Supplementary Material

Subspace-constrained deconvolution of auditory evoked potentials

Angel de la Torre[1], Joaquin T. Valderrama[2,3], Jose C. Segura[1], Isaac M. Alvarez[1], and Jesús García-Miranda[4]

[1]*Department of Signal Theory, Telematics and Communications, University of Granada, Granada, Spain.*
[2]*National Acoustic Laboratories, Sydney, Australia.*
[3]*Department of Linguistics, Macquarie University, Sydney, Australia.*
[3]*Department of Algebra, University of Granada, Granada, Spain.*
E-mail: atv@ugr.es (A. de la Torre); joaquin.valderrama@nal.gov.au (J.T. Valderrama); segura@ugr.es (J.C. Segura); isamaru@ugr.es (I.M. Alvarez); jesusgm@ugr.es (J. García-Miranda)

Contents

1 Derivation of the LS solution for an over-determined system of linear equations

This derivation could be useful for those readers not familiar with linear regression, multivariate linear regression, approximate methods for solving over-determined linear systems of equations, or least squares methods.

Let us consider the matrix equation:

$$\mathbf{y} = A\mathbf{x} \tag{1}$$

where $\mathbf{y}$ is a vector of N observations, $\mathbf{x}$ is a vector of J unknowns, and A is a $(N \times J)$ matrix of coefficients. This matrix equation is equivalent to a system of linear equations:

$$\begin{aligned}
y_0 &= a(0,0)x_0 + a(0,1)x_1 + \ldots + a(0,J-1)x_{J-1} \\
y_1 &= a(1,0)x_0 + a(1,1)x_1 + \ldots + a(1,J-1)x_{J-1} \\
y_2 &= a(2,0)x_0 + a(2,1)x_1 + \ldots + a(2,J-1)x_{J-1} \\
y_3 &= a(3,0)x_0 + a(3,1)x_1 + \ldots + a(3,J-1)x_{J-1} \\
&\ldots \\
y_{N-1} &= a(N-1,0)x_0 + a(N-1,1)x_1 + \ldots + a(N-1,J-1)x_{J-1} \qquad (2) \\
& \qquad (3)
\end{aligned}$$

Assuming that $N > J$, this system of linear equations is in general over-determined and has no solution (i.e. there is no solution satisfying simultaneously all the equations). To illustrate the situation, let us consider a system with two unknowns, x_0 and x_1. The possible values of the unknowns define a two dimension space, i.e. a plane. Each of the equations in the linear system, with the form:

$$y_n = a(n,0)x_0 + a(n,1)x_1 \tag{4}$$

defines a straight line in the plane (x_0, x_1). If there are only two equations, then there are only two lines in the plane and there is a unique solution (if the lines are not parallel). But in general, if we have more than two equations, the linear system has no solution, because there is no point (x_0, x_1) belonging simultaneously to all the lines.

Therefore, in general, we cannot find an exact solution (because it does not exists) but we can find an approximate solution, i.e. a solution closely satisfying all the equations. If closeness is defined in terms of least squares of the differences between the left- and right-hand sides of the equations, then we obtain the least squares (LS) approximation to the solution.

Note that this approximate resolution of the over-determined linear system is equivalent to the resolution of a linear system assuming that the exact solution exists but the equations (or the observations y_n) are affected by noise:

$$\mathbf{y} = A\mathbf{x} + \mathbf{n} \tag{5}$$

or equivalently:

$$\begin{aligned}
y_0 &= a(0,0)x_0 + a(0,1)x_1 + \ldots + a(0,J-1)x_{J-1} + n_0 \\
y_1 &= a(1,0)x_0 + a(1,1)x_1 + \ldots + a(1,J-1)x_{J-1} + n_1 \\
y_2 &= a(2,0)x_0 + a(2,1)x_1 + \ldots + a(2,J-1)x_{J-1} + n_2 \\
y_3 &= a(3,0)x_0 + a(3,1)x_1 + \ldots + a(3,J-1)x_{J-1} + n_3 \\
&\ldots \\
y_{N-1} &= a(N-1,0)x_0 + a(N-1,1)x_1 + \ldots + a(N-1,J-1)x_{J-1} + n_{N-1} \qquad (6) \\
& \qquad (7)
\end{aligned}$$

where the observations $\mathbf{y}$ and the coefficients A are known, the unknowns to be estimated are the components of $\mathbf{x}$, and the noise affecting the observations can be statistically described (usually through its covariance matrix) but the specific values affecting the observation are also unknown.

The LS solution is obtained by minimizing the squared distance between $\mathbf{y}$ and $A\mathbf{x}$ with respect to $\mathbf{x}$:

$$\hat{\mathbf{x}}_{LS} = \arg\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|^2 = \arg\min_{\mathbf{x}} (\mathbf{y} - A\mathbf{x})^T(\mathbf{y} - A\mathbf{x}) \tag{8}$$

and the minimization can be obtained by canceling the gradient with respect to $\mathbf{x}$:

$$\frac{\partial(\mathbf{y} - A\mathbf{x})^T(\mathbf{y} - A\mathbf{x})}{\partial\mathbf{x}} = 0 \tag{9}$$

where the gradient can be expanded (according to equation (84) of [1]) as:

$$\frac{\partial(\mathbf{y}-A\mathbf{x})^T(\mathbf{y}-A\mathbf{x})}{\partial\mathbf{x}} = -2A^T(\mathbf{y}-A\mathbf{x}) \tag{10}$$

Therefore, the cancellation of the gradient provides the equation:

$$A^T\mathbf{y} = A^T A\mathbf{x} \tag{11}$$

and the well known LS solution is obtained as (see eq. 2.0.4 in [2] or eq. 1.2.5 in [3]):

$$\hat{\mathbf{x}}_{LS} = (A^T A)^{-1} A^T\mathbf{y} \tag{12}$$

which requires that the $(J \times J)$ matrix $(A^T A)$ is non-singular (and can therefore be inverted).

This estimation is commonly found in the literature as the “ordinary least squares” method, in order to distinguish it from the weighted least squares, the generalized least squares or the the non-linear least squares methods (which can be considered as different generalizations of the ordinary least squares method) [3].

The LS solution provided in the last equation is optimal (in the sense that it provides an unbiased and minimum variance estimation of $\mathbf{x}$) under several assumptions: linearity (the model is appropriately described by a system of linear equations); exogeneity (in the deconvolution problem the exogeneity is verified if the noise and the stimulation sequence are uncorrelated, which can usually be assumed); homocedasticity (in the deconvolution problem it implies that the noise is stationary and white, and therefore its covariance matrix is spherical, or equivalently it is a constant multiplied by the $(N \times N)$ identity matrix). This last assumption is not verified in most practical situations, and the LS criterion usually provides a good solution (better as the SNR is higher), but not the optimal solution. There are optimization criteria appropriate for colored stationary noise (for example, the minimum mean square error criterion), even though they are out of the scope of this study.

2 Effect of an inappropriate subspace selection

Subspace-constrained least squares deconvolution provides an optimal least squares deconvolution, better than the non-constrained solution, assumed that the response $\mathbf{x}$ is appropriately represented in the subspace, i.e., that can be written as $\mathbf{x} = V_r^T\mathbf{x}_r$.

In order to understand the importance of the appropriate selection of the subspace, let us consider that V_r (with J_r rows and J columns) is a submatrix of an orthonormal $(J \times J)$ matrix V that can be decomposed as:

$$V = \begin{bmatrix} V_r \\ V_c \end{bmatrix} \tag{13}$$

where V_c is an orthonormal projector representing the orthogonal complement of the subspace given by V_r. If the response $\mathbf{x}$ is not contained in the subspace given by V_r, then its projection over the orthogonal complement $\mathbf{x}_c = V_c\mathbf{x}$ is not null, and $\mathbf{x}$ can be decomposed as:

$$\mathbf{x} = V_r^T\mathbf{x}_r + V_c^T\mathbf{x}_c \tag{14}$$

In this case, the synchronous averaging of the EEG can be written as:

$$\mathbf{z}_0 = S_k\,\mathbf{y} = S_k\,S\,\mathbf{x} + S_k\,\mathbf{n}_0 = R_s\mathbf{x} + \mathbf{n}_A = R_s\left(V_r^T\,\mathbf{x}_r + V_c^T\,\mathbf{x}_c\right) + \mathbf{n}_A \tag{15}$$

and therefore (according to equation (12) of the main article) the subspace-constrained deconvolution is:

$$\hat{\mathbf{x}}_{rLS} = \left(V_r\,R_s\,V_r^T\right)^{-1}\,V_r\,\mathbf{z}_0 = \left(V_r\,R_s\,V_r^T\right)^{-1}\,V_r\,R_s\left(V_r^T\,\mathbf{x}_r + V_c^T\,\mathbf{x}_c\right) + \left(V_r\,R_s\,V_r^T\right)^{-1}\,V_r\,\mathbf{n}_A$$

$$\hat{\mathbf{x}}_{rLS} = \mathbf{x}_r + \left(V_r\,R_s\,V_r^T\right)^{-1}\,V_r\,R_sV_c^T\,\mathbf{x}_c + \mathbf{n}_{rLS} \tag{16}$$

where we can observe two contributions to the error: the first one associated to the component $\mathbf{x}_c$ from the orthogonal complement (out of the subspace) and, the other one associated to the noise. The term associated to $\mathbf{x}_c$, causes a biased estimation of the evoked response, even in the absence of noise (i.e. for high SNR, or for an experiment with a very large number of stimuli K), unless either $(V_rR_sV_c^T)$ or $\mathbf{x}_c$ are null (because $(V_rR_sV_r^T)^{-1}$ is a positive definite matrix). The matrix product $(V_rR_sV_c^T)$ is null if R_s is a block-diagonal matrix with respect to both subspaces. However, since R_s is Toeplitz, symmetric and positive definite, it is hardly block-diagonal unless it is the identity matrix (but if $R_s = I$ then a deconvolution is not necessary, and the response can be directly obtained by synchronous averaging). Therefore, if the response is estimated with a subspace-constrained deconvolution, the response must lie in the subspace defined by the orthonormal projector V_r (i.e. $\mathbf{x}_c$ must be null). Otherwise, the solution is biased (even in absence of noise, or even in the case of infinite number of stimuli). This bias is caused by the transference of energy from the orthogonal complement towards the reduced subspace due to the R_s autocorrelation matrix. The equation (16) provides an estimate of the bias.

3 Noise reduction provided by the subspace-constrained LS estimation

In this section we demonstrate that, under the assumption of linearity, exogeneity and homocedasticity (valid for a deconvolution problem affected by uncorrelated-stationary-white noise), the LS-based deconvolution constrained to the subspace given by V_r provides a better solution than that obtained with a deconvolution performed in the complete representation space. In other words, we demonstrate that the trace of the covariance matrix of the error affecting the solutions is smaller when the deconvolution is constrained to the subspace:

$$\mathrm{tr}(\Sigma_{e_r LS}) \leq \mathrm{tr}(\Sigma_{e_{LS}}) \tag{17}$$

Before this demonstration, we present some previous definitions and demonstrations.

3.1 The orthonormal projector V_r and some properties

Let us assume an original representation space with J dimensions, and a reduced representation space with J_r dimensions (with $J_r < J$), given by an orthonormal transformation V_r. The matrix V_r is a $(J_r \times J)$ matrix, with J_r rows and J columns:

$$V_r = [v_{i,m}] \qquad m \in \{0, \ldots, J-1\} \qquad i \in \{0, \ldots, J_r - 1\} \tag{18}$$

where each row is a unitary vector orthogonal to all the others:

$$\sum_{m=0}^{J-1} v_{i,m} v_{j,m} = \delta_{i,j} \tag{19}$$

being $\delta_{i,j}$ the Kronecker's delta. Equivalently we can write:

$$\sum_{m=0}^{J-1} v_{i,m}^2 = 1 \qquad \sum_{m=0}^{J-1} v_{i,m} v_{j,m} = 0 \quad \text{if } i \neq j \tag{20}$$

The orthonormal projector V_r can be considered one submatrix of a complete orthogonal matrix V:

$$V = \begin{bmatrix} V_r \\ V_c \end{bmatrix} \tag{21}$$

where V_c is an orthonormal projector over the orthogonal complement of the subspace defined by V_r:

$$V_c = [v_{i,m}] \qquad m \in \{0, \ldots, J-1\} \qquad i \in \{J_r, \ldots, J-1\} \tag{22}$$

and again each row is a unitary vector orthogonal to all the others (either to the other rows of V_c or to any row of V_r).

Since the complete matrix V is orthonormal, its transpose is equal to its inverse:

$$V^T = V^{-1} \qquad V\,V^T = V^T\,V = I \tag{23}$$

and the transpose is also orthonormal, and the columns of V are also orthonormal vectors:

$$\sum_{i=0}^{J-1} v_{i,m} v_{i,n} = \delta_{m,n} \tag{24}$$

Finally, since the columns of V are normalized (because both V and V^T are orthonormal) we can write:

$$\sum_{i=0}^{J-1} v_{i,m}^2 = 1 \tag{25}$$

and since the orthonormal projector V_r is a submatrix of V including only the first J_r rows, the norm of the columns of V_r are less than or equal to 1:

$$\sum_{i=0}^{J_r-1} v_{i,m}^2 \leq 1 \qquad \forall m \in \{0, \ldots, J-1\} \tag{26}$$

3.2 Properties of the covariance and autocorrelation matrices

In addition to the orthonormal projector, the matrices involved in the LS deconvolution procedure (formulated either in the original or in the reduced representation space) are the normalized autocorrelation matrix of the stimulation sequence R_s or the covariance matrix of the noise Σ_{n_0} or the matrices resulting from the corresponding operations (inversion, projection over the reduced subspace, synchronous averaging, etc.).

Covariance and autocorrelation matrices are always symmetric and positive semidefinite (i.e. with all the eigenvalues positive or null). For stationary processes these matrices are, in addition, Toeplitz (all the elements in each direct diagonal are identical). Therefore, Σ_{n_0} is symmetric and positive semidefinite (also Toeplitz in the case of stationary noise); $\Sigma_{n_A} = (S_k \Sigma_{n_0} S_k^T)$ is also symmetric and positive semidefinite (since it is the covariance matrix of the process $\mathbf{n}_A = S_k \mathbf{n}_0$), as well as $(V_r \Sigma_{n_A} V_r^T)$, $\Sigma_{e_{LS}}$ and $\Sigma_{e_{rLS}}$. Finally, R_s is symmetric, positive semidefinite and Toeplitz (indeed positive definite, since we assume it is invertible); R_s^{-1} is also symmetric and positive definite; and in general $(V_r R_s V_r^T)$, $(V_r R_s^{-1} V_r^T)$ and their respective inverses are also symmetric and positive definite (and usually not Toeplitz due to the orthonormal transformation V_r).

3.3 Demonstration 1: for all A symmetric and positive definite, $\operatorname{tr}((V_r A V_r^T)^{-1}) \leq \operatorname{tr}(V_r A^{-1} V_r^T)$

Let A be a symmetric and positive definite $(J \times J)$ matrix, and V_r an orthonormal projector described as a $(J_r \times J)$ matrix with $J_r < J$, and let us compare the trace of the projected inverse matrix $\operatorname{tr}(V_r A^{-1} V_r^T)$ and the trace of the inversion of the projected matrix $\operatorname{tr}((V_r A V_r^T)^{-1})$.

Since the traces are rotationally invariant, the original representation space can be chosen, without loss of generality, verifying that A is diagonal (therefore with the eigenvalues as the elements in the principal diagonal, all of them positive since A is assumed to be positive definite):

$$A = [a_{m,n}] \qquad a_{m,m} = \lambda_m > 0 \qquad a_{m,n} = 0 \quad \text{if } m \neq n \tag{27}$$

Let the matrices B, C and D be defined as:

$$B = V_r A V_r^T \qquad C = V_r A^{-1} V_r^T \qquad D = B^{-1} = \left(V_r A V_r^T\right)^{-1} \tag{28}$$

which are $(J_r \times J_r)$ matrices. The elements of B and C are, respectively:

$$B = [b_{i,j}] \qquad b_{i,j} = \sum_{m=0}^{J-1} \lambda_m v_{i,m} v_{j,m} \qquad i,j \in \{0, \ldots, J_r - 1\} \tag{29}$$

$$C = [c_{i,j}] \qquad c_{i,j} = \sum_{m=0}^{J-1} \lambda_m^{-1} v_{i,m} v_{j,m} \qquad i,j \in \{0, \ldots, J_r - 1\} \tag{30}$$

and, as can be noted, the elements of the matrix C can easily be computed because, since A is diagonal, its inverse is also diagonal and its elements are the inverse of the eigenvalues of A.

Again, since the traces are rotationally invariant, the reduced representation space can be chosen, without loss of generality, verifying that B is diagonal (it implies a rotation within the reduced representation space)[1]. For this particular representation of the subspace, since B is diagonal, its inverse D is also diagonal, and the elements of D are:

$$D = [d_{i,j}] \qquad d_{i,i} = b_{i,i}^{-1} \qquad d_{i,j} = 0 \quad \text{if } i \neq j \tag{31}$$

Therefore, the i^{th} element of the diagonal for matrices D and C are, respectively:

$$d_{i,i} = \frac{1}{\sum\limits_{m=0}^{J-1} \lambda_m v_{i,m}^2} \qquad\qquad c_{i,i} = \sum_{m=0}^{J-1} \lambda_m^{-1} v_{i,m}^2 \tag{32}$$

and the ratio between the corresponding elements in the diagonal $c_{i,i}/d_{i,i}$ is:

$$\frac{c_{i,i}}{d_{i,i}} = \left(\sum_{m=0}^{J-1} \lambda_m^{-1} v_{i,m}^2\right) \left(\sum_{m=0}^{J-1} \lambda_m v_{i,m}^2\right) = \sum_m \sum_{m'} \lambda_m^{-1} \lambda_{m'} v_{i,m}^2 v_{i,m'}^2 \tag{33}$$

[1]Note that we could select a particular representation of the subspace where B is diagonal, or a particular representation where C is diagonal, but in general it is not possible satisfying both conditions simultaneously.

$$\frac{c_{i,i}}{d_{i,i}} = \sum_m \frac{\lambda_m}{\lambda_m} v_{i,m}^4 + \sum_m \sum_{m'>m} \left(\frac{\lambda_m}{\lambda_{m'}} + \frac{\lambda_{m'}}{\lambda_m} \right) v_{i,m}^2 v_{i,m'}^2 \tag{34}$$

Since the matrix A is positive definite, all its eigenvalues are positive, and the factor involving the eigenvalues has a lower bound:

$$\frac{\lambda_m}{\lambda_{m'}} + \frac{\lambda_{m'}}{\lambda_m} \geq 2 \tag{35}$$

because:

$$x + \frac{1}{x} \geq 2 \qquad \forall x > 0 \tag{36}$$

and therefore, the ratio between the elements in the diagonal has also a lower limit:

$$\frac{c_{i,i}}{d_{i,i}} \geq \sum_m v_{i,m}^4 + 2 \sum_m \sum_{m'>m} v_{i,m}^2 v_{i,m'}^2 = \left(\sum_m v_{i,m}^2 \right)^2 = (1)^2 = 1 \qquad \Rightarrow \qquad \frac{c_{i,i}}{d_{i,i}} \geq 1 \tag{37}$$

where the last sum is equal to 1 because the rows of V_r are normalized (are unitary vectors).

Therefore, for each element of the diagonal of the matrices D and C we can write:

$$d_{i,i} \leq c_{i,i} \tag{38}$$

and since each element in the diagonal is smaller for matrix D, the inequality also applies for their traces:

$$\mathrm{tr}(D) \leq \mathrm{tr}(C) \tag{39}$$

and taking into account the definitions of C and D, we finally demonstrate the relationship between the traces of $(V_r A V_r^T)^{-1}$ and $(V_r A^{-1} V_r^T)$:

$$\mathrm{tr}((V_r A V_r^T)^{-1}) \leq \mathrm{tr}(V_r A^{-1} V_r^T) \tag{40}$$

The equality of the traces requires that:

$$\frac{\lambda_m}{\lambda_{m'}} + \frac{\lambda_{m'}}{\lambda_m} = 2 \quad \forall\, m, m' \tag{41}$$

and it requires that all the eigenvalues of A are equal, or equivalently, that A is a spherical matrix (i.e. the identity matrix multiplied by a constant). In this case, obviously both traces are equal (because the matrices are identical). In any other case, the trace of the inversion of the projected matrix is strictly smaller.

3.4 Demonstration 2: for all A symmetric and positive definite, $\mathrm{tr}(V_r A V_r^T) < \mathrm{tr}(A)$

Let A be a symmetric and positive definite $(J \times J)$ matrix, and V_r an orthonormal projector described as a $(J_r \times J)$ matrix with $J_r < J$, and let us compare the trace of the matrix $\mathrm{tr}(A)$ and the trace of the projected matrix $\mathrm{tr}(V_r A V_r^T)$.

The orthonormal projector V_r can be considered one submatrix of a complete orthogonal matrix V:

$$V = \begin{bmatrix} V_r \\ V_c \end{bmatrix} \tag{42}$$

and since the traces are rotationally invariant, the trace of A can be decomposed as:

$$\mathrm{tr}(A) = \mathrm{tr}(V\,A\,V^T) = \mathrm{tr}\left(\begin{bmatrix} V_r \\ V_c \end{bmatrix} A \begin{bmatrix} V_r^T \, V_c^T \end{bmatrix} \right) = \mathrm{tr}\left(\begin{bmatrix} V_r\,A\,V_r^T & V_r\,A\,V_c^T \\ V_c\,A\,V_r^T & V_c\,A\,V_c^T \end{bmatrix} \right) = \mathrm{tr}(V_r\,A\,V_r^T) + \mathrm{tr}(V_c\,A\,V_c^T) \tag{43}$$

Finally, since the matrix A is assumed to be positive definite, the traces are positive for both projections, and therefore:

$$\mathrm{tr}(A) = \mathrm{tr}(V_r\,A\,V_r^T) + \mathrm{tr}(V_c\,A\,V_c^T) > \mathrm{tr}(V_r\,A\,V_r^T) \tag{44}$$

or equivalently:

$$\mathrm{tr}(V_r\,A\,V_r^T) < \mathrm{tr}(A) \tag{45}$$

Additionally, if the matrix A is symmetric and positive definite, the matrix A^{-1} is also symmetric and postivie definite, and therefore we can also write:

$$\mathrm{tr}(V_r A^{-1} V_r^T) < \mathrm{tr}(A^{-1}) \tag{46}$$

3.5 Demonstration of the noise reduction provided by the subspace-constrained LS deconvolution

In order to demonstrate that the subspace-constrained deconvolution provides a better solution than the non-constrained deconvolution, we have to demonstrate that the trace of the covariance matrix of the error in the subspace-constrained deconvolution is less than or equal to that in the non-constrained deconvolution:

$$\mathrm{tr}(\Sigma_{e_{rLS}}) \leq \mathrm{tr}(\Sigma_{e_{LS}}) \tag{47}$$

As described in the main article, the covariance matrix of the error affecting the LS estimate of $\mathbf{x}$ (in the original representation space) is given by:

$$\Sigma_{e_{LS}} = R_s^{-1}\, S_k\, \Sigma_{n_0}\, S_k^T\, R_s^{-1} \tag{48}$$

and in the case of the subspace-constrained LS deconvolution, by:

$$\Sigma_{e_{rLS}} = (V_r\, R_s\, V_r^T)^{-1}\, V_r\, S_k\, \Sigma_{n_0}\, S_k^T\, V_r^T (V_r\, R_s\, V_r^T)^{-1} \tag{49}$$

where R_s is the normalized autocorrelation matrix of the stimulation sequence and Σ_{n_0} is the covariance matrix of the noise. If, according to the least squares assumptions, the noise is white and stationary, its covariance matrix is spherical, i.e., it is the identity matrix multiplied by a constant σ_n^2 (the variance of the noise), and we can then write:

$$\Sigma_{e_{LS}} = R_s^{-1}\, S_k\, \sigma_n^2\, I\, S_k^T\, R_s^{-1} = \sigma_n^2\, R_s^{-1}\, S_k\, S_k^T\, R_s^{-1} \tag{50}$$

$$\Sigma_{e_{rLS}} = (V_r\, R_s\, V_r^T)^{-1}\, V_r\, S_k\, \sigma_n^2\, I\, S_k^T\, V_r^T (V_r\, R_s\, V_r^T)^{-1} = \sigma_n^2\, (V_r\, R_s\, V_r^T)^{-1}\, V_r\, S_k\, S_k^T\, V_r^T (V_r\, R_s\, V_r^T)^{-1} \tag{51}$$

and since $S_k S_k^T = R_s/K$ these expressions can be rewritten as:

$$\Sigma_{e_{LS}} = \frac{\sigma_n^2}{K}\, R_s^{-1}\, R_s\, R_s^{-1} = \frac{\sigma_n^2}{K}\, R_s^{-1} \tag{52}$$

$$\Sigma_{e_{rLS}} = \frac{\sigma_n^2}{K}\, (V_r\, R_s\, V_r^T)^{-1}\, V_r\, R_s\, V_r^T (V_r\, R_s\, V_r^T)^{-1} = \frac{\sigma_n^2}{K}\, (V_r\, R_s\, V_r^T)^{-1} \tag{53}$$

According to the equation (40), $\mathrm{tr}((V_r A V_r^T)^{-1}) \leq \mathrm{tr}(V_r A^{-1} V_r^T)$ and therefore, since R_s is symmetric and positive definite, the trace of $\Sigma_{e_{rLS}}$ is bounded:

$$\mathrm{tr}(\Sigma_{e_{rLS}}) = \frac{\sigma_n^2}{K}\, \mathrm{tr}\left((V_r\, R_s\, V_r^T)^{-1}\right) \leq \frac{\sigma_n^2}{K}\, \mathrm{tr}\left(V_r\, R_s^{-1}\, V_r^T\right) \tag{54}$$

Moreover, the equality requires that R_s is a spherical matrix (since it is a normalized autocorrelation matrix, it would imply that R_s is the identity matrix). This will never occur in a deconvolution problem, because if $R_s = I$ then the optimal solution is estimated as the synchronous averaging.

On the other hand, since R_s^{-1} is symmetric and positive definite, and according to the equation (46), $\mathrm{tr}(V_r A^{-1} V_r^T) < \mathrm{tr}(A^{-1})$, the right hand of the previous inequality is also bounded:

$$\frac{\sigma_n^2}{K}\, \mathrm{tr}\left(V_r\, R_s^{-1}\, V_r^T\right) < \frac{\sigma_n^2}{K}\, \mathrm{tr}\left(R_s^{-1}\right) = \mathrm{tr}(\Sigma_{e_{LS}}) \tag{55}$$

Finally, taking into account the inequalities in equations (54) and (55), we can write:

$$\mathrm{tr}(\Sigma_{e_{rLS}}) < \mathrm{tr}(\Sigma_{e_{LS}}) \tag{56}$$

being therefore demonstrated that the subspace-constrained least squares deconvolution reduces the error in the estimation of the evoked response.

4 Comparison of subspace-constrained deconvolution vs. LDFDS after deconvolution

The non-constrained LS deconvolution is given by:

$$\hat{\mathbf{x}}_{LS} = R_s^{-1}\, S_k\, \mathbf{y} = \mathbf{x} + R_s^{-1}\, S_k\, \mathbf{n}_0 = \mathbf{x} + R_s^{-1}\, \mathbf{n}_A \tag{57}$$

and the covariance matrix of the error affecting the LS estimation is:

$$\Sigma_{e_{LS}} = R_s^{-1}\, S_k\, \Sigma_{n_0}\, S_k^T\, R_s^{-1} = R_s^{-1}\, \Sigma_{n_A}\, R_s^{-1} \tag{58}$$

Similarly, the subspace-constrained LS deconvolution is given by:

$$\hat{\mathbf{x}}_{rLS} = \left(V_r\, R_s\, V_r^T\right)^{-1}\, V_r\, S_k\, \mathbf{y} = \mathbf{x}_r + (V_r\, R_s\, V_r^T)^{-1}\, V_r\, S_k\, \mathbf{n}_0 = \mathbf{x}_r + (V_r\, R_s\, V_r^T)^{-1}\, V_r\, \mathbf{n}_A \tag{59}$$

and the covariance matrix of the error affecting the subspace-constrained LS estimation is:

$$\Sigma_{e_{rLS}} = (V_r\, R_s\, V_r^T)^{-1}\, V_r\, S_k\, \Sigma_{n_0}\, S_k^T\, V_r^T (V_r\, R_s\, V_r^T)^{-1} = (V_r\, R_s\, V_r^T)^{-1}\, V_r\, \Sigma_{n_A}\, V_r^T (V_r\, R_s\, V_r^T)^{-1} \tag{60}$$

Finally, if we apply the latency-dependent filtering and down-sampling (LDFDS), represented by the V_r matrix operator, to the non-constrained LS solution, the resulting estimation is:

$$\left(\hat{\mathbf{x}}_{LS}\right)_r = V_r\, \hat{\mathbf{x}}_{LS} = V_r\, R_s^{-1}\, S_k\, \mathbf{y} = V_r\, \mathbf{x} + V_r\, R_s^{-1}\, S_k\, \mathbf{n}_0 = \mathbf{x}_r + V_r\, R_s^{-1}\, \mathbf{n}_A \tag{61}$$

and the covariance matrix of the error affecting this estimation is:

$$\Sigma_{(e_{LS})_r} = V_r R_s^{-1}\, S_k\, \Sigma_{n_0}\, S_k^T\, R_s^{-1} V_r^T = V_r R_s^{-1} \Sigma_{n_A} R_s^{-1} V_r^T \tag{62}$$

and in order to demonstrate that the subspace constrained deconvolution provides a better solution than the application of LDFDS after the deconvolution, we must demonstrate that the trace of the former is smaller or equal:

$$\operatorname{tr}(\Sigma_{e_{rLS}}) \leq \operatorname{tr}(\Sigma_{(e_{LS})_r}) \tag{63}$$

or, equivalently:

$$\operatorname{tr}\left((V_r\, R_s\, V_r^T)^{-1}\, V_r\, S_k\, \Sigma_{n_0}\, S_k^T\, V_r^T (V_r\, R_s\, V_r^T)^{-1}\right) \leq \operatorname{tr}\left(V_r R_s^{-1}\, S_k\, \Sigma_{n_0}\, S_k^T\, R_s^{-1} V_r^T\right) \tag{64}$$

4.1 Comparison of the traces for white noise

If, according to the least squares assumptions, the noise is white and stationary, its covariance matrix is spherical and can be written as $\Sigma_{n_0} = \sigma_n^2 I$. In this case, the covariance matrix for the subspace-constrained solution is:

$$\Sigma_{e_{rLS}} = \sigma_n^2\, (V_r\, R_s\, V_r^T)^{-1}\, V_r\, S_k\, I\, S_k^T\, V_r^T (V_r\, R_s\, V_r^T)^{-1} = \frac{\sigma_n^2}{K}\, (V_r\, R_s\, V_r^T)^{-1}\, (V_r\, R_s\, V_r^T)(V_r\, R_s\, V_r^T)^{-1}$$

$$\Sigma_{e_{rLS}} = \frac{\sigma_n^2}{K}\, (V_r\, R_s\, V_r^T)^{-1} \tag{65}$$

and the covariance matrix for the LDFDS applied after the deconvolution is:

$$\Sigma_{(e_{LS})_r} = \sigma_n^2\, V_r\, R_s^{-1}\, S_k\, I\, S_k^T\, R_s^{-1}\, V_r^T = \frac{\sigma_n^2}{K}\, V_r\, R_s^{-1}\, R_s\, R_s^{-1}\, V_r^T$$

$$\Sigma_{(e_{LS})_r} = \frac{\sigma_n^2}{K}\, V_r R_s^{-1}\, V_r^T \tag{66}$$

According to the equation (40), $\operatorname{tr}((V_r A V_r^T)^{-1}) \leq \operatorname{tr}(V_r A^{-1} V_r^T)$ for whatever A symmetric positive definite, and therefore, since R_s is symmetric and positive definite, the trace of $\Sigma_{e_{rLS}}$ is smaller:

$$\operatorname{tr}(\Sigma_{e_{rLS}}) = \frac{\sigma_n^2}{K}\, \operatorname{tr}\left((V_r\, R_s\, V_r^T)^{-1}\right) \leq \frac{\sigma_n^2}{K}\, \operatorname{tr}\left(V_r\, R_s^{-1}\, V_r^T\right) = \operatorname{tr}(\Sigma_{(e_{LS})_r}) \tag{67}$$

and indeed, the inequality is strict unless $R_s = I$. Therefore, as expected, for white noise (and if all the other assumptions for the LS criterion are verified) the subspace-constrained deconvolution is optimal and this solution is better than that obtained by applying LDFDS after the deconvolution.

4.2 Comparison of the traces for non-constrained noise

The case of a noise with a non-constrained covariance matrix is out of the least squares assumptions. However, some analysis of this situation is interesting.

If the error covariance matrices are expressed as a function of Σ_{n_A}:

$$\Sigma_{e_{rLS}} = (V_r\, R_s\, V_r^T)^{-1}\, V_r\, \Sigma_{n_A}\, V_r^T (V_r\, R_s\, V_r^T)^{-1} \qquad\qquad \Sigma_{(e_{LS})_r} = V_r R_s^{-1} \Sigma_{n_A} R_s^{-1} V_r^T \tag{68}$$

the study of the traces for a unconstrained noise can be analyzed by considering random $(J \times J)$ symmetric positive definite matrices for R_s and Σ_{n_A} and random $J_r \times J$ orthonormal matrices for V_r, and comparing the traces for a sufficiently large number of repetitions. A Monte Carlo simulation has been prepared to compare the traces for a number of repetitions with randomly generated R_s, Σ_{n_A} and V_r matrices (see below the MatLab/Octave code implementing it), and it has been observed that most of the times the inequality $\mathrm{tr}(\Sigma_{e_{rLS}}) \leq \mathrm{tr}(\Sigma_{(e_{LS})_r})$ is verified, but sometimes it is not. Therefore, the proposed subspace-constrained deconvolution is guaranteed to improve the solution based on LDFDS after deconvolution only for stationary white noise, and even though the former usually improves the result provided by the later, the improvement is not always guaranteed in a general case. This is not surprising, since the least squares criterion assumes white noise (and the optimal solution for non-white noise requires the minimum mean square error criterion instead of the least squares error, which is out of the scope of this study).

4.3 Comparison of the traces when $R_s \approx I$

An interesting particular case (for unconstrained noise) is obtained when $R_s \approx I$. As previously discussed, the autocorrelation matrix is only equal to the identity matrix in the case of responses not overlapping in the evoked potentials experiment (but in this case, the optimal LS solution would be obtained as a synchronous averaging of the EEG). However, the analysis of the traces is interesting in the case of $R_s \approx I$, because in the case of a randomized (and not resonant) stimulation sequence, we usually can assume that R_s is close to the identity matrix.

If we introduce the substitution $R_s = I$ in the error covariance matrices, for the subspace-constrained solution we have:

$$\Sigma_{e_{rLS}} = (V_r\, I\, V_r^T)^{-1}\, V_r\, S_k\, \Sigma_{n_0}\, S_k^T\, V_r^T (V_r\, I\, V_r^T)^{-1} = (V_r\, V_r^T)^{-1}\, V_r\, S_k\, \Sigma_{n_0}\, S_k^T\, V_r^T (V_r\, V_r^T)^{-1} = I\, V_r\, S_k\, \Sigma_{n_0}\, S_k^T\, V_r^T\, I$$

$$\Sigma_{e_{rLS}} = V_r\, S_k\, \Sigma_{n_0}\, S_k^T\, V_r^T \tag{69}$$

and similarly, for the LDFDS applied after the deconvolution we have:

$$\Sigma_{(e_{LS})_r} = V_r\, I\, S_k\, \Sigma_{n_0}\, S_k^T\, I\, V_r^T = V_r\, S_k\, \Sigma_{n_0}\, S_k^T\, V_r^T \tag{70}$$

and, as can be observed, both covariance matrices are, in this case, identical (independently of the noise statistics), and therefore, the traces are equal. For this reason, since usually R_s is close to the identity matrix, one could expect that both solutions are close (i.e. the difference between both solutions is expected to be small compared with the error due to the noise).

4.4 Code implementing a Monte Carlo simulation for comparing $\mathrm{tr}(\Sigma_{e_{rLS}})$ and $\mathrm{tr}(\Sigma_{(e_{LS})_r})$

The following MatLab/Octave code (function "func_trace_SigmaA_SigmaB.m") generates random matrices R_s, Σ_{n_A} and V_r with configurable sizes according to J and J_r, and compares the traces of the error covariance matrices $\mathrm{tr}(\Sigma_{e_{rLS}})$ and $\mathrm{tr}(\Sigma_{(e_{LS})_r})$ for a number of repetitions. In order to allow R_s to be close to a spherical matrix, a configuration parameter "White_K" is included (if the parameter is set to 0, R_s is random; as the parameter increases, R_s is closer to a spherical matrix, i.e. with more homogeneous eigenvalues).

The function can be run with the script "script_trace_SigmaA_SigmaB.m", also provided below, which configures the parameters and call the function for several sizes of the subspace. Figure 1 includes some histograms of the trace ratio generated with this code. As observed, the trace ratio is usually smaller than one, but not always.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% func_trace_SigmaA_SigmaB.m
% MatLab/Octave function for comparing the traces of the covariance matrices SigmaA and SigmaB associated to the
% error for these cases:
%  * Solution A: LS deconvolution and then subspace projection
%  * Solution B: subspace-constrained LS deconvolution
%  SigmaA = V1*inv(Rs) * SigNA * inv(Rs)*V1'
%  SigmaB = inv(V1*Rs*V1') * (V1*SigNA*V1') * inv(V1*Rs*V1')
%   - Rs, SigNA symmetric, positive definite, JxJ matrices
%   - V1 orthonormal projector, JrxJ matrix (with Jr<J) (V' is transpose)
% This function generates random matrices Rs, V1, SigNA according to the configuration, computes SigmaA and SigmaB
% and computes the ratio tr(SigmaB)/tr(SigmaA). The process is repeated acording to the parameter Nrep, and a
% histogram of the trace ratio is provided. The probability P(tr(SigmaB)>tr(SigmaA)) and the trace-ratio distribution
% are provided as output parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Arguments:
%  J and Jr: number of dimensions of the original/reduced repres. space
%  Nrep:     number of repetitions in the simulation
%  White_K:  a matrix White_K*I (identity) is added to the random Rs (as White_K is larger, Rs is closer to spherical).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [p,Tr_ratio]=func_trace_SigmaA_SigmaB(J,Jr,Nrep,White_K)

Tr_ratio=zeros(Nrep,1); % for saving the trace ratio of each case
for n=1:Nrep
    if mod(n,1000)==0, fprintf('n=%d of %d\n',n,Nrep); end;
    % generation of random Rs and SigNA (symmetric and positive definite):
    Rs=func_matrix_covar(J); SigNA=func_matrix_covar(J);
    Rs=Rs+eye(J)*White_K;    % as White_K larger, Rs closer to spherical
    % generation of randon V1, (Jr x J) orthonormal projector to subspace:
    V1=func_matrix_ON(J); V1=V1(1:Jr,:);
    % SigmaA = V1*inv(Rs) * SigNA * inv(Rs)*V1'
    A = V1/Rs;   % efficient implementation of A = V1*inv(Rs);
    SigmaA = A*(SigNA*A');
    % SigmaB = inv(V1*Rs*V1') * (V1*SigNA*V1') * inv(V1*Rs*V1')
    Rsr = V1*(Rs*V1'); SigNAr=V1*(SigNA*V1');  % Rsr = V1*Rs*V1'; SigNAr = V1*SigNA*V1';
    SigmaB=(Rsr\SigNAr)/Rsr; % efficient implementation of SigmaB=inv(B)*SigNAr*inv(B);
    % trace ratio
    ratio = trace(SigmaB)/trace(SigmaA); Tr_ratio(n)=ratio;
end
figure(1); clf; hist(Tr_ratio,40);
xlabel('tr(\Sigma_B)/tr(\Sigma_A)'); ylabel('number of cases'); grid on;
p=sum(Tr_ratio>1)/Nrep; % pci=sqrt(p*(1-p)/Nrep)*1.96; pci: 95% confidence interval of p
result=sprintf('J=%d   Jr=%d   Wh-K=%g   N=%d    p(trB>trA)=%f %%',J,Jr,White_K,Nrep,p*100);
title(result); fprintf('%s\n',result);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function M=func_matrix_covar(J)  % random JxJ sym positive definite matrix
V=func_matrix_ON(J);     % random orthonormal matrix
rs=abs(randn(J,1));      % random positive eigenvalues
M0=diag(rs);             % corresponding diagonal matrix (posit defin)
M=V*(M0*V');             % rotated with V: symm. posit. definite.
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function V=func_matrix_ON(J)  % provides a random JxJ orthonormal matrix V
A0=randn(J); A0=A0+A0';   % random symmetric matrix
[V,~]=eig(A0);            % the eigenvectors are an Orthonormal basis
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% script_trace_SigmaA_SigmaB.m
% MatLab/Octave script using the function: func_trace_SigmaA_SigmaB(J,Jr,Nrep,White_K) to compare the traces of
% SigmaA and SigmaB defined as:
%    SigmaA = V1*inv(Rs) * SigNA * inv(Rs)*V1'
%    SigmaB = inv(V1*Rs*V1') * (V1*SigNA*V1') * inv(V1*Rs*V1')
% for different configurations, using "func_trace_SigmaA_SigmaB()"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
Nrep=10000; White_K=0.8; lim_x=0.6;
for Jr=1:4
    func_trace_SigmaA_SigmaB(8,Jr,Nrep,White_K);
    figure(1); xlim([0 2]); a1=gca;
    f2=figure(Jr+10); clf; copyobj(a1,f2); close(1);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

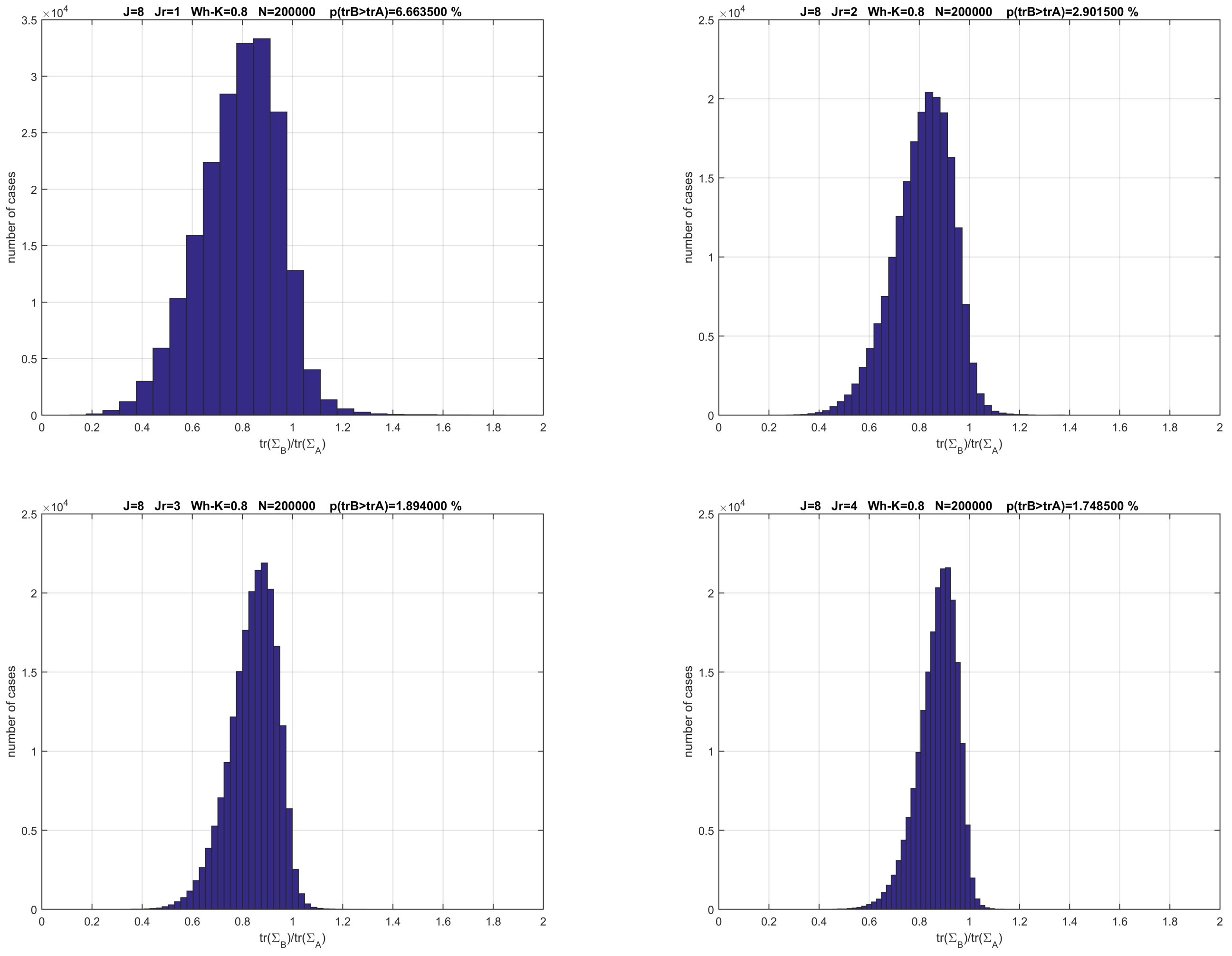


Figure 1: Histograms of the trace ratio $\mathrm{tr}(\Sigma_{e_{rLS}})/\,\mathrm{tr}(\Sigma_{(e_{LS})_r})$ for random R_s, Σ_{n_A} and V_r matrices simulated with the function "func_trace_SigmaA_SigmaB.m". The probability of observing cases where $\mathrm{tr}(\Sigma_{e_{rLS}}) > \mathrm{tr}(\Sigma_{(e_{LS})_r})$ is indicated.

5 Code implementing the LS, LS-R and SC-LS procedures

The three criteria for least squares deconvolution (LS, LS-R and SC-LS) have been implemented as MatLab/Octave functions, either with matrix division (as proposed in the RSLSD algorithm) or with the iterative estimation (as proposed in the IRSA algorithm). Therefore, 6 MatLab/Octave functions have been prepared:

- Function "RSLSD_LS.m" (conventional LS deconvolution implemented with matrix division)
- Function "IRSA_LS.m" (conventional LS deconvolution implemented with iterative estimation)
- Function "RSLSD_LSR.m" (LS deconvolution implemented with matrix division, and LDFDS-based dimensionality reduction)
- Function "IRSA_LSR.m" (LS deconvolution implemented with iterative estimation, and LDFDS-based dimensionality reduction)
- Function "RSLSD_SCLS.m" (subspace-constrained LS deconvolution implemented with matrix division)
- Function "IRSA_SCLS.m" (subspace-constrained LS deconvolution implemented with iterative estimation)

The code is provided in the next sections. The input of these functions are the EEG, the trigger vector (a list of the sample indexes corresponding to the beginning of each event in the evoked potential experiment) and some other parameters depending on the particular implementation (the length of the response J in the case of LS; the V_r transformation in the case of LS-R or SC-LS; the maximum number of iterations and convergence criterion in the case of IRSA estimations). The output of these functions are the estimated response (in the original representation space in the case of LS, in the reduced representation space in the case of LS-R or SC-LS). The functions also provide the execution time in the case of the RSLSD implementations. In the case of the IRSA implementation, in addition, the execution time per iteration, the convergence parameter α and the required number of iterations are also provided. The functions "RSLSD_LS.m" and "IRSA_LS.m" are similar to those presented in [4], but including an optimization in the initialization: some computations are performed with 32-bit fixed-point precision (instead of 64-bit floating-point). This does not affect the accuracy, since the EEG is a digitized signal represented as a 16-bit fixed-point integer (and is strongly affected by the background noise, significantly greater than the quantization noise); additionally, the stimulation signal can be accurately represented with 1-bit fixed-point precision. This optimization saves around 10 seconds in the execution time of the complete test (far all the ISI configurations).

In addition to these functions, the code of the function providing the LDFDS transformation from the original representation to the reduced representation, V_r, is also included (function "Basis_LinLog_RRC.m"). This function was proposed and described in detail in [5].

A MatLab/Octave script providing a demonstration using these functions is also included ("script_simulation_SCLSDec.m"). The script reads a binary file with an AEP response to be used in the simulations. A noisy EEG is synthesized according to the configuration of the script (ISI configuration, number of stimuli, amplitude of the noise, length of the response, etc.) and the deconvolution is performed according to the different methods. The script provides several figures (with the response used as reference, the synthesized noisy EEG and the AEP estimations obtained with the different deconvolution procedures) and some results related to the quality of the AEP estimations and the execution time.

The functions, together with the script, a binary file with the reference AEP response used for the simulations and a "Readme.txt" file are provided within a directory, allowing the readers to perform their simulations according to different configurations. In order to run the simulation, the script must be run with MatLab or Octave in the directory where the functions, the script and the binary file are stored. The script provides the results in the command-line, as well as the output figures.

The results and the figure with the estimated AEPs are reported at the end of this section for one execution of the script. Since the simulation includes the addition of random noise, and since the execution time depends on the computational load of the computer where the simulation is executed, the results provided by the script fluctuate for each simulation.

5.1 Function “RSLSD_LS.m” (conventional LS, with matrix division)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [xi ,t_run] = RSLSD_LS(y,m,J)
% Randomized Stimulation with Least Squares Deconvolution: direct
% deconvolution (infinite iterations) with matrix inversion
%     xi =  Rs^(-1) z0         xi = Rs\z0;
%  Input parameters:  y (Recorded EEG)
%                     m (Trigger vector)
%                     J (Length of the averaging window in samples)
%  Output parameters: xi (AEP estimate)
%                     t_run (time required for algorithm execution)
% Angel de la Torre, Jose Carlos Segura, Joaquin Valderrama 2021
%     University of Granada (Spain)
%     National Acoustic Laboratories, Macquarie University (Australia)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xi ,t_run] = RSLSD_LS(y,m,J)
% Initialization
tic;                 % time-stamp beginning of function
N=length(y);
Es=length(m);        % energy of the stimulation signal (number of stimuli)
z0_int=int32(zeros(J,1)); rs0_int=uint32(zeros(J,1));
s(N,1) = uint8(0); m=cast(m,'uint32'); s(m)=1;  % stimulation signal
gain=1e6/max(abs(y));
y1=int32(y*gain);
for j=1:J
    idx=j+m-1;
    z0_int(j)=sum(y1(idx));  % cross-corr between EEG and stim. signal
    rs0_int(j)=sum(s(idx));  % autocorrelation of stim. signal
end
z0=double(z0_int)/Es/gain;   % first averaged response
rs0=double(rs0_int)/Es;      % normalized autocorrelation stim. signal
Rs=zeros(J,J);
for i=1:J
    j=1:J; idx=abs(j-i)+1;
    Rs(i,j)=rs0(idx); % autocorrelation matrix
end
xi=(Rs\z0);           % direct deconvolution by matrix inversion
t_run=toc;            % total execution time
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.2 Function “IRSA_LS.m” (conventional LS, with iterative estimation)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [xi ,t_run,t_iter,Niter,alpha] = IRSA_LS(y,m,I,SNR,J,OUTPUT)
% IRSA, matrix implementation with matrix product in fft domain
%    (this is possible because Rs is a Toeplitz-symmetric matrix)
%    Fast version: alpha / converg. criterion optimized (120dB num. error)
%  Input parameters:  y (Recorded EEG)
%                     m (Trigger vector)
%                     I (maximum number of iterations)
%                     SNR (SNR for numerical error in convergence criterion)
%                     J (Length of the averaging window in samples)
%                     OUTPUT (flag for presenting results at iterations)
%  Output parameters: xi (AEP estimate)
%                     t_run (time required for algorithm execution)
%                     t_iter (time required for each iteration)
% Angel de la Torre, Jose Carlos Segura, Joaquin Valderrama 2021
%     University of Granada (Spain)
%     National Acoustic Laboratories, Macquarie University (Australia)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xi ,t_run,t_iter,Niter,alpha] = IRSA_LS(y,m,I,SNR,J,OUTPUT)
% Initialization
tic;                 % time-stamp beginning of function
N=length(y);
Es=length(m);        % energy of the stimulation signal (number of stimuli)
z0_int=int32(zeros(J,1)); rs0_int=uint32(zeros(J,1));
s(N,1) = uint8(0); m=cast(m,'uint32'); s(m)=1;  % stimulation signal
gain=1e6/max(abs(y));
y1=int32(y*gain);
for j=1:J
    idx=j+m-1;
    z0_int(j)=sum(y1(idx));  % cross-corr between EEG and stim. signal
    rs0_int(j)=sum(s(idx));  % autocorrelation of stim. signal
end
z0=double(z0_int)/Es/gain;   % first averaged response
rs0=double(rs0_int)/Es;      % normalized autocorrelation stim. signal
RS=real(fft([rs0; 0; flipud(rs0(2:end))]));       % FT of autocorrelation
Z0=fft([z0; zeros(J,1)]); Zi=Z0; Xi=zeros(2*J,1); % FT of z0, zi and xi
lambda_1=max(RS(1:2:J)); lambda_2=max(RS(2:2:J));
max_mu=0.5*(lambda_1+lambda_2); % bound for max eigenvalue of autoc. matrix
alpha=1.9/max_mu;        % selected alpha (close to maximum value)
ener_z0=Z0'*Z0;          % energy of RSA solution used for convergence
thr_conv=10^(-SNR/20);   % threshold for convergence criterion
% Iterations
t_iter=toc;          % time-stamp for iterations
for i=1:I            % loop for iterations
    Xi=Xi+alpha*Zi;    % AEP estimate in freq. domain
    P=RS.*Xi;          % matrix product in freq. domain
    P1=real(ifft(P));  % this two lines are important in order to truncate
    P=fft([P1(1:J); zeros(J,1)]); % the estimation of the P in time domain
    Zi=Z0-P;
    ener_zi=Zi'*Zi;              % energy of the correction at current it.
    ratio=sqrt(ener_zi/ener_z0); % ratio of correction vs initialization
    if ratio<thr_conv, break; end; % convergence criterion (loop broken)
    if OUTPUT==1, fprintf('It.%d: ratio:%.16f alpha=%.5f\n',i,ratio,alpha); end;
end
Niter=i;
xi=real(ifft(Xi));    % the result is transformed to time domain
xi=xi(1:J);           % ...and truncated to remove the non-causal part
t_run=toc;               % total execution time
t_iter=(t_run-t_iter)/i; % execution time for each iteration
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.3 Function “RSLSD_LSR.m” (LS and LDFDS reduction, with matrix division)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [xi ,t_run] = RSLSD_LSR(y,m,V)
% Randomized Stimulation with Least Squares Deconvolution: direct
% deconvolution with matrix inversion
% The AEP is transformed to the reduced representation space given by V
%     xi =  Rs^(-1) z0         xi = Rs\z0;
%  Input parameters:  y (Recorded EEG)
%                     m (Trigger vector)
%                     V orthonormal transform. for reduced representation
%  Output parameters: xi (AEP estimate)
%                     t_run (time required for algorithm execution)
% Angel de la Torre, Jose Carlos Segura, Joaquin Valderrama 2021
%     University of Granada (Spain)
%     National Acoustic Laboratories, Macquarie University (Australia)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xi_ssp ,t_run] = RSLSD_LSR(y,m,V)
% Initialization
tic;                   % time-stamp beginning of function
[~,J]=size(V);
N=length(y);
Es=length(m);          % energy of the stimulation signal (number of stimuli)
z0_int=int32(zeros(J,1)); rs0_int=uint32(zeros(J,1));
s(N,1) = uint8(0); m=cast(m,'uint32'); s(m)=1;  % stimulation signal
gain=1e6/max(abs(y));
y1=int32(y*gain);
for j=1:J
    idx=j+m-1;
    z0_int(j)=sum(y1(idx));  % cross-corr between EEG and stim. signal
    rs0_int(j)=sum(s(idx));  % autocorrelation of stim. signal
end
z0=double(z0_int)/Es/gain;   % first averaged response
rs0=double(rs0_int)/Es;      % normalized autocorrelation stim. signal
Rs=zeros(J,J);
for i=1:J
    j=1:J; idx=abs(j-i)+1;
    Rs(i,j)=rs0(idx); % autocorrelation matrix
end
xi=(Rs\z0);             % direct deconvolution by matrix inversion
xi_ssp=V*xi;            % projection
t_run=toc;              % total execution time
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.4 Function “IRSA_LSR.m” (LS and LDFDS reduction, with iterative estimation)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [xi ,t_run,t_iter,Niter,alpha] = IRSA_LSR(y,m,V,I,SNR,J,OUTPUT)
% IRSA, matrix implementation with matrix product in fft domain
%    (this is possible because Rs is a Toeplitz-symmetric matrix)
%    Fast version: alpha / converg. criterion optimized (120dB num. error)
%    The AEP is transformed to the reduced representation space given by V
%  Input parameters:  y (Recorded EEG)
%                    m (Trigger vector)
%                    V orthonormal transform. for reduced representation
%                    I (maximum number of iterations)
%                    SNR (SNR for numerical error in convergence criterion)
%                    J (Length of the averaging window in samples)
%                    OUTPUT (flag for presenting results at iterations)
%  Output parameters: xi (AEP estimate)
%                    t_run (time required for algorithm execution)
%                    t_iter (time required for each iteration)
% Angel de la Torre, Jose Carlos Segura, Joaquin Valderrama 2021
%     University of Granada (Spain)
%     National Acoustic Laboratories, Macquarie University (Australia)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xi_ssp ,t_run,t_iter,Niter,alpha] = IRSA_LSR(y,m,V,I,SNR,OUTPUT)
% Initialization
tic;                 % time-stamp beginning of function
[~,J]=size(V);
N=length(y);
Es=length(m);        % energy of the stimulation signal (number of stimuli)
z0_int=int32(zeros(J,1)); rs0_int=uint32(zeros(J,1));
s(N,1) = uint8(0); m=cast(m,'uint32'); s(m)=1;  % stimulation signal
gain=1e6/max(abs(y));
y1=int32(y*gain);
for j=1:J
    idx=j+m-1;
    z0_int(j)=sum(y1(idx));  % cross-corr between EEG and stim. signal
    rs0_int(j)=sum(s(idx));  % autocorrelation of stim. signal
end
z0=double(z0_int)/Es/gain;       % first averaged response
rs0=double(rs0_int)/Es;          % normalized autocorrelation stim. signal
RS=real(fft([rs0; 0; flipud(rs0(2:end))]));       % FT of autocorrelation
Z0=fft([z0; zeros(J,1)]); Zi=Z0; Xi=zeros(2*J,1); % FT of z0, zi and xi
lambda_1=max(RS(1:2:J)); lambda_2=max(RS(2:2:J));
max_mu=0.5*(lambda_1+lambda_2); % bound for max eigenvalue of autoc. matrix
alpha=1.9/max_mu;        % selected alpha (close to maximum value)
ener_z0=Z0'*Z0;          % energy of RSA solution used for convergence
thr_conv=10^(-SNR/20);   % threshold for convergence criterion
% Iterations
t_iter=toc;          % time-stamp for iterations
for i=1:I            % loop for iterations
    Xi=Xi+alpha*Zi;    % AEP estimate in freq. domain
    P=RS.*Xi;          % matrix product in freq. domain
    P1=real(ifft(P));  % this two lines are important in order to truncate
    P=fft([P1(1:J); zeros(J,1)]); % the estimation of the P in time domain
    Zi=Z0-P;
    ener_zi=Zi'*Zi;              % energy of the correction at current it.
    ratio=sqrt(ener_zi/ener_z0); % ratio of correction vs initialization
    if ratio<thr_conv, break; end; % convergence criterion (loop broken)
    if OUTPUT==1, fprintf('It.%d: ratio:%.16f alpha=%.5f\n',i,ratio,alpha); end;
end
Niter=i;
xi=real(ifft(Xi));    % the result is transformed to time domain
xi=xi(1:J);           % ...and truncated to remove the non-causal part
xi_ssp=V*xi;          % projection
t_run=toc;               % total execution time
t_iter=(t_run-t_iter)/i; % execution time for each iteration
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.5 Function “RSLSD_SCLS.m” (subspace constrained LS, with matrix division)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [xi_ssp ,t_run] = RSLSD_SCLS(y,m,V)
% Randomized Stimulation with Least Squares Deconvolution
% Subspace Constrained version (fast implementation)
% Direct deconvolution with matrix inversion
%     Rs_ssp = V Rs V'    z0_ssp = V z0
%     xi_ssp =  Rs_ssp^(-1) z0_ssp         xi_ssp = Rs_ssp\z0_ssp;
%  Input parameters:  y (Recorded EEG)
%                     m (Trigger vector)
%                     V (Transformation matrix, Jred x J components)
%  Output parameters: xi_ssp (AEP estimate in the subspace representation)
%                     t_run (time required for algorithm execution)
% Angel de la Torre, Jose Carlos Segura, Joaquin Valderrama 2021
%     University of Granada (Spain)
%     National Acoustic Laboratories, Macquarie University (Australia)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xi_ssp ,t_run] = RSLSD_SCLS(y,m,V)
% Initialization
tic;                  % time-stamp beginning of function
[Jred,J]=size(V);     % dimensions of the representation space and subspace
N=length(y);
Es=length(m);         % energy of the stimulation signal (number of stimuli)
z0_int=int32(zeros(J,1)); rs0_int=uint32(zeros(J,1));
s(N,1) = uint8(0); m=cast(m,'uint32'); s(m)=1;  % stimulation signal
gain=1e6/max(abs(y));
y1=int32(y*gain);
for j=1:J
    idx=j+m-1;
    z0_int(j)=sum(y1(idx));  % cross-corr between EEG and stim. signal
    rs0_int(j)=sum(s(idx));  % autocorrelation of stim. signal
end
z0=double(z0_int)/Es/gain;   % first averaged response
rs0=double(rs0_int)/Es;      % normalized autocorrelation stim. signal
rs0ext=[flipud(rs0(2:end)); rs0]; % extended autocorrelation
% Estimation of the matrix to be inverted Rs_ssp
Aux=zeros(J,Jred);
for i=1:Jred
    a=conv(V(i,:),rs0ext,'same');
    Aux(:,i)=a';
end
Rs_ssp=V*(Aux);
% Subspace constrained deconvolution
z0_ssp=V*z0;
xi_ssp=(Rs_ssp\z0_ssp);
t_run=toc;            % total execution time
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.6 Function “IRSA_SCLS.m” (subspace constrained LS, with iterative estimation)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [xi_ssp ,t_run, t_iter] = IRSA_SCLS(y,m,V,I,SNR,OUTPUT)
% IRSA, matrix implementation
%    Fast version: alpha / converg. criterion optimized (120dB num. error)
%    Subspace Constrained version:
%    Subspace version of Rs and z0 used in the recursion
%     Rs_ssp = V Rs V'    z0_ssp = V z0
%  Input parameters: y (Recorded EEG)
%                    m (Trigger vector)
%                    V (Transformation matrix, Jred x J components)
%                    I (maximum number of iterations)
%                    SNR (SNR for numerical error in convergence criterion)
%                    OUTPUT (flag for presenting results at iterations)
%  Output parameters: xi_ssp (AEP estimate in the subspace representation)
%                     t_run (time required for algorithm execution)
%                     t_iter (time required for each iteration)
% Angel de la Torre, Jose Carlos Segura, Joaquin Valderrama 2021
%     University of Granada (Spain)
%     National Acoustic Laboratories, Macquarie University (Australia)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xi_ssp ,t_run, t_iter, Niter, alpha] = IRSA_SCLS(y,m,V,I,SNR,OUTPUT)
% Initialization
tic;                 % time-stamp beginning of function
[Jred,J]=size(V);    % dimensions of the representation space and subspace
N=length(y);
Es=length(m);        % energy of the stimulation signal (number of stimuli)
z0_int=int32(zeros(J,1)); rs0_int=uint32(zeros(J,1));
s(N,1) = uint8(0); m=cast(m,'uint32'); s(m)=1;  % stimulation signal
gain=1e6/max(abs(y));
y1=int32(y*gain);
for j=1:J
    idx=j+m-1;
    z0_int(j)=sum(y1(idx));  % cross-corr between EEG and stim. signal
    rs0_int(j)=sum(s(idx));  % autocorrelation of stim. signal
end
z0=double(z0_int)/Es/gain;        % first averaged response
rs0=double(rs0_int)/Es;           % normalized autocorrelation stim. signal
rs0ext=[flipud(rs0(2:end)); rs0]; % extended autocorrelation
% estimation of eigenvalues of Rs and convergence parameter alpha
RS=real(fft([rs0; 0; flipud(rs0(2:end))]));       % FT of autocorrelation
lambda_1=max(RS(1:2:J)); lambda_2=max(RS(2:2:J));
max_mu=0.5*(lambda_1+lambda_2); % bound for max eigenvalue of autoc. matrix
alpha=1.9/max_mu;        % selected alpha (close to maximum value)
% Estimation of the matrix to be inverted Rs_ssp
Aux=zeros(J,Jred);
for i=1:Jred
    a=conv(V(i,:),rs0ext,'same');
    Aux(:,i)=a';
end
Rs_ssp=V*(Aux);
% Subspace constrained deconvolution
z0_ssp=V*z0;
ener_z0=z0_ssp'*z0_ssp;  % energy of RSA solution used for convergence
thr_conv=10^(-SNR/20);   % threshold for convergence criterion
% Iterations
xi=zeros(Jred,1);     % initial estimation of the response
zi=z0_ssp;
t_iter=toc;          % time-stamp for iterations
for i=1:I            % loop for iterations
    xi=xi+alpha*zi;  % AEP estimate
    zi=z0_ssp-Rs_ssp*xi; % Average residual estimation
    ener_zi=zi'*zi;  % energy of the correction at current iteration
    ratio=sqrt(ener_zi/ener_z0); % ratio of correction vs initialization
    if ratio<thr_conv, break; end; % convergence criterion (loop broken)
    if OUTPUT==1
        fprintf('It.%d: ratio:%.16f alpha=%.5f\n',i,ratio,alpha);
    end
end
Niter=i;
xi_ssp=xi;
t_run=toc;               % total execution time
t_iter=(t_run-t_iter)/i; % execution time for each iteration
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.7 Function “Basis_LinLog_RRC.m” (transformation from original representation to reduced subspace)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [V] = Basis_LinLog_RRC(J,Kdec)     (revised Nov-2019)
%
% Basis_LinLog_RRC() constructs an orthonormal basis of functions (V),
% uniformly distributed in the lin-log-scaled time, with one function
% per sample at small latency and Kdec functions per decade at large
% latency.
% Root-raised-cosine (RRC) function is used for each element of the basis.
% This function is used in digital communications because it provides
% an appropriate limitation of the bandwidth with a relatively short
% duration in the impulsive response.
% The resulting functions are orthonormalized with Gram-Schmidt.
% The resulting basis is contained in a [J_red,J] matrix, with J_red<J.
% The matrix V contains J_red rows, each one with a vector of J components.
% The application of the basis V to a function x (V*x) provides a
% representation in a reduced representation space. The application of the
% transpose V' to the reduced representation (V'*(V*x)) provides a
% reconstruction from the reduced representation into the original time
% representation that is equivalent to a latency-dependent low-pass
% filtering of the response.
%
% INPUT:  [J]      Number of samples in the original representation
%         [Kdec]   Number of functions per decade
% OUTPUT: [V]      Orthonormal basis of functions provided as a matrix
%
% Example: V = Basis_LinLog_RRC(500,25);
%
% Angel de la Torre, Jose Carlos Segura, Joaquin Valderrama (2019)
%     University of Granada (Spain)
%     National Acoustic Laboratories, Macquarie University (Australia)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [V] = Basis_LinLog_RRC(J,Kdec)
%%%% Check for correct input of data
if(J<40||J<Kdec||Kdec<5||Kdec>500)
    error('Error: V=Basis_LinLog_RRC(J,Kdec); (J>=40,J>=Kdec,5<=Kdec<=500)')
end
%%%% Initialization of variables
j=0:(J-1);           % Time-axis, linear scale (in samples)
jr_samp=Kdec*log10(j*log(10)/Kdec+1);  % Time-axis, compressed scale
jr=0:(jr_samp(end)-1); % Samples in compressed scale (functions in basis)
K=length(jr);        % K: Number of functions of the base
%%%% Function template: raised cosine function
Npt_sym=40;           % number of points per symbol period in raised-cosine
N_per=14.6;           % number of periods to each side of the raised-cosine
alpha=0.20;           % roll-off factor (low-pass filtering effect)
[h,tau]=sr_rcos(Npt_sym,N_per,alpha);
%%%% Set of functions of the base before amplitude normalization (V0)
V0 = zeros(length(jr),length(j)); % V0: K functions with J*5 samples
for k=1:K
    % Functions are placed at jr(k) latency (linearly distr in compr. scale)
    V0(k,:)=interp1(tau,h,jr(k)-jr_samp,'linear',0);
end
%%%% Gram-Schmidt orthonormalization
V=OrthoNorm_Gauss_fast(V0,K);
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [h,tau] = sr_rcos(Npt_sym,n_per,alpha)
%
% This function prepares the samples of a square root raised cosine filter
%  See Proakis, J. (1995). Digital Communications. McGraw-Hill Inc. or
%  https://en.wikipedia.org/wiki/Root-raised-cosine_filter)
% Proakis and wikipedia formulas are identical except for a factor Ts
% In this implementation Ts=1;
% Input parameters:
```

(continue in the next page....)

```
%   Npt_sym samples per symbol (or symbol period expressed in samples)
%   n_per   number of periods to the left and to the right of maximum
%   alpha   roll off factor (between 0 and 1)
% output:
%   h       impulsive response
%   tau     normalized time, t/Ts
% example:
%   [h,tau]=sr_rcos(20,6,0.35)
function [h,tau] = sr_rcos(Npt_sym,N_per,alpha)
if alpha == 0, alpha = realmin; end
tau = (-N_per:1/Npt_sym:N_per); h=zeros(size(tau));
% (A) response for t=0:
cond1=tau==0;
h(cond1)=1+alpha*(4/pi-1);
% (B) response for denominator=0:
denom0=1-(4.*alpha.*tau).^2;
cond2=abs(denom0) < sqrt(eps);
phi=pi/(4*alpha);
h(cond2)=alpha/(sqrt(2))*( (1+2/pi)*sin(phi) + (1-2/pi)*cos(phi) );
% (C) response for all the other samples:
cond3=~(cond2|cond1);
t1=tau(cond3);
phi1=pi*t1*(1-alpha); phi2=pi*t1*(1+alpha);
denom=pi*t1.*denom0(cond3);
h(cond3)=(sin(phi1) + 4*alpha*t1.*cos(phi2))./(denom);
return;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% V=OrthoNorm_Gauss_fast(V0,K)
% Orthonormalization, Gauss Method (fast implementation)
% Input parameters:
%    V0  matrix with vectors to be orthonormalized
%    K   number of vectors
% Output paramter:
%    V   matrix with orthonormalized vectors
function V=OrthoNorm_Gauss_fast(V0,K)
% prenormalization
for k=1:K,  v=V0(k,:); V0(k,:)=v/sqrt(dot(v,v)); end;
P=V0*V0';       % Gaussian elimination of [V0*V0' | V] = [P | V0]
M_all=eye(K);   % Matrix operations to be applied to P for Gaussian elimination
for j1=1:K-1
    M_tmp=eye(K);
    for j2=(j1+1):K
        M_tmp(j2,j1)=-P(j2,j1); M_tmp(j2,j2)=P(j1,j1);
        P(j2,:)=P(j2,:)*P(j1,j1)-P(j1,:)*P(j2,j1);
    end
    M_all=M_tmp*M_all;
    M_tmp=eye(K);
    M_tmp(j1+1,j1+1)=1/P(j1+1,j1+1);
    M_all=M_tmp*M_all;
    P(j1+1,:)=P(j1+1,:)/P(j1+1,j1+1);
end
V=M_all*V0;    % Matrix operations applied to V0
% Final normalization
for k=1:K, v=V(k,:);  v=v/sqrt(dot(v,v));  V(k,:)=v;  end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.8 Script "script_simuation_SCLSDec.m" (comparison of different deconvolution algorithms)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% script_simuation_SCLSDec.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script compares different criteria and implementations for least
% squares (LS) deconvolution:
%     * conventional LS deconvolution (LS),
%     * LS deconv. transformed to the reduced representation space (LS-R)
%     * and subspace-constrained LS deconv. (SC-LS)
% The criteria are implemented with matrix division (RSLSD) and with
% iterative estimation (IRSA).
% The execution of this script requires the following functions:
%      Basis_LinLog_RRC.m   Provides the transformation V for the subspace
%      IRSA_LS.m            IRSA (iterative) implementation of LS
%      IRSA_LSR.m           IRSA (iterative) implementation of LS-R
%      IRSA_SCLS.m          IRSA (iterative) implementation of SC-LS
%      RSLSD_LS.m           RSLSD (matrix div.) implementation of LS
%      RSLSD_LSR.m          RSLSD (matrix div.) implementation of LS-R
%      RSLSD_SCLS.m         RSLSD (matrix div.) implementation of SC-LS
% And additionally, a response used for the simulation
%      response_30_60_subj1.mat
% A noisy EEG is synthesized from the reference response according to the
% script configuration; the different deconvolution methods are compared.
% It requires signal processing toolbox
% For Octave users, run:    pkg load signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear,clc;

%% Configuration parameters for simulation
fs=14700;          % sampling rate 14700 Hz
NOISE_GAIN=4;      % noise level
%J=round(1000e-3*fs);   % length of the response 14700, for 1 second
J=round(500e-3*fs);    % length of the response: 7350 for 500 ms
isi_min=30e-3; % 30 ms
isi_max=60e-3; % 60 ms
%n_stim=15200;  % number of stimuli 15200
n_stim=500;    % number of stimuli 500
OUTPUT_IRSA=0; % flag for reporing algorithm evolution
CONVERG_CRITERION=120; % convergence criterion for IRSA_matrix_fft_fast, 120 dB
Imax=10000;    % maximum number of iterations for IRSA
Kdec=40;       % resolution of the LDFDS transformation (samples/decade)
V=Basis_LinLog_RRC(J,Kdec); % LDFDS transformation for subspace

%% Stimulation sequence
isi=rand(n_stim,1)*(isi_max-isi_min)+isi_min; % random distribution of isi, uniform
stim_ind = round(cumsum(isi)*fs); % indexes of stimulus start
N=max(stim_ind)+fs; % number of samples
s=zeros(N,1);
s(stim_ind)=1;

%% Response to be used in simulation (response to be estimated)
load('response_30_60_subj1.mat');
x(J+1)=0; x=x(1:J); % this guarantees a response length equal to J
x=V'*(V*x); % LDFDS applied to x to remove components out of the subspace
t_plot=(0:(J-1))/fs;
figure(1)
semilogx(t_plot*1000,x); xlabel('time (ms)'); ylabel('amplitude');
title('Response to be estimated (time log-scaled)');
grid on; xlim([0.1 1000*J/fs]);
figure(2)
plot(t_plot*1000,x); xlabel('time (ms)'); ylabel('amplitude');
title('Response to be estimated (time in linear scale)');
grid on; xlim([0 1000*J/fs]);
```

(continue in the next page....)

```
%% Simulation of EEG: y=s*x+noise
y0=filter(x,1,s);
y=y0+randn(size(y0))*std(y0)*NOISE_GAIN;
SNR_EEG=10*log10(var(y0)/var(y-y0));
figure(3)
t_plotN=(0:(N-1))/fs;
figure(3)
plot(t_plotN,y); xlabel('time (s)'); ylabel('amplitude');
title(sprintf('EEG     (SNR-EEG: %.2f dB)',SNR_EEG))
figure(4)
plot(t_plotN,y); xlim([0 1])
xlabel('time (s)'); ylabel('amplitude');
title(sprintf('EEG (first second)     (SNR-EEG: %.2f dB)',SNR_EEG))

%% Deconvolution (with predefined alpha and number of iterations)
fprintf('Running RSLSD_LS....\n')
[x1,t_run1]    = RSLSD_LS(y,stim_ind,J);
fprintf('Running IRSA_LS....\n')
[x2,t_run2]    = IRSA_LS(y,stim_ind,Imax,CONVERG_CRITERION,J,OUTPUT_IRSA);
fprintf('Running RSLSD_LSR....\n')
[x3,t_run3]    = RSLSD_LSR(y,stim_ind,V);
fprintf('Running IRSA_LSR....\n')
[x4,t_run4]    = IRSA_LSR(y,stim_ind,V,Imax,CONVERG_CRITERION,OUTPUT_IRSA);
fprintf('Running RSLSD_SCLS....\n')
[x5,t_run5]    = RSLSD_SCLS(y,stim_ind,V);
fprintf('Running IRSA_SCLS....\n')
[x6,t_run6]    = IRSA_SCLS(y,stim_ind,V,Imax,CONVERG_CRITERION,OUTPUT_IRSA);
fprintf('\n--------END OF IRSA/RSLSD ALGORITHMS---------------\n\n')

%% Results: figure
figure(5)
x3=V'*x3; x4=V'*x4; x5=V'*x5; x6=V'*x6; % transformed to original representation
semilogx(t_plot*1000,[x+3.5 x6+2.5 x5+2 x4+1.5 x3+1 x2+0.5 x1 ]);
xlabel('time (ms)'); ylabel('amplitude');
title('Estimated responses');
legend('Response (ref.)','SC-LS iter.','SC-LS mat.div.', ...
    'LS-R iter.','LS-R mat.div.','LS iter.','LS mat.div.');
legend('Location','eastoutside');
grid on; xlim([0.2 1000*J/fs]);

%% Results: difference among responses
% SNR using the clean response as reference
SNR1_a=10*log10(var(x)/var(x1-x)); SNR2_a=10*log10(var(x)/var(x2-x));
SNR3_a=10*log10(var(x)/var(x3-x)); SNR4_a=10*log10(var(x)/var(x4-x));
SNR5_a=10*log10(var(x)/var(x5-x)); SNR6_a=10*log10(var(x)/var(x6-x));
% SNR using SC-LS mat.div. (RSLSD_SCLS) as reference
SNR1_b=10*log10(var(x6)/var(x1-x6)); SNR2_b=10*log10(var(x6)/var(x2-x6));
SNR3_b=10*log10(var(x6)/var(x3-x6)); SNR4_b=10*log10(var(x6)/var(x4-x6));
SNR5_b=10*log10(var(x6)/var(x5-x6));
fprintf('SNR of estimated responses, using the clean response as reference:\n')
fprintf('  SC-LS iter.:      %.5f dB\t\t  SC-LS mat.div.:   %.5f dB\n',SNR6_a,SNR5_a)
fprintf('  LS-R iter.:       %.5f dB\t\t  LS-R mat.div.:    %.5f dB\n',SNR4_a,SNR3_a)
fprintf('  LS iter.:         %.5f dB\t\t  LS mat.div.:      %.5f dB\n',SNR2_a,SNR1_a)
fprintf('SNR of estimated responses, using SC-LS iter. as reference:\n')
fprintf('  SC-LS mat.div.:   %.5f dB\n',SNR5_b)
fprintf('  LS-R iter.:       %.5f dB\t\t  LS-R mat.div.:    %.5f dB\n',SNR4_b,SNR3_b)
fprintf('  LS iter.:         %.5f dB\t\t  LS mat.div.:      %.5f dB\n',SNR2_b,SNR1_b)
fprintf('Total execution time:\n')
fprintf('  SC-LS iter.:      %.5f s\t\t  SC-LS mat.div.:   %.5f s\n',t_run6,t_run5)
fprintf('  LS-R iter.:       %.5f s\t\t  LS-R mat.div.:    %.5f s\n',t_run4,t_run3)
fprintf('  LS iter.:         %.5f s\t\t  LS mat.div.:      %.5f s\n',t_run2,t_run1)

return
```

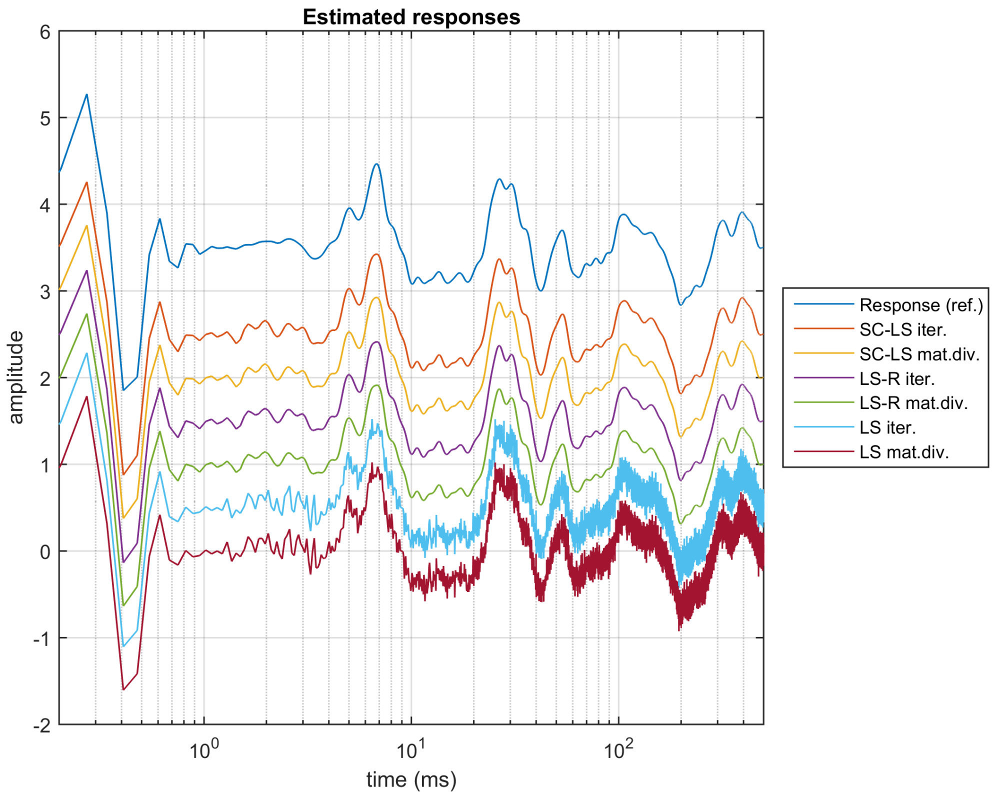


Figure 2: Output figure provided by the script “script_simuation_SCLSDec.m” with the estimated AEP responses. The figure includes the clean AEP response used as reference for the simulations, and the LS, LS-R and SC-LS deconvolutions provided either with the matrix division procedure (RSLSD) or the iterative procedure (IRSA).

5.9 Output of the script “script_simuation_SCLSDec.m”

The script provides some figures with the reference AEP response used for the simulation, a portion of the synthesized noisy EEG, and the AEP estimations provided by the deconvolution procedures. Figure 2, generated by the script, represents the clean AEP response used as reference for synthesizing the EEG in the simulations, and the AEP estimations provided by the different methods. Since the LS-R and the SC-LS estimations provide the response $\hat{\mathbf{x}}_r$ in the reduced representation space, this figure represents the LS-R and SC-LS estimations transformed to the original representation space, i.e. $\hat{\mathbf{x}} = V_r^T\,\hat{\mathbf{x}}_r$.

The script also provides some results in the command line, including a SNR estimation using the clean AEP response as reference (in order to evaluate the quality of the estimations), a SNR estimation using the SC-LS iterative estimation as reference (in order to evaluate the differences among the deconvolution procedures) and the execution time. An example of the script output is provided below:

```
      SNR of estimated responses, using the clean response as reference:
        SC-LS iter.:      27.22830 dB   SC-LS mat.div.:    27.22909 dB
        LS-R iter.:       27.17594 dB   LS-R mat.div.:     27.17683 dB
        LS iter.:         11.09427 dB   LS mat.div.:       11.09429 dB
      SNR of estimated responses, using SC-LS iter. as reference:
        SC-LS mat.div.:   96.89309 dB
        LS-R iter.:       45.92197 dB   LS-R mat.div.:     45.92412 dB
        LS iter.:         11.11060 dB   LS mat.div.:       11.11060 dB
      Total execution time:
        SC-LS iter.:      0.24781 s   SC-LS mat.div.:   0.23361 s
        LS-R iter.:       0.90770 s   LS-R mat.div.:    3.67420 s
        LS iter.:         0.86884 s   LS mat.div.:      3.73644 s
```

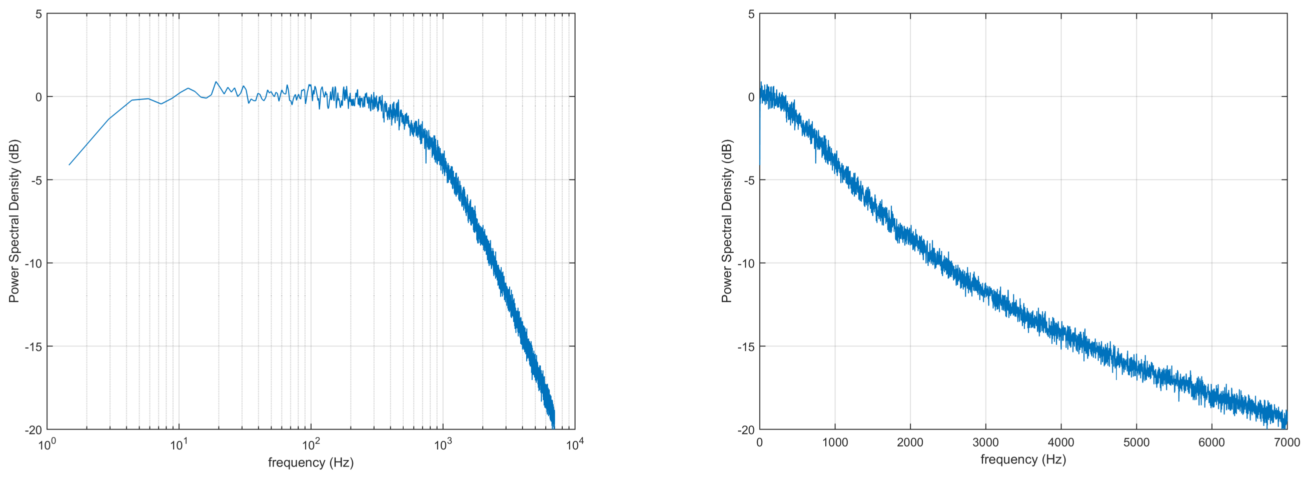


Figure 3: Power spectral density of the noise used for the simulations. The noise has a 3 dB pass-band in the range [1.5 - 800] Hz. In the left panel, the frequency axis is logarithmically scaled; in the right panel, the frequency axis is in linear scale.

6 Complementary results: experiments with simulations

6.1 Description of the noise used for the simulations

The figure 3 represents the power spectral density of the noise used for the simulations. This is a band-pass noise, with a pass-band [1.5 800] Hz, obtained from a white noise filtered with an appropriate 1st order Butterworth band-pass filter (the slope of the power spectral density is +20 dB/decade at low frequencies, -20 dB/decade at high frequencies, and the response is flat in the pass-band, as can be observed in the left panel of the figure).

6.2 Comparison of the estimations provided by the different deconvolution methods

In this section, the responses obtained with the different deconvolution methods in the simulation experiments are compared. The figure 4 shows the AEP responses for one repetition of the simulation, including the clean AEP responses (used as reference), and the LS, LS-R and SC-LS estimations. Responses corresponding to the different ISI configuration (from 480-960 ms to 15-30 ms) are included. Since the LS-R and the SC-LS estimations provide the response $\hat{\mathbf{x}}_r$ in the reduced representation space, this figure represents the LS-R and SC-LS estimations transformed to the original representation space, i.e. $\hat{\mathbf{x}} = V_r^T \hat{\mathbf{x}}_r$. As can be observed, the LS estimation is more affected by noise than the LS-R and SC-LS estimations, and the estimations provided by LS-R and SC-LS are very similar.

Figure 5 represents the error observed for the LS (top), LS-R (center) and SC-LS (bottom) estimations, i.e. the difference between the estimation and the reference (clean) response. The errors plotted in this figure corresponds to the estimations represented in the figure 4. As can again be observed, the error is more important for the LS estimation than for the LS-R and SC-LS estimations, and very similar between these two estimations.

In order to compare the LS, LS-R and SC-LS estimations, the difference between SC-LS and LS, and that between SC-LS and LS-R have been plotted in figure 6. As observed, the SC-LS and the LS estimations are quite different, while the difference between the SC-LS and the LS-R estimations is significantly smaller than the error affecting them (the figure in the center represents the differences in the same scale, while the figure in the bottom represents a detail, with an appropriate scale). The error affecting the SC-LS and LS-R estimations is more than 20 times larger than the difference between them (or equivalently, the difference is more than 26 dB below the estimation error). This suggests that, regarding the quality of the estimated responses, both SC-LS and LS-R provide results that are equivalent from a practical point of view.

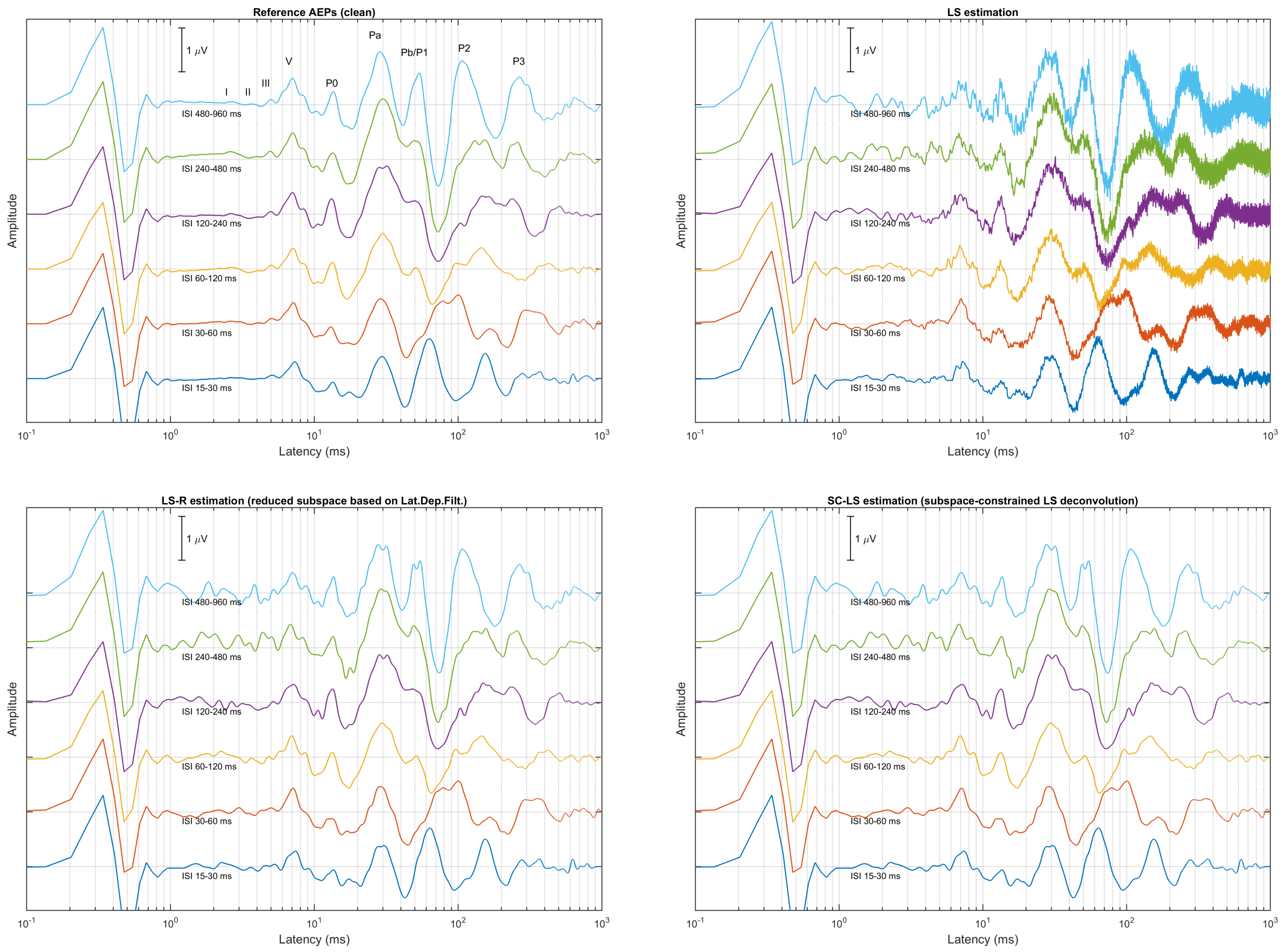
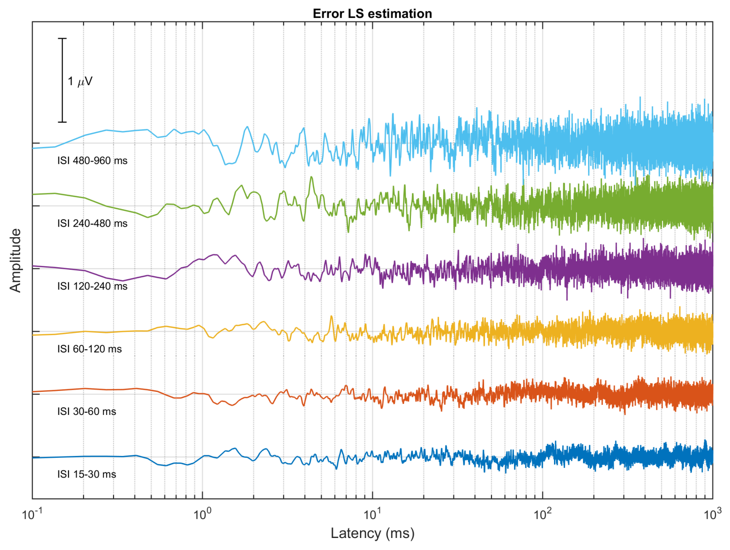
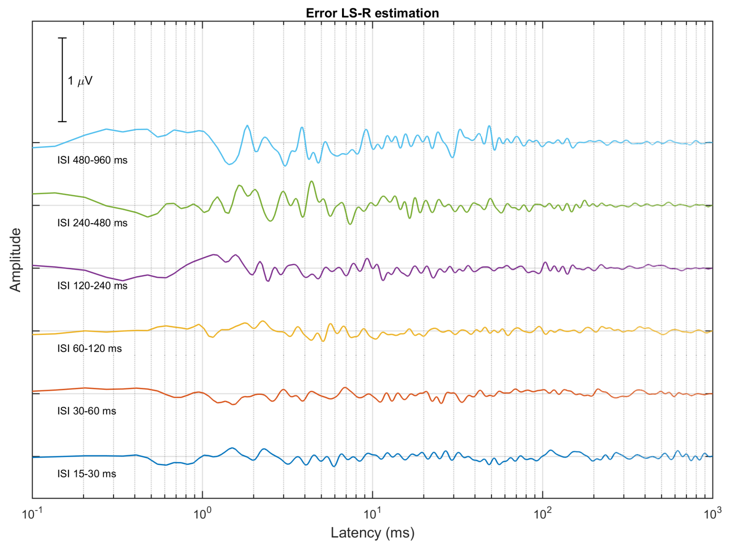


Figure 4: AEP responses obtained in the simulations for one of the 100 repetitions. The figure includes the clean responses used as reference, the LS estimations (conventional LS deconvolution in the original representation space), the LS-R estimations (by applying the dimensionality reduction based on LDFDS to the LS estimation) and the SC-LS estimations (subspace-constrained LS deconvolution).





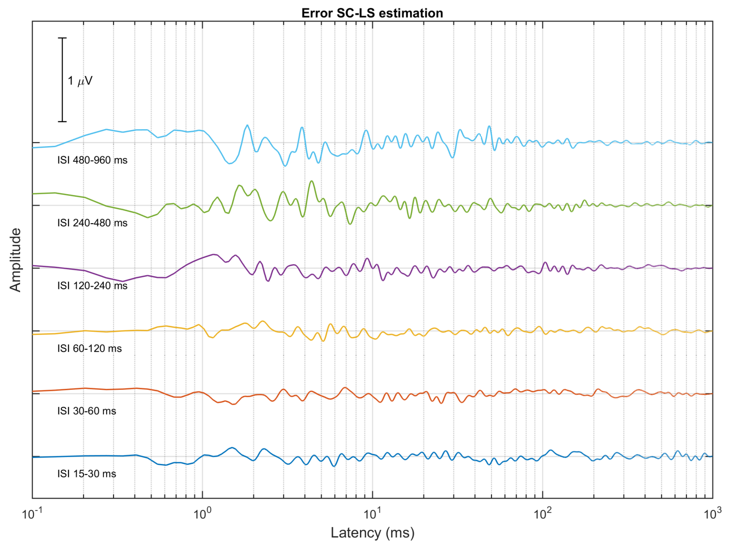
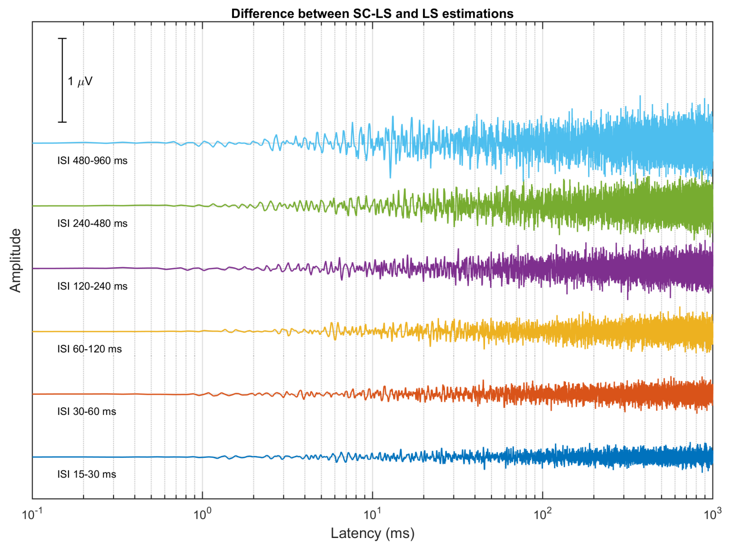
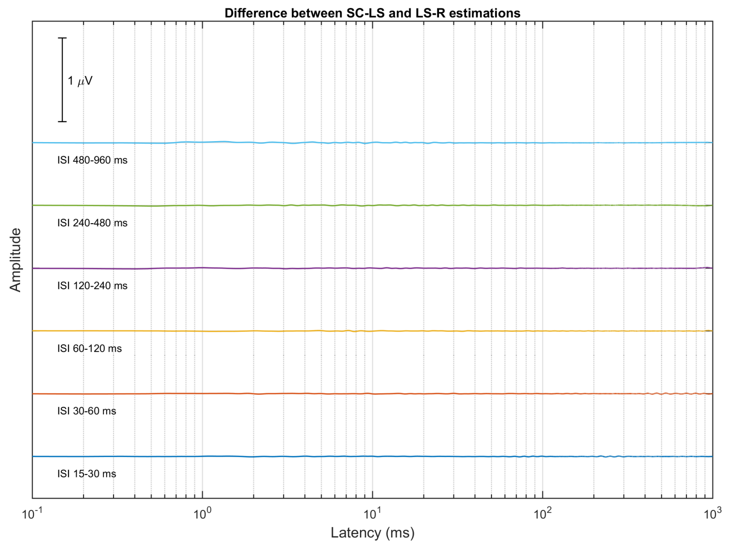


Figure 5: Error for the LS (top), LS-R (center) and SC-LS (bottom) estimations obtained in the simulations for one repetition. Error is obtained by subtracting the reference (clean) response to the estimations provided by each method.





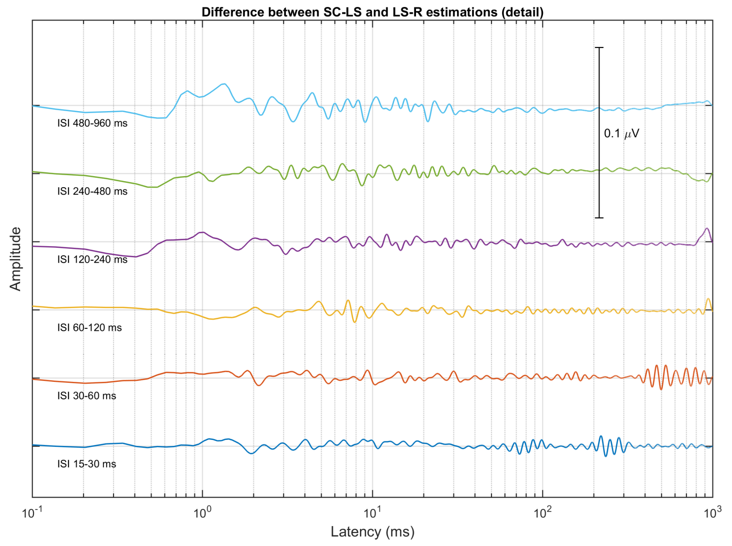


Figure 6: Difference between the SC-LS and the LS estimations (top) and between the SC-LS and the LS-R estimations (center). The plot in the bottom is a detail of the last comparison (note that the scale represents an amplitude of 0.1 μV).

6.3 Statistical characterization of the noise and the error

The statistical characterization of the error is obtained by transforming the covariance matrix of the noise Σ_{n_0}, according to the matrix operations applied to the EEG signal $\mathbf{y}$, in order to obtain the covariance matrix Σ_e of the error affecting the estimation. This requires the estimation of the $(N \times N)$ covariance matrix Σ_{n_0}, its transformation (with the averaging matrix S_k) into the $(J \times J)$ covariance matrix of the averaged noise Σ_{n_A}, and the manipulation of this matrix using R_s, its inverse, and the V_r matrices, according to the equations (58), (60) and (62).

The manipulation of a $(N \times N)$ matrix is not practical with the typical lengths of an EEG (several million samples). However, assumed that the noise is a stationary process, its covariance matrix is Toeplitz and therefore it can be represented with its covariance function (or its autocorrelation function assumed it is a zero-mean process). Additionally, since the matrix S is a convolution matrix, the product $\Sigma_{n_A} = S_k \, \Sigma_{n_0} \, S_k^T$ is also Toeplitz, and therefore Σ_{n_A} can be represented from its autocorrelation function, which can be estimated by convolution products involving the stimulation sequence and the autocorrelation function of the noise.

The figure 7 represents the autocorrelation function of the noise (corresponding to Σ_{n_0}, in the top) and that of the averaged noise (corresponding to Σ_{n_A}, in the bottom). These plots correspond to 1 of the 100 repetitions in the simulation experiments, and for the 15-30 ms ISI condition. The shape of the autocorrelation noise is consistent with the 1st order 1.5-800 Hz bandpass filtered white noise used in the simulations. The shape of the averaged autocorrelation noise is consistent with the ISI interval of this example. From the autocorrelation function of the averaged noise, a reasonable estimation of Σ_{n_A} can be obtained, and this estimation can be used for the estimation of the covariance matrix of the error Σ_e under the different approaches.

The figure 8 represents the diagonal of the Σ_e matrices and the response estimations corresponding to this example. In the case of the LS estimation, the covariance matrix of the error is a $(J \times J)$ matrix, and the diagonal contains J components (14700 in this example), as observed in the plot in the top. The trace is 40.38 μV^2 in this case. In the case of the LS-R and SC-LS estimations, the covariance matrices are $(J_r \times J_r)$ matrices and the diagonals contain 117 components. The traces are 9.44 and 9.39 μV^2, respectively, for LS-R and SC-LS, as indicated in the plot in the center. These estimations of the expected energy of the error are significantly smaller than that for the LS estimation. The traces are, in addition, very similar for LS-R and SC-LS, even though slightly smaller for SC-LS (i.e. the expected error energy is slightly smaller for SC-LS than for LS-R).

The plot in the bottom represents the reference AEP response, as well as the LS, LS-R and SC-LS estimations. The legend includes the SNR corresponding to each estimation. The SNRs have been calculated either from the traces (i.e. using the expected energy error from the trace of the corresponding Σ_e covariance matrices) or directly from the signals (i.e. using the error signal calculated as the difference between the estimated and the reference signals). In this example, the SNRs estimated from the traces are 11.59, 17.91 and 17.93 dB for LS, LS-R and SC-LS, respectively. The SNRs directly measured from the signals are 11.08, 16.48 and 16.57 dB, respectively for LS, LS-R and SC-LS. These results show the consistency between the SNR estimations (or the error energy estimations) derived either from the traces of the error covariance matrices or directly measured from the signals.

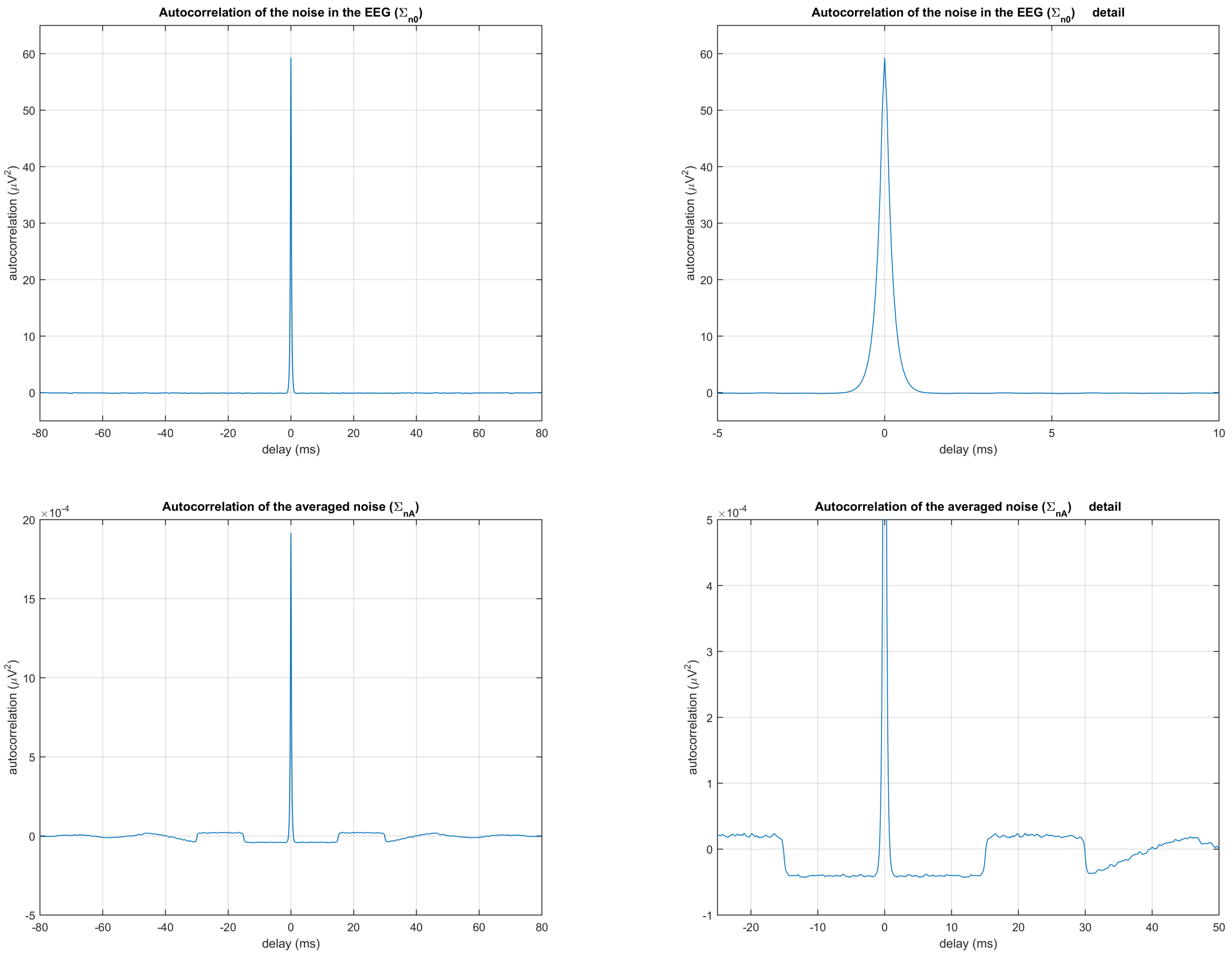


Figure 7: Autocorrelation functions of the noise (top) and the averaged noise (bottom) estimated for one of the 100 repetitions in the simulation experiments, for the 15-30 ms ISI condition. The plots in the right side include a detail of the autocorrelation functions.

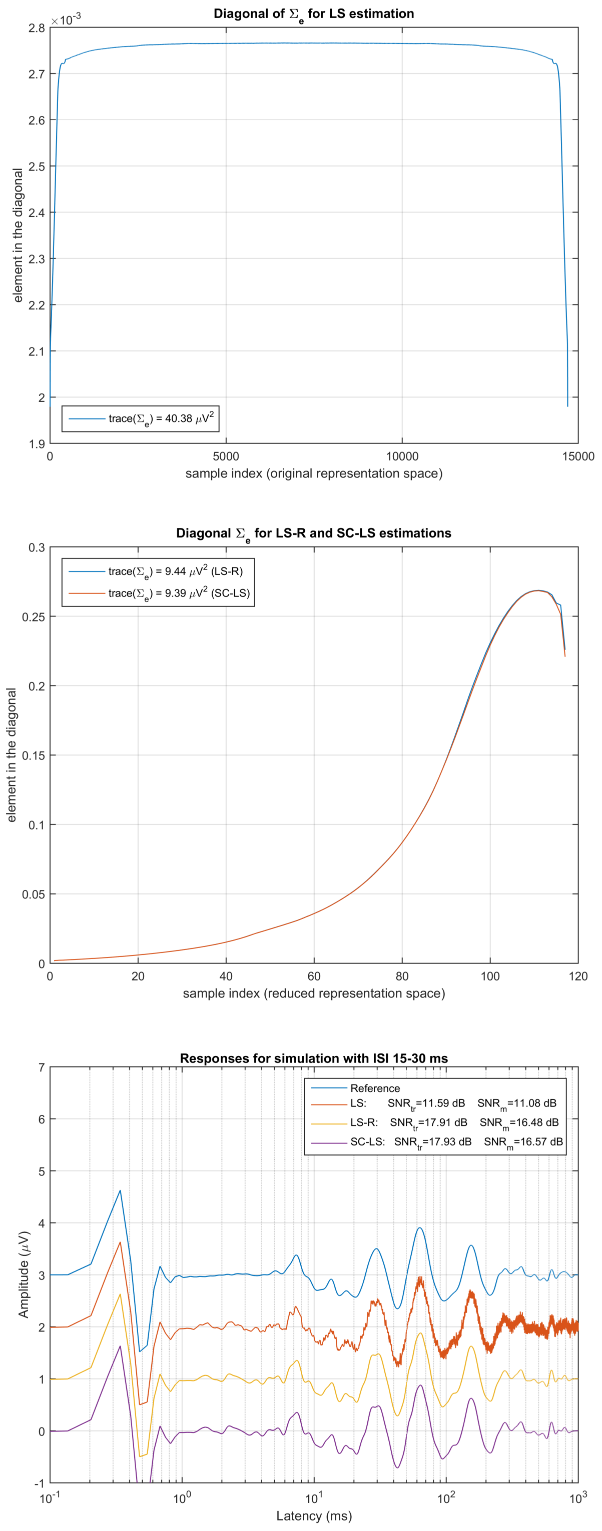


Figure 8: Diagonal of the covariance matrix of the error for the LS estimation (top) and for the LS-R and SC-LS estimations (center). The traces of these covariance matrices are also indicated. The reference response, and the estimations provided by the LS, LS-R and SC-LS estimations are represented in the bottom. The legend includes the SNR of each solution estimated from the expected value of the error energy (using the traces) or directly measured from the responses (using the difference between the estimated response and the reference response). These plots corresponds to the example in the previous figures, for ISI configuration 15-30 ms.

Table 1: SNR mean (standard deviation in parenthesis) in dB, for the LS, LS-R and SC-LS estimations, obtained in simulations with 100 repetitions for each ISI condition. The SNR measurements are based on the observed error energy (top table) and the expected error energy (using the trace of the covariance matrix of the error, bottom table).

(A) SNR based on measured error energy

ISI condition	LS mean (std.dev) dB	LS-R mean (std.dev) dB	SC-LS mean (std.dev) dB
480-960 ms	9.335 (0.101)	23.288 (0.624)	23.285 (0.622)
240-480 ms	9.236 (0.096)	22.765 (0.622)	22.791 (0.614)
120-240 ms	9.599 (0.114)	21.905 (1.059)	21.959 (1.063)
60-120 ms	7.617 (0.152)	17.758 (1.192)	17.793 (1.164)
30-60 ms	11.508 (0.208)	19.434 (1.076)	19.483 (1.076)
15-30 ms	11.580 (0.268)	17.926 (1.014)	17.951 (1.018)
Average	**9.812 (1.391)**	**20.513 (2.438)**	**20.544 (2.430)**

(B) SNR based on expected error energy (covariance matrix trace)

ISI condition	LS mean (std.dev) dB	LS-R mean (std.dev) dB	SC-LS mean (std.dev) dB
480-960 ms	9.351 (0.002)	23.181 (0.014)	23.188 (0.014)
240-480 ms	9.252 (0.012)	22.672 (0.023)	22.686 (0.023)
120-240 ms	9.589 (0.008)	21.778 (0.035)	21.810 (0.035)
60-120 ms	7.614 (0.004)	17.645 (0.041)	17.688 (0.041)
30-60 ms	11.472 (0.008)	19.180 (0.045)	19.209 (0.045)
15-30 ms	11.610 (0.008)	17.928 (0.030)	17.948 (0.030)
Average	**9.815 (1.379)**	**20.397 (2.238)**	**20.421 (2.230)**

6.4 Comparison of the observed error energy and the expected error energy

In the simulation experiments, where the reference clean response $\mathbf{x}$ is available, the error can easily be estimated as $\mathbf{e} = \hat{\mathbf{x}} - \mathbf{x}$. and the different procedures can easily be compared in terms of the measured or observed error energy. Interestingly, since the transformation V_r is orthonormalized, for the estimations involving the dimensionality reduction (LS-R and SC-LS), the energy of the error can equivalently be estimated in the original representation ($\mathbf{e} = V^T\hat{\mathbf{x}}_r - \mathbf{x}$) or in the reduced representation ($\mathbf{e}_r = \hat{\mathbf{x}}_r - V\mathbf{x}$). Alternatively the expected error energy can be estimated as the trace of the covariance matrix of the error $\mathrm{tr}(\Sigma_e)$, that can be estimated from Σ_{n_A} and the equations (58), (60) and (62), respectively, for LS, SC-LS and LS-R estimations.

Table 1 represents the SNR estimations for the simulation experiments obtained with the measured error energy (top) and with the expected error energy, derived from the traces of the covariance matrices, (bottom). The results are consistent for both, the measured and the expected error energies, but those based on the expected energy errors are significantly more stable (the standard deviations are significantly smaller). As can be observed in this table, the LS-R and SC-LS methods provide a significant improvement in the quality of the estimated responses with respect to the LS method (more than 10 dB). These results also suggest a slight improvement of SC-LS with respect to LS-R.

Table 2 compares the quality of LS-R vs. LS and SC-LS vs. LS-R. The comparison is based in the SNR increase, and is evaluated with a paired Student's t-test. The results in the top correspond to SNR comparison based on the measured error energy, while the results in the bottom correspond to SNR based on expected error energy (using covariance matrices of the error). The comparison of LS-R vs. LS reveals an average improvement greater than 10 dB, consistent for both, measured and expected estimations of the error energy. The comparison of SC-LS vs. LS-R shows a very slight improvement (around 0.025 dB), which is not useful in practice, but is statistically significant (statistical significance evaluated with p is not clear for some ISI conditions with the measured error estimations, but the improvement are always significant with the expected error energies).

Even though the SNR increase observed for SC-LS with respect to LS-R is statistically significant, this improvement is irrelevant in practice. The relevance of the SNR improvement can be evaluated in terms of the required increase of recording time necessary to achieve the observed SNR improvement: for example an improvement of 3 dB can be obtained by doubling the EEG length, since $10 \cdot \log_{10}(2)$ =3 dB; similarly, an improvement of 0.025 dB can be obtained by increasing the length of the EEG in a factor 1.006, or equivalently by increasing the EEG length in a 0.6 % (in our case, by recording 688 seconds of EEG instead of 684 seconds).

Table 2: SNR improvement provided by LS-R with respect to LS (left) and by SC-LS with respect to LS-R (right), in dB, obtained in simulations with 100 repetitions for each ISI condition. The SNR improvements are based on the observed error energy (top table) and the expected error energy (using the trace of the covariance matrix of the error, bottom table). The table includes mean, standard deviation and the p parameter in a paired Student's t-test.

(A) SNR comparison based on measured error energy

ISI	LS-R vs. LS		SC-LS vs. LS-R	
condition	mean (std.dev) dB	p	mean (std.dev) dB	p
480-960 ms	13.953 (0.596)	1.1e-137	-2.5e-3 (0.061)	0.68
240-480 ms	13.529 (0.603)	8.3e-136	0.025 (0.175)	0.15
120-240 ms	12.307 (1.003)	5.5e-110	0.054 (0.203)	9.4e-3
60-120 ms	10.140 (1.070)	5.5e-99	0.035 (0.176)	0.050
30-60 ms	7.926 (0.894)	3.8e-96	0.049 (0.108)	1.5e-5
15-30 ms	6.346 (0.799)	1.9e-91	0.025 (0.077)	1.7e-3
Average	**10.700 (2.958)**	**<1e-320**	**0.031 (0.144)**	**2.2e-07**

(B) SNR comparison based on expected error energy (covariance matrix trace)

ISI	LS-R vs. LS		SC-LS vs. LS-R	
condition	mean (std.dev) dB	p	mean (std.dev) dB	p
480-960 ms	13.830 (0.013)	1.4e-300	0.006 (4.4e-4)	4.0e-117
240-480 ms	13.419 (0.018)	2.0e-287	0.014 (8.1e-4)	3.8e-124
120-240 ms	12.188 (0.038)	6.1e-250	0.033 (1.1e-3)	1.2e-149
60-120 ms	10.031 (0.037)	2.1e-242	0.043 (4.9e-4)	6.2e-194
30-60 ms	7.708 (0.040)	9.2e-228	0.028 (5.6e-4)	4.5e-171
15-30 ms	6.318 (0.023)	2.7e-244	0.020 (4.2e-4)	7.5e-168
Average	**10.582 (2.83)**	**<1e-320**	**0.024 (0.012)**	**7.5e-211**

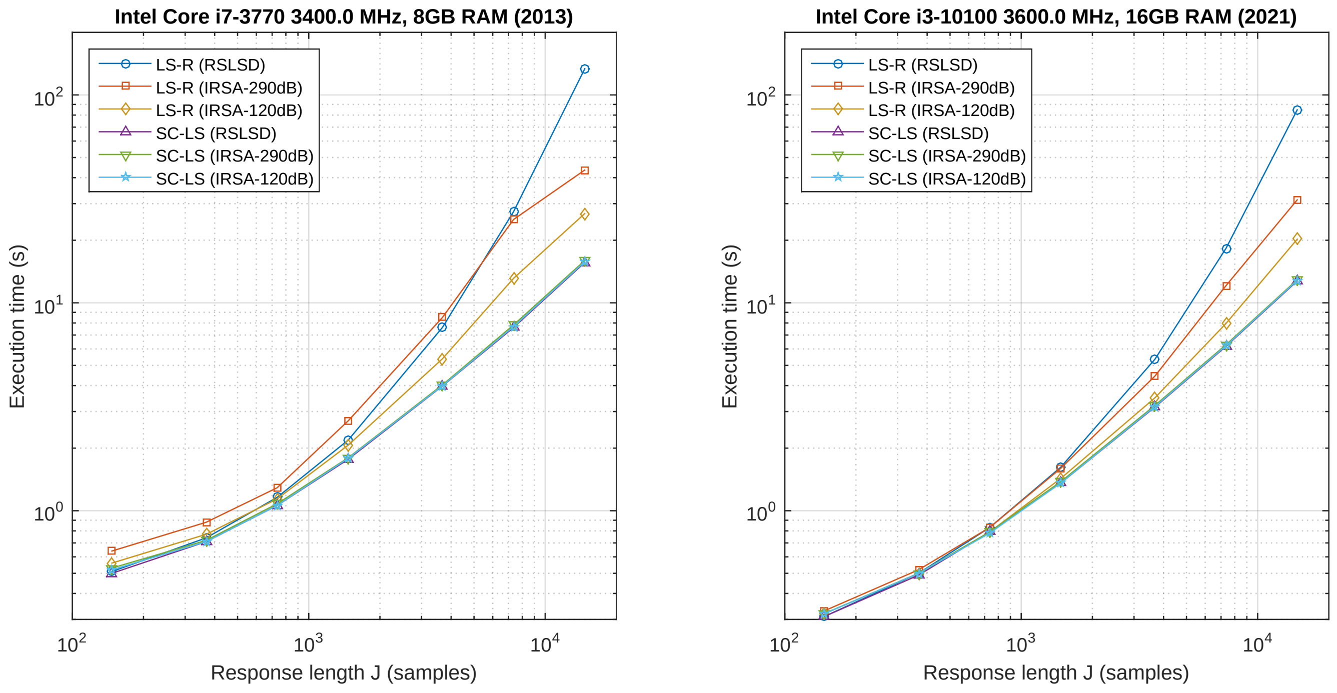


Figure 9: Total execution time per subject of the different algorithms as a function of the response length J. In the left panel, execution times measured with Computer 1 (a 2013 system with an Intel Core i7-3770 3400.0 MHz processor and 8GB RAM); in the right panel, execution times measured with Computer 2 (a 2021 system with an Intel Core i3-10100 3600.0 MHz and 16GB RAM).

7 Complementary results: experiments with real EEGs

7.1 Execution time

The execution time for the experiments involving real EEGs is reported in the figure 9 and the table table 3. The execution times were measured with two different computers: the first one is a system manufactured in 2013, with an Intel Core i7-3770 processor at 3400.0 MHz and with 8 GB of memory installed in the RAM, while the second one is a system manufactured in 2021, with an Intel Core i3-10100 processor at 3600.0 MHz and with 16 GB of memory installed in the RAM.

The figure 9 compares the total execution time per subject as a function of the response length, for both computers. The total execution time includes the execution time for all the test configuration (for the different ISI conditions). As can be observed in these plots, the subspace-constrained deconvolution provides a reduction in the execution time with both computers. As expected, the deconvolution procedure is faster with the new computer (even though the improvement is very small in the case of subspace-constrained deconvolution).

The table 3 includes the execution time required for the algorithm initialization and for each iteration (in the case of iterative algorithms). Results include mean execution time per subject and standard deviations. The execution times are very similar for the LS and LS-R algorithms (because the computational load for the LDFDS dimensionality reduction is very small compared with the the LS estimation). The execution times for the subspace-constrained deconvolution are significantly smaller that those for LS and LS-R. The execution times for the different implementation of SC-LS are very similar among them (around 16 seconds for estimating the AEP responses of each subject in all the ISI conditions with Computer 1; around 13 seconds with computer 2).

Table 3: Execution time per subject in the experiments with real EEGs. The columns correspond to the different deconvolution algorithms and the rows to the different ISI conditions. The tables provide (A) the initialization time, (B) average time per iteration (in the iterative algorithms) and (C) total execution time, measured with Computer 1 (a 2013 system with an Intel Core i7-3770 3400.0 MHz processor and 8GB RAM) and Computer 2 (a 2021 system with an Intel Core i3-10100 3600.0 MHz and 16GB RAM).

Computer 1 - (A) Initialization time, in seconds: mean (std.dev)

ISI	LS_{MD}	LS_{It}	LS_{It}	LS-R_{MD}	LS-R_{It}	LS-R_{It}	SC-LS_{MD}	SC-LS_{It}	SC-LS_{It}
(ms)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)
480-960	20.23 (0.22)	0.51 (0.01)	0.51 (0.01)	20.50 (0.31)	0.50 (0.01)	0.50 (0.01)	0.59 (0.01)	0.59 (0.01)	0.59 (0.01)
240-480	20.91 (0.32)	0.82 (0.02)	0.80 (0.01)	20.74 (0.26)	0.81 (0.02)	0.79 (0.01)	1.01 (0.01)	1.02 (0.02)	1.01 (0.01)
120-240	21.58 (0.40)	1.29 (0.01)	1.26 (0.01)	21.34 (0.29)	1.28 (0.02)	1.26 (0.01)	1.74 (0.02)	1.75 (0.02)	1.75 (0.02)
60-120	22.10 (0.71)	2.02 (0.02)	1.99 (0.01)	22.18 (0.84)	2.01 (0.03)	1.98 (0.02)	2.47 (0.03)	2.47 (0.02)	2.47 (0.02)
30-60	22.99 (0.32)	3.08 (0.03)	3.05 (0.02)	22.99 (0.43)	3.07 (0.02)	3.05 (0.02)	3.52 (0.03)	3.53 (0.03)	3.55 (0.03)
15-30	25.87 (0.53)	5.98 (0.22)	5.86 (0.24)	25.82 (0.26)	5.85 (0.26)	5.76 (0.28)	6.31 (0.26)	6.29 (0.26)	6.24 (0.27)

Computer 1 - (B) Time for each iteration, in microseconds: mean (std.dev)

ISI	LS_{MD}	LS_{It}	LS_{It}	LS-R_{MD}	LS-R_{It}	LS-R_{It}	SC-LS_{MD}	SC-LS_{It}	SC-LS_{It}
(ms)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)
480-960	-	1251 (164)	1229 (170)	-	1102 (139)	1217 (130)	-	11.39 (0.12)	14.03 (3.37)
240-480	-	1195 (48)	1176 (85)	-	1155 (38)	1200 (116)	-	15.25 (4.18)	11.18 (0.60)
120-240	-	1169 (55)	1151 (55)	-	1150 (29)	1129 (51)	-	12.41 (2.57)	11.78 (2.20)
60-120	-	1141 (37)	1116 (38)	-	1130 (27)	1149 (43)	-	13.36 (2.52)	11.44 (1.41)
30-60	-	1130 (20)	1131 (26)	-	1130 (24)	1123 (18)	-	13.01 (1.65)	13.35 (3.36)
15-30	-	1117 (24)	1123 (21)	-	1115 (13)	1125 (27)	-	12.51 (1.97)	12.68 (2.41)

Computer 1 - (C) Execution time (including initialization and iterations), in seconds: mean (std.dev)

ISI	LS_{MD}	LS_{It}	LS_{It}	LS-R_{MD}	LS-R_{It}	LS-R_{It}	SC-LS_{MD}	SC-LS_{It}	SC-LS_{It}
(ms)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)
480-960	20.23 (0.22)	0.63 (0.02)	0.55 (0.01)	20.50 (0.31)	0.61 (0.01)	0.53 (0.01)	0.59 (0.01)	0.59 (0.01)	0.59 (0.01)
240-480	20.91 (0.32)	1.79 (0.06)	1.09 (0.05)	20.74 (0.26)	1.75 (0.04)	1.09 (0.05)	1.01 (0.01)	1.03 (0.02)	1.02 (0.01)
120-240	21.58 (0.40)	3.75 (0.16)	2.03 (0.07)	21.34 (0.29)	3.71 (0.10)	2.01 (0.06)	1.74 (0.02)	1.77 (0.02)	1.75 (0.02)
60-120	22.10 (0.71)	7.05 (0.20)	3.60 (0.10)	22.18 (0.84)	6.99 (0.22)	3.65 (0.12)	2.47 (0.03)	2.52 (0.02)	2.49 (0.02)
30-60	22.99 (0.32)	13.48 (0.33)	6.66 (0.14)	22.99 (0.43)	13.47 (0.36)	6.63 (0.12)	3.52 (0.03)	3.65 (0.04)	3.59 (0.03)
15-30	25.87 (0.53)	17.16 (0.26)	13.05 (0.20)	25.82 (0.26)	17.00 (0.30)	12.97 (0.28)	6.31 (0.26)	6.42 (0.24)	6.32 (0.26)
All	**133.69 (1.92)**	**43.86 (0.47)**	**26.98 (0.36)**	**133.58 (1.64)**	**43.53 (0.46)**	**26.88 (0.34)**	**15.63 (0.30)**	**15.98 (0.28)**	**15.76 (0.27)**

Computer 2 - (A) Initialization time, in seconds: mean (std.dev)

ISI	LS_{MD}	LS_{It}	LS_{It}	LS-R_{MD}	LS-R_{It}	LS-R_{It}	SC-LS_{MD}	SC-LS_{It}	SC-LS_{It}
(ms)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)
480-960	12.39 (0.08)	0.37 (0.01)	0.28 (0.01)	12.38 (0.05)	0.29 (0.01)	0.29 (0.01)	0.34 (0.00)	0.34 (0.00)	0.34 (0.00)
240-480	13.02 (0.22)	0.50 (0.01)	0.49 (0.01)	12.85 (0.06)	0.50 (0.01)	0.49 (0.01)	0.62 (0.00)	0.62 (0.01)	0.61 (0.01)
120-240	13.22 (0.10)	0.80 (0.01)	0.78 (0.01)	13.20 (0.05)	0.79 (0.01)	0.78 (0.01)	1.07 (0.01)	1.06 (0.01)	1.06 (0.01)
60-120	13.40 (0.05)	1.30 (0.01)	1.29 (0.01)	13.50 (0.26)	1.31 (0.03)	1.28 (0.01)	1.57 (0.01)	1.57 (0.02)	1.57 (0.01)
30-60	14.54 (0.07)	2.44 (0.01)	2.41 (0.02)	14.53 (0.07)	2.43 (0.01)	2.40 (0.01)	2.71 (0.03)	2.70 (0.02)	2.70 (0.02)
15-30	18.35 (0.14)	6.19 (0.02)	6.02 (0.05)	18.33 (0.15)	6.19 (0.08)	6.12 (0.03)	6.43 (0.02)	6.43 (0.04)	6.44 (0.04)

Computer 2 - (B) Time for each iteration, in microseconds: mean (std.dev)

ISI	LS_{MD}	LS_{It}	LS_{It}	LS-R_{MD}	LS-R_{It}	LS-R_{It}	SC-LS_{MD}	SC-LS_{It}	SC-LS_{It}
(ms)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)
480-960	-	754 (25)	757 (47)	-	749 (18)	774 (45)	-	8.19 (1.25)	8.65 (1.50)
240-480	-	748 (5.9)	756 (17)	-	747 (11)	753 (17)	-	7.69 (1.23)	7.71 (1.07)
120-240	-	746 (5.6)	743 (10)	-	744 (3.9)	741 (8.2)	-	7.46 (1.31)	6.98 (1.19)
60-120	-	747 (3.5)	741 (7.8)	-	748 (3.7)	742 (4.1)	-	7.45 (0.66)	6.72 (0.53)
30-60	-	739 (1.9)	738 (4.3)	-	740 (2.4)	739 (4.3)	-	7.45 (0.50)	7.67 (0.71)
15-30	-	742 (2.6)	741 (2.0)	-	742 (4.4)	742 (1.5)	-	7.47 (0.29)	7.54 (0.65)

Computer 2 - (C) Execution time (including initialization and iterations), in seconds: mean (std.dev)

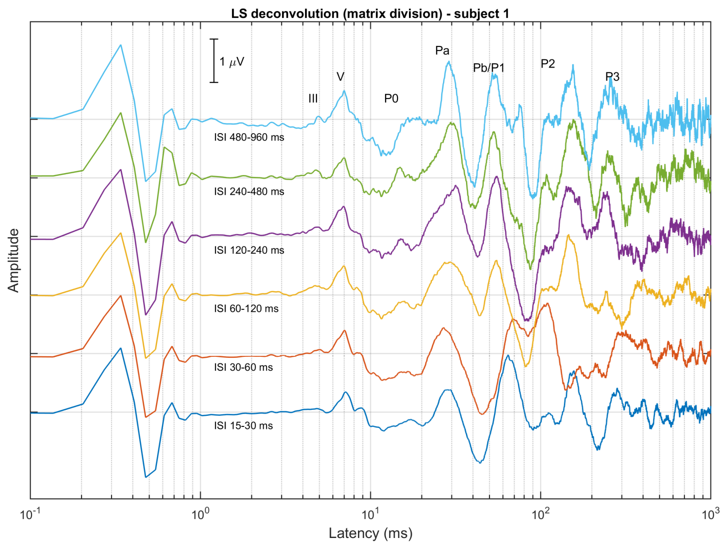
ISI	LS_{MD}	LS_{It}	LS_{It}	LS-R_{MD}	LS-R_{It}	LS-R_{It}	SC-LS_{MD}	SC-LS_{It}	SC-LS_{It}
(ms)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)	(RSLSD)	(IRSA-290dB)	(IRSA-120dB)
480-960	12.39 (0.08)	0.37 (0.01)	0.31 (0.01)	12.38 (0.05)	0.36 (0.01)	0.32 (0.02)	0.34 (0.00)	0.34 (0.00)	0.34 (0.00)
240-480	13.02 (0.22)	1.11 (0.03)	0.67 (0.03)	12.85 (0.06)	1.11 (0.02)	0.67 (0.03)	0.62 (0.00)	0.62 (0.01)	0.62 (0.01)
120-240	13.22 (0.10)	2.37 (0.07)	1.28 (0.04)	13.20 (0.05)	2.36 (0.06)	1.27 (0.04)	1.07 (0.01)	1.08 (0.01)	1.07 (0.01)
60-120	13.40 (0.05)	4.60 (0.11)	2.36 (0.06)	13.50 (0.26)	4.61 (0.09)	2.36 (0.06)	1.57 (0.01)	1.61 (0.02)	1.58 (0.01)
30-60	14.54 (0.07)	9.23 (0.12)	4.77 (0.08)	14.53 (0.07)	9.23 (0.12)	4.76 (0.09)	2.71 (0.03)	2.77 (0.02)	2.72 (0.02)
15-30	18.35 (0.14)	13.61 (0.04)	10.76 (0.20)	18.33 (0.15)	13.61 (0.10)	10.85 (0.19)	6.43 (0.02)	6.50 (0.04)	6.49 (0.04)
All	**84.92 (0.44)**	**31.29 (0.21)**	20.15 (0.26)	**84.79 (0.28)**	**31.29 (0.23)**	**20.24 (0.27)**	**12.74 (0.04)**	**12.92 (0.03)**	**12.82 (0.05)**

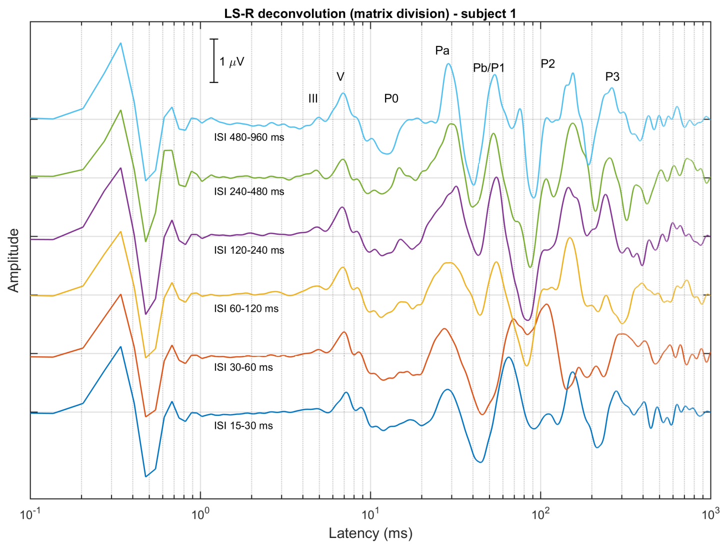
7.2 Comparison of the estimations provided by the different deconvolution methods

Figure 10 represents the responses obtained for the LS, LS-R and SC-LS estimations (using the matrix-division based algorithms) for subject 1 and different ISI conditions. As in the simulations, these AEP estimations are more affected by the noise in the case of LS, and are similar in the case of LS-R and SC-LS.

In order to compare the estimations provided by the LS, LS-R ad SC-LS methods, the differences are represented in figure 11. The figure in the top represents the difference between the SC-LS and the LS estimations, and the figure in the bottom represents the difference between SC-LS and the LS-R estimations. The SNR of the difference has been averaged for the different ISI conditions. This SNR has been estimated as the ratio between the energy of the signal and the energy of the difference, and is 10.39 dB in the comparison SC-LS vs. LS, and 29.71 dB in the comparison SC-LS vs. LS-R (note that the amplitude scale is 1 μV in the top figure, but 0.1 μV in the bottom figure). According to the estimated SNR for the difference, we can say that the difference between the SC-LS and LS-R estimations are about 30 dB below the signal energy for the estimations based on real EEGs, which is in accordance with the results obtained with the simulations (reported in figure 6). The difference is significantly smaller than the expected background noise.

Figure 12 compares the SC-LS estimations obtained for subject 1 with the different algorithms (based on matrix-division or iterative). The top figure represents the difference between the matrix-division estimations and the iterative estimations using 120 dB as convergence criterion, while in the bottom figure the iterative estimations apply 290 dB as convergence criterion. As can be observed, the iterative estimations converge to the matrix-division estimations, and in average, the energy of the differences are 95 and 265 dB below the energy of the signal, when the 120 and 290 dB criteria are considered, respectively.





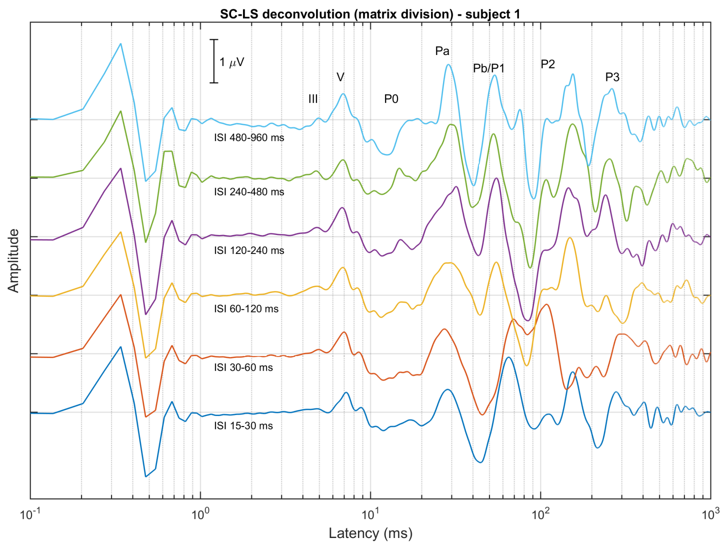
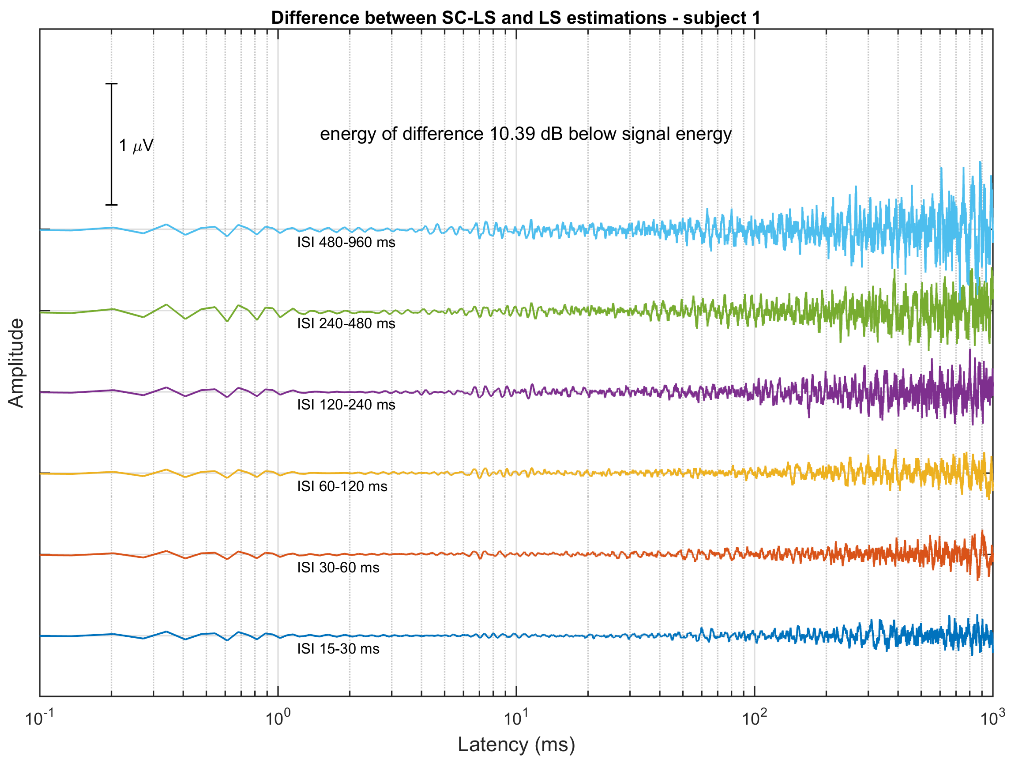


Figure 10: AEP responses for subject 1, estimated with LS_{MD} (top), LS-R_{MD} (center) and SC-LS_{MD} (bottom).



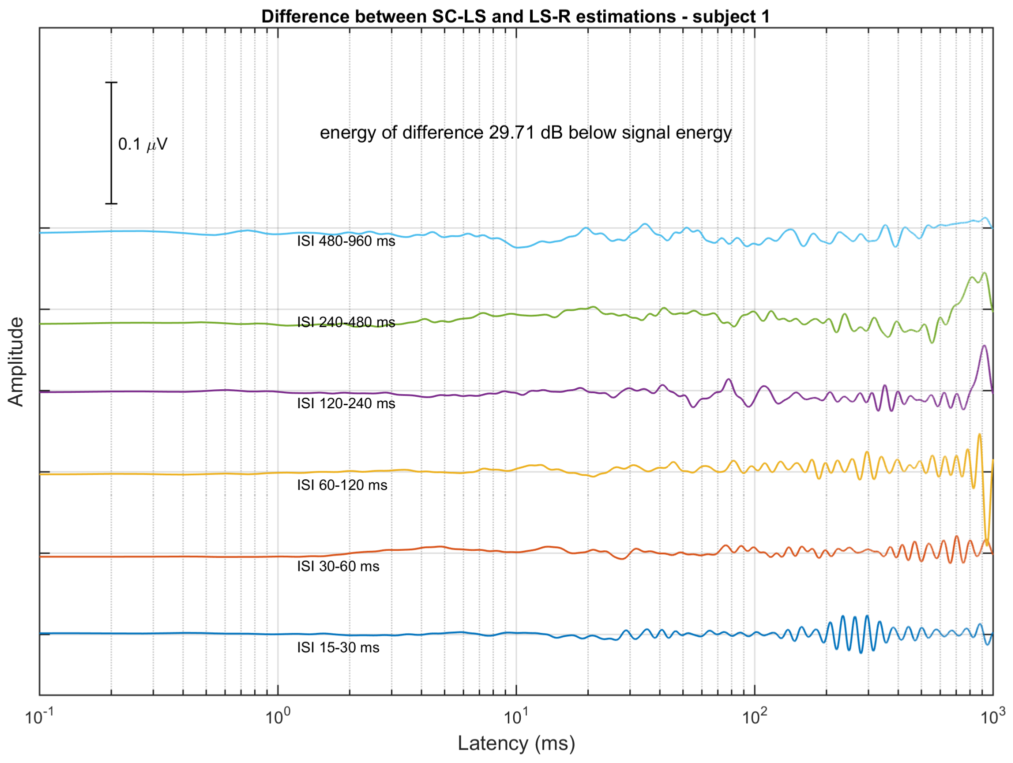
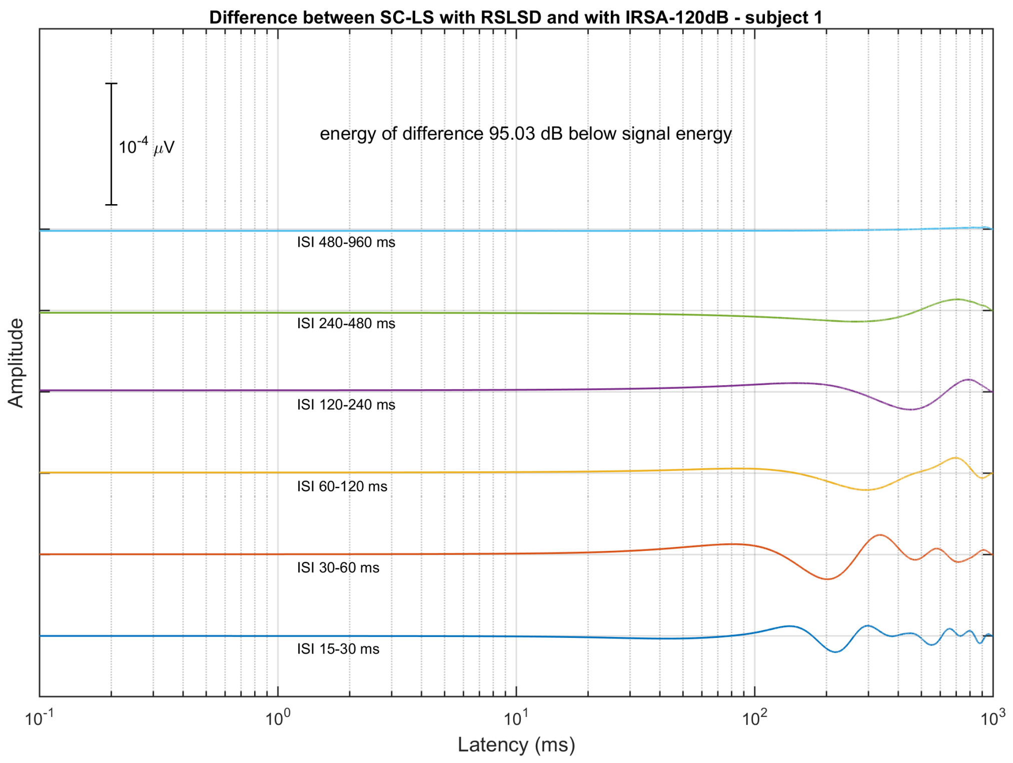


Figure 11: Difference between the SC-LS and the LS estimations (top) and between SC-LS and LS-R estimations (bottom) obtained with the algorithms based on matrix division. These plots correspond to subject 1 (and the responses presented in the figure 10). Note that the scale is 1 μV for the top figure but 0.1 μV for the bottom figure.



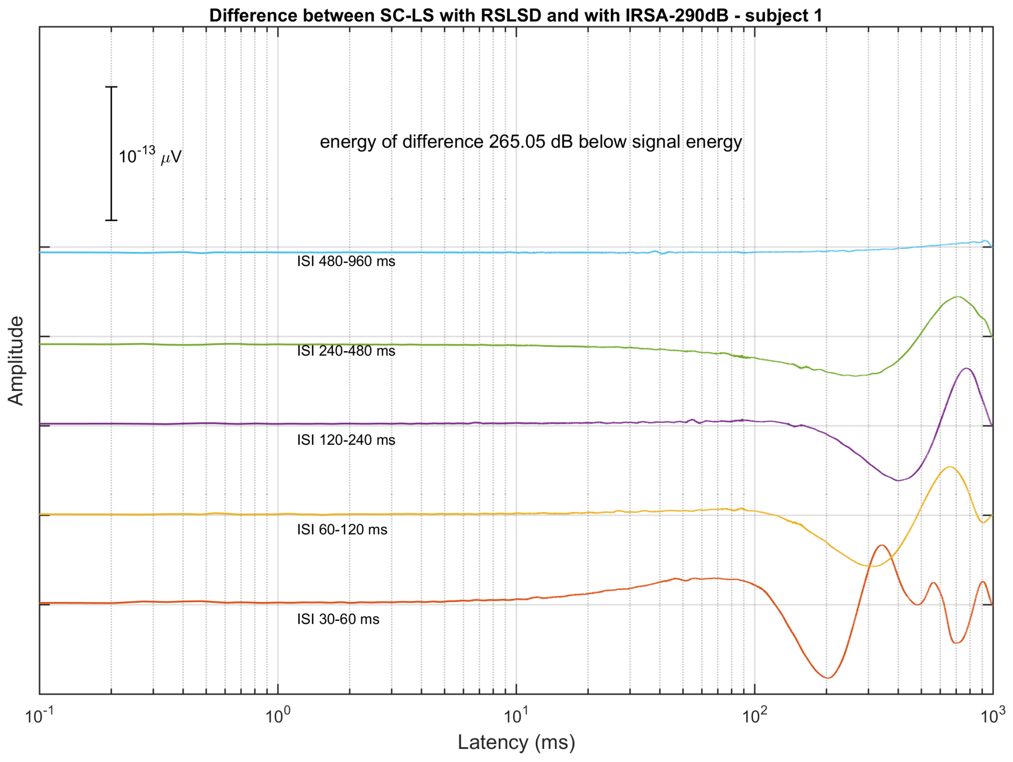


Figure 12: Difference for the SC-LS based estimations. Comparison between the estimations obtained with SC-LS$_{\text{MD}}$ and SC-LS$_{\text{It-120dB}}$ (top). Comparison between the estimations obtained with SC-LS$_{\text{MD}}$ and SC-LS$_{\text{It-290dB}}$ (bottom). These plots correspond to subject 1. Note that the scale is 10^{-4} μV for the top figure and 10^{-13} μV for the bottom figure.

Table 4: SNR relative to SC-LS$_{MD}$ (i.e. using this AEP estimation as reference). The SNR is estimated as the ratio between the signal energy and the energy of the difference between the estimations to be compared. The table includes mean and standard deviation (in parenthesis) of the estimated SNR.

ISI (ms)	LS$_{MD}$ (RSLSD)	LS$_{It}$ (IRSA-290dB)	LS$_{It}$ (IRSA-120dB)	LS-R$_{MD}$ (RSLSD)	LS-R$_{It}$ (IRSA-290dB)	LS-R$_{It}$ (IRSA-120dB)	SC-LS$_{MD}$ (RSLSD)	SC-LS$_{It}$ (IRSA-290dB)	SC-LS$_{It}$ (IRSA-120dB)
480-960	10.18 (2.42)	10.18 (2.42)	10.18 (2.42)	35.06 (5.61)	35.06 (5.61)	35.06 (5.61)	-	282.52 (1.22)	112.27 (0.70)
240-480	9.63 (2.55)	9.63 (2.55)	9.63 (2.55)	32.16 (5.50)	32.16 (5.50)	32.16 (5.50)	-	266.61 (0.58)	96.59 (0.55)
120-240	10.67 (3.71)	10.67 (3.71)	10.67 (3.71)	24.97 (6.49)	24.97 (6.49)	24.97 (6.49)	-	265.05 (1.49)	95.31 (1.46)
60-120	12.24 (2.32)	12.24 (2.32)	12.24 (2.32)	28.13 (7.31)	28.13 (7.31)	28.13 (7.31)	-	264.26 (1.25)	94.45 (1.23)
30-60	13.44 (2.24)	13.44 (2.24)	13.44 (2.24)	30.30 (5.24)	30.30 (5.24)	30.30 (5.24)	-	265.18 (2.21)	95.60 (2.41)
15-30	12.30 (2.70)	12.30 (2.70)	12.30 (2.70)	34.22 (3.88)	34.22 (3.88)	34.22 (3.88)	-	146.92 (3.24)	96.70 (3.08)
Average	**11.41 (2.89)**	**11.41 (2.89)**	**11.41 (2.89)**	**30.81 (6.50)**	**30.81 (6.50)**	**30.81 (6.50)**	**-**	**248.42 (46.35)**	**98.49 (6.51)**

Table 5: Comparison among the estimations provided by the different procedures. For each pair of methods the SNR associated to the difference between the respective estimations is evaluated as the ratio between the signal energy and the energy of the difference. For example, the energy of the difference between LS$_{It\text{-}290dB}$ and LS$_{It\text{-}120dB}$ is 98.19 dB below the energy of the AEP response.

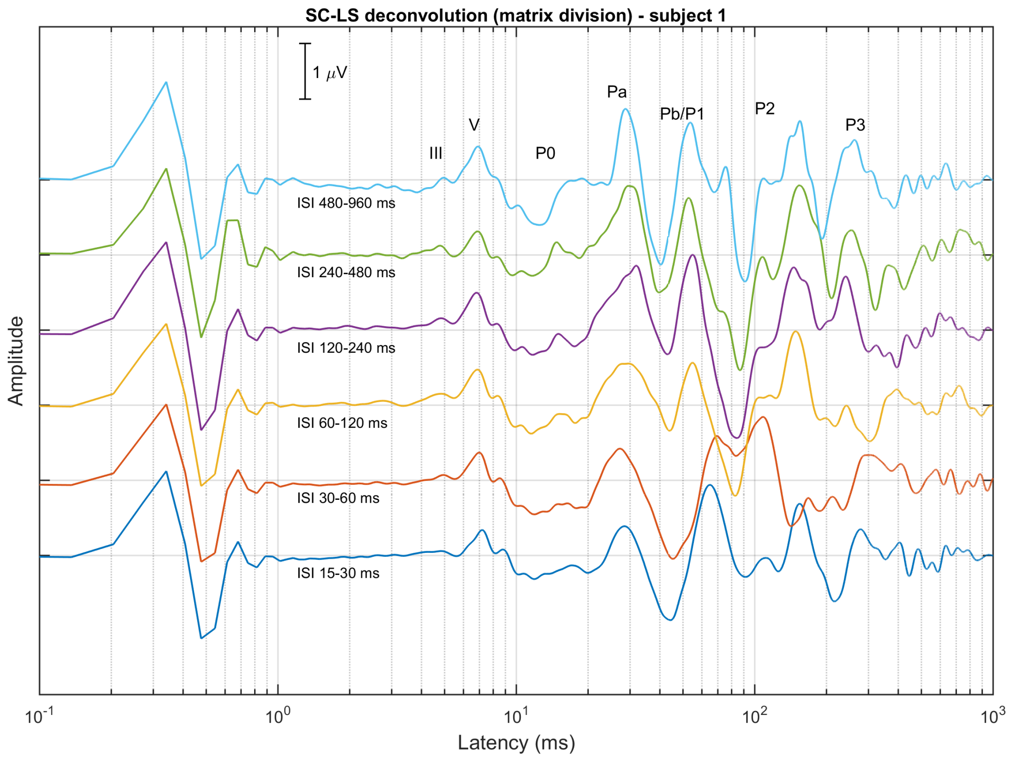
	LS$_{MD}$ (RSLSD)	LS$_{It}$ (IRSA-290dB)	LS$_{It}$ (IRSA-120dB)	LS-R$_{MD}$ (RSLSD)	LS-R$_{It}$ (IRSA-290dB)	LS-R$_{It}$ (IRSA-120dB)	SC-LS$_{MD}$ (RSLSD)	SC-LS$_{It}$ (IRSA-290dB)	SC-LS$_{It}$ (IRSA-120dB)
LS$_{MD}$ (RSLSD)	-	248.02	98.19	11.56	11.56	11.56	11.41	11.41	11.41
LS$_{It}$ (IRSA-290dB)	248.02	-	98.20	11.56	11.56	11.56	11.41	11.41	11.41
LS$_{It}$ (IRSA-120dB)	98.19	98.20	-	11.56	11.56	11.56	11.41	11.41	11.41
LS-R$_{MD}$ (RSLSD)	11.56	11.56	11.56	-	247.79	97.87	30.81	30.81	30.81
LS-R$_{It}$ (IRSA-290dB)	11.56	11.56	11.56	247.79	-	97.88	30.81	30.81	30.81
LS-R$_{It}$ (IRSA-120dB)	11.56	11.56	11.56	97.87	97.88	-	30.81	30.81	30.81
SC-LS$_{MD}$ (RSLSD)	11.41	11.41	11.41	30.81	30.81	30.81	-	248.42	98.49
SC-LS$_{It}$ (IRSA-290dB)	11.41	11.41	11.41	30.81	30.81	30.81	248.42	-	98.49
SC-LS$_{It}$ (IRSA-120dB)	11.41	11.41	11.41	30.81	30.81	30.81	98.49	98.49	-

The quality of the responses cannot be evaluated in the experiments with real EEGs, because there is no reference to be considered as clean signal. Instead, the differences among the AEP estimations can be evaluated, as in the figures 11 and 12. Table 4 reports the SNR estimated using the SC-LS$_{MD}$ responses as reference. As can be observed, the energy of the difference between the LS estimations and the SC-LS estimations is only about 11 dB below the response energy, while that for the LS-R estimations is about 31 dB below the response energy. The SC-LS estimations obtained with the iterative algorithm are very close to those obtained with the matrix division algorithm, with energy of the difference 98 and 248 dB below the response energy (for 120 and 290 dB in the convergence criterion, respectively). These results, averaged for all the subjects, are consistent with those in the figures 11 and 12. In order to provide a more complete comparison, table 5 compares each pair of procedures in terms of the ratio between the response energy and the energy of the difference of the responses to be compared. These results are consistent with those in table 4.

7.3 Individual AEP responses estimated for each subject and grand average

The responses provided by the SC-LS$_{MD}$ algorithm for each subject included in the EEG database are presented in figures 13, 14, 15 and 16.

Finally, figure 17 represents the grand average AEP responses (including also the individual AEP responses) for the different ISI conditions. Since subject 7 present a particularly large PAM response at 15 ms, (and also a particularly large negative peak around 70 ms), the grand average has also been estimated excluding this particular case.



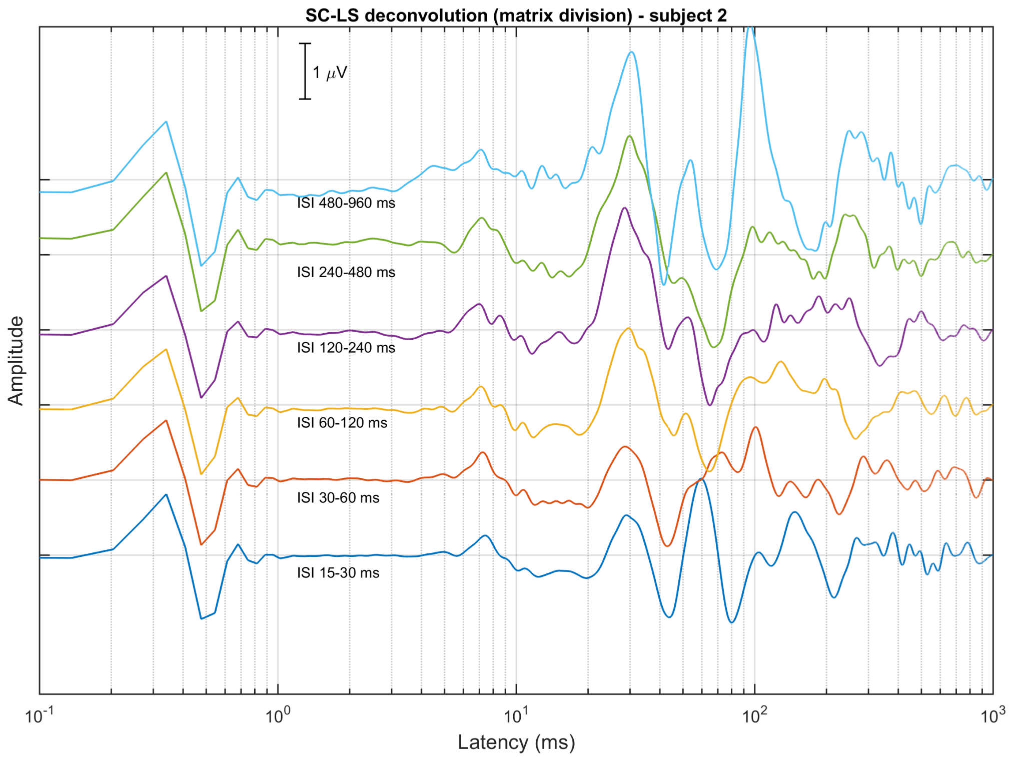
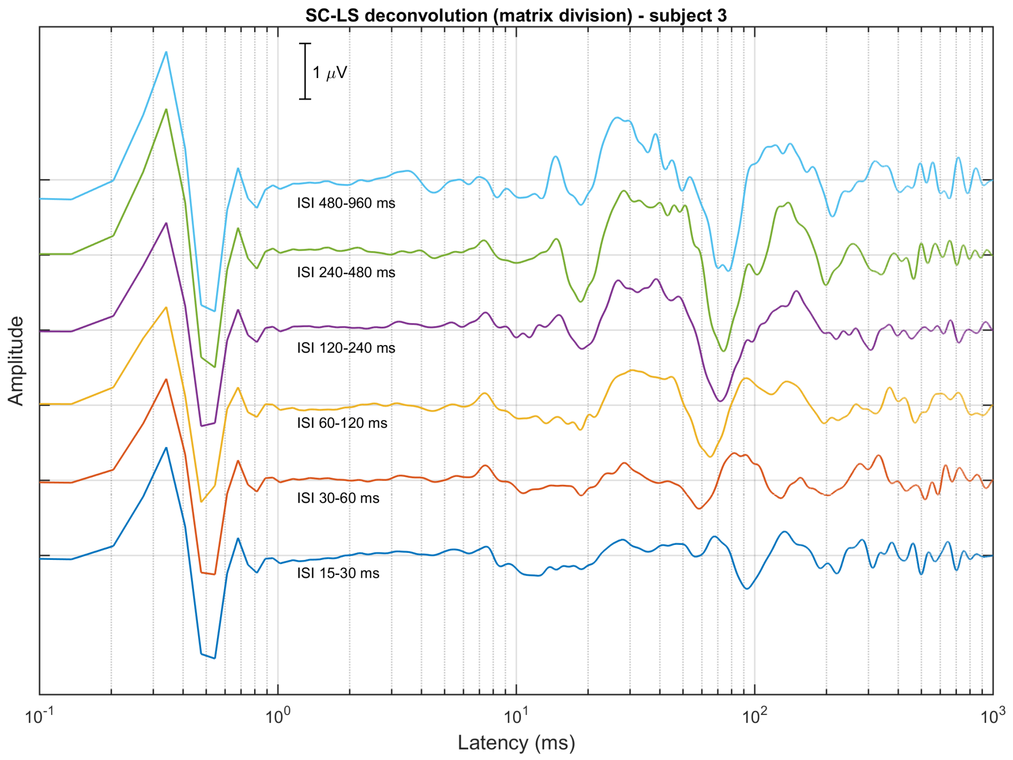


Figure 13: SC-LS$_{\text{MD}}$ estimations of the AEP responses for subjects 1 and 2.



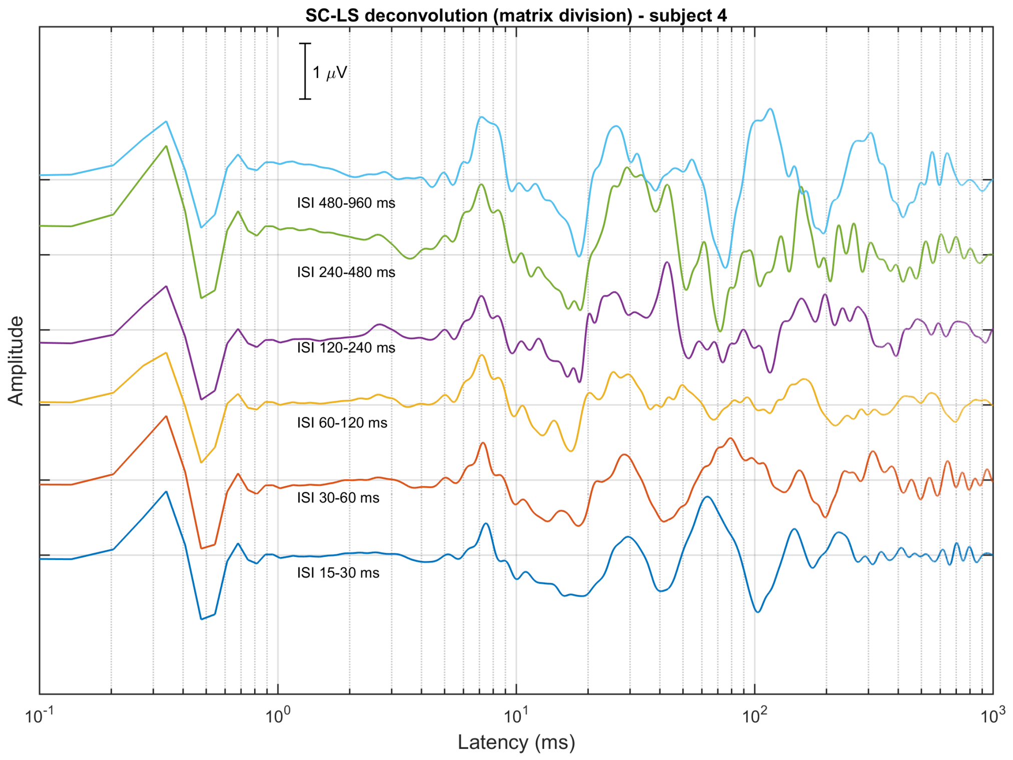
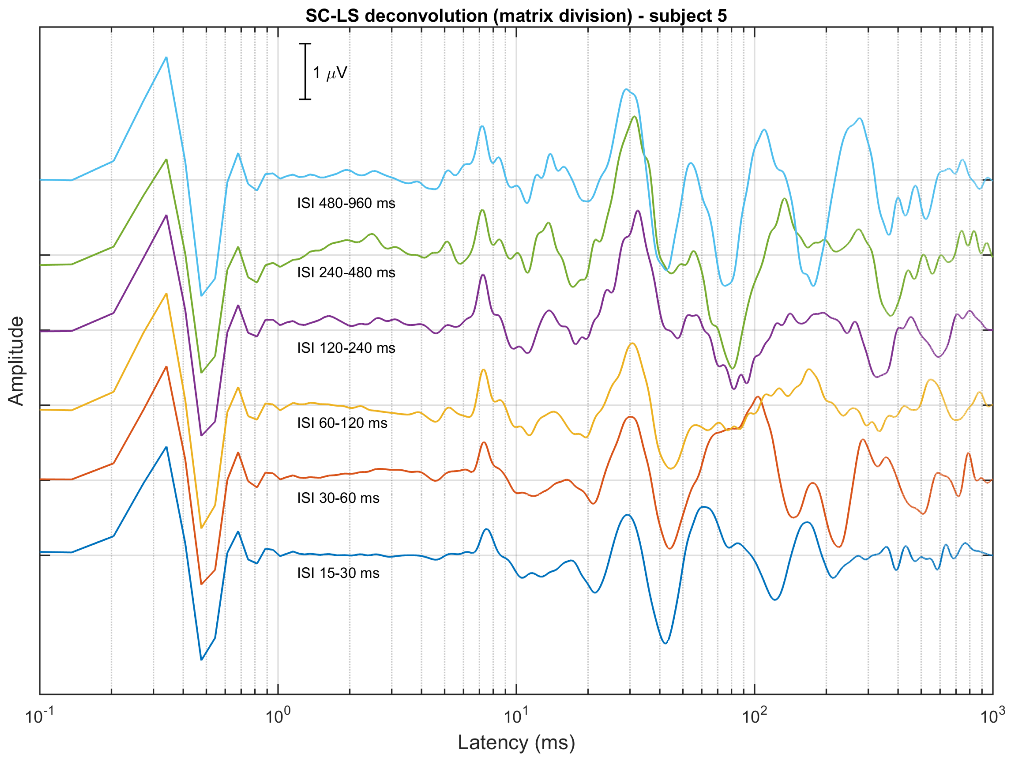


Figure 14: SC-LS$_{MD}$ estimations of the AEP responses for subjects 3 and 4.



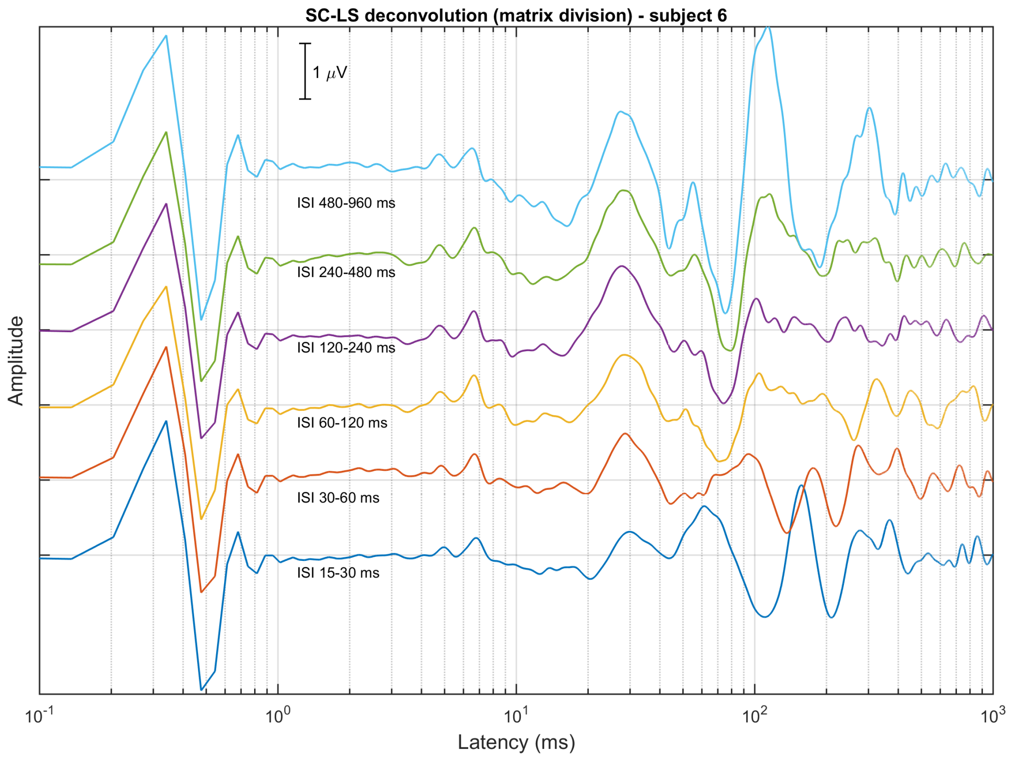
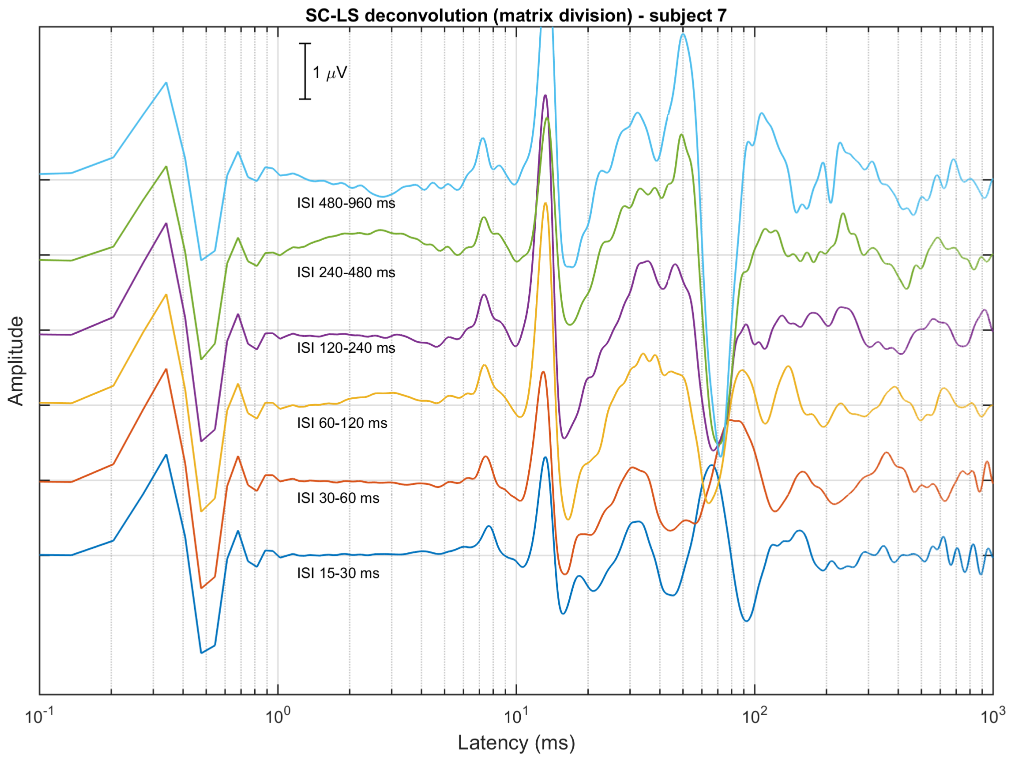


Figure 15: SC-LS$_{MD}$ estimations of the AEP responses for subjects 5 and 6.



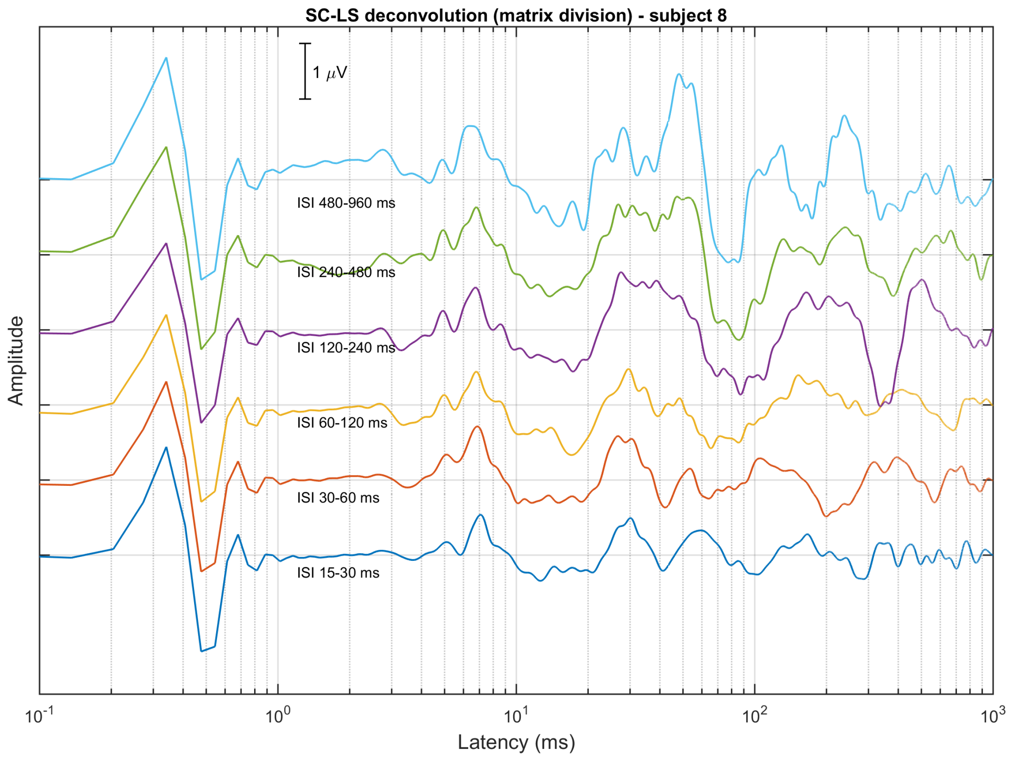


Figure 16: SC-LS$_{\mathrm{MD}}$ estimations of the AEP responses for subjects 7 and 8.

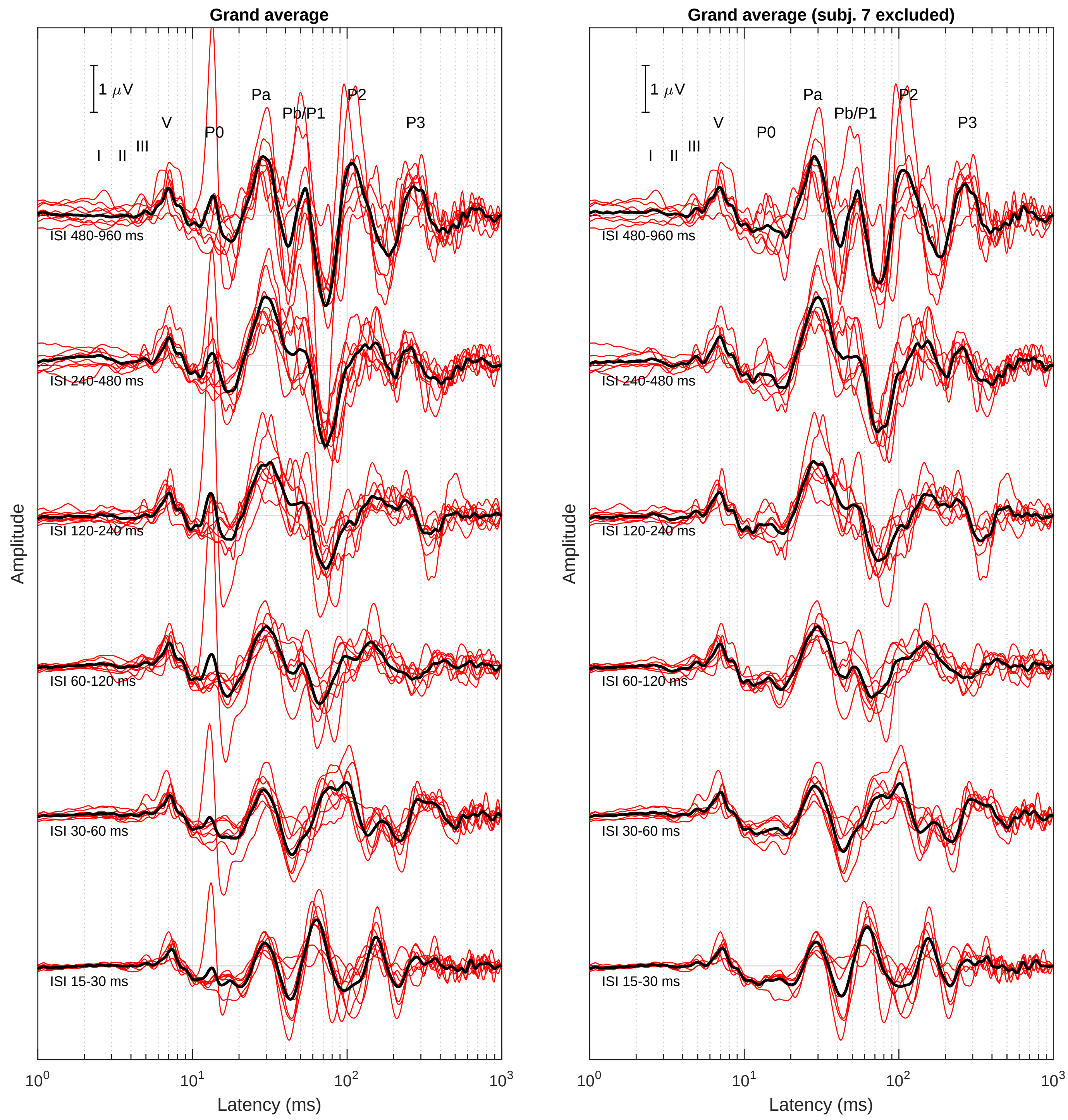


Figure 17: Left panel: grand average and individual AEP responses including the 8 subjects of the database. Right panel: grand average and individual AEP responses including all the subjects except subject 7.

References

[1] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*. http://matrixcookbook.com: Nov 15, 2012.

[2] W. H. Press, S. A. Teutolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The art of Scientific Computing.* New York: Cambridge University Press, 2nd ed., 2002.

[3] F. Hayashi, *Econometrics*. New Jersey: Princeton University Press, 2000.

[4] A. de la Torre, J. T. Valderrama, J. C. Segura, and I. M. Alvarez, "Matrix-based formulation of the iterative randomized stimulation and averaging method for recording evoked potentials," *Journal of the Acoustical Society of America*, vol. 146, pp. 4545–4556, 2019.

[5] A. de la Torre, J. T. Valderrama, J. C. Segura, and I. M. Alvarez, "Latency-dependent filtering and compact representation of the complete auditory pathway response," *Journal of the Acoustical Society of America*, vol. 148, pp. 599–613, 2020.