by partial least squares regression, *Chemometrics and Intelligent Laboratory Systems*, 2 (1987) 209–219.

[15] H.A.L. Kiers and W.P. Krijnen, An efficient algorithm for PARAFAC of three-way data with large number of observation units, *Psychometrika*, 56 (1992) 147–152.

[16] H.A.L. Kiers, P.M. Kroonenberg and J.M. ten Berge, An efficient algorithm for Tuckals3 on data with large number of observation units, *Psychometrika*, 57 (1992) 415–422.

[17] M.J.S. Dewar, E.G. Zoebisch, F.H. Eamonn and J.P. Stewart, AM1: A new general purpose quantum mechanical molecular model, *Journal of the American Chemical Society*, 107 (1985) 3902–3909.

[18] C. de Boor, *Spline Toolbox for Use with MATLAB. User's Guide*, The MathWorks Inc., South Natick, MA, 1990.

[19] 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*, (1994) in press.

[20] B.K. Alsberg and O.M. Kvalheim, Speed improvement of multivariate algorithms by the method of postponed basis matrix multiplication. Part II. Three-mode principal component analysis, *Chemometrics and Intelligent Laboratory Systems*, (1994) in press.

[21] Y. Meyer, *Wavelets. Algorithms and Applications*, SIAM Book, 1993.

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

[23] J.R. Magnus and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics*, Wiley, Essex, 1988.

[24] R. Henrion, N-way principal component analysis — theory, algorithms and applications, *Chemometrics and Intelligent Laboratory Systems*, (1993) submitted.

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

[26] J.R. Wilson, *Introduction to Graph Theory*, Longman Scientific and Technical, Essex, 3rd edn., 1985.