JOURNAL OF CHEMOMETRICS
J. Chemometrics 2004; **18**: 441–450
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cem.889

Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems

Mark H. Van Benthem and Michael R. Keenan*
Sandia National Laboratories, Albuquerque, NM 87185-0886, USA

Received 1 July 2004; Revised 15 January 2005; Accepted 20 January 2005

Algorithms for multivariate image analysis and other large-scale applications of multivariate curve resolution (MCR) typically employ constrained alternating least squares (ALS) procedures in their solution. The solution to a least squares problem under general linear equality and inequality constraints can be reduced to the solution of a non-negativity-constrained least squares (NNLS) problem. Thus the efficiency of the solution to any constrained least square problem rests heavily on the underlying NNLS algorithm. We present a new NNLS solution algorithm that is appropriate to large-scale MCR and other ALS applications. Our new algorithm rearranges the calculations in the standard active set NNLS method on the basis of combinatorial reasoning. This rearrangement serves to reduce substantially the computational burden required for NNLS problems having large numbers of observation vectors. Copyright © 2005 John Wiley & Sons, Ltd.

KEYWORDS: NNLS; non-negativity; MCR; ALS

1. INTRODUCTION

The method of non-negativity-constrained least squares (NNLS) is an important tool in the chemometrician's toolbox. Often it is used in alternating least squares (ALS) algorithms when the variables to be estimated are known *a priori* to be non-negative. Emission spectra and chemical concentrations represent two examples of such variables that are constrained by physical principles to be non-negative [1–3]. It is common in such applications to solve a single specific model for large numbers of observation vectors (e.g., spectra). For instance, a single set of pure component spectra may be used to estimate the respective concentrations at each pixel in a spectral image. In the current context, each vector of observations (e.g., a spectrum) will be denoted a right-hand-side (RHS) vector, and the efficient solution of the multiple-RHS NNLS problem is the focus of the present work. We will demonstrate that improved computational performance can be achieved by exploiting the structure that exists in such large-scale problems to minimize the number of arithmetic operations required to accomplish the solution.

Given its fundamental role in solving a variety of linearly constrained optimization problems, much attention has been paid to methods for estimating least squares models subject to non-negativity constraints. A simple, approximate way to implement these constraints is to solve the corresponding unconstrained least squares problem and then overwrite any negative values with zeros. Unfortunately, this technique does not employ an appropriate least squares criterion, so one is left with a solution that has ill-defined mathematical properties. ALS algorithms based on the overwriting method, for instance, do not necessarily lower the cost function on successive iterations and therefore are not guaranteed to converge to a least squares minimum. Methods do exist, however, that solve the NNLS problems in a mathematically rigorous fashion [4,5]. These methods impose non-negativity on the solution while minimizing the sum of squared residuals between the data being fitted and their estimates in a true least squares sense.

While rigorous methods ensure true least squares solutions that satisfy the non-negativity constraints, they can be computationally slow and demanding. The original NNLS algorithm presented by Lawson and Hanson [4] in their landmark text was designed to solve the NNLS problem for the case of a single vector of observations. When applied in a straightforward manner to large-scale, multiple-RHS problems, however, performance is found to be unacceptably slow owing to the need to perform the equivalent of a full pseudoinverse calculation for each observation vector. Recently, Bro and de Jong [5] made a substantial speed improvement to Lawson and Hanson's algorithm for the multiple-RHS case. Bro and de Jong's fundamental realization was that large parts of the pseudoinverse could be computed once but used repeatedly. Specifically, their algorithm precomputes the cross-product matrices that appear in the normal equation formulation of the least squares solution. Bro and de Jong also observed that, during ALS

*Correspondence to: M. R. Keenan, Sandia National Laboratories, MS0886, Albuquerque, NM 87185-0886, USA.
E-mail: mrkeena@sandia.gov
Contract/grant sponsors: Sandia National Laboratories; United States Department of Energy; contract/grant number: DE-ACO4-94AL85000.

procedures, solutions tend to change only slightly from iteration to iteration. In an extension to their NNLS algorithm that they characterized as being for 'advanced users', Bro and de Jong retained information about the previous iteration's solution and were able to extract further performance improvements in ALS applications that employ NNLS. These innovations led to a substantial performance increase when analyzing large multivariate, multiway data sets.

We have devised a new algorithm that further improves the performance of NNLS for multivariate data. This new method rigorously solves the constrained least squares problem while exacting essentially no performance penalty as compared with the overwriting method. While Bro and de Jong's algorithm precomputes parts of the pseudoinverse, the algorithm still requires work to complete the pseudoinverse calculation once for each RHS. We also precompute the cross-product matrices. It is often the case in large-scale problems, however, that the number of *unique* pseudoinverses that are required is much smaller than the number of RHS vectors. Thus our new algorithm employs combinatorial reasoning to identify and group together all RHS vectors that share a common pseudoinverse at each stage in the NNLS iteration. The complete pseudoinverse is then computed just once per group and, subsequently, is applied individually to each observation in the group. As a result, the computational burden is significantly reduced and the time required to perform ALS operations is likewise reduced.

The notation used in this paper is typical of this type of work: scalars are lowercase italic, column vectors are lowercase bold, matrices are uppercase bold and transposes of vectors (i.e., row vectors) and matrices are indicated by a superscript T. We depict columns of matrices using lowercase bold with a dot–index subscript, e.g., column one of matrix $\mathbf{A}$ is the column vector $\mathbf{a}_{\cdot 1}$. We depict rows of matrices using lowercase bold with an index–dot subscript, e.g., row one of matrix $\mathbf{A}$ is the row vector $\mathbf{a}_{1\cdot}$ (note that it is not shown as a transpose). Prepended superscripts represent the ordinal in a series of iterations. We also use notation related to absorption spectroscopy so as to be consistent with our previous work on the subject of MCR and constrained ALS [6]. Thus the letter 'A' represents the measured absorbances, the letter 'C' represents the pure component concentrations and the letter 'K' represents the pure component path-normalized absorption coefficients. We represent sets of indices by script type, e.g., $\mathcal{P}$, and sets containing sets of indices by bold script type, e.g., $\boldsymbol{\mathcal{P}}$. Finally, we represent submatrices as pre- and/or post-subscripted bold capital letters. Thus ${}_{\mathcal{P}}\mathbf{K}_{\mathcal{E}}$ is a submatrix of $\mathbf{K}$ composed of those rows and columns of $\mathbf{K}$ whose indices are contained in sets $\mathcal{P}$ and $\mathcal{E}$, respectively.

2. LEAST SQUARES PROBLEMS WITH MULTIPLE RIGHT-HAND SIDES

2.1. Solving the unconstrained least squares problem

In the present context the least squares problem with a single right-hand side (RHS) is framed as

$$\mathbf{Ck} \cong \mathbf{a} \tag{1}$$

where $\mathbf{a}$ is a vector of absorbances for n samples measured at one frequency, $\mathbf{C}$ is a matrix of concentrations for n samples and l chemical components and $\mathbf{k}$ is an l-vector of absorption coefficients for which we want to solve. The least squares method seeks to find a solution vector $\mathbf{k}$ that satisfies

$$\min_{\mathbf{k}} \|\mathbf{Ck} - \mathbf{a}\|_2^2 \tag{2}$$

which is the squared 2-norm of the residuals. Formally, the unconstrained least squares solution to Equation (1) can be written as

$$\hat{\mathbf{k}} = \mathbf{C}^{+}\mathbf{a} \tag{3}$$

where $\mathbf{C}^{+}$ is the pseudoinverse of $\mathbf{C}$. In practice the pseudoinverse is rarely computed explicitly and applied directly as suggested by Equation (3); several approaches to solving Equation (2) can be taken that are more numerically efficient [7]. However, the pseudoinverse does provide a convenient conceptual framework for explaining the least squares method and discussing performance improvements, and we will use it in that capacity for the remainder of this discussion.

2.2. Solving unconstrained least squares problems with multiple RHS

We now consider a problem where the number of columns of observations is greater than one. Thus the absorbances for n samples measured at p frequencies are now represented as a matrix $\mathbf{A} = [\mathbf{a}_{\cdot 1}\ \mathbf{a}_{\cdot 2}\ \cdots\ \mathbf{a}_{\cdot p}]$. Similarly, $\mathbf{K} = [\mathbf{k}_{\cdot 1}\ \mathbf{k}_{\cdot 2}\ \cdots\ \mathbf{k}_{\cdot p}]$ is an $l \times p$ matrix of absorption coefficients such that one column of $\mathbf{K}$ has the same dimensions as $\mathbf{k}$. This is a minor departure from our previous work [6], which had $\mathbf{K}$ as a $p \times l$ matrix, in order to simplify the explanation of our techniques. The single-RHS least squares problem in Equation (1) now becomes the multiple-RHS problem

$$\mathbf{CK} \cong \mathbf{A} \tag{4}$$

which, analogously to Equation (3), can be solved as

$$\hat{\mathbf{K}} = \mathbf{C}^{+}\mathbf{A} \tag{5}$$

This formulation is appropriate for the unconstrained least squares problem where the pseudoinverse is the same for all columns of $\mathbf{K}$ and $\mathbf{A}$. As we will show, this is not the case for constrained problems. Thus it is convenient to express the multiple-RHS least squares solution equivalently as the solutions to a series of single-RHS problems

$$\hat{\mathbf{k}}_{\mathcal{G}j} = \mathbf{C}^{+}\mathbf{a}_{\mathcal{G}j}, \quad j = 1, 2, \ldots, p \tag{6}$$

which minimize the objective function

$$\begin{aligned} \sum_{j=1}^{p} \|\mathbf{Ck}_{\cdot j} - \mathbf{a}_{\cdot j}\|_2^2 &= \sum_{i=1}^{n}\sum_{j=1}^{p} \|\mathbf{c}_{i\cdot}\mathbf{k}_{\cdot j} - a_{ij}\|_2^2 \\ &\equiv \|\mathbf{CK} - \mathbf{A}\|_F^2 \end{aligned} \tag{7}$$

The last expression is the squared Frobenius norm of the residuals. When solving for multiple RHS, considerable computational effort can be saved by computing the pseudoinverse of $\mathbf{C}$ only once and then applying it to each column of $\mathbf{A}$ in whatever column order one chooses. Of course, this is a common practice for solving unconstrained

least squares problems [7]. As noted earlier, Bro and de Jong's method partially achieves this efficiency by precomputing the cross-product matrices.

3. SOLVING THE NON-NEGATIVE LEAST SQUARES PROBLEM

3.1. Solving NNLS problems with a single RHS

The NNLS problem with a single observation is written identically to the unconstrained problem with the added condition of non-negativity, specifically

$$\begin{aligned} &\min_{\mathbf{k}} \|\mathbf{Ck} - \mathbf{a}\|_2^2 \\ &\text{subject to} \quad \mathbf{k} \geq \mathbf{0} \end{aligned} \tag{8}$$

Commonly employed NNLS algorithms utilize the active/passive set method to provide mathematically rigorous solutions to least squares problems in which all variables are required to be non-negative. The basic idea underlying this method is to convert the inequality-constrained problem represented by Equation (8) into a sequence of equality-constrained problems that can be solved by standard means [6]. Full details of how the algorithm specifies the particular sequence of subproblems are available elsewhere [4,5] and we will not describe them here. It is important, however, to understand the basic active/passive set solution strategy.

By way of preliminaries, a consistent set of inequality constraints defines a volume in multidimensional space that is denoted the feasible region. Any potential solution to the constrained least squares problem must reside in the feasible region, and any vector that satisfies the constraints is called a feasible solution. In two dimensions, for instance, non-negativity constraints compel feasible solutions to lie in the first quadrant of a Cartesian co-ordinate system. Furthermore, for any feasible solution, two classes of variables can be identified, those that reside within the interior of the feasible region and those that fall on a boundary (or co-ordinate axis in the case of non-negativity). The latter variables constitute the so-called active set and represent those variables that can only lower the cost function by moving outside of the feasible region. Such variables must be *actively* forced to assume their boundary values to maintain solution feasibility. The remaining variables, those which reside within the interior of the feasible region, constitute the passive set. The optimal solution is the particular feasible solution that minimizes the sum of the squared residuals in Equation (8). Thus, given any initial feasible solution, the active/passive set strategy for solving the inequality-constrained least squares problem is to iteratively move variables between the active and passive sets, typically one at a time, in such a way that the cost function steadily decreases while the solution remains feasible. After each repartitioning of variables between the active and passive sets, the passive set variables are obtained as the solution to an equality-constrained least squares problem where all active set variables are constrained to equal their boundary values. Since the cost function is strictly decreasing with each iteration and there are a limited number of ways to partition variables between the active and passive sets, convergence of this process is guaranteed. Again, full details of the variable selection criteria and optimality tests are given in the references for the specific case of NNLS [4,5].

3.2. Use of the active/passive set method for solving problem NNLS

Letting $\mathcal{N} = \{1, 2, \ldots, l\}$ be the set that indexes the columns of $\mathbf{C}$, the elements of the active ($\mathcal{A}$) and passive ($\mathcal{P}$) sets are subsets that index the rows of $\mathbf{k}$ as well as the columns of $\mathbf{C}$. The variables in $\mathbf{k}$ that are indexed in $\mathcal{P}$ are those that are solved for in an unconstrained least squares sense. $\mathcal{A}$ is merely the complement of $\mathcal{P}$ (thus $\mathcal{A} \cup \mathcal{P} = \mathcal{N}$) and it contains the indices of variables that are constrained to equal zero. The challenge resides in selecting variables to assign to one set or the other, and this is the purpose of Lawson and Hanson's algorithm NNLS. Consider the following example to help clarify this principle.

We desire to model the following data after Equation (1):

$$\mathbf{a} = \begin{bmatrix} 92 \\ 74 \\ 18 \\ 41 \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} 95 & 89 & 82 \\ 23 & 76 & 44 \\ 61 & 46 & 62 \\ 42 & 2 & 79 \end{bmatrix}$$

The unconstrained least squares solution is shown below:

$$\hat{\mathbf{k}} = \begin{bmatrix} -0.45 \\ 0.71 \\ 0.68 \end{bmatrix} \tag{9}$$

If we solve the problem using the NNLS algorithm, taking the zero vector as the initial feasible solution, the solution evolves as follows:

$${}^0\hat{\mathbf{k}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \Rightarrow \quad {}^1\hat{\mathbf{k}} = \begin{bmatrix} 0 \\ 0 \\ 0.81 \end{bmatrix} \quad \Rightarrow \quad {}^2\hat{\mathbf{k}} = \begin{bmatrix} 0 \\ 0.63 \\ 0.35 \end{bmatrix} \tag{10}$$

Thus we obtain the correct NNLS solution after $r = 2$ iterations of the algorithm. The passive and active sets for the three solution vectors shown above are

$$\begin{aligned} {}^0\mathcal{P} = \emptyset, {}^0\mathcal{A} = \{1,2,3\} &\Rightarrow {}^1\mathcal{P} = \{3\}, {}^1\mathcal{A} = \{1,2\} \\ &\Rightarrow {}^2\mathcal{P} = \{2,3\}, {}^2\mathcal{A} = \{1\} \end{aligned} \tag{11}$$

The first and second iterations are merely equality-constrained least squares problems that can be solved by the method of direct elimination [6]. The first step when solving a simple direct elimination problem of this type is to partition the matrices into the constrained and unconstrained parts. Taking the final iteration shown in Equation (11) as an example, this becomes

$$\begin{aligned} \mathbf{a} = \mathbf{Ck} &= [\mathbf{C}_{\mathcal{A}} \ \mathbf{C}_{\mathcal{P}}] \begin{bmatrix} {}_{\mathcal{A}}\mathbf{k} \\ {}_{\mathcal{P}}\mathbf{k} \end{bmatrix} \\ &= [\mathbf{c}_{\cdot 1}]k_1 + [\mathbf{c}_{\cdot 2} \ \mathbf{c}_{\cdot 3}] \begin{bmatrix} k_2 \\ k_3 \end{bmatrix} \end{aligned} \tag{12}$$

$$\mathbf{a} = [\mathbf{c}_{\cdot 2} \ \mathbf{c}_{\cdot 3}] \begin{bmatrix} k_2 \\ k_3 \end{bmatrix} \tag{13}$$

where Equation (13) follows once the actively constrained variable k_1 has been set to zero. The remaining part of the solution, then, is simply the unconstrained least squares solution of the passive set vectors, i.e.

$$\mathbf{a} = \begin{bmatrix} 92 \\ 74 \\ 18 \\ 41 \end{bmatrix} \cong \begin{bmatrix} 89 & 82 \\ 76 & 44 \\ 46 & 62 \\ 2 & 79 \end{bmatrix} \begin{bmatrix} k_2 \\ k_3 \end{bmatrix} \qquad \therefore \begin{bmatrix} \hat{k}_2 \\ \hat{k}_3 \end{bmatrix} = \begin{bmatrix} 0.63 \\ 0.35 \end{bmatrix} \tag{14}$$

and the complete NNLS solution is given in Equation (10). Clearly now, NNLS can be viewed in terms of a projection of the data in **a** into the subspace defined by the columns of **C** whose indices are in the passive set, i.e., $\mathbf{C}_{\mathcal{P}}$. The following equations summarize the general solution to the zero-equality-constrained problem:

$$\begin{aligned} {}_{\mathcal{P}}\mathbf{k} &= \mathbf{C}_{\mathcal{P}}^{+}\mathbf{a} \\ {}_{\mathcal{A}}\mathbf{k} &= \mathbf{0} \end{aligned} \tag{15}$$

The most important point here is that the pseudoinverse is a function of the particular passive set $\mathcal{P}$.

3.3. Solving NNLS problems with multiple RHS

Given the nature of NNLS, we must emphasize that a problem with multiple RHS cannot be treated in the same way as an identical unconstrained least squares problem. Since the pseudoinverse can vary from column to column, one cannot simply precompute it once and apply it to each column of **A**, as is the case with the unconstrained least squares solution. Consider the data above augmented by additional observations

$$\mathbf{A} = \begin{bmatrix} 92 & 99 & 80 \\ 74 & 19 & 43 \\ 18 & 41 & 51 \\ 41 & 61 & 39 \end{bmatrix}$$

The rigorous, non-negativity-constrained solution is given by

$$\hat{\mathbf{K}} = \begin{bmatrix} 0 & 0.82 & 0.30 \\ 0.63 & 0 & 0.30 \\ 0.35 & 0.15 & 0.30 \end{bmatrix}$$

with the corresponding passive sets

$$\mathcal{P} = \{\{2,3\} \;\; \{1,3\} \;\; \{1,2,3\}\}$$

Here $\mathcal{P}$ is an ordered set containing the passive sets $\mathcal{P}_j$ for the corresponding jth columns of **K**. Since the passive sets differ among the three columns of **K**, separate pseudoinverse calculations are required for each solution. Given that the elements of the passive sets are not known without executing the NNLS algorithm, the traditional method of solving NNLS is to treat each column independently as a matter of course. Thus the columns of **K** are solved sequentially in a manner which we will term 'column-serial'. The column-serial approach completely solves the NNLS problem for the jth column of **K** before beginning to solve for the $(j+1)$th column. The evolution of the solution for the three-RHS example above is then

$${}^{0}\hat{\mathbf{k}}_{\cdot 1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{1} {}^{1}\hat{\mathbf{k}}_{\cdot 1} = \begin{bmatrix} 0 \\ 0 \\ 0.81 \end{bmatrix} \xrightarrow{2} {}^{2}\hat{\mathbf{k}}_{\cdot 1} = \begin{bmatrix} 0 \\ 0.63 \\ 0.35 \end{bmatrix}$$

$${}^{0}\hat{\mathbf{k}}_{\cdot 2} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{1} {}^{1}\hat{\mathbf{k}}_{\cdot 2} = \begin{bmatrix} 0 \\ 0 \\ 0.87 \end{bmatrix} \xrightarrow{2} {}^{2}\hat{\mathbf{k}}_{\cdot 2} = \begin{bmatrix} 0.82 \\ 0 \\ 0.15 \end{bmatrix}$$

$${}^{0}\hat{\mathbf{k}}_{\cdot 3} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{1} {}^{1}\hat{\mathbf{k}}_{\cdot 3} = \begin{bmatrix} 0 \\ 0 \\ 0.78 \end{bmatrix} \xrightarrow{2} {}^{2}\hat{\mathbf{k}}_{\cdot 3} = \begin{bmatrix} 0 \\ 0.36 \\ 0.52 \end{bmatrix}$$

$$\xrightarrow{3} {}^{3}\hat{\mathbf{k}}_{\cdot 3} = \begin{bmatrix} 0.30 \\ 0.30 \\ 0.30 \end{bmatrix}$$

This methodology, while straightforward, tends to be computationally inefficient, because it can result in redundant calculations. In the present case, seven pseudoinverses must be computed to complete the solution. However, if we examine the progression of passive sets by iteration, i.e.

$$\begin{aligned} {}^{0}\boldsymbol{\mathcal{P}} = \{\emptyset \;\; \emptyset \;\; \emptyset\} \quad &\xrightarrow{1} \quad {}^{1}\boldsymbol{\mathcal{P}} = \{\{3\} \;\; \{3\} \;\; \{3\}\} \\ &\xrightarrow{2} \quad {}^{2}\boldsymbol{\mathcal{P}} = \{\{2,3\} \;\; \{1,3\} \;\; \{2,3\}\} \\ &\xrightarrow{3} \quad {}^{3}\boldsymbol{\mathcal{P}} = \{\{2,3\} \;\; \{1,3\} \;\; \{1,2,3\}\} \end{aligned}$$

we note that all the columns share the same passive set on the first iteration and that two of the three columns share the same passive set on the second iteration. Thus one would like to compute the pseudoinverse corresponding to $\mathcal{P} = \{3\}$, for instance, a single time and apply it to all three columns. Continuing in this manner would reduce the total count of pseudoinverse calculations in this example to four: one in step 1 (for all three columns), two in step 2 (one for columns 1 and 3 and one for column 2) and one in step 3 (for column 3 only). These considerations suggest that it would be advantageous to execute the NNLS algorithm in a 'column-parallel' fashion in which the ith iteration of NNLS is performed for all columns before the $(i+1)$th iteration is begun for any column. Taking this approach, the example NNLS solution evolves as

$${}^{0}\hat{\mathbf{K}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \xrightarrow{1} {}^{1}\hat{\mathbf{K}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.81 & 0.87 & 0.78 \end{bmatrix}$$

$$\xrightarrow{2} {}^{2}\hat{\mathbf{K}} = \begin{bmatrix} 0 & 0.82 & 0 \\ 0.63 & 0 & 0.36 \\ 0.35 & 0.15 & 0.52 \end{bmatrix}$$

$$\xrightarrow{3} {}^{3}\hat{\mathbf{K}} = \begin{bmatrix} 0 & 0.82 & 0.30 \\ 0.63 & 0 & 0.30 \\ 0.35 & 0.15 & 0.30 \end{bmatrix}$$

An added benefit of the column-parallel way of organizing the solution steps is that it facilitates the design of algorithms that are rich in matrix–matrix operations. This minimizes data movement and makes more effective use of the cache memories on modern processors, both leading to enhanced computational performance.

In our new algorithm we reorganize the calculation in *two* ways. First we group together problems that share a common passive set and solve them together; and second, recognizing that the passive sets vary from iteration to

iteration, we perform each NNLS iteration for all columns in parallel rather than performing all iterations for each column in series. Of course, as in this example, not all columns will require the same number of iterations to achieve optimality. It is a simple matter, however, to remove a given column from further consideration once its optimal solution has been obtained.

One final factor that will have, perhaps, the largest impact on performance is the choice of a feasible solution to initialize the NNLS algorithm. In the example above we have used zero vectors. While this appears to be the usual choice, it would seem to be a particularly poor choice in the context of curve resolution applications. As pointed out by Gemperline and Cash [8] in their discussion of the 'self-modeling curve resolution research hypothesis', we should expect to find very few actively constrained variables in the fitted model if the constraints accurately and precisely reflect the physical reality of the given experiment. To the extent that the non-rigorous overwriting method for approximating NNLS provides a reasonable estimate of the true active and passive sets, it provides a good initial feasible solution for rigorous NNLS and is the one we use exclusively. Given the foregoing example together with its overwriting solution ${}^0\hat{\mathbf{K}}$, we obtain the rigorous NNLS solution in a single iteration with a total of three pseudoinverse calculations:

$$ {}^0\hat{\mathbf{K}} = \begin{bmatrix} 0 & 0.92 & 0.30 \\ 0.71 & 0 & 0.30 \\ 0.69 & 0.14 & 0.30 \end{bmatrix} \xrightarrow{1} {}^1\hat{\mathbf{K}} = \begin{bmatrix} 0 & 0.82 & 0.30 \\ 0.63 & 0 & 0.30 \\ 0.35 & 0.15 & 0.30 \end{bmatrix} $$

In fact, a column of ${}^0\hat{\mathbf{K}}$ is itself the optimal solution in the commonly encountered case that all elements are non-negative as in the third column above. Viewed this way, the rigorous least squares solutions obtained by our NNLS algorithm are seen to be refinements of those obtained by the approximate overwriting method. We will also note that this choice of initializer is similar in spirit to Bro and de Jong's [5] advanced user mode in the sense that approximate active and passive sets are given to the NNLS algorithm up front. The current approach has the advantage, however, that the estimates are self-generated during the course of obtaining the overwriting solution, so it avoids the overhead associated with retaining copies of active and passive sets during ALS procedures.

The final example shows a modest decrease in the number of pseudoinverses that must be computed, about a factor of two, as compared with the one-column-at-a-time approach. It should be noted, however, that the decrease can become more dramatic as the number of RHS becomes much larger. For any l-pure component system there exist 2^l possible passive sets. For the three-component example above there are eight possible $\mathcal{P}$, i.e. {1, 2, 3}, {1, 2}, {1, 3}, {2, 3}, {1}, {2}, {3}, and $\emptyset$. If one is solving a problem where the number of RHS is greater than 2^l, one can necessarily save computational effort by grouping the columns that have identical passive sets and computing one pseudoinverse per passive set. In fact, if one initializes the NNLS algorithm with the overwriting solution, then the number of possible passive sets that need to be considered decreases by one, since the all-positive columns of $\mathbf{K}$ are already solved in the NNLS sense. Even when $2^l - 1$ is a number that is large compared with the number of RHS, if the constrained linear model is truly appropriate for the data, one might expect the number of unique passive sets to be smaller than the number of RHS. Thus we can still use our new techniques to great advantage to solve these problems.

4. FAST COMBINATORIAL NNLS (FC-NNLS) ALGORITHM

At a very basic level the active set/passive set method solves an inequality-constrained least squares problem as a sequence of equality-constrained problems. The overall NNLS algorithm is responsible for defining that sequence. It governs the flow of the solution and defines the details of each individual problem by choosing the particular passive set variables that must be estimated at each step in the sequence. In our column-parallel, combinatorial approach we identify the columns of $\mathbf{K}$ whose passive sets are identical, collect them together, compute the appropriate pseudoinverses and solve for the passive variables. While the foregoing examples have discussed performance in terms of the number of pseudoinverse computations, there are certainly costs associated with identifying and grouping identical passive sets. Efficiently performing this activity is critical to obtaining good performance of the overall FC-NNLS algorithm.

During the initialization phase of the FC-NNLS algorithm we precompute parts of the pseudoinverse that will remain constant throughout the calculation. We then use them to solve the unconstrained, multiple-RHS least squares problem. This description is general, since there are many methods available to perform the minimization (e.g., normal equations and orthogonal factorization) [9]. We use the unconstrained solution to initialize our algorithm. It determines the initial passive set variables by simply identifying the locations of the non-negative entries in $\mathbf{K}$ and building a passive set for each column. It also indexes the columns of $\mathbf{K}$ whose solutions are not optimal, specifically, columns containing negative values. We put those column indices in the set $\mathcal{F}$, which throughout the algorithm will maintain the current list of column indices for suboptimal solutions. An auxiliary index set $\mathcal{H}$ is a subset of $\mathcal{F}$ containing the indices for solutions that are currently infeasible.

The remainder of the algorithm has a main loop/inner loop structure similar to that of the original Lawson and Hanson [4] NNLS algorithm. The main loop serves to generate trial least squares solutions given the current passive set and to test them for optimality. In the case that a solution is not optimal, it selects a variable to be moved from the active set to the passive set for the subsequent iteration. The inner loop accounts for the fact that the unconstrained least squares solution for a passive set variable may become negative at some point and provides a mechanism for moving variables from the passive set to the active set. Set $\mathcal{H}$ indexes the infeasible solutions, and the inner loop terminates when $\mathcal{H}$ has been emptied. The optimality criteria as well as details of the variable selection and movement strategies have been thoroughly described elsewhere [4,5]. Once a column is found to contain an optimal solution, its index is removed from set $\mathcal{F}$. The algorithm terminates when

$\mathcal{F}$ becomes empty. Algorithm FC-NNLS includes a subroutine called algorithm CSSLS (combinatorial subspace least squares), which applies the combinatorial strategy to obtain solutions for the passive set variables. The details of algorithm CSSLS follow shortly after algorithm FC-NNLS.

4.1. Algorithm FC-NNLS

%Initialize

1 Index the columns (RHS) of the solution matrix: $\mathcal{M} := \{1, 2, \ldots, p\}$
2 Index the rows of the solution matrix: $\mathcal{N} := \{1, 2, \ldots, l\}$
3 Precompute constant parts of the pseudoinverse, e.g., $\mathbf{C}^{\mathrm{T}}\mathbf{C}$ and $\mathbf{C}^{\mathrm{T}}\mathbf{A}$
4 Compute $\min_{\mathbf{K}} \| \mathbf{CK} - \mathbf{A} \|_{\mathrm{F}}$, the unconstrained estimate for $\mathbf{K}$
5 Initialize the set of passive sets: $\boldsymbol{\mathcal{P}} = \{\mathcal{P}_1\ \mathcal{P}_2\ \cdots\ \mathcal{P}_p\}$, where $\mathcal{P}_j = \{x \in \mathcal{N} : {}_x\mathbf{K}_j > 0\}$
6 Find the set of columns that is yet to be optimized: $\mathcal{F} = \{j \in \mathcal{M} :\ \mathcal{P}_j \neq \mathcal{N}\}$
7 Complete the overwriting solution: $\forall j \in \mathcal{F}$, $\forall x \notin \mathcal{P}_j$, ${}_x\mathbf{K}_j = 0$

% Main loop

8 If $\mathcal{F} = \emptyset$, go to step 19, you have finished
% Obtain unconstrained least squares solutions for the passive variables
9 Compute $\min_{\mathbf{K}_F} \| \mathbf{C}\,\mathbf{K}_F - \mathbf{A}_F \|$ using **algorithm CSSLS** and $\mathcal{P}_{\mathcal{F}}$

% Inner loop

% Find any infeasible solutions and make them feasible
10 Put indices of columns with negative variables into set $\mathcal{H} : \mathcal{H} = \{j \in \mathcal{F} : \min_{x \in \mathcal{P}_j}({}_x\mathbf{K}_j) < 0\}$
11 If $\mathcal{H} = \emptyset$, go to step 15
12 $\forall i \in \mathcal{H}$, select the variables to move out of the passive sets $\boldsymbol{\mathcal{P}}_{\mathcal{H}}$
13 Compute $\min_{\mathbf{K}_{\mathcal{H}}} \| \mathbf{C}\,\mathbf{K}_{\mathcal{H}} - \mathbf{A}_{\mathcal{H}} \|$ using **algorithm CSSLS** and $\boldsymbol{\mathcal{P}}_{\mathcal{H}}$
14 Go to step 10
15 $\forall i \in \mathcal{F}$, test solution $\mathbf{K}_i$ for optimality
16 Remove from $\mathcal{F}$ indices of columns whose solutions are optimal
17 $\forall i \in \mathcal{F}$, select the variables to move into the passive sets $\boldsymbol{\mathcal{P}}_{\mathcal{F}}$
18 Go to step 8
19 End of computation

The major innovation explicitly illustrated in algorithm FC-NNLS is the structuring of the multiple-RHS NNLS solution process in a column-parallel manner. A single iteration of the FC-NNLS algorithm is completed for all the non-optimal solutions indexed by set $\mathcal{F}$ before starting subsequent iterations. We have omitted the details in steps 12, 15 and 17 that explain the optimality testing and how variables are moved between the active and passive sets. These details, while important, are not particularly relevant to the innovations of our new algorithm. We point interested readers to the discussions by Lawson and Hanson [4] and Bro and de Jong [5] for in-depth explanations.

Algorithm CSSLS is the engine of FC-NNLS that finds, identifies and groups identical passive sets. It then performs the least squares solution subject to a zero-equality constraint on the active set variables. Consider the subsets of the matrix $\mathbf{K}$ that we want to estimate in a least squares sense in steps 9 and 13 of algorithm FC-NNLS. We will do this using the corresponding subsets of the measured absorbances in $\mathbf{A}$ and passive sets in $\boldsymbol{\mathcal{P}}$, and the concentrations in $\mathbf{C}$. For clarity, we will drop, for the moment, the set subscripts on $\mathbf{K}$, $\mathbf{A}$ and $\boldsymbol{\mathcal{P}}$. After initializing $\mathbf{K}$ to zero, we define a new set $\mathcal{M}$ containing the column indices of $\mathbf{K}$ and partition it into k subsets corresponding to the k *unique* passive sets in $\boldsymbol{\mathcal{P}}$. Then for each unique passive set we perform an unconstrained least squares solution for the passive set variables using the submatrices constructed from columns of $\mathbf{C}$ and rows of $\mathbf{K}$ corresponding to variables in the passive set, and the columns of $\mathbf{K}$ and $\mathbf{A}$ corresponding to all the column indices sharing the same passive set. The unconstrained solution routine executes as many times as there are unique passive sets in $\boldsymbol{\mathcal{P}}$. For problems with very many RHS, this is typically a small number compared with the number of solutions required by the one-column-at-a-time approach.

4.2. Algorithm CSSLS: output(K), input(C, A, $\boldsymbol{\mathcal{P}}$)

1 Initialize: $\mathbf{K} = \mathbf{0}$
2 Index the columns of $\mathbf{K}$: $\mathcal{M} = \{1, 2, \ldots, p\}$
3 Find the set of k unique passive sets: $\boldsymbol{\mathcal{U}} = \{\mathcal{U}_1\ \mathcal{U}_2\ \cdots \mathcal{U}_k\} =$ unique($\boldsymbol{\mathcal{P}}$)
4 Index the columns of $\mathbf{K}$ with identical passive sets into $\mathcal{E}$: $\mathcal{E}_j = \{i \in \mathcal{M} :\ \mathcal{P}_i = \mathcal{U}_j\}$
5 $\forall j \in \{1, 2, \ldots, k\}$, solve: $\min_{\mathcal{U}_j \mathbf{K}_{\mathcal{E}_j}} \| \mathbf{C}_{\mathcal{U}_j \mathcal{U}_j} \mathbf{K}_{\mathcal{E}_j} - \mathbf{A}_{\mathcal{E}_j} \|$
6 End computation

There are certainly many different ways that algorithm CSSLS can be implemented, and there are variations on the algorithm that can accomplish the same task. It is crucial, however, that the identification and grouping of like passive sets in steps 3 and 4 be performed as efficiently as possible. Without paying close attention to the details, the time needed for the extra bookkeeping can easily overwhelm any time saved by solving like RHS in parallel. Exemplary Matlab® implementations of the complete FC-NNLS and CSSLS algorithms are furnished in the Appendix.

5. PERFORMANCE COMPARISONS ON REAL DATA

This new methodology promises to reduce the computational burden for rigorous NNLS solutions to problems with many more RHS than factors. Thus we chose to compare the performance of our new algorithm against the previous fast algorithms [5], which are the apparent standards of excellence. To do so, we downloaded [10] both Bro's fast algorithm (FNNLS) and his alternative, advanced user routine (FNNLSb). The solution in Bro's standard FNNLS algorithm is initialized with all zeros. The FNNLSb algorithm, on the other hand, is initialized with estimates of the active and passive sets. In our comparisons we supplied the known true passive and active sets to initialize FNNLSb. We also obtained times for the overwriting method by terminating our algorithm after step 7. The overwriting method, while it

Table I. Data used for algorithm comparison

Data	Data type[a]	n	p	$n \times p$	l
Data set 1	EDS	16384	1024	16777216	15
Data set 2	EDS	262144	256	67108864	10
Data set 3	EDS	65536	1024	67108864	13
Data set 4	EDS	65536	1024	67108864	14
Data set 5	EDS	65536	1024	67108864	15
Data set 6	EDS	1024	1024	1048576	7
Data set 7	EDS	16384	2048	33554432	11
Data set 8	EDS	16384	1012	16580608	3
Data set 9	EDS	1024	2048	2097152	7
Data set 10	FLU	37500	57	2137500	6
Data set 11	EDS	16384	1024	16777216	8
Data set 12	EDS	16384	1024	16777216	5
Data set 13	EDS	16384	1024	16777216	7
Data set 14	EDS	131072	512	67108864	6
Data set 15	EDS	262144	256	67108864	8

[a]EDS, energy-dispersive X-ray spectroscopy; FLU, fluorescence.

does not provide the true least squares solution, should provide the fastest benchmark against which to measure our new algorithm. All programs were coded in Matlab® [11] and executed on a 1 GHz Pentium III computer containing 1 GB of main memory. In all cases we used the method of normal equations to obtain least squares solutions. We precomputed the cross-product matrices for the rigorous NNLS algorithms and included those computation times in the totals, since frequently this was the most time-consuming step in the entire calculation.

We compared the performance of each of these methods with our new FC-NNLS algorithm on 15 data sets. The test data were primarily Sandia National Laboratories-acquired energy-dispersive X-ray spectroscopy (EDS) spectral images. Details regarding several of these data sets have been presented in Reference [3], which also provides a general description applicable to all the EDS data. One data set is a visible fluorescence image that has also been described previously [12]. Table I contains specific information about our data, including type, dimensions (n and p) and estimated rank (l). Data sets 3–5 are the same actual data, but we analyzed them for three different rank estimates. In all cases, **C** and **K** were taken from previous MCR-ALS analyses when they were used as *a priori* known coefficient matrices.

In order to make a more extensive and realistic comparison, we solved least squares problems in both the spectral and concentration domains. That is, we solved for both the long and short dimension of each data set. In the case of data set 1, for example, the data matrix **A** is 16 384 samples by 1024 spectral channels and has an estimated 15 chemical components. We first solved for **K** given **C** for its 1024 RHS; we then solved for **C** given **K** for its 16 384 RHS. Table II contains the execution times for each data set and each method.

Looking first at the cases of solving **K** given **C**, where the number of RHS is relatively small, the performance improvement using our new algorithm is modest, although it is advantageous in all cases: the ratios of the FNNLS solution times to those obtained by our FC-NNLS algorithm range from 1.14 to 16.3. Analogous ratios for the advanced user algorithm FNNLSb range from 1.01 to 3.98. Note that the latter comparisons are best case for the performance of FNNLSb. Since we used the true active and passive sets to initialize the algorithm, no solutions were ever found to be infeasible (i.e., the inner loop was never entered), and computed solutions were always optimal. This would not be the case when only approximate active and passive sets are supplied, so the relative performance of FNNLSb in a real-world application would be worse. Perhaps the most telling performance comparison is with the overwriting method. For these smaller data sets there is essentially no practical difference in the computation times between the approximate overwriting method and our rigorous FC-NNLS algorithm.

Predictably, as the number of RHS becomes much larger, the advantages of the combinatorial strategy become more impressive. When estimating the abundances of each component for all pixels in the spectral images (i.e., solving for **C** given **K**), the timing ratios range from 13.0 to 107 for FNNLS and from 3.13 to 24.5 for FNNLSb. Figure 1 is a revealing graphical presentation of the data in Tables I and II. In general, computation times increase linearly with problem size, defined here to be the product of the number of factors and the number of RHS. Notably, the computation times obtained using FC-NNLS scale very slowly with problem

Table II. Time used to solve non-negative least squares problem for each data set for the spectral and concentration domains

Data set	Time to solve for **K** given **C** (s)				Time to solve for **C** given **K** (s)			
	Overwrite	FC-NNLS	FNNLS	FNNLSb	Overwrite	FC-NNLS	FNNLS	FNNLSb
1	1.3	1.5	4.8	1.6	1.3	1.6	63.2	7.4
2	3.7	3.7	4.2	3.8	3.7	9.1	504.5	81.4
3	5.3	5.5	8.4	5.7	4.9	5.9	207.4	26.7
4	5.6	5.8	8.8	6.0	4.9	6.0	226.4	27.5
5	6.0	6.3	9.4	6.4	5.3	6.9	242.2	28.2
6	0.1	0.1	1.5	0.3	0.1	0.1	1.5	0.3
7	2.0	2.2	6.7	2.7	2.0	2.2	43.6	7.1
8	0.6	0.6	1.2	0.8	0.7	0.7	11.0	4.7
9	0.1	0.2	2.1	0.6	0.1	0.1	1.6	0.4
10	0.1	0.1	0.2	0.1	0.2	0.4	45.1	10.3
11	0.8	0.9	2.6	1.1	0.9	1.2	26.0	5.7
12	0.7	0.7	1.8	1.0	0.8	0.9	17.6	5.0
13	0.8	0.8	2.4	1.1	0.9	1.0	24.2	5.4
14	2.6	2.7	3.3	2.8	2.9	4.1	157.7	38.4
15	3.2	3.2	3.6	3.2	3.1	7.2	390.2	77.8

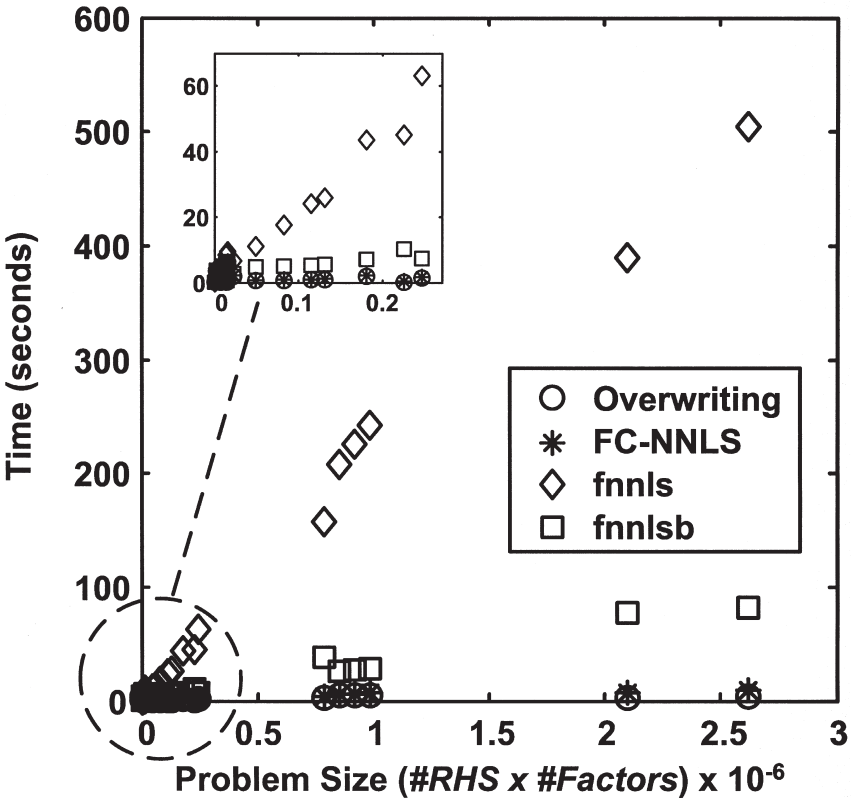

Figure 1. Plot of computation execution time versus the product of number of RHS and *l*, the number of chemical components sought.

size and, in fact, are only marginally slower than the overwriting method for all but the very largest problems (data sets 2 and 15).

Taken together, solving for **K** given an estimate of **C** and then solving for **C** given the new estimate of **K** represent the overwhelming majority of calculations in a complete iteration cycle of a typical alternating least squares algorithm (e.g., MCR-ALS). Thus, clearly, the new fast algorithm can be used to advantage in such applications. Algorithm FC-NNLS, like the overwriting method and algorithm FNNLS, makes no use of information from prior iterations in an ALS sequence. We find, as a result, that the time per ALS iteration is approximately constant. The advanced user algorithm FNNLSb, on the other hand, does make use of prior information. Consequently, ALS algorithms based on FNNLSb would be expected to execute more and more quickly as they approach convergence, when fewer and fewer changes in the active and passive sets are required. In this context we note that the timing results shown for FNNLSb in Table II would be representative of the *final* and quickest iteration in an ALS procedure, since we initialized it with the *a priori* known active and passive sets. It is to be expected, then, that the performance advantage of FC-NNLS as compared with FNNLSb would be even greater over the course of a full ALS calculation than is suggested by the results in the table.

6. CONCLUSION

We have presented a new algorithm that can significantly reduce the computational burden when solving the non-negativity-constrained least squares problem with a large number of RHS. The algorithm provides a mathematically rigorous solution and operates at great speed by (1) initializing the algorithm with the unconstrained least squares solution and (2) reorganizing the calculations to take advantage of the combinatorial nature of the problems. Using large data sets that are typical of spectral imaging applications, we have obtained order-of-magnitude performance improvements as compared with other fast NNLS algorithms. This same combinatorial approach is not limited to NNLS but is usable to great advantage for more general equality- and inequality-constrained least squares problems. We provide an example implementation of the FC-NNLS algorithm, written as a Matlab® m-file, in the Appendix.

Acknowledgements

This work was supported by Sandia National Laboratories and the United States Department of Energy. Sandia is a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-ACO4-94AL85000.

REFERENCES

1. Tauler R, Izquierdo-Ridorsa A, Casassas E. Simultaneous analysis of several spectroscopic titrations with self-modelling curve resolution. *Chemometrics Intell. Lab. Syst.* 1993; **18**: 293–300.
2. Andrew JJ, Hancewicz TM. Rapid analysis of Raman image data using two-way multivariate curve resolution. *Appl. Spectrosc.* 1998; **52**: 797–807.

3. Kotula P, Keenan M, Michael J. Automated analysis of SEM X-ray spectral images: a powerful new microanalysis tool. *Microsc. Microanal.* 2003; **9**: 1–17.
4. Lawson CL, Hanson RJ. *Solving Least Squares Problems*. Prentice-Hall: Englewood Cliffs, NJ, 1974.
5. Bro R, de Jong S. A fast non-negativity-constrained least squares algorithm. *J. Chemometrics* 1997; **11**: 393–401.
6. Van Benthem M, Keenan M, Haaland D. Application of equality constraints on variables during alternating least squares procedures. *J. Chemometrics* 2002; **16**: 613–622.
7. Golub GH, Van Loan CF. *Matrix Computations* (3rd edn). Johns Hopkins University Press: Baltimore, MD, 1996.
8. Gemperline PJ, Cash E. Advantages of soft versus hard constraints in self-modeling curve resolution problems. Alternating least squares with penalty functions. *Anal. Chem.* 2003; **75**: 4236–4243.
9. Bjorck A. *Numerical Methods for Least Squares Problems*. SIAM: Philadelphia, PA, 1996.
10. Bro R. [Online]. Available: http://www.models.kvl.dk/source/ [14 April 2004].
11. *MATLAB, Ver. 6.5.1*. The MathWorks: Natick, MA, 2003.
12. Keenan M, Timlin J, Van Benthem M, Haaland D. Algorithms for constrained linear unmixing with application to the hyperspectral analysis of fluorophore mixtures. *Proc. SPIE* 2002; **4816**: 193–202.

APPENDIX

```
function [K, Pset] = fcnnls(C, A)
% NNLS using normal equations and the fast combinatorial strategy
%
% I/O: [K, Pset] = fcnnls(C, A);
% K = fcnnls(C, A);
%
% C is the nObs × lVar coefficient matrix
% A is the nObs × pRHS matrix of observations
% K is the lVar × pRHS solution matrix
% Pset is the lVar × pRHS passive set logical array
%
% M. H. Van Benthem and M. R. Keenan
% Sandia National Laboratories

% Pset: set of passive sets, one for each column
% Fset: set of column indices for solutions that have not yet converged
% Hset: set of column indices for currently infeasible solutions
% Jset: working set of column indices for currently optimal solutions

% Check the input arguments for consistency and initialize
error(nargchk(2,2,nargin))
[nObs, lVar] = size(C);

if size(A,1)~= nObs, error('C and A have imcompatible sizes'), end
pRHS = size(A,2);
W = zeros(lVar, pRHS);
iter = 0; maxiter = 3*lVar;

% Precompute parts of pseudoinverse
CtC = C'*C; CtA = C'*A;
% Obtain the initial feasible solution and corresponding passive set
K = cssls(CtC, CtA);
Pset = K > 0;
K(~Pset) = 0;
D = K;
Fset = find(~all(Pset));

% Active set algorithm for NNLS main loop
while ~isempty(Fset)
  % Solve for the passive variables (uses subroutine below)
  K(:,Fset) = cssls(CtC, CtA(:,Fset), Pset(:,Fset));
  % Find any infeasible solutions
  Hset = Fset(find(any(K(:,Fset) < 0)));
  % Make infeasible solutions feasible (standard NNLS inner loop)
  if ~isempty(Hset)
    nHset = length(Hset);
    alpha = zeros(lVar, nHset);
```



```
    while ~isempty(Hset) & (iter < maxiter)
      iter = iter + 1;
      alpha(:,1:nHset) = Inf;
      % Find indices of negative variables in passive set
      [i, j] = find(Pset(:,Hset) & (K(:,Hset) < 0));
      hIdx = sub2ind([lVar nHset], i, j);
      negIdx = sub2ind(size(K), i, Hset(j)');
      alpha(hIdx) = D(negIdx)./(D(negIdx) - K(negIdx));
      [alphaMin,minIdx] = min(alpha(:,1:nHset));
      alpha(:,1:nHset) = repmat(alphaMin, lVar, 1);
      D(:,Hset) = D(:,Hset)-alpha(:,1:nHset).*(D(:,Hset)-K(:,Hset));
      idx2zero = sub2ind(size(D), minIdx, Hset);
      D(idx2zero) = 0;
      Pset(idx2zero) = 0;
      K(:, Hset) = cssls(CtC, CtA(:,Hset), Pset(:,Hset));
      Hset = find(any(K < 0)); nHset = length(Hset);
    end
  end
  % Make sure the solution has converged
  if iter == maxiter, error('Maximum number iterations exceeded'), end
  % Check solutions for optimality
  W(:,Fset) = CtA(:,Fset)-CtC*K(:,Fset);
  Jset = find(all(~Pset(:,Fset).*W(:,Fset) <= 0));
  Fset = setdiff(Fset, Fset(Jset));
  % For non-optimal solutions, add the appropriate variable to Pset
  if ~isempty(Fset)
    [mx, mxidx] = max(~Pset(:,Fset).*W(:,Fset));
    Pset(sub2ind([lVar pRHS], mxidx, Fset)) = 1;
    D(:,Fset) = K(:,Fset);
  end
end

% ****************************** Subroutine
****************************
function [K] = cssls(CtC, CtA, Pset)
% Solve the set of equations CtA = CtC*K for the variables in set Pset
% using the fast combinatorial approach
K = zeros(size(CtA));
if (nargin == 2) || isempty(Pset) || all(Pset(:))
  K = CtC\CtA;
else
  [lVar pRHS] = size(Pset);
  codedPset = 2.^(lVar-1:-1:0)*Pset;
  [sortedPset, sortedEset] = sort(codedPset);
  breaks = diff(sortedPset);
  breakIdx = [0 find(breaks) pRHS];
  for k = 1:length(breakIdx)-1
    cols2solve = sortedEset(breakIdx(k)+1:breakIdx(k+1));
    vars = Pset(:,sortedEset(breakIdx(k)+1));
    K(vars,cols2solve) = CtC(vars,vars)\CtA(vars,cols2solve);
  end
end
```