Optimized multidimensional nonoscillating deconvolution

M. Morháč *, V. Matoušek, J. Kliman

Institute of Physics, Slovak Academy of Sciences, Dúbravská cesta 9, SK-842 28 Bratislava, Slovak Republic

Received 20 August 2000; received in revised form 27 February 2001

Abstract

The paper presents a nonoscillating iterative method of Gold deconvolution. The method is generalized for multidimensional data. From computational point of view the Gold deconvolution is time-consuming operation. It requires a great number of numerical operations. A new optimized algorithm aimed to reduce number of computer operations has been derived in the paper. © 2002 Elsevier Science B.V. All rights reserved.

MSC: 15A06; 15A29

Keywords: Linear equations; Inverse problems; Deconvolution; Gold deconvolution; Multidimensional systems

1. Introduction

The deconvolution methods are widely applied in various fields of data processing. Recently they found many applications in various domains of experimental science, e.g., image and signal restoration, determination of thickness of multilayer structures, determination of positions and intensities of peaks in nuclear histograms. The deconvolution method can be successfully applied also for the decomposition of multiplets in γ-ray spectroscopy [2].

From numerical point of view the deconvolution belongs to one of the most critical problems. It is so-called ill-posed problem, which means that many different functions solve convolution equation within error bounds of the experimental data. The estimates of solution are extremely sensitive to errors in the measured data [8]. When employing standard algorithms to solve linear convolution system small changes in measured data (noise) cause enormous oscillations in the result. It implies that the suitable method of regularization must be employed. Tikhonov first treated this problem on a strict mathematical basis by introducing the regularization theory and methods [7,4]. The regularization encompasses a class of solution techniques that modify an ill-posed problem into a well-posed one by approximation so that a physically acceptable approximate solution can be obtained, and

* Corresponding author.
E-mail address: fyzimiro@savba.sk (M. Morháč).
the solution is sufficiently stable from the computational viewpoint [6]. Therefore the methods of regularization can be classified from different aspects:

- quality of deconvoluted signal (smoothness, positive solution, oscillations, etc.);
- computational complexity;
- convergence speed.

The Gold deconvolution algorithm proved to work as the most stable with very good results. Its basic property is that the solution is always nonnegative. When processing data (e.g. histograms - spectra) where negative solutions are senseless this is very important property.

On the other hand, from the computational point of view the Gold deconvolution is an extremely time-consuming operation. This problem is becoming relevant for big sizes of data and for multidimensional data where the number of operations grows exponentially with the sizes. The implementation of the method requires optimization from the point of view of time (redundant operations can be omitted) and memory storage (data during computation may be stored in memory more efficiently).

2. Theory

The relationship between a measured value \( x(t) \) and the raw result of measurement \( y(t) \) can be described by a convolution-type integral equation

\[
y(t) = \int_{-\infty}^{\infty} x(\tau) h(t - \tau) \, d\tau,
\]

where \( h(t) \) is an impulse response. Knowledge of the instrumental function \( h(t) \) is usually required. For a discrete system, (1) can be written as

\[
y(i) = \sum_{k=0}^{N-1} h(i - k) x(k), \quad i = 0, 1, \ldots, 2N - 2,
\]

where \( N \) is the number of samples of vectors \( h, x \).

The impulse response has finite length. Therefore we will suppose that \( h(i) = 0 \) for \( i < 0 \) and \( i \geq N \). Then (2) can be written in matrix form as

\[
\begin{bmatrix}
y(0) \\
y(1) \\
y(2) \\
\vdots \\
y(2N - 2)
\end{bmatrix} =
\begin{bmatrix}
h(0) & 0 & 0 & \cdots \\
h(1) & h(0) & 0 & \cdots \\
h(2) & h(1) & h(0) & \cdots \\
\vdots & \vdots & \vdots & \ddots \\
h(N - 1) & h(N - 2) & h(N - 3) & \cdots \\
0 & h(N - 1) & h(N - 2) & \cdots \\
0 & 0 & h(N - 1) & \cdots \\
\vdots & \vdots & \vdots & \ddots 
\end{bmatrix}
\begin{bmatrix}
x(0) \\
x(1) \\
x(2) \\
\vdots \\
x(N - 1)
\end{bmatrix}
\]
or
\[ y = Hx. \] (4)

It means that the columns of \( H \) are represented by vectors \( h \) mutually shifted by one position. Multiplying both sides of (4) by \( H^T \) gives
\[
H^T y = H^T H x
\] (5)
or
\[ y_1 = H_1 x, \] (6)

where \( H_1 \) is Toeplitz matrix [2,5]. Solution of the linear equation system (6) (vector \( x \)), under the condition that the output vector \( y \) and matrix of impulse response \( H \) is known, is a problem of deconvolution. The output vector of system \( y \) is affected by noise that accompanies each measurement. The existence of this noise strongly affects the process of deconvolution, and can lead to difficulties in solving the linear equation system (6).

The above-formulated problem of input reconstruction is a rule ill-conditioned, i.e., the estimates \( \hat{x}(t) \) of \( x(t) \) satisfying (1) are extremely sensitive to errors in the measured data \( y(t) \). It is expressed by the fact, that the matrix \( H_1 \) is almost a singular one. The direct inversion of \( H_1 \) for solving \( x \) cannot lead to a stable solution.

Therefore, in order to solve this problem, the method of regularization must be included. This means that original problem is replaced by an approximate one whose solutions are significantly less sensitive to errors in data \( y(t) \).

Van Cittert iterative method of deconvolution is widely applied in different areas, for example in spectroscopy or in image processing [2,9]. Van Cittert algorithm of deconvolution is described in detail in [9], so we describe it only very briefly. Its basic form for a general linear discrete system is
\[
x^{(k+1)} = x^{(k)} + \mu (y - A x^{(k)}),
\] (7)
where \( A \) is system matrix, \( k \) represents the number of iterations and \( \mu \) is the relaxation factor. The convergence condition of (7) is that the diagonal elements of the matrix \( A \) satisfy
\[
A_{ii} > \sum_{j=0,j\neq i}^{N-1} A_{ij}, \quad i = 0, 1, \ldots, N - 1.
\] (8)

It is obvious that such a diagonal element dominance is rare in physical problems. However, the deconvolution algorithm in (7) can be modified in such a way that it will satisfy the conditions of convergence. Hence, (7) becomes
\[
x^{(k+1)} = \mu y + (E - \mu A) x^{(k)} = \mu y + Dx^{(k)},
\] (9)
where \( E \) is a unit matrix and
\[
D = E - \mu A.
\] (10)
Under the condition that $x^{(0)} = y$, the successive substitutions give
\[ x^{(k)} = \mu y + \mu D y + \cdots + \mu D^{k-1} y + \mu D^k y = \mu (E + D + \cdots + D^k) y. \] (11)

Supposing that $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$ are eigenvalues of $A$, then $(1 - \mu \lambda_0), (1 - \mu \lambda_1), \ldots, (1 - \mu \lambda_{N-1})$ are eigenvalues of $D$. If
\[ \lim_{k \to \infty} (1 - \mu \lambda_i)^k = 0, \quad i = 0, 1, \ldots, N - 1 \] (12)
then
\[ \lim_{k \to \infty} D^k = [0] \]
and
\[ \lim_{k \to \infty} x^{(k)} = A^{-1} y = x. \] (13)

From (12) this implies that the necessary and sufficient conditions of convergence are
\[ |1 - \mu \lambda_i| < 1, \quad i = 0, 1, \ldots, N - 1. \] (14)

If we define $\lambda_i$ and its conjugate $\lambda_i^*$ as
\[ \lambda_i = a_i + j b_i, \quad \lambda_i^* = a_i - j b_i \]
then the convergence condition (14) becomes
\[ \mu [\mu (a_i^2 + b_i^2) - 2a_i] < 0, \quad i = 0, 1, \ldots, N - 1. \] (15)

Inequality (15) gives two bounds for $\mu$
\[ \mu = 0, \quad \mu = \frac{2a_i}{a_i^2 + b_i^2}, \quad i = 0, 1, \ldots, N - 1. \] (16)

$N$ conditions determine the bounds of the $\mu$ coefficient. Unfortunately these conditions are not fulfilled for all cases. However, if the system matrix $A$ is positive definite the convergent solution always exists. So we settle the algorithm in such a way that the eigenvalues $\lambda_i$ will be positive, real numbers.

Let us return to (5). Matrix $H^TH$ is symmetric, so its eigenvalues are real. The eigenvalues of matrix $(H^TH)(H^TH)$ are squares of eigenvalues of matrix $H^TH$ and therefore must be positive. (5) becomes
\[ (H^THH^T)y = (H^THH^T)x \] (17)
and the iterative algorithm of deconvolution becomes
\[ x^{(k+1)} = x^{(k)} + \mu [(H^THH^T)y - (H^THH^T)x^{(k)}] \] (18)
or
\[ x^{(k+1)} = x^{(k)} + \mu [y' - H'x^{(k)}]. \] (19)
By this we ensure the existence of a common interval of solution for inequality (14) and convergence of the deconvolution algorithm. Eigenvalues \( \lambda_i \) are real, positive numbers, so from (16) for \( \mu \) we can write
\[
0 < \mu < \frac{2}{\lambda_{\text{max}}},
\]
where \( \lambda_{\text{max}} \) is the greatest eigenvalue of \( H' \)
\[
\lambda_{\text{max}} = \max(\lambda_0, \lambda_1, \ldots, \lambda_{N-1}).
\]
Now we determine the maximum eigenvalue \( \lambda_{\text{max}} \). For eigenvalues of system (17) we can write
\[
H'x = \lambda_i x, \quad i = 0, 1, \ldots, N - 1.
\]
If \( x_j \) is the greatest absolute element in \( x \), then from (22) we get
\[
\sum_{m=0}^{N-1} H'_{jm} x_m = \lambda_i x_j
\]
or
\[
\lambda_i = \frac{\sum_{m=0}^{N-1} H'_{jm} x_m}{x_j}.
\]
Then
\[
\lambda_i \leq \sum_{m=0}^{N-1} |H'_{jm}|, \quad i = 0, 1, \ldots, N - 1.
\]
In practical cases we do not know the greatest element in \( x \). We determine the value of \( \lambda_{\text{max}} \) as the maximum value from \( \lambda_i \), determined by (25) e.g., from the sum of absolute values of rows in matrix \( H' \). This is a base of Van Cittert algorithm of deconvolution. Now we introduce, in analogy with (24), local variable relaxation factor
\[
\mu_i = \frac{x_i^{(k)}}{\sum_{m=0}^{N-1} H'_{im} x_m^{(k)}}
\]
and we use it in (19). For \( i \)th element of vector \( x^{(k+1)} \) we get
\[
x_i^{(k+1)} = x_i^{(k)} + \frac{x_i^{(k)}}{\sum_{m=0}^{N-1} H'_{im} x_m^{(k)}} \left[ y_i' - \sum_{m=0}^{N-1} H'_{im} x_m^{(k)} \right]
\]
or
\[
x_i^{(k+1)} = \frac{y_i'}{\sum_{m=0}^{N-1} H'_{im} x_m^{(k)}} x_i^{(k)},
\]
where \( x_i^{(0)} = 1, \ i \in (0, N - 1) \). Eq. (28) is the Gold algorithm of deconvolution [3]. It is an extension of Van Cittert’s iterative method. One can observe that if \( h_i \geq 0, \ y_i \geq 0, \ i \in (0, N - 1) \) then \( x_i^{(k+1)}, k = 0, 1, 2, \ldots, \) are also always positive.
3. Multidimensional Gold deconvolution

The relation between the input value of a linear time-invariant system and its output value can be described by convolution integral equation. Subsequently for one-, two-, \ldots, \( n \)-dimensional continuous linear system one can write

\[
y_1(t_1) = \int_{-\infty}^{+\infty} h_1(t_1 - \tau)x_1(\tau) \, d\tau
\]

\[
y_2(t_1, t_2) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h_2(t_1 - \tau_1, t_2 - \tau_2)x_2(\tau_1, \tau_2) \, d\tau_1 \, d\tau_2
\]

\[\vdots\]

\[
y_n(t_1, \ldots, t_n) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} h_n(t_1 - \tau_1, \ldots, t_n - \tau_n)x_n(\tau_1, \ldots, \tau_n) \, d\tau_1 \, d\tau_2 \cdots d\tau_n,
\]

where \( h_i, i \in \langle 1, n \rangle \) is impulse response function of \( i \)-dimensional convolution system, \( y_i \) and \( x_i \) are output and input \( i \)-dimensional signals, respectively. Analogously for discrete signals the (29) will become

\[
y_1(i) = \sum_{k=0}^{N_1-1} h_1(i - k)x_1(k), \quad i = 0, 1, \ldots, 2N_1 - 2,
\]

\[
y_2(i_1, i_2) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} h_2(i_1 - k_1, i_2 - k_2)x_2(k_1, k_2),
\]

\[i_1 \in \langle 0, 2N_1 - 2 \rangle, \quad i_2 \in \langle 0, 2N_2 - 2 \rangle,
\]

\[\vdots\]

\[
y_n(i_1, \ldots, i_n) = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \cdots \sum_{k_n=0}^{N_n-1} h_n(i_1 - k_1, \ldots, i_n - k_n)x_n(k_1, \ldots, k_n),
\]

\[i_j \in \langle 0, 2N_j - 2 \rangle, \quad j \in \langle 1, n \rangle.
\]

We shall assume the knowledge of the impulse response function (resolution instrumental function) and the measured output values \( y \). Based on this, in deconvolution procedure, we are looking for the solution of the corresponding system of linear equations (32).
The impulse response function of \(j\)-dimensional system is supposed to have finite length in all dimensions. It implies that \(h_j(i_1, i_2, \ldots, i_j) = 0\) for \(i_k < 0\) and for \(i_k \geq N_k\), \(k \in \{1, j\}\). Then for one-dimensional system (30) can be written in matrix form

\[
y_1(0) \begin{bmatrix}
y_1(1) \\
y_1(2) \\
\vdots \\
y_1(2N_1 - 2)
\end{bmatrix} = 
\begin{bmatrix}
h_1(0) & 0 & 0 & \cdots \\
h_1(1) & h_1(0) & 0 & \cdots \\
h_1(2) & h_1(1) & h_1(0) & \cdots \\
\vdots & \vdots & \vdots & \ddots \\
h_1(N_1 - 1) & h_1(N_1 - 2) & h_1(N_1 - 3) & \cdots \\
0 & h_1(N_1 - 1) & h_1(N_1 - 2) & \cdots \\
0 & 0 & h_1(N_1 - 1) & \cdots \\
\vdots & \vdots & \vdots & \ddots 
\end{bmatrix} 
\begin{bmatrix}
x_1(0) \\
x_1(1) \\
x_1(2) \\
\vdots \\
x_1(N_1 - 1)
\end{bmatrix}
\tag{33}
\]

For two-dimensional convolution system a similar procedure can be used. Let us assume that we denote the shifted \(i\)th column of the two-dimensional response as submatrix

\[
h_2^{(1)}(i_2) = 
\begin{bmatrix}
h_2(0, i_2) & 0 & 0 & \cdots \\
h_2(1, i_2) & h_2(0, i_2) & 0 & \cdots \\
h_2(2, i_2) & h_2(1, i_2) & h_2(0, i_2) & \cdots \\
\vdots & \vdots & \vdots & \ddots \\
h_2(N_1 - 1, i_2) & h_2(N_1 - 2, i_2) & h_2(N_1 - 3, i_2) & \cdots \\
0 & h_2(N_1 - 1, i_2) & h_2(N_1 - 2, i_2) & \cdots \\
0 & 0 & h_2(N_1 - 1, i_2) & \cdots \\
\vdots & \vdots & \vdots & \ddots 
\end{bmatrix}
\tag{34}
\]

and the \(i\)th columns of the input and output matrices, respectively, as vectors

\[
x_2^{(1)}(i_2) = [x_2(0, i_2), x_2(1, i_2), \ldots, x_2(N_1 - 1, i_2)]^T, \tag{35}
\]

\[
y_2^{(1)}(i_2) = [y_2(0, i_2), y_2(1, i_2), \ldots, y_2(N_1 - 1, i_2)]^T. \tag{36}
\]
where \( i \in \langle 0, N_2 - 1 \rangle \). Then we have

\[
\begin{bmatrix}
  y_2^{(1)}(0) \\
  y_2^{(1)}(1) \\
  \vdots \\
  y_2^{(1)}(2N_2 - 2)
\end{bmatrix} =
\begin{bmatrix}
  h_2^{(1)}(0) & 0 & 0 & \cdots \\
  h_2^{(1)}(1) & h_2^{(1)}(0) & 0 & \cdots \\
  h_2^{(1)}(2) & h_2^{(1)}(1) & h_2^{(1)}(0) & \cdots \\
  \vdots & \vdots & \vdots & \ddots \\
  h_2^{(1)}(N_2 - 1) & h_2^{(1)}(N_2 - 2) & h_2^{(1)}(N_2 - 3) & \cdots & \vdots \\
  0 & h_2^{(1)}(N_2 - 1) & h_2^{(1)}(N_2 - 2) & \cdots & \vdots \\
  0 & 0 & h_2^{(1)}(N_2 - 1) & \cdots & \vdots \\
  \vdots & \vdots & \vdots & \ddots & \ddots
\end{bmatrix}
\begin{bmatrix}
  x_2^{(1)}(0) \\
  x_2^{(1)}(1) \\
  \vdots \\
  x_2^{(1)}(N_2 - 1)
\end{bmatrix}.
\]

The system matrix in (37) consists of shifted submatrices \( h_2^{(1)}(i) \) given by (34).

We can continue in an analogous way for third, fourth dimension up to \( n \)-dimensional convolution system. Let us denote the submatrix of \( j \)th order of the \( n \)-dimensional response

\[
h_n^{(j)}(i_{j+1}, \ldots, i_n) =
\begin{bmatrix}
  h_n^{(j-1)}(0, i_{j+1}, \ldots, i_n) & 0 & 0 & \cdots \\
  h_n^{(j-1)}(1, i_{j+1}, \ldots, i_n) & h_n^{(j-1)}(0, i_{j+1}, \ldots, i_n) & 0 & \cdots \\
  h_n^{(j-1)}(2, i_{j+1}, \ldots, i_n) & h_n^{(j-1)}(1, i_{j+1}, \ldots, i_n) & h_n^{(j-1)}(0, i_{j+1}, \ldots, i_n) & \cdots \\
  \vdots & \vdots & \vdots & \ddots \\
  h_n^{(j-1)}(N_j - 1, i_{j+1}, \ldots, i_n) & h_n^{(j-1)}(N_j - 2, i_{j+1}, \ldots, i_n) & h_n^{(j-1)}(N_j - 3, i_{j+1}, \ldots, i_n) & \cdots \\
  0 & h_n^{(j-1)}(N_j - 1, i_{j+1}, \ldots, i_n) & h_n^{(j-1)}(N_j - 2, i_{j+1}, \ldots, i_n) & \cdots \\
  0 & 0 & h_n^{(j-1)}(N_j - 1, i_{j+1}, \ldots, i_n) & \cdots \\
  \vdots & \vdots & \vdots & \ddots
\end{bmatrix}
\]

and the submatrices of the \( j \)th order of the input and the output signals, respectively, as

\[
x_n^{(j)}(i_{j+1}, \ldots, i_n) = [x_n^{(j-1)}(0, i_{j+1}, \ldots, i_n), x_n^{(j-1)}(1, i_{j+1}, \ldots, i_n), \ldots, x_n^{(j-1)}(N_j - 1, i_{j+1}, \ldots, i_n)]^T,
\]

\[
y_n^{(j)}(i_{j+1}, \ldots, i_n) = [y_n^{(j-1)}(0, i_{j+1}, \ldots, i_n), y_n^{(j-1)}(1, i_{j+1}, \ldots, i_n), \ldots, y_n^{(j-1)}(N_j - 1, i_{j+1}, \ldots, i_n)]^T.
\]
One can observe that the submatrix of \( j \)th order in (38) consists of shifted submatrices of \((j - 1)\)st order. Taking into account (39), (40) finally one can write

\[
\begin{bmatrix}
  y^{(n-1)}(0) \\
  y^{(n-1)}(1) \\
  \vdots \\
  y^{(n-1)}(2N_n - 2)
\end{bmatrix} =
\begin{bmatrix}
  h^{(n-1)}_n(0) & 0 & 0 & \cdots \\
  h^{(n-1)}_n(1) & h^{(n-1)}_n(0) & 0 & \cdots \\
  h^{(n-1)}_n(2) & h^{(n-1)}_n(1) & h^{(n-1)}_n(0) & \cdots \\
  \vdots & \vdots & \vdots & \ddots \\
  h^{(n-1)}_n(N_n - 1) & h^{(n-1)}_n(N_n - 2) & h^{(n-1)}_n(N_n - 3) & \cdots \\
  0 & h^{(n-1)}_n(N_n - 1) & h^{(n-1)}_n(N_n - 2) & \cdots \\
  0 & 0 & h^{(n-1)}_n(N_n - 1) & \cdots \\
  \vdots & \vdots & \vdots & \ddots 
\end{bmatrix}
\begin{bmatrix}
  x^{(n)}(0) \\
  x^{(n)}(1) \\
  \vdots \\
  x^{(n)}(N_n - 1)
\end{bmatrix}
\]  

(41)

or

\[
y^{(n)}(i) = h^{(n)}_n \cdot x^{(n)}(i).
\]  

(42)

The dimension of the matrix \( h^{(n)}_n \) equals

\[
\prod_{i=1}^{n} (2N_i - 1) \cdot \prod_{j=1}^{n} N_j.
\]  

(43)

One can conclude that using above given procedure the convolution system of any dimension can be converted to the product of matrix with vector given by (41). However from (43) it is observable that the size of the matrix for the multidimensional convolution system would be enormous.

The Gold algorithm for the deconvolution of the one-dimensional spectra (nonoptimized) is described in [1]. As the multidimensional convolution system can be expressed in the form of linear equations according to (42) the algorithm of multidimensional Gold deconvolution is analogous. Therefore we present only its final form. For details we refer to [1]. We calculate

\[
C = h^{(n)\top}_n \cdot h^{(n)}_n \cdot h^{(n)\top}_n \cdot h^{(n)}_n
\]  

(44)

and

\[
y' = h^{(n)\top}_n \cdot h^{(n)}_n \cdot h^{(n)\top}_n \cdot y^{(n)}
\]  

(45)

where the matrix \( h^{(n)}_n \) and the vector \( y^{(n)} \) are defined by (41), (42). Then the Gold algorithm of multidimensional deconvolution is

\[
(k+1)x^{(n)}_n(i) = \frac{y'(i)}{\sum_{m=0}^{M-1} C(i, m) \cdot (k)x^{(n)}_n(m)} \cdot (k)x^{(n)}_n(i),
\]  

(46)

where \( k \) represents number of iterations, \( i \in \{0, M - 1\} \), \( M = \prod_{j=1}^{n} N_j \). One can easily imagine that for number of iterations \( \approx 1000 \) the realization of the algorithm (46) with respect to (41), (42), (44),
(45) requires enormous number of operations. From the computational point of view the algorithm is becoming nonrealizable in reasonable time even for small two-dimensional convolution systems. On these grounds the optimization of the Gold deconvolution algorithm is unavoidable.

4. Optimization of the multidimensional Gold deconvolution

Let us start with one-dimensional deconvolution. In the previous chapter we supposed the length of the impulse response to be equal to \( N \), which is also the length of the sought vector \( x \). The length of impulse response is much more less than the length of both input and output vectors (spectra). Outside this interval the impulse response counts vanish to zero. Let us denote the length of the impulse response as \( L \). Then, if \( N \) is the length of the input vector \( x \) the length of the output vector \( y \) is \( N + L - 1 \).

The algorithm of one-dimensional deconvolution resides in calculation of the matrix \( C \) (44), vector \( y' \) (45), before starting iterations, and in successive corrections of the vector \( x \) according to (46). Each correction is done by the multiplication of the matrix \( C \) with the particular solution \( \psi \).

Exactly this multiplication is critical. In one-dimensional deconvolution \( N^2 \) multiplications must be carried out. However in \( n \)-dimensional deconvolution the number of multiplications increases to \( M^2 \), where \( M = \prod_{j=1}^{n} N_j \).

Let us illustrate the situation for one-dimensional Gold deconvolution for \( L = 3 \) and \( N = 7 \). According to (33) we obtain

\[
\begin{bmatrix}
y_1(0) \\
y_1(1) \\
y_1(2) \\
y_1(3) \\
y_1(4) \\
y_1(5) \\
y_1(6) \\
y_1(7) \\
y_1(8)
\end{bmatrix} =
\begin{bmatrix}
h_1(0) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
h_1(1) & h_1(0) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
h_1(2) & h_1(1) & h_1(0) & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & h_1(2) & h_1(1) & h_1(0) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & h_1(2) & h_1(1) & h_1(0) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & h_1(2) & h_1(1) & h_1(0) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & h_1(2) & h_1(1) & h_1(0) & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & h_1(2) & h_1(1) & h_1(0) & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & h_1(2) & h_1(1) & h_1(0)
\end{bmatrix}
\begin{bmatrix}
x_1(0) \\
x_1(1) \\
x_1(2) \\
x_1(3) \\
x_1(4) \\
x_1(5) \\
x_1(6)
\end{bmatrix}
\]

or

\[ y = H \cdot x. \]  

To calculate the matrix \( C \) and the vector \( y' \) one needs the matrix

\[ B = H^T H. \]
In our example we have

\[
\mathbf{B} = \begin{bmatrix}
    b_0 & b_1 & b_2 & 0 & 0 & 0 \\
    b_1 & b_0 & b_1 & b_2 & 0 & 0 \\
    b_2 & b_1 & b_0 & b_1 & b_2 & 0 \\
    0 & b_2 & b_1 & b_0 & b_1 & b_2 \\
    0 & 0 & b_2 & b_1 & b_0 & b_1 \\
    0 & 0 & 0 & b_2 & b_1 & b_0
\end{bmatrix},
\]

(50)

where

\[
b_0 = h_1^2(0) + h_1^2(1) + h_1^2(2),
\]
\[
b_1 = h_1(0)h_1(1) + h_1(1)h_1(2),
\]
\[
b_2 = h_1(0)h_1(2).
\]

(51)

One can observe that the matrix \( \mathbf{B} \) is 5-diagonal symmetrical matrix. In general case it is \((2L-1)\)-diagonal symmetrical matrix. However, to store its elements the vector \( \mathbf{b} \) of the length \( L \) (in our example 3) is completely sufficient. The element \( i \) of the vector \( \mathbf{b} \) is

\[
b_i = \sum_{j=0}^{L-1-i} h_1(j) \cdot h_1(i + j), \quad i \in \langle 0, L-1 \rangle.
\]

(52)

Regarding to (44), (49), the matrix \( \mathbf{C} \) is

\[
\mathbf{C} = \mathbf{B}^T \cdot \mathbf{B} = \mathbf{B} \cdot \mathbf{B} =
\]

\[
\begin{bmatrix}
    d_{00} & d_{01} & c_2 & c_3 & c_4 & 0 & 0 \\
    d_{01} & d_{11} & c_1 & c_2 & c_3 & c_4 & 0 \\
    c_2 & c_1 & c_0 & c_1 & c_2 & c_3 & c_4 \\
    c_3 & c_2 & c_1 & c_0 & c_1 & c_2 & c_3 \\
    c_4 & c_3 & c_2 & c_1 & c_0 & c_1 & c_2 \\
    0 & c_4 & c_3 & c_2 & c_1 & d_{11} & d_{01} \\
    0 & 0 & c_4 & c_3 & c_2 & d_{01} & d_{00}
\end{bmatrix},
\]

(53)
where

\[
\begin{align*}
  c_0 &= b_0^2 + 2b_1^2 + 2b_2^2, \\
  c_1 &= 2b_0b_1 + 2b_1b_2, \\
  c_2 &= 2b_0b_2 + b_1^2, \\
  c_3 &= 2b_1b_2, \\
  c_4 &= b_2^2, \\
  d_{00} &= c_0 - b_1^2 - b_2^2, \\
  d_{01} &= c_1 - b_1b_2, \\
  d_{11} &= c_0 - b_2^2.
\end{align*}
\]

Now let us extend the system (47) by padding zeros at the beginning and end of vectors \(x, y\). Let the length of extensions is \(L - 1\). Then we obtain

\[
\begin{bmatrix}
  0 \\
  0 \\
  y_1(0) \\
  y_1(1) \\
  y_1(2) \\
  y_1(3) \\
  y_1(4) \\
  y_1(5) \\
  y_1(6) \\
  y_1(7) \\
  y_1(8) \\
  0
\end{bmatrix} =
\begin{bmatrix}
  h_1(0) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  h_1(1) & h_1(0) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  h_1(2) & h_1(1) & h_1(0) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & h_1(2) & h_1(1) & h_1(0) & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & h_1(2) & h_1(1) & h_1(0) & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & h_1(2) & h_1(1) & h_1(0) & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & h_1(2) & h_1(1) & h_1(0) & 0 & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & h_1(2) & h_1(1) & h_1(0) & 0 \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & h_1(2) & h_1(1) & h_1(0) \\
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_1(2) & h_1(1)
\end{bmatrix}
\begin{bmatrix}
  0 \\
  0 \\
  x_1(0) \\
  x_1(1) \\
  x_1(2) \\
  x_1(3) \\
  x_1(4) \\
  x_1(5) \\
  x_1(6) \\
  x_1(7) \\
  0
\end{bmatrix},
\]

(54)
where the old system of equations is denoted by dashed lines. Then according to (49), (53) the extended system (55) can be expressed

\[
\begin{bmatrix}
  y_1'(-2) \\
y_1'(-1) \\
y_1'(0) \\
y_1'(1) \\
y_1'(2) \\
y_1'(3) \\
y_1'(4) \\
y_1'(5) \\
y_1'(6) \\
y_1'(7) \\
y_1'(8)
\end{bmatrix} =
\begin{bmatrix}
d_{00} & d_{01} & c_2 & c_3 & c_4 & 0 & 0 & 0 & 0 & 0 \\
d_{01} & d_{11} & c_1 & c_2 & c_3 & c_4 & 0 & 0 & 0 & 0 \\
c_2 & c_1 & c_0 & c_1 & c_2 & c_3 & c_4 & 0 & 0 & 0 \\
c_3 & c_2 & c_1 & c_0 & c_1 & c_2 & c_3 & c_4 & 0 & 0 \\
c_4 & c_3 & c_2 & c_1 & c_0 & c_1 & c_2 & c_3 & 0 & 0 \\
0 & c_4 & c_3 & c_2 & c_1 & c_0 & c_1 & c_2 & c_3 & 0 \\
0 & 0 & c_4 & c_3 & c_2 & c_1 & c_0 & c_1 & c_2 & c_3 \\
0 & 0 & 0 & c_4 & c_3 & c_2 & c_1 & c_0 & c_1 & c_2 \\
0 & 0 & 0 & 0 & c_4 & c_3 & c_2 & c_1 & d_{11} & d_{01} \\
0 & 0 & 0 & 0 & 0 & c_4 & c_3 & c_2 & d_{01} & d_0
\end{bmatrix}
\begin{bmatrix}
0 \\
x_1(0) \\
x_1(1) \\
x_1(2) \\
x_1(3) \\
x_1(4) \\
x_1(5) \\
x_1(6) \\
x_1(7) \\
x_1(8)
\end{bmatrix}.
\]  

(56)

Again, the old system is denoted by dashed lines. Comparing to (53) one can observe that the system matrix is symmetrical. It does not contain the submatrices \( \mathbf{d} \) in its corners. This fact allows to simplify significantly the Gold deconvolution algorithm mainly for multidimensional data.

Obviously the matrix \( \mathbf{C} \) is 9-diagonal (in general case \( (4L - 3) \)-diagonal) symmetrical matrix. Its elements can be stored in vector \( \mathbf{c} \) of the length \( 2L - 1 \) (in our example 5). These elements are

\[
c_i = \sum_{j=\max(0, i-L+1)}^{\min(N-i, 0)} b_{|j|} \cdot b_{|i-|j|}, \quad i \in \langle 0, 2L - 2 \rangle.
\]  

(57)

We introduce also the relations to calculate vector \( \mathbf{y}' \). According to (45) and with respect to (48), (49) we have

\[
\mathbf{y}' = \mathbf{H}^T \mathbf{H} \mathbf{y} = \mathbf{B} \cdot \mathbf{H}^\top \mathbf{y} = \mathbf{B} \cdot \mathbf{p}.
\]  

(58)

where the vector \( \mathbf{p} \) is

\[
p_i = \sum_{j=0}^{L-1} h_1(j) \cdot y_1(i + j), \quad i \in \langle -L + 1, N + L - 2 \rangle,
\]  

(59)

where for \( k < 0 \) and \( k \geq N \), \( y_k = 0 \). This represents extension of vectors \( \mathbf{x}, \mathbf{y} \) according to (55). Then the vector \( \mathbf{y}' \) is

\[
\mathbf{y}' = \sum_{j=-L+1}^{L-1} b_{|j|} \cdot p_{i+j}, \quad i \in \langle 0, N - 1 \rangle.
\]  

(60)

Eqs. (52), (57), (59) and (60) represent optimum algorithm to compute vectors \( \mathbf{b}, \mathbf{c}, \mathbf{p}, \mathbf{y}' \). These computations can be carried out before starting the deconvolution iterative algorithm itself. From
Table 1  
Number of multiplications for optimized and nonoptimized one-dimensional Gold deconvolution

<table>
<thead>
<tr>
<th></th>
<th>Optimized</th>
<th>Nonoptimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector ( b )</td>
<td>( (L^2 + L)/2 )</td>
<td>( \begin{bmatrix} N^2 &amp; (N + L - 1) \end{bmatrix} )</td>
</tr>
<tr>
<td>Vector ( c )</td>
<td>( 2L^2 - L )</td>
<td>( N^3 )</td>
</tr>
<tr>
<td>Vector ( p )</td>
<td>( N \cdot L )</td>
<td>( N(N + L - 1) )</td>
</tr>
<tr>
<td>Vector ( y' )</td>
<td>( N(2L - 1) )</td>
<td>( N^2 )</td>
</tr>
<tr>
<td>Vector ( C \cdot (k)x )</td>
<td>( N(4L - 3) - 4L^2 + 6L - 2 )</td>
<td>( N^2 )</td>
</tr>
</tbody>
</table>

the point of view of number of numerical operations (multiplications and additions), and thus the execution time, the multiplication of the matrix \( C \) with the vector \( (k)x_1 \) (see denominator in (46)) is the most critical. This operation must be carried out in each iteration step.

However, from the above given example it is worth noticing that the matrix \( C \) is composed of shifted vector \( C \) and contains zero elements. By removing redundant multiplications with zeros in the matrix \( C \), the operation \( C \cdot (k)x_1 \) can be substantially sped up. The optimal algorithm of the operation

\[
z_i = C \cdot (k)x_1,
\]

where \( k \) is iteration step, can be expressed

\[
z_i = \sum_{j=-\min(2L-2)}^{\min(N-1-i, 2L-2)} c_{i+j} \cdot (k)x_1(i+j), \quad i \in \{0, N - 1\}.
\]

Table 1 presents the number of needed multiplications for the optimized and nonoptimized algorithms to carry out one-dimensional Gold deconvolution. Obviously the optimized algorithm substantially speeds up the calculation of Gold deconvolution.

Let us go ahead in these considerations for two-dimensional Gold deconvolution. Without loss of generality we shall suppose that the length of response in both directions is smaller than the size of array \( X_{N_1, N_2} \). Again let us illustrate it using a small example for \( L_1 = 3, L_2 = 3, N_1 = 7, N_2 = 7 \). Then expressing \( x, y \) using (35)–(37), analogously to (48), one can write the matrix \( H \)

\[
H = \begin{bmatrix}
H_0 & 0 & 0 & 0 & 0 & 0 & 0 \\
H_1 & H_0 & 0 & 0 & 0 & 0 & 0 \\
H_2 & H_1 & H_0 & 0 & 0 & 0 & 0 \\
0 & H_2 & H_1 & H_0 & 0 & 0 & 0 \\
0 & 0 & H_2 & H_1 & H_0 & 0 & 0 \\
0 & 0 & 0 & H_2 & H_1 & H_0 & 0 \\
0 & 0 & 0 & 0 & H_2 & H_1 & H_0 \\
0 & 0 & 0 & 0 & 0 & H_2 & H_1 \\
0 & 0 & 0 & 0 & 0 & 0 & H_2
\end{bmatrix},
\]
where

\[
H_i = \begin{bmatrix}
h_2(0,i) & 0 & 0 & 0 & 0 & 0 & 0 \\
h_2(1,i) & h_2(0,i) & 0 & 0 & 0 & 0 & 0 \\
h_2(2,i) & h_2(1,i) & h_2(0,i) & 0 & 0 & 0 & 0 \\
0 & h_2(2,i) & h_2(1,i) & h_2(0,i) & 0 & 0 & 0 \\
0 & 0 & h_2(2,i) & h_2(1,i) & h_2(0,i) & 0 & 0 \\
0 & 0 & 0 & h_2(2,i) & h_2(1,i) & h_2(0,i) & 0 \\
0 & 0 & 0 & 0 & h_2(2,i) & h_2(1,i) & h_2(0,i)
\end{bmatrix}, \quad i = 0, 1, 2. \tag{64}
\]

This time the matrix \(B\) is

\[
B = H^T \cdot H = \begin{bmatrix}
B_0 & B_1 & B_2 & 0 & 0 & 0 \\
B_1^T & B_0 & B_1 & B_2 & 0 & 0 \\
B_2^T & B_1^T & B_0 & B_1 & B_2 \\
0 & B_2^T & B_1^T & B_0 & B_1 & B_2 \\
0 & 0 & B_2^T & B_1^T & B_0 & B_1 \\
0 & 0 & 0 & B_2^T & B_1^T & B_0
\end{bmatrix}, \tag{65}
\]

where

\[
B_0 = H_0^T H_0 + H_1^T H_1 + H_2^T H_2,
\]

\[
B_1 = H_1^T H_0 + H_2^T H_1,
\]

\[
B_2 = H_2^T H_0
\]

and 0 represent submatrices of the size 7 \(\times\) 7. The matrix \(B\) has the size 49 \(\times\) 49. The submatrices \(B_0, B_1, B_2\) are 5-diagonal (in this case nonsymmetrical) matrices

\[
B_i = \begin{bmatrix}
b_{0,i} & b_{1,i} & b_{2,i} & 0 & 0 & 0 & 0 \\
b_{-1,i} & b_{0,i} & b_{1,i} & b_{2,i} & 0 & 0 & 0 \\
b_{-2,i} & b_{-1,i} & b_{0,i} & b_{1,i} & b_{2,i} & 0 & 0 \\
0 & b_{-2,i} & b_{-1,i} & b_{0,i} & b_{1,i} & b_{2,i} & 0 \\
0 & 0 & b_{-2,i} & b_{-1,i} & b_{0,i} & b_{1,i} & b_{2,i} \\
0 & 0 & 0 & b_{-2,i} & b_{-1,i} & b_{0,i} & b_{1,i} \\
0 & 0 & 0 & 0 & b_{-2,i} & b_{-1,i} & b_{0,i}
\end{bmatrix}, \quad i = 0, 1, 2. \tag{67}
\]
The storage of submatrices $B_i$ are in general not symmetrical. The storage of the whole matrix $B$ can be accomplished in its dense form in a matrix $B'$ with the size $L_2 \cdot (2L_1 - 1)$. It is not necessary to store zeros. Then the element of $i_1, i_2$ of the matrix $B_1$ is

$$b_{i_1,i_2} = \sum_{j_2=0}^{L_2 - i_2 - 1} U_{i_1} \sum_{j_1=i_1}^{L_1 - 1} h(j_1,j_2)h(i_1 + j_1, i_2 + j_2),$$

(68)

where $i_1 \in (-L_1 + 1, L_1 - 1)$, $i_2 \in (0, L_2 - 1)$ and $l_1 = \max(0, i_1)$, $U_1 = \min(L_1 - i_1 - 1, L_1 - 1)$.

Now we shall draw an analogy with one-dimensional deconvolution. We define the matrix $C$

$$C = B^T \cdot B = B \cdot B = \begin{bmatrix} D_{00} & D_{01} & C_2 & C_3 & C_4 & 0 & 0 \\ D_{01}^T & D_{11} & C_1 & C_2 & C_3 & C_4 & 0 \\ C_2^T & C_1^T & C_0 & C_1 & C_2 & C_3 & 0 \\ C_3^T & C_2^T & C_1^T & C_0 & C_1 & C_2 & 0 \\ C_4^T & C_3^T & C_2^T & C_1^T & D_{11}' & D_{01}' \\ 0 & C_4^T & C_3^T & C_2^T & D_{01}' & D_{00}' \end{bmatrix}. \quad (69)$$

The submatrices $C, D$ of the size $7 \times 7$ are

$$C_0 = B_2^T B_2 + B_1^T B_1 + B_0 B_0 + B_1 B_1^T + B_2 B_2^T,$$

$$C_1 = B_1^T B_2 + B_0 B_1 + B_1 B_0 + B_2 B_1^T,$$

$$C_2 = B_0 B_2 + B_1 B_1 + B_2 B_0,$$

$$C_3 = B_1 B_2 + B_2 B_1,$$

$$C_4 = B_2 B_2,$$

$$D_{00} = B_0 B_0 + B_1 B_1^T + B_2 B_2^T = C_0 - B_1^T B_1 - B_2^T B_2,$$

$$D_{01} = B_0 B_1 + B_1 B_0 + B_2 B_1^T = C_1 - B_1^T B_2,$$

$$D_{11} = B_1^T B_1 + B_0 B_0 + B_1 B_1^T + B_2 B_2^T = C_0 - B_2^T B_2 \quad (70)$$

and

$$D_{i,j}'(l,k) = D_{i,j}(N_1 - 1 - k, N_1 - 1 - l), \quad (71)$$

where $k, l \in (0, N_1 - 2)$ and $i, j \in (0, L_2 - 2)$. If we extend the system of linear equations in both directions analogously to one-dimensional case then we do not need to take care about submatrices
D. It simplifies substantially the whole algorithm. Again the matrix \( C \) can be stored in its dense form in a matrix \( C' \) defined (size \((2L_2 - 1) \cdot (4L_1 - 3)\))

\[
C_{i_1,i_2} = \sum_{j_2=-L_2+i_2+1}^{L_2-1} \sum_{j_1=L_1}^{L_1} b_{j_1,j_2} \cdot b_{j_1-i_1,j_2-i_2},
\]

(72)

where \( i_1 \in (-2L_1 + 2, 2L_1 - 2) \), \( i_2 \in (0, 2L_2 - 2) \) and \( l_1 = \max(-L_1 + i_1 + 1, -L_1 + 1) \), \( U_1 = \min(L_1 - 1, L_1 + i_1 - 1) \). In this case the \( C \) is 9-diagonal matrix consisting of submatrices \( C_i \), \( D_j \), 0. Each of the submatrices \( C_i \), \( D_j \) is 9-diagonal matrix. In general case the \( C \) is \((4L_2 - 3)\)-diagonal matrix of submatrices from which each is \((4L_1 - 3)\)-diagonal matrix, i.e., Analogously to one-dimensional deconvolution (see (58), (59)) we calculate the matrix \( p' \) with elements

\[
p_{i_1,i_2} = \sum_{j_2=0}^{L_2-1} \sum_{j_1=0}^{L_1-1} h_2(j_1,j_2)y_2(i_1 + j_1, i_2 + j_2),
\]

(73)

where \( i_1 \in (-L_1 + 1, N_1 + L_1 - 2) \), \( i_2 \in (-L_2 + 1, N_2 + L_2 - 2) \), where we suppose \( y_2(k_1,k_2) = 0 \) if \( k_1 \) or \( k_2 < 0 \) or \( k_1 \geq N_1 \) or \( k_2 \geq N_2 \). Subsequently we calculate matrix \( y' \)

\[
y'(i_1,i_2) = \sum_{j_2=-L_2+1}^{L_2-1} \sum_{j_1=-L_1+1}^{L_1-1} b_{j_1,j_2}p_{i_1+j_1,i_2+j_2},
\]

(74)

where \( i_1 \in (0, N_1 - 1) \), \( i_2 \in (0, N_2 - 1) \).

Again, from the point of view of time, the multiplication of the square matrix \( C \) of the size \( N_2N_1 \cdot N_2N_1 \) with the vector \((k)x_2^{(2)}\) (see (46)) is the most critical. From (69) it is apparent that on the one hand the matrix \( C \) contains zero submatrices and on the other hand each of the submatrices \( C_i \) contains zero elements. The multiplications with zeros are redundant and therefore can be omitted. Consequently the optimal algorithm of multiplication matrix \( C \) with the vector of particular solution \((k)x_2^{(2)}\)

\[
z = C \cdot (k)x_2^{(2)}
\]

(75)

can be expressed

\[
Z_{i_1,i_2} = \sum_{j_2=-\min(i_2,2L_2-2)}^{\min(N_2-1-i_2,2L_2-2)-\min(N_1-1-i_1,2L_1-2)} \sum_{j_1=-\min(i_1,2L_1-2)}^{\min(N_1-1-i_1,2L_1-2)-\min(i_1,2L_1-2)} (k)x_2(i_1 + j_1, i_2 + j_2),
\]

(76)

where \( k \) is iteration step, \( i_1 \in (0, N_1 - 1) \), \( i_2 \in (0, N_2 - 1) \).

The memory requirements are given in the Table 2. Again, in analogy with one-dimensional deconvolution we have analyzed the number of needed multiplications (Table 3) to calculate appropriate matrices for both optimized and nonoptimized algorithms.

The algorithm of two-dimensional Gold can be directly extended to \( n \)-dimensional case. Let us suppose we want to compute \( n \)-dimensional input signal with the lengths \( N_1, N_2, \ldots, N_n \). We know \( n \)-dimensional impulse response with the lengths \( L_1, L_2, \ldots, L_n \), where \( L_1 \ll N_1 \), \( L_2 \ll N_2, \ldots L_n \ll N_n \). We know also output signal of (32) \( y_n(i_1,i_2,\ldots,i_n) \). Then the Gold algorithm of \( n \)-dimensional
deconvolution is as follows.

- Calculate array $B'$:
  \[
  b_{i_1,i_2,\ldots,i_n}’ = \sum_{j_n=0}^{L_n-1} \sum_{j_{n-1}=i_n+1}^{U_n-1} \cdots \sum_{j_1=1}^{U_1} h(j_1,j_2,\ldots,j_n) h(i_1+j_1,i_2+j_2,\ldots,i_n+j_n),
  \]
  where $i_k \in \langle -L_k + 1, L_k - 1 \rangle$, $k \in \langle 1, n - 1 \rangle$, $i_n \in \langle 0, L_n - 1 \rangle$, and $l_k = \max(0, i_k)$, $U_k = \min(L_k - i_k, L_k - 1)$, $k \in \langle 1, n - 1 \rangle$.

- Calculate array $C'$:
  \[
  c_{i_1,i_2,\ldots,i_n}’ = \sum_{j_n=-L_n+i_n}^{L_n-i_n-1} \sum_{j_{n-1}=i_n+1}^{U_n-1} \cdots \sum_{j_1=1}^{U_1} b_{j_1,j_2,\ldots,j_n} \cdot b_{j_1-i_1,j_2-i_2,\ldots,j_n-i_n},
  \]
  where $i_k \in \langle -2L_k + 2, 2L_k - 2 \rangle$, $k \in \langle 1, n - 1 \rangle$, $i_n \in \langle 0, 2L_n - 2 \rangle$, and $l_k = \max(-L_k + i_k + 1, -L_k + 1)$, $U_k = \min(L_k - 1, L_k + i_k - 1)$, $k \in \langle 1, n - 1 \rangle$.

- Calculate array $P'$:
  \[
  p_{i_1,i_2,\ldots,i_n}’ = \sum_{j_n=0}^{L_n-1} \sum_{j_{n-1}=0}^{L_{n-1}-1} \cdots \sum_{j_1=0}^{L_1-1} h_n(j_1,j_2,\ldots,j_n) y_n(i_1+j_1,i_2+j_2,\ldots,i_n+j_n),
  \]
  where $i_k \in \langle -L_k + 1, N_k + L_k - 2 \rangle$, $k \in \langle 1, n \rangle$. We suppose that $y_n(l_1,l_2,\ldots,l_n) = 0$ if any of $l_k < 0$ or any of $l_k \geq N_k$, $k \in \langle 1, N \rangle$.
The arrays $C$ to (46) the correction of array redundant multiplications from the calculation of the denominator of (46) can be optimized by employing the following algorithm. It removes where $l_k$.

Number of multiplications for optimized and nonoptimized $n$-dimensional Gold deconvolution are given in Table 4.

The number of multiplications can be substantially decreased. This is important mainly due to finite length of impulse response in all dimensions the number of multiplications can be substantially decreased. This is important mainly in the iteration step $X$.

Comparison of computational complexity for optimized and nonoptimized $n$-dimensional Gold deconvolution is given in Table 5.

From Tables 1, 3, 5 one can observe that due to finite length of impulse response in all dimensions the number of multiplications can be substantially decreased.

### Table 4

<table>
<thead>
<tr>
<th>Array</th>
<th>Memory requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B'$</td>
<td>$L_n \cdot \prod_{k=1}^{n-1}(2L_k - 1)$</td>
</tr>
<tr>
<td>$C'$</td>
<td>$(2L_N - 1) \cdot \prod_{k=1}^{n-1}(4L_k - 3)$</td>
</tr>
<tr>
<td>$P'$</td>
<td>$\prod_{k=1}^{n-1}(N_k + 2L_k - 2)$</td>
</tr>
<tr>
<td>$y'$</td>
<td>$\prod_{k=1}^{n-1}N_k$</td>
</tr>
<tr>
<td>$Z$</td>
<td>$\prod_{k=1}^{n-1}N_k$</td>
</tr>
</tbody>
</table>

### Table 5

Number of multiplications for optimized and nonoptimized $n$-dimensional Gold deconvolution

<table>
<thead>
<tr>
<th>Array</th>
<th>Optimized</th>
<th>Nonoptimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B'$</td>
<td>$(L^2_n + L_n)/2 \prod_{k=1}^{n-1}L_k^2$</td>
<td>$\prod_{k=1}^{n}N_k^3(N_k + L_k - 1)$</td>
</tr>
<tr>
<td>$C'$</td>
<td>$(2L^2_n - L_n) \prod_{k=1}^{n-1}(4L^2_k - 4L_k + 1)$</td>
<td>$\prod_{k=1}^{n}N_k^3$</td>
</tr>
<tr>
<td>$P'$</td>
<td>$\prod_{k=1}^{n}N_k L_k$</td>
<td>$\prod_{k=1}^{n}N_k(N_k + l_k - 1)$</td>
</tr>
<tr>
<td>$y'$</td>
<td>$\prod_{k=1}^{n}N_k(2L_k - 1)$</td>
<td>$\prod_{k=1}^{n}N_k^2$</td>
</tr>
<tr>
<td>$Z$</td>
<td>$\prod_{k=1}^{n}(N_k(4L^2_k - 3) - 4L^2_k + 6L_k - 2)$</td>
<td>$\prod_{k=1}^{n}N_k^2$</td>
</tr>
</tbody>
</table>

- Calculate array $y'$:
  \[
y'_{i_1,j_2,\ldots,j_n} = \sum_{j_a=-L_a+1}^{U_a} \sum_{j_{a-1}=-L_{a-1}+1}^{U_{a-1}} \cdots \sum_{j_1=-L_1+1}^{U_1}  b_{j_1,j_2,\ldots,j_n} \cdot P_{i_1+j_1,i_2+j_2,\ldots,i_n+j_n}, \quad (80)
\]
  where $i_k \in \{0,N_k-1\}$, $k \in \{1,n\}$.

The arrays $C'$ and $y'$ are calculated only once before the iterations of the Gold deconvolution start. The array $y'$ can be immediately used in the calculation of the nominator of (46). The calculation of the denominator of (46) can be optimized by employing the following algorithm. It removes redundant multiplications from the calculation

\[
z_{i_1,j_2,\ldots,j_n} = \sum_{j_n=0}^{U_n} \sum_{j_{n-1}=0}^{U_{n-1}} \cdots \sum_{j_1=0}^{U_1} c_{j_1,j_2,\ldots,j_n} \cdot (k)x(i_1 + j_1,i_2 + j_2,\ldots,i_n + j_n), \quad (81)
\]

where $l_k = \min(i_k,2L_k - 2)$, $U_k = \min(N_k - i_k,2L_k - 2)$, $i_k \in \{0,N_k-1\}$, $k \in \{1,n\}$. Then according to (46) the correction of array $X$ in the iteration step $k + 1$ can be expressed

\[
(k+1)x(i_1,i_2,\ldots,i_n) = \frac{y'(i_1,i_2,\ldots,i_n) \cdot (k)x(i_1,i_2,\ldots,i_n)}{z_{i_1,i_2,\ldots,i_n}}, \quad (82)
\]

where $(0)x(i_1,i_2,\ldots,i_n) = 1$. The memory requirements to store arrays during the process of $n$-dimensional Gold deconvolution are given in Table 4.
in the calculation of (81) which is repeated as many times as number of iterations. Another optimization can be also achieved according to the following fact. If in (82) for a point \(i_1, i_2, \ldots, i_n\) in iteration step \(k, (k)_{x_{i_1, i_2, \ldots, i_n}}\) is zero or sufficiently close to zero then for all following iteration steps \(j > k, (j)_{x_{i_1, i_2, \ldots, i_j}}\) remains zero. It means that for such a point one can stop corrections. This allows to reduce dramatically the computing time of (82).

5. Conclusions

The paper presents nonoscillating Gold deconvolution method, which proved from all tested methods to work as the best one. We have generalized the Gold deconvolution algorithm to \(n\)-dimensional case. The direct extension of one-dimensional Gold deconvolution algorithm to two- , three- and multidimensional systems leads to exponential growth of needed memory space as well as number of numerical operations.

In the paper we have derived optimized multidimensional Gold deconvolution algorithm. The algorithm is optimal from both memory space and numerical point of view. It benefits from the fact that the impulse response has only limited, relatively small number of discrete points with nonzero values. The speed and memory requirements of both optimized and nonoptimized algorithms are in detail analyzed for one- , two- and \(n\)-dimensional cases. From the analysis presented it is clear that without optimization the deconvolution of multidimensional data would not be realizable.

References