A FAST NON-NEGATIVITY-CONSTRAINED LEAST SQUARES ALGORITHM

RASMUS BRO[1]* AND SIJMEN DE JONG[2]

[1] *Chemometrics Group, Food Technology, Department of Dairy and Food Science, Royal Veterinary and Agricultural University, Rolighedsvej 30, iii, DK-1958 Frederiksberg C, Denmark*

[2] *Unilever Research Laboratorium, PO Box 114, NL-3130 AC Vlaardingen, Netherlands*

SUMMARY

In this paper a modification of the standard algorithm for non-negativity-constrained linear least squares regression is proposed. The algorithm is specifically designed for use in multiway decomposition methods such as PARAFAC and *N*-mode principal component analysis. In those methods the typical situation is that there is a high ratio between the numbers of objects and variables in the regression problems solved. Furthermore, very similar regression problems are solved many times during the iterative procedures used. The algorithm proposed is based on the *de facto* standard algorithm NNLS by Lawson and Hanson, but modified to take advantage of the special characteristics of iterative algorithms involving repeated use of non-negativity constraints. The principle behind the NNLS algorithm is described in detail and a comparison is made between this standard algorithm and the new algorithm called FNNLS (fast NNLS). © 1997 John Wiley & Sons, Ltd.

Journal of Chemometrics, Vol. **11,** 393–401 (1997) (No. of Figures: 0 No. of Tables: 2 No. of References: 22)

KEY WORDS non-negativity; multiway; PARAFAC; NNLS

INTRODUCTION

Estimation of models subject to non-negativity constraints is of practical importance in chemistry. A case in point is the estimation of concentrations from spectral data.[1–4] The time required for estimating true least squares non-negativity-constrained models is typically many times longer than for estimating unconstrained models. In certain applications, e.g. multiway analysis of spectral data, such constrained models are estimated repeatedly during the iterative process. Computation time then becomes formidable and sometimes resort is taken to approximate procedures.

Some authors have used a very intuitive approach for imposing non-negativity using unconstrained estimation and then subsequently setting negative values to zero. This approach, however, cannot be recommended for several reasons. First of all, one does not obtain least squares estimates, which means that there is no guarantee whatsoever for the quality of the model. This can be demonstrated very easily by an example. Consider a matrix **Z** of independent variables and a matrix **x** of dependent variables. Let

$$\mathbf{Z}=\begin{bmatrix}\mathbf{73} & \mathbf{71} & \mathbf{52}\\ \mathbf{87} & \mathbf{74} & \mathbf{46}\\ \mathbf{72} & \mathbf{2} & \mathbf{7}\\ \mathbf{80} & \mathbf{89} & \mathbf{71}\end{bmatrix}, \qquad \mathbf{x}=\begin{bmatrix}49\\ 67\\ 68\\ 20\end{bmatrix}$$

* Correspondence to: R. Bro.

Contract grant sponsor: Nordic Industry Foundation
Contract grant number: P93149

CCC 0886–9383/97/050393–09 $17.50 *Received October 1996*
Accepted February 1997

The regression vector of the least sqares problem

$$\min \|\mathbf{x} - \mathbf{Zd}\|^2$$

can be found as $\mathbf{d}=(\mathbf{Z}^\mathrm{T}\mathbf{Z})^{-1}\mathbf{Z}^\mathrm{T}\mathbf{x}=[1{\cdot}123 \;\; 0{\cdot}917 \;\; -2{\cdot}068]^\mathrm{T}$. Setting the negative element to zero, the estimated solution under non-negativity constraint is $\mathbf{d}=[1{\cdot}123 \;\; 0{\cdot}917 \;\; 0]^\mathrm{T}$. The root mean square (RMS) error corresponding to this regression vector equals 103. If the non-negativity-constrained regression vector is properly estimated, the result is $\mathbf{d}=[0{\cdot}650 \;\; 0 \;\; 0]^\mathrm{T}$, yielding an RMS error of 20. Thus the approximate regression vector is not even close to an optimal solution. Another serious problem with this approximate approach is that when included in a multiway algorithm, it can cause the algorithm to diverge, i.e. successive iterations yield models that describe the data progressively more poorly. This can happen because the approximate estimates are not truly least squares. Iterative multiway algorithms are most often based on alternating least squares (ALS). This means that every substep in the iterations is a least squares estimation of a part of the parameters temporarily fixing the remaining parameters. Because the estimates are conditional least squares, the algorithm will always improve (or not change) the fit in each iteration and therefore converge. When incorporating new substeps in an ALS algorithm, it is therefore very important to make sure that these substeps also give conditional least squares estimates to retain the convergence properties of the ALS algorithm. Especially when modelling data that are very noisy or otherwise difficult to model, the problem of divergence can be troublesome.

Clearly there is a need for faster algorithms for non-negativity-constrained least squares regression. In this paper we propose a modification of the current standard algorithm that cuts execution time considerably. We will use the three-way PARAFAC model and algorithm as an example, but many other multiway models can benefit from the proposed algorithm, e.g. INDSCAL[5] or *N*-mode principal component analysis.[6, 7] The original NNLS algorithm[8] will be described first and then the PARAFAC algorithm will be discussed briefly. Next we show how to accommodate the NNLS algorithm to the PARAFAC algorithm. Finally some results on different data sets are shown.

NOTATION

Scalars, including elements of vectors and matrices, are indicated by lowercase italics, vectors by bold lowercase characters, bold capitals are used for two-way matrices and underlined bold capitals for three-way arrays. The letters I, J and K are reserved for indicating the dimension of the first, second and third mode of a three-way array and i, j and k are used as indices for each of these modes. An $I\times J\times K$ array $\underline{\mathbf{X}}$ is also equivalently named z_{ijk}, implicitly assuming that the indices run from one to the respective dimensionalities. A subarray of a three-way array $\underline{\mathbf{X}}$ is called $\mathbf{X}_k$ if it is the kth $I\times J$ layer in the third mode. An $I\times J$ matrix $\mathbf{X}$ is sometimes referred to by its column vectors as $[\mathbf{x}_1 \;\; \mathbf{x}_2 \;\; \ldots \;\; \mathbf{x}_J]$.

NON-NEGATIVITY-CONSTRAINED LINEAR LEAST SQUARES

The NNLS problem will be stated using the following nomenclature. Given $\mathbf{Z}$ $(L\times M)$ and $\mathbf{x}$ $(L\times 1)$, a vector $\mathbf{d}$ $(M\times 1)$ is sought to minimize the expression

$$\|\mathbf{x} - \mathbf{Zd}\|^2$$

subject to

$$d_m \geq 0 \quad \text{for all } m$$

where d_m is the mth element of $\mathbf{d}$. The solution can be found by the algorithm NNLS.[8] This is an *active* *set algorithm*. There are M inequality constraints in the above-stated problem. The mth constraint is said to be active, if the mth regression coefficient will be negative (or zero) if estimated unconstrained, otherwise the constraint is passive. An active set algorithm is based on the fact that if the true active set is known, the solution to the least squares problem will simply be the unconstrained least squares solution to the problem using only the variables corresponding to the passive set, setting the regression coefficients of the active set to zero. This can also be stated as: if the active set is known, the solution to the NNLS problem is obtained by treating the active constraints as equality constraints rather than inequality constraints. To find this solution, an alternating least squares algorithm is applied. An initial feasible set of regression coefficients is found. A feasible vector is a vector ($M \times 1$) with no elements violating the constraints. In this case the vector containing only zeros is a feasible starting vector as it contains no negative values. In each step of the algorithm, variables are identified and removed from the active set in such a way that the fit strictly decreases. After a finite number of iterations the true active set is found and the solution is found by simple linear regression on the unconstrained subset of the variables. More general descriptions of active set algorithms can be found in Reference 9.

Algorithmn NNLS

```
A. Initialization
  A1. $\mathrm{P} = \varnothing$
  A2. $\mathrm{R} = \{1, 2, \ldots, M\}$
  A3. $\mathbf{d} = \mathbf{0}$
  A4. $\mathbf{w} = \mathbf{Z}^{\mathrm{T}}(\mathbf{x} - \mathbf{Zd})$

B. Main loop
  B1. Proceed if $\mathrm{R} \neq \varnothing \wedge [\max_{n \in \mathrm{R}}(w_n) > \text{tolerance}]$
  B2. $m = \operatorname{argmax}_{n \in \mathrm{R}}(w_n)$
  B3. Include the index $m$ in P and remove it from R
  B4. $\mathbf{s}^{\mathrm{P}} = [(\mathbf{Z}^{\mathrm{P}})^{\mathrm{T}}\mathbf{Z}^{\mathrm{P}}]^{-1}(\mathbf{Z}^{\mathrm{P}})^{\mathrm{T}}\mathbf{x}$

C. Inner loop
  C1. Proceed if $\min(\mathbf{s}^{\mathrm{P}}) \leq 0$
  C2. $\alpha = -\min_{n \in \mathrm{P}}[d_n/(d_n - s_n)]$
  C3. $\mathbf{d} := \mathbf{d} + \alpha(\mathbf{s} - \mathbf{d})$
  C4. Update R and P
  C5. $\mathbf{s}^{\mathrm{P}} = [(\mathbf{Z}^{\mathrm{P}})^{\mathrm{T}}\mathbf{Z}^{\mathrm{P}}]^{-1}(\mathbf{Z}^{\mathrm{P}})^{\mathrm{T}}\mathbf{x}$
  C6. $\mathbf{s}^{\mathrm{R}} = \mathbf{0}$
  end C

  B5. $\mathbf{d} = \mathbf{s}$
  B6. $\mathbf{w} = \mathbf{Z}^{\mathrm{T}}(\mathbf{x} - \mathbf{Zd})$
  end B
```

A. Initialization

The set P comprises all indices of the M variables which are currently not fixed: the passive set. Likewise, R holds indices of those coefficients that are currently fixed at the value zero: the active set.

To ensure that the initial solution vector **d** is feasible, it is set equal to the $M \times 1$ zero vector (step A3) and all constraints are active (steps A1 and A2). The vector **w** defined in step A4 can be interpreted as a set of Lagrange multipliers. We will show that when the optimal solution has been found, the following holds:

$$w_m = 0, \quad m \in \mathrm{P}$$

$$w_m < 0, \quad m \in \mathrm{R}$$

To realize this, observe the following. The function to be minimized is

$$f(\mathbf{d}) = \|\mathbf{x} - \mathbf{Zd}\|^2$$

subject to the non-negativity constraints on the regression coefficients. The derivative of $f(\mathbf{d})$ is

$$f'(\mathbf{d}) = 2\mathbf{Z}^{\mathrm{T}}(\mathbf{Zd} - \mathbf{x})$$

One necessary condition for optimality of a solution is that the derivatives with respect to the parameters of the passive set be zero. Note that $w_m = -\frac{1}{2}f'(d_m)$, hence the mth element of **w** is zero for all $m \in \mathrm{P}$. For the elements d_m constrained to zero at the optimal solution, the sum of square errors increase when d_m takes positive values. Hence the derivative $f'(d_m)$ is positive and w_m must be negative. Rigorous proof of this is given in Reference 8.

B. Main loop

If the set R is empty, all constraints are passive and all coefficients must be positive and thus the problem is solved. If R is not empty but all w_m, $m \in \mathrm{R}$, are negative, no fixed variable can be set free as the regression coefficient of that variable would then have to be negative to lower the error sum of squares. If any w_m, $m \in \mathrm{R}$, is positive, transferring the corresponding varible to the set of free variables will yield a new positive regression coefficient. The variable with the highest w_m is included in the set P (step B3) and the intermediate regression vector **s** is calculated on this new set (step B4). The superscript P indicates the passive set. The matrix $\mathbf{Z}^{\mathrm{P}}$ is a matrix containing only the variables currently in the passive set P. In practice the elements of **w** are not tested to be positive but to be above a certain low tolerance to ensure that numerical inaccuracies do not cause a non-feasible variable to enter the passive set. If all regression coefficients in **s** are non-negative the regression vector **d** is set equal to **s** (step B5) and a new set of Lagrange multipliers **w** is calculated (step B6). The main loop is continued until no more active constraints can be set free. It is theoretically possible to drop more than one active constraint at a time in step B3. However, the general experience is that it is not advantageous to do so for several reasons; for example, too many constraints may be dropped in each step, giving extra workload in activating these currently passive constraints (see e.g. Reference 10).

C. Inner loop

When a new variable has been included in the passive set P, there is a chance that in the unconstrained solution to the new least squares problem (step B4) some of the regression coefficients of the free variables will turn negative (step C1). Calling the new estimates **s** and the former **d**, it is possible to adjust the new estimate to satisfy the constraints. The old estimate **d** is feasible but with a worse fit (higher loss) than the new estimate **s** which, however, is not feasible. Somewhere along the line segment $\mathbf{d} + \alpha(\mathbf{s} - \mathbf{d})$, $0 \le \alpha \le 1$, there is an interval for which the inequalities are not violated. As the fit is strictly decreasing as $\alpha \rightarrow 1$, it is possible to find that particular value of α which minimizes the fit and yet retains as many variables in the passive set as possible (step C2). For the optimal α, one or several variables will become zero and hence they are removed from the set P (step C4). however, the fit of the model will be improved compared with the prior passive set of variables. After adjusting the active and passive sets, the unconstrained solution using the current passive set is calculated (step C5). The regression coefficients of variables removed from the set P are set to zero (step C6). The inner loop is repeated until all violating variables have been moved to the set R. In practice the NNLS algorithm seldom enters the inner loop C.

This concludes the description of the general NNLS algorithm. It can be proved that the algorithm will consist of only a finite number of iterations as both the inner and outer loops converge after finite iterations. Further comments on and variations of this algorithm can be found in e.g. References 11–14.

MODIFIED NNLS: FAST NNLS

In most cases where the NNLS algorithm is used, the number of observations is much higher than the number of variables. it is customary to use a QR decomposition to reduce the problem. In the context of e.g. PARAFAC another type of reduction seems more appropriate. The PARAFAC model has been described in detail in several papers and the reader is referred to those for details.[3, 15–19] For an $I \times J \times K$ array $\underline{\mathbf{X}}$ the three-way PARAFAC model is defined as

$$x_{ijk} = \sum_{f=1}^{F} a_{if} b_{jf} c_{kf} + e_{ijk}$$

where x_{ijk} is the element of $\underline{\mathbf{X}}$ in the ith row, jth column and kth tube. The element of the ith row and fth column of the loading matrix $\mathbf{A}$ is called a_{if}. The loading matrices of the second and third modes are defined likewise from b_{jf} and c_{kf}, and e_{ijk} is the residual part of the ijkth element of $\underline{\mathbf{X}}$. The number of components is equal to F. The loading matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are estimated in a least squares sense. The estimation method is most often based on iteratively solving conditional least squares problems of the type.

$$\mathbf{X} = \mathbf{Z}\mathbf{A}^{\mathrm{T}} + \mathbf{E}$$

The matrix $\mathbf{Z}$ is of size $JK \times F$ and its fth column is the Kronecker product of the fth columns of $\mathbf{B}$ and $\mathbf{C}$. The matrix $\mathbf{X}$ is a $JK \times I$ matrix obtained by unfolding the $I \times J \times K$ array of $\underline{\mathbf{X}}$, and $\mathbf{E}$ is the residual matrix of the same size. The matrix $\mathbf{A}$ is the $I \times F$ loading matrix to be estimated. Similar problems are solved when $\mathbf{B}$ and $\mathbf{C}$ are estimated. Computationally simple expressions for estimating the loading matrices can be derived by using the structure of $\mathbf{Z}$ explicitly. For the first-mode loadings the following holds:

$$\begin{gathered}\mathbf{Z}^{\mathrm{T}}\mathbf{Z} = (\mathbf{C}^{\mathrm{T}}\mathbf{C}) * (\mathbf{B}^{\mathrm{T}}\mathbf{B}) \\ \mathbf{X}^{\mathrm{T}}\mathbf{Z} = \mathbf{X}_1\mathbf{B}\mathbf{D}_1 + \mathbf{X}_2\mathbf{B}\mathbf{D}_2 + \cdots + \mathbf{X}_K\mathbf{B}\mathbf{D}_K \\ \mathbf{A} = \mathbf{X}^{\mathrm{T}}\mathbf{Z}(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\end{gathered}$$

Similar expressions can be derived for estimating $\mathbf{B}$ and $\mathbf{C}$. The matrices $\mathbf{X}_k$ in the above steps correspond to the $I \times J$ submatrices obtained by fixing the third mode of the array on the kth level. The operator $*$ signifies the Hadamard product.[20] The Hadamard product of two matrices $\mathbf{M}$ and $\mathbf{N}$ of equal size is defined through

$$h_{ij} = m_{ij} n_{ij}$$

where h_{ij} is the ijth element of the matrix $\mathbf{H} = \mathbf{M} * \mathbf{N}$. The matrix $\mathbf{D}_k$ is the diagonal matrix containing the kth row of $\mathbf{C}$ in its diagonal.

When the non-negativity-constrained least squares estimate of e.g. **A** is sought, the above simplified expressions for $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ and $\mathbf{X}^{\mathrm{T}}\mathbf{Z}$ are not applicable. For NNLS to work, the problem must be expressed in terms of individual rows of **A**. To estimate $\mathbf{a}^i$, the *i*th row of **A** transposed, the following model holds:

$$\mathbf{x}_i = \mathbf{Z}\mathbf{a}^i + \mathbf{e}_i$$

where the superscript indicates the *i*th row transposed and the subscript denotes the *i*th column. The explicit calculation of **Z** is given by

$$\mathbf{Z} = [\mathbf{b}_1 \otimes \ \mathbf{c}_1 \ \ \mathbf{b}_2 \otimes \ \mathbf{c}_2 \ \ldots \ \mathbf{b}_F \otimes \ \mathbf{c}_F]$$

with $\otimes$ indicating the Kronecker product. The *i*th row of **A** can thus be estimated by replacing the dependent variable **x** in the NNLS algorithm with the *i*th column of **X**. However, resorting to the explicit calculation of **Z** instead of the cross-product terms in itself will slow down the PARAFAC algorithm. If non-negativity-constrained least squares estimation is to be used in the faster PARAFAC algorithm, the problem is that the matrix **Z** is not available for the regression problem to be solved. Only $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ and $\mathbf{Z}^{\mathrm{T}}\mathbf{x}$ are available. This, however, turns out to be an advantage, as these cross-product matrices are much smaller in size than the original data and the NNLS algorithm can be modified to handle the cross-product matrices instead of the raw data.

To modify the NNLS algorithm so that it accepts the cross-products as input instead of the raw data, the following changes in algorithm NNLS are required: steps A4 and B6 are changed to

$$\mathbf{w} = (\mathbf{Z}^{\mathrm{T}}\mathbf{x}) - (\mathbf{Z}^{\mathrm{T}}\mathbf{Z})\mathbf{d}$$

and steps B4 and C5 are changed to

$$\mathbf{s}^{\mathrm{P}} = [(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{\mathrm{P}}]^{-1}(\mathbf{Z}^{\mathrm{T}}\mathbf{x})^{\mathrm{P}}$$

where $(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{\mathrm{P}}$ is the submatrix of $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ containing only the rows and columns corresponding to the set P. Likewise, the column vector $(\mathbf{Z}^{\mathrm{T}}\mathbf{x})^{\mathrm{P}}$ contains the elements of $\mathbf{Z}^{\mathrm{T}}\mathbf{x}$ corresponding to the set P. It is easily verified that the vector $\mathbf{s}^{\mathrm{P}}$ defined in this way equals $\mathbf{s}^{\mathrm{P}}$ as defined in algorithm NNLS. When **Z** has many more rows than columns, using the cross-products instead of the raw data improves the speed.

To further decrease the time spent in the FNNLS algorithm, it is worth noting that during the estimation of loadings in the PARAFAC algorithm, typically only small changes occur from iteration to iteration. Elements that are forced to zero will usually stay zero in subsequent iterations. Instead of starting each FNNLS estimation with all variables forced to zero (step A3), it is therefore sensible to define R and P according to their optimal values in the previous PARAFAC iteration of that particular set of loadings. To do this, the FNNLS algorithm must be changed so that the appropriateness of these predefined sets of active and passive variables is checked initially before entering the main loop. This is done by calculating the regression vector corresponding to the current passive set P and then entering a loop similar to the inner loop in NNLS before entering the main loop. If the sets R and P initially given are correct, this algorithm will converge very fast, because after estimation of the regression vector only a check for negative regression coefficients and a check for positive *w*-values has to be performed.

For completeness, two possible modifications of FNNLS will be discussed here. It is possible to do weighted non-negativity-constrained linear least squares regression by simply using $\mathbf{Z}^{\mathrm{T}}\mathbf{V}\mathbf{Z}$ and $\mathbf{Z}^{\mathrm{T}}\mathbf{V}\mathbf{x}$ instead of $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ and $\mathbf{Z}^{\mathrm{T}}\mathbf{x}$, the diagonal matrix **V** containing the weights. It is also possible to only require non-negativity of individual regression coefficients by simply keeping unconstrained regression coefficients passive for good and ignoring their value when testing for negative regression coefficients (step C1). Requiring non-negativity only for a subset of the variables can be useful in some types of modelling if only some of the variables (e.g. spectral) are known to be intrinsically positive, while for others (e.g. temperature in °C) no such *a priori* theory is given.

RESULTS ON REAL AND SIMULATED DATA

To test the FNNLS algorithm, the following five combinations of PARAFAC and NNLS algorithms were constructed: PFa–NNLS, PFa–FNNLSa, PFb–FNNLSa, PFb–FNNLSb and PFb. The abbreviation PFa stands for PARAFAC implemented by estimating the full **Z**, PFb for an implentation where only $\mathbf{Z}^T\mathbf{Z}$ are $\mathbf{X}^T\mathbf{Z}$ are estimated, NNLS is the ordinary NNLS as implemented in the MATLAB software (see ‘Materials and methods’), FNNLSa is the new NNLS algorithm using the default zero vector as initial estimate in the algorithm, and FNNLSb is the new algorithm using the formerly found passive variables for initialization. Algorithm PFa–NNLS is the ordinary non-negativity-constrained PARAFAC algorithm to test the new algorithm against. To be able to distinguish between the influence of the PARAFAC and the NNLS algorithm, the FNNLSa algorithm is used together with both the PFa and the PFb algorithm to show the influence of using the cross-product terms instead of the original data in the PARAFAC algorithm. Finally the PFb algorithm is used together with the FNNLSb algorithm to show the effect of the full version of the new NNLS algorithm. These four algorithms are numerically identical and will converge after the same number of iterations if started by the same initial guesses on the loading matrices. An unconstrained PARAFAC model (PFb) is estimated for the same number of iterations as the non-negativity-constrained models for comparison.

Three different experiments were performed. First a simulation study was performed using randomly generated arrays of equal dimensions in each mode of either eight or 20 and estimating either a three- or a five-component model. For each setting the following procedure was performed 30 times. An array $\underline{\mathbf{X}}$ of proper size was generated by using uniformly distributed random numbers. The initial matrices **A**, **B** and **C** were generated randomly and all five algorithms were run with identical initial estimates. For the non-negativity-constrained PARAFAC algorithms the convergence criterion was a relative change in fit of less than 10^{-6}. The unconstrained PARAFAC model was subsequently estimated using the same number of iterations as the non-negativity-constrained models. For each model estimation the time spent was saved and averaged over the 30 repetitions. The results are shown in Table 1. As can be seen, the gain of using the FNNLS algorithm in conjunction with the PFb implementation is approximately a fivefold to a 20-fold improvement in time. The primary gain is obtained by using the cross-product matrices instead of the raw data in the non-negativity-constrained least squares algorithm (columns PFa–NNLS versus PFa–FNNLSa). However, changing to algorithm PFb and FNNLSb also provides significant improvements and actually makes the algorithm almost as fast as the corresponding unconstrained algorithm (columns PFb–FNNLSb versus PFb). For some

Table 1. Time consumption of non-negativity-constrained PARAFAC algorithms averaged over 30 runs. Data matrices generated randomly with dimensions Dim × Dim × Dim and estimating an *F*-component model. For PFa–NNLS the (average) absolute time for estimating the models is given in seconds. The remaining columns show the time consumption for all algorithms relative to PFa–NNLS (standard deviation of relative time consumption in all cases less than 0·8)

Dim	*F*	PFa–NNLS (s)	PFa–NNLS (%)	PFa–NNLSa (%)	PFb–NNLSa (%)	PFb–NNLSb (%)	PFb (%)
8	3	20	100	49	36	20	16
8	5	47	100	41	29	12	12
20	3	230	100	25	18	10	8
20	5	675	100	15	10	4	5

settings the non-negativity-constrained algorithm is actually faster than the unconstrained algorithm. This can be explained by the fact that if many variables are forced to zero, smaller regression problems are to be solved in the non-negativity-constrained than in the unconstrained algorithm.

Two data sets from fluorescence spectroscopy were used for testing the algorithm on real data. The first, called AMINO, is a small data set of five samples with different amounts of tryptophan, phenylalanine and tyrosine. Each sample has been measured spectrofluorometrically at excitation 250–300 nm and emission 250–450 nm in 1 nm intervals. The data were kindly provided by Claus A. Andersson[21] and have also been described in Reference 3. The data array is of dimensions 5 (sample) $\times$ 51 (excitation) $\times$ 201 (emission).

The second data set is a set of 32 sugar samples, here called SUGAR. The sugar was dissolved in water and measured spectrofluorometrically according to Reference 22. The data array here is of dimension $32\times4\times371$, corresponding to 32 samples each excited at four wavelengths (239, 240, 290 and 340 nm) and emission measured from 375 to 560 nm in 0·5 nm intervals. Both data sets can be considered as typical examples of spectroscopic arrays suitable for multiway analysis.

A three-component PARAFAC model was chosen for both AMINO and SUGAR. For the AMINO data this is undoubtedly correct, while for the SUGAR data more components might be appropriate. However, this is not essential in the present context, where only the time consumption is of interest. The results from the two real data sets are shown in Table 2. The results have been averaged over four runs in each case. For both data sets the conclusions remain the same as for the simulated data. The FNNLS algorithm is extremely fast compared with the standard NNLS algorithm. Note that for the data set SUGAR a further improvement in speed could be obtained by using the modified PARAFAC algorithm suggested by Kiers and Krijnen.[17]

For all the simulations mentioned above, the number of flops (floating point operations) used was also determined using the FLOPS command of MATLAB. It can be argued that the flops usage is a more objective indicator of the workload of an algorithm. In this case the flops used by all algorithms except PFa–NNLS were almost identical and much lower than the flops used by PFa–NNLS (typically only 1%–5% of the PFa–NNLS algorithm). However, as MATLAB is a widely used programming software in chemometrics, it seems more relevant to judge the actual time consumption instead of a theoretical measure that does not tell us anything about the time consumption of an actual implementation of an algorithm. In terms of flops the PFb–NNLSb algorithm was the least demanding algorithm in general, even compared with the PFb unconstrained algorithm.

MATERIALS AND METHODS

The algorithms have been implemented in MATLAB for Windows v. 4.2c.1 (MathWorks, Inc.) and can be obtained from the Internet at http://newton.foodsci.kvl.dk/foodtech.html. All calculations were performed on a 200 MHz Pentium Dell PC with 64 MB RAM. The data used are also available from the first author.

Table 2. Time consumption of non-negativity-constrained PARAFAC algorithms averaged over four runs. Data matrices SUGAR and AMINO are described in the next. The number of components used in each case is three. Standard deviation of relative time consumption in all cases less than 0·5. For further explanation see Table 1

Data set	PFa–NNLS (s)	PFa–NNLS (%)	PFa–NNLSa (%)	Pfb–NNLSa (%)	PFb–NNLSb (%)	PFb (%)
AMINO	466	100	19	14	9	7
SUGAR	4358	100	23	18	10	2

ACKNOWLEDGEMENTS

Professor Lars Munck, Food Technology, Department of Dairy and Food Science, Royal Veterinary and Agricultural University, Denmark, is thanked for financial support through the Nordic Industry Foundation project P93149 and the FØTEK foundation.

REFERENCES

1. S. R. Durell, C. Lee, R. T. Ross and E. L. Gross, *Arch. Biochem. Biophys.* **278**, 148–160 (1990).
2. P. Paatero, *Chemometrics Intell. Lab. Syst.* in press.
3. R. Bro, *Chemometrics Intell. Lab. Syst.* in press.
4. J. M. F. ten Berge, H. A. L. Kiers and W. P. Krijnen, *J. Classif.* **10**, 115–124 (1994).
5. J. B. Kruskal, *Proc. Symp. Appl. Math.* **28**, 75–104 (1983).
6. P. M. Kroonenberg, *Three-Mode Principal Component Analysis. Theory and Applications.* DSWO Press, Leiden (1983).
7. R. Henrion, *Chemometrics Intell. Lab. Syst.* **25**, 1–23 (1993).
8. C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ (1974); also available as *Classics in Applied Mathemaatics*, Vol. 15, SIAM, Philadelphia, PA (1995).
9. P. E. Gill, W. Murray and M. H. Wright, *Practical Optimization*, Academic, London (1981).
10. A. Dax, *ACM Trans. Math. Softw.* **17**, 64–73 (1991).
11. J. Stoer, *SIAM J. Numer. Anal.* **8**, 382–411 (1971).
12. K. Schittkowski and J. Stoer, *Numer. Math.* **31**, 431–463 (1979).
13. K. H. Haskell and R. J. Hanson, *Math. Prog.* **21**, 98–118 (1981).
14. R. J. Hanson, *SIAM J. Sci. Stat. Comput.* **7**, 826–834 (1986).
15. R. A. Harshman, *UCLA Working Papers Phonet.* **16**, 1–84 (1970).
16. R. A. Harshman, *UCLA Working Papers Phonet.* **22**, 111–117 (1972).
17. H. A. L. Kiers and W. P. Krijnen, *Psychometrika*, **56**, 147–152 (1991).
18. S. Leurgans, R. T. Ross and R. B. Abel, *SIAM J. Matrix Anal. Appl.* **14**, 1064–1083 (1993).
19. R. A. Harshman and M. E. Lundy, *Comput. Stat. Data Anal.* **18**, 39–72 (1994).
20. G. P. H. Styan, *Linear Algebra Appl.* **6**, 217–240 (1973).
21. C. A. Andersson, *Internal Rep.*, Food Technology, Department of Dairy and Food Science, Royal Veterinary and Agricultural University, Frederiksberg (1996).
22. L. Nørgaard, *Zuckerind.* **120**, 970–981 (1995).