CCIT Report #467

February 2004

Sparse Multisensor Signal Reconstruction

Dmitry Model, Michael Zibulevsky

February 4, 2004

1 Abstract

We propose a technique of multisensor signal reconstruction based on the assumption, that source signals are spatially sparse, as well as have sparse [wavelet-type] representation in time domain. This leads to a large scale convex optimization problem, which involves l_1 norm minimization. The optimization is carried by the Truncated Newton method, using preconditioned Conjugate Gradients in inner iterations. The byproduct of reconstruction is the estimation of source locations.

2 Introduction

The solution of the "Cocktail Party" problem is the active research field. However none of the developed techniques provides an ideal solution. Yet another active research area is source localization. In this paper we propose to benefit from both fields in order to receive a more precise and stable solution.

Our technique is based on the assumption, that incoming signals can be sparsely represented in an appropriate basis or frame (e.g., via the short time Fourier transform, Wavelet transform, Wavelet Packets, etc.). This idea is exploited, for example, in [1],[2]. We also assume that there are few stationary sources, and that they are sparsely located in space. The last assumption is used in [3] and [4]. The combination of both assumptions can lead to an improved performance, as demonstrated by our simulations. An additional advantage of our method, is that it deals with the sensor array model in time domain, and thus is applicable for both narrowband and wideband signals.

The solution of our problem is the restored signals in each location. Only the locations, from which the signals have actually arrived, will contain signals with relatively large energy, others will contain only noise, suppressed by our method and, hence, relatively low energy. Thus, the byproduct of our solution is an estimate of the source locations.

3 Observation Model

Consider several source signals impinging upon an array of n sensors. Let $\{\theta_1, \ldots, \theta_m\}$ be a discrete grid of all source locations. The arriving signals are sampled and represented in discrete time by T time samples. Hence, the sources can be represented by an $m \times T$ matrix S, whose i-th row represents the signal from the i-th direction. In the same manner, we can introduce the sensor measurement matrix, Y.

Signal from different source position arrives to each sensor with different delay and, possibly, different attenuation. This leads to the following observation model:

$$Y = \mathcal{A}S + N \tag{1}$$

where N stands for the measurement noise matrix; $\mathcal{A}$ denotes 'mixing operator', which *shifts*, *attenuates* and *sums* incoming signals modelling the real environment. Note, that the operator $\mathcal{A}$ written in an explicit matrix form will have the huge dimensions of $n \times mT$, hence, for optimization, it is more convenient to implement the product $Y = \mathcal{A}S$ by a series of *shifts*, *multiplications* and *sums* actions:

$$y_i = \sum_{j=1}^{m} \alpha_{ji} U_{\Delta_{ji}}(s_j) \tag{2}$$

where y_i is the i-th row of the sensor measurement matrix, Y; s_j is the j-th row of sources' matrix S; α_{ji} represents attenuation of the j-th source toward the i-th sensor; $U_{\Delta_{ji}}$ is a shifting operator and Δ_{ji} is the delay of the j-th source toward the i-th sensor.

In the same manner we can implement the application of the adjoint op-

erator $X = \mathcal{A}^* Y$ by a series of *shifts*, *multiplications* and *sums* actions:

$$x_j = \sum_{i=1}^{n} \alpha_{ji} y_i (1 + \Delta_{ji} : T + \Delta_{ji}) \tag{3}$$

x_j and y_i refer to j-th and i-th rows of X and Y respectively. Matlab-like $y_i(1 + \Delta_{ji} : T + \Delta_{ji})$ stands for the T-length subvector of y_i, starting at $1 + \Delta_{ji}$ position.

As mentioned above, we work with the discrete-time signals. Therefore, a problem arises when Δ_{ji} is not integer. A straightforward solution is to replace the fractional delays with the rounded ones. However, this approach significantly limits the spatial resolution. A better approach suggests upsampling of signals prior to applying the $\mathcal{A}$ operator. The upsampling may be produced using some interpolation kernel.

Let $\mathcal{I}_{N_{up}}$ denote upsampling by factor N_{up} operator, and if S is an $m \times T$ matrix, then $S_{up} = \mathcal{I}_{N_{up}} S$ is $m \times TN_{up}$ matrix. Note, that the $1 + N_{up}(i-1)$ -th column of S_{up} is equal to the i-the column of S ($1 \leq i \leq T$). Other columns should be calculated using interpolation.

Suppose, we want to calculate the j-th column, S^j_{up}, of the matrix S_{up}. This column corresponds to the time point, laying between the samples $k = \lceil \frac{j}{N_{up}} \rceil$ and $k+1$ of the original signal ($\lceil\ \rceil$ is the ceiling operator). The distances between the above time point and the closest samples of original signal are $d^- = (j - (k-1)N_{up} - 1)/N_{up}$ to the left sample and $d^+ = 1 - d^-$ to the right sample (measured in sampling periods T_s). Finally, if h is the interpolation kernel, N_{io} is an interpolation order and S^k is the k-th column of S, then:

$$S^j_{up} = \sum_{l=-N_{io}+1}^{N_{io}} h(l - d^-) S^{k+l} \tag{4}$$

The adjoint operator $\mathcal{I}^*_{N_{up}}$ translates an $m \times TN_{up}$ matrix S_{up} into $m \times T$ matrix $S_r = \mathcal{I}^*_{N_{up}} S_{up}$. Using the above notations, we can write the following formula for the S^k_r - the k-th column of matrix S_r:

$$S^k_r = \sum_{l=-N_{io}*N_{up}}^{N_{io}*N_{up}} h\left(\frac{l}{N_{up}}\right) S^{N_{up}(k-1)+l}_{up} \tag{5}$$

Now, in our model we will use the modified operators

$$\hat{\mathcal{A}} = \mathcal{A} \cdot \mathcal{I}_{N_{up}} \qquad \hat{\mathcal{A}}^* = \mathcal{I}^*_{N_{up}} \cdot \mathcal{A}^* \tag{6}$$

instead of $\mathcal{A}$ and $\mathcal{A}^*$, but for simplicity, we will continue to denote the modified operators as $\mathcal{A}$ and $\mathcal{A}^*$. Note, that after upsampling, we should adjust Δ_{ji} to be $\Delta_{ji} * N_{up}$. We will still need to round $\Delta_{ji} * N_{up}$ to the closest integer, but now the rounding error is N_{up} times less. In our simulations we used $N_{up} = 10$ and the 'sinc' interpolation kernel.

4 Method Description

We assume that the sources S are sparsely representable in some basis or overcomplete system of functions [5] (e.g. Gabor, wavelet, wavelet packet, etc.). In other words, there exists some operator Φ, such that $S = C\Phi$, and the matrix of coefficients, C, is sparse. We use the objective function of the following form:

$$F(C) = F_1(C) + F_2(C) + F_3(C) \tag{7}$$

where $F_1(C)$ is the l_2-norm-based data fidelity term; $F_2(C)$ is the temporal sparsity regularizing term, which is intended to prefer sparsely representable signals; $F_3(C)$ is the spatial sparsity regularizing term, which is intended to prefer solutions with the source signals concentrated in a small number of locations. $F_2(C)$ is based on the l_1-norm, which is proved to be effective in forcing sparsity [5]. Then, the objective function can be written as:

$$F(C) = \frac{1}{2} \left\| Y - \mathcal{A}(C\Phi) \right\|_F^2 + \mu_1 \sum_{i,j} |c_{ij}| + \mu_2 \sum_{i=1}^{m} \|c_i\|_2 \tag{8}$$

where c_i denotes the i-th row of the matrix C (the i-th source' coefficients), and c_{ij} is the j-th element in c_i. The scalars μ_1 and μ_2 are used to regulate the weight of each term. And $\|X\|_F = \sqrt{\sum_{ij} X_{ij}^2}$ denotes a Frobenius norm of matrix X.

In order to minimize the objective (8) numerically, we use a smooth approximation of the l_2-norm, having the following form:

$$\psi(x) = \sqrt{\sum_i x_i^2 + \epsilon} \approx \|x\|_2 \tag{9}$$

the approximation becomes more precise as $\epsilon \to 0$. It can be easily seen, that if ψ is applied to a single element of x - it becomes the smooth approximation of absolute value:

$$\psi(x_i) = \sqrt{x_i^2 + \epsilon} \approx |x| \tag{10}$$

Using (9) and (10), we obtain the following objective function:

$$F(C) = \frac{1}{2} \|Y - \mathcal{A}(C\Phi)\|_F^2 + \mu_1 \sum_{i,j} \psi(c_{ij}) + \mu_2 \sum_{i=1}^{m} \psi(c_i) \tag{11}$$

We can efficiently calculate both the $\mathcal{A}S$ and the $\mathcal{A}^*Y$ products, which enables us to calculate the gradient matrix G and the product of the Hessian operator $\mathcal{H}$ with an arbitrary matrix X (see appendix A). Hence, the objective (11) can be minimized by one of the numerical optimization methods, for example the *Quasi Newton* method. A problem arises when the dimension of the problem growths. The memory consumption and iteration cost grow as $(mT)^2$. This circumstance leads us to the usage of the *Truncated Newton* method [6],[7]. In the *Truncated Newton* method the Newton direction d is found by the approximate solution of the system of linear equations $Hd = -g$. This is done by the *linear Conjugate-Gradients* method. We use diagonal preconditioning in order to further speed up the optimization [8]. Note that in *Truncated Newton* method, the memory consumption growth linearly with the number of variables. This enables us to solve large problems with fair performance. See Appendix B for detailed description of *Truncated Newton* and preconditioned *Conjugate-Gradients* algorithms.

5 Computational Experiments

Our simulations were restricted to $2D$ model, far field and sensors lined up with constant distances. The delay of the j-th source location toward the i-th sensor is easy to calculate, given the geometrical position of each sensor and assuming that the source is far enough, so that signal arrives as a planar wave (far field assumption). Note that it is straightforward to extend our simulations to the general case. It only requires to recalculate the delay from each location to each sensor.

The experiment setup is as following: 8 sensors are lined up with $\frac{\lambda_{min}}{2} = \frac{1}{2}\frac{\mathcal{C}}{f_{max}}$ distance (we assume our signal to be band limited, and f_{max} denoting the

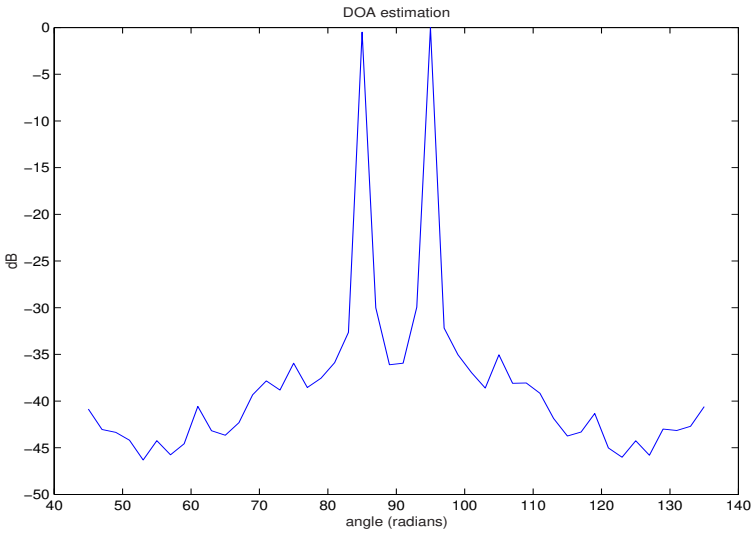

Figure 1: DOA estimation, (no noise)

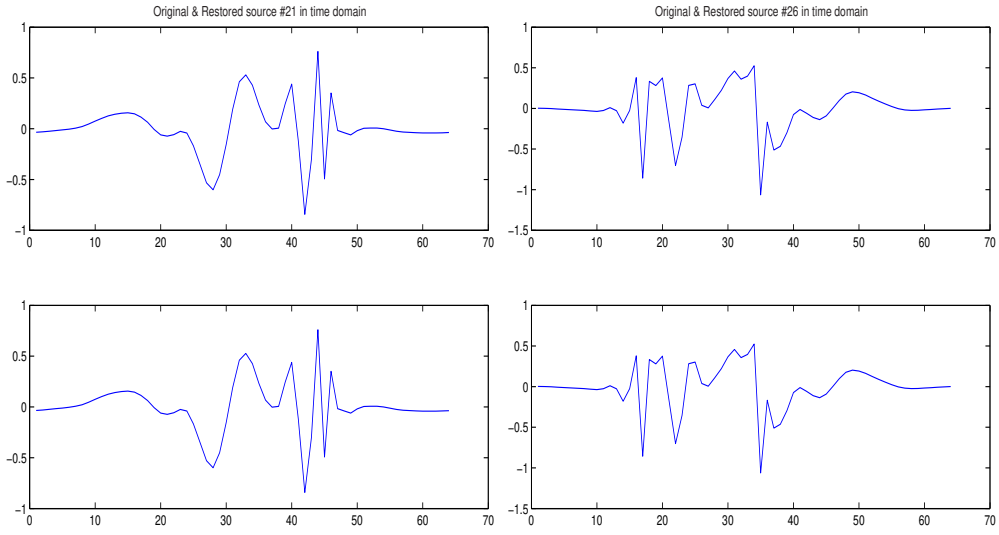

Figure 2: Top: sources from 2 active directions, bottom: restored sources

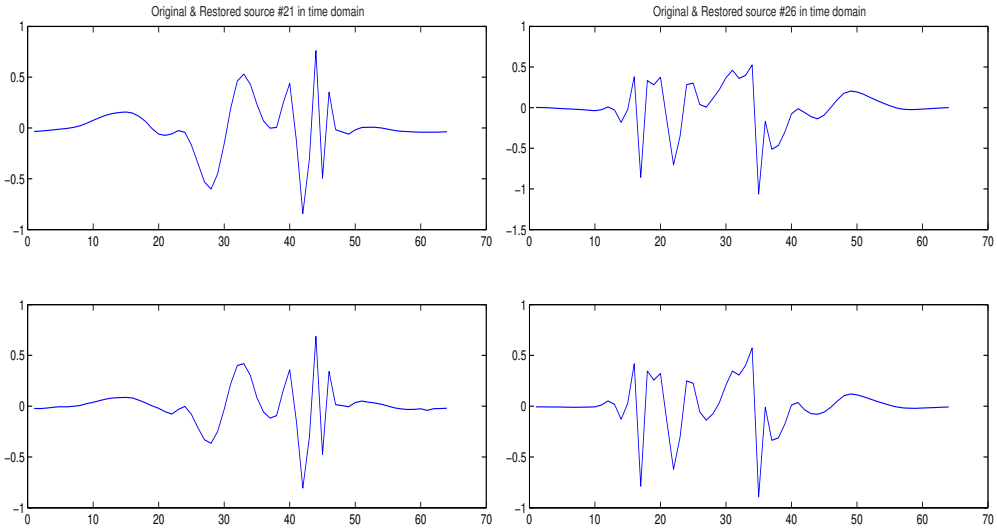

Figure 3: Top: sources from 2 active directions, bottom: restored sources

highest frequency). Signals are upcoming from 45 possible directions, and they are 64 time samples-long. The environment is noisy, with $SNR = 5dB$. There are only 2 active sources, located very close to each other - 10^o. In these conditions conventional methods, such as beamforming and MUSIC fail to superresolve them (as shown in [3],[4]).

We have generated the sensors' measurement matrix Y in the following way: at first, we have generated the sparse coefficients matrix C. Next, the source signals were created $S = C\Phi$ and finally $Y = \mathcal{A}S$ ($\mathcal{A}$ defined in (6) and $N_{up} = 10$).

In the first experiment we have checked that our algorithm can reconstruct signals in noise-free environment. The experiment was successful, and the algorithm has correctly determined the source positions (Figure 1) and has produced reconstruction with less than $5 * 10^{-3}$ reconstruction error (Figure 2). The error was calculated according to $\frac{\|s_{init} - s_{rec}\|_2}{\|s_{init}\|_2}$.

In the second experiment, we have also added white Gaussian noise to the matrix Y. The contaminated by the noise matrix Y was used as an input to our algorithm. After successful optimization, we have checked the signals

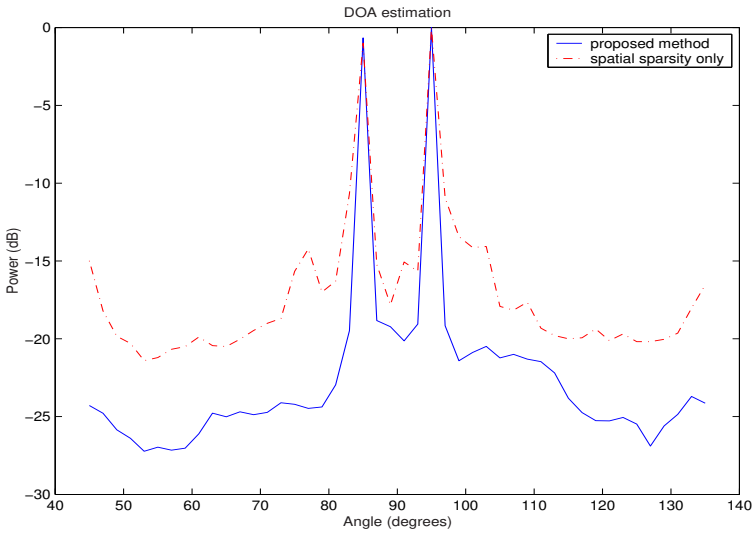

Figure 4: DOA estimation, (SNR=5 db)

(original vs. reconstructed) from the active directions. As one can see in Figure 3 the active signals were restored rather accurately.

In addition, we have checked our method for DOA estimation, by computing the energy of the restored signal at each direction. We have compared our technique with the method based on spatial sparsity only, in spirit of ([3],[4]), by setting $\mu_1 = 0$ in (11). It can be seen from Figure 4 that both methods correctly identify the active directions, however sidelobes are about $5dB$ lower when temporal sparsity is enforced along with spatial sparsity.

6 Conclusions

We have presented a method for reconstruction of multiple source signals from multi-sensor observations, based on temporal-spatial sparsity. We derive the expressions for efficient computation of the gradient, multiplication by Hessian and diagonal preconditioning, necessary for *Truncated Newton* programming.

Computational experiments showed the feasibility of our method. The use of temporal sparsity along with spatial sparsity further lowers the sidelobes.

However, more simulations and comparison to other methods should be completed before we can judge the method's performance.

We are planning to test our method in the case of near field sources. As well we wish to further speed up the optimization.

7 Appendix A. Gradient and Hessian of the objective function

In order to use the *Truncated Newton* method, we need to calculate the gradient G of the objective (11), as well as to implement the product of the Hessian $\mathcal{H}$ with an arbitrary matrix X. Note that $\mathcal{H}$ is a tensor, but if we parse the matrix variable C into a long vector, then a Hessian will be represented by a matrix H. We will use these notations throughout this appendix. We also derive multiplication by the diagonal of H, required for preconditioned *Conjugate-Gradients*.

Let us start with the first term in (11). We will define a new operator $\mathcal{B}$ in the following way:

$$\mathcal{B}C = \mathcal{A}(C\Phi) \qquad \mathcal{B}^*X = (\mathcal{A}^*X)\Phi^* \tag{12}$$

This enables us to write the first term in (11) as: $F_1 = \frac{1}{2}\left\|\mathcal{B}C - Y\right\|_F^2$. If we introduce new variable $U = \mathcal{B}C - Y$, then $F_1 = \frac{1}{2}\left\|U\right\|_F^2 = \frac{1}{2}Tr(U^TU)$. Hence, $dF_1 = \frac{1}{2}\left(Tr(U^TdU) + Tr(dU^TU)\right) = Tr(U^TdU)$. Substituting U and $dU = \mathcal{B}dC$ yields $dF_1 = Tr\left((\mathcal{B}C - Y)^T\mathcal{B}dC\right) = \langle \mathcal{B}C - Y, \mathcal{B}dC\rangle = \langle \mathcal{B}^*(\mathcal{B}C - Y), dC\rangle$. Recall that $dF = \langle G, dC\rangle$, and we get the gradient

$$G_1(C) = \mathcal{B}^*(\mathcal{B}C - Y) \tag{13}$$

Now we can substitute the expressions for $\mathcal{B}$ and $\mathcal{B}^*$ from (12) and we will receive:

$$G_1(C) = \left(\mathcal{A}^*\left(\mathcal{A}\left(C\Phi\right) - Y\right)\right)\Phi^* \tag{14}$$

In order to calculate the multiplication of the Hessian operator $\mathcal{H}$ by an arbitrary matrix X we need to recall that $dG(C) = \mathcal{H}dC$. By (13) $dG_1(C) = \mathcal{B}^*(\mathcal{B}dC)$, and thus for an arbitrary X

$$\mathcal{H}_1X = \mathcal{B}^*(\mathcal{B}X) \tag{15}$$

which gives after substituting $\mathcal{B}$ and $\mathcal{B}^*$ from (12):

$$\mathcal{H}_1 X = \left(\mathcal{A}^* \left(\mathcal{A}\left(X\Phi\right)\right)\right)\Phi^* \tag{16}$$

Parentheses are used to ensure correct order of multiplications, $\mathcal{A}X$ and $\mathcal{A}^*X$ are defined in (2),(3),(4),(5),(6).

In order to proceed with the second and the third terms in (11), we need to use the gradient and Hessian of (9):

$$\nabla\psi(x) = \frac{1}{\psi(x)}x \tag{17}$$

$$\begin{aligned}\left(\nabla^2\psi(x)\right)_{ii} &= -\frac{1}{\psi^3(x)}x_i^2 + \frac{1}{\psi(x)} \\ \left(\nabla^2\psi(x)\right)_{ij} &= -\frac{1}{\psi^3(x)}x_i x_j \qquad (i \neq j)\end{aligned} \tag{18}$$

where $(\nabla^2\psi(x))_{ii}$ and $(\nabla^2\psi(x))_{ij}$ are diagonal and off diagonal elements elements of $\nabla^2\psi(x)$ respectively. Now, by straightforward calculations we can write down the gradients of the second and the third term in (11):

$$(G_2)_{ij} = \mu_1 \frac{1}{\psi(c_{ij})}c_{ij} \tag{19}$$

$$(G_3)_{ij} = \mu_2 \frac{1}{\psi(c_i)}c_{ij} \tag{20}$$

note, that the gradient of (11) is a matrix, because our variable C is also a matrix (hence G_1,G_2 and G_3 are also matrices). It can be noticed in (19), that all elements of G_2 are independent, and thus the H_2 matrix will be diagonal. It is convenient to 'pack' the diagonal of $\mathcal{H}_2$ into a matrix with the same size as C row by row. Let us denote the packed matrix as $\tilde{H}_2$:

$$\tilde{H}_{2_{ij}} = \mu_1 \left(-\frac{1}{\psi^3(c_{ij})}c_{ij}^2 + \frac{1}{\psi(c_{ij})}\right) \tag{21}$$

it is obvious, that

$$\mathcal{H}_2 X = \tilde{H}_2 \odot X \tag{22}$$

where $\odot$ is element-wise multiplication.

In order to define the multiplication $\mathcal{H}_3 X$ we need to rewrite the equation (18):

$$\nabla^2 \psi(c_i^T) = \frac{1}{\psi^3(c_i^T)} c_i^T c_i + \frac{1}{\psi(c_i^T)} I \tag{23}$$

where I represents the identity matrix. Now it is easy to define the i-th row of $\mathcal{H}_3 X$:

$$(\mathcal{H}_3 X)_i = \mu_2 \left(-\frac{1}{\psi^3(c_i^T)} c_i (c_i x_i^T) + \frac{1}{\psi(c_i^T)} x_i \right) \tag{24}$$

where x_i is the i-th row of matrix X.

This calculus is sufficient for the *Truncated Newton* method. However, in order to use *Preconditioned* Conjugate Gradients method for inner iterations, we need to define the diagonal of the Hessian of (11).

We will calculate the elements in the diagonal of $\mathcal{H}_1$ in the following manner: let E be a zero matrix with only one non-zero element equal to 1 at an arbitrary location - i-th row and j-th column. Then:

$$\left(\tilde{H}_1\right)_{ij} = \langle E, \mathcal{H}_1 E \rangle \tag{25}$$

where $\tilde{H}_1$ is a diagonal of $\mathcal{H}_1$ packed in the same manner as a diagonal of H_2 in (21).

It follows from (15) that $\langle E, \mathcal{H}_1 E \rangle = \langle E, \mathcal{B}^*(\mathcal{B}E) \rangle = \langle \mathcal{B}E, \mathcal{B}E \rangle = \|\mathcal{B}E\|_F^2$, and if we substitute the expression for $\mathcal{B}$ from (12) we will receive $\left\langle E^T, \mathcal{H}_1 E \right\rangle = \|\mathcal{A}(E\Phi)\|_F^2$. The elements of $E\Phi$ will be all zeros, except for the i-th row which will be equal to the j-th row of Φ. After applying the operator $\mathcal{A}$ as described in (2),(3),(4),(5),(6), we will receive a *shifted*, *attenuated* and *upsampled* copy of j-th row of Φ in each row of $\mathcal{A}(E\Phi)$. And, finally, after taking the norm and using (2) and (25), we will receive:

$$\left(\tilde{H}_1\right)_{ij} = \left\| \left(\mathcal{I}_{N_{up}} \Phi\right)_j \right\|_2^2 \sum_{j=1}^{n} \alpha_{ij}^2 \tag{26}$$

where $\left(\mathcal{I}_{N_{up}} \Phi\right)_j$ is the j-th row of upsampled Φ.

The diagonal of $\mathcal{H}_2$ is already defined in (21). Finally, the diagonal of $\mathcal{H}_3$, packed in the same manner as a diagonal of $\mathcal{H}_2$, is given by:

$$(\tilde{H}_3)_{ij} = \mu_2 \left(-\frac{1}{\psi^3(c_i)} c_{ij}^2 + \frac{1}{\psi^3(c_i)} \right) \tag{27}$$

8 Appendix B. Truncated Newton method

In the algorithm description we will use the following notations: $f(C)$ - the objective function (11). G and $\mathcal{H}$ are the gradient and the Hessian of $f(C)$, respectively. The *Truncated Newton* method applied to the objective (11) has the following iterative scheme:

1. Start with an initial estimate C_0 of source coefficients
2. For $k = 1, 2, ...$ until convergence
   (a) Compute the current direction D_k by approximate solution of system of linear equations $\mathcal{H}D_k = -G_k$
   (b) Compute the step size α_k by exact or inexact line search: $\alpha_k = arg\min_\alpha f(C_k + \alpha D_k)$
   (c) $C_{k+1} = C_k + \alpha_k D_k$
3. End of loop

The step 2a is performed by the **preconditioned** *linear Conjugate Gradients*. We use the diagonal operator $\mathcal{W}$ for preconditioning. $\mathcal{W}$ has the same size and the diagonal as $\mathcal{H}$ - the Hessian of (11). Since $\mathcal{W}$ is diagonal, the calculation of $\mathcal{W}^{-1}$ is straightforward. Moreover, the optimization algorithm doesn't differ much from the regular *CG*:

1. Start with D_0, $R_0 = \mathcal{H}D_0 + G_k$, $\beta_0 = 0$, $P_0 = 0$
2. For $k = 1, 2, ...$
   (a) $P_k = -\mathcal{W}^{-1}R_k + \beta_{k-1}P_{k-1}$
   (b) $\gamma_k = \frac{\langle R_k, \mathcal{W}^{-1}R_k \rangle}{\langle P_k, \mathcal{H}P_k \rangle}$
   (c) $D_{k+1} = D_k + \gamma_k P_k$

(d) $R_{k+1} = R_k + \gamma_k \mathcal{H} P_k$

(e) $\beta_k = \frac{\langle R_{k+1}, \mathcal{W}^{-1} R_{k+1} \rangle}{\langle R_k, \mathcal{W}^{-1} R_k \rangle}$

3. End of loop

where $\langle A, B \rangle = Tr\left(A^T B\right) = \sum_{ij} a_{ij} b_{ij}$ is an inner product of two matrices A and B.

Note, that we are not looking for the exact solution of step 2a of *Truncated Newton* algorithm. Hence, we should stop our *CG* algorithm when we are close enough to the solution. One of the stop criteria may be a fixed number of steps. Other possible criteria is when the $\frac{\|R_k\|_2}{\|R_0\|_2}$ is low enough - say 10^{-3}.

References

[1] M. Zibulevsky and B. A. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary," *Neural Computations*, vol. 13, no. 4, pp. 863–882, 2001.

[2] M. Zibulevsky, B. A. Pearlmutter, P. Bofill, and P. Kisilev, "Blind source separation by sparse decomposition," in *Independent Components Analysis: Princeiples and Practice* (S. J. Roberts and R. M. Everson, eds.), Cambridge University Press, 2001.

[3] M. Çetin, D. M. Malioutov, and A. S. Willsky, "A variational technique for source localization based on a sparse signal reconstruction perspective," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 2965–2968, May 2002.

[4] D. M. Malioutov, M. Çetin, J. W. FisherIII, and A. S. Willsky, "Superresolution source localization through data-adaptive regularization," in *IEEE Sensor Array and Multichannel Signal Processing Workshop*, pp. 194–198, Aug. 2002.

[5] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1998.

[6] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, "Inexact newton methods," *SIAM Journal on Numerical Analysis*, vol. 19, pp. 400–408, 1982.

[7] S. Nash, "A survey of truncated-newton methods," *Journal of Computational and Applied Mathematics*, vol. 124, pp. 45–59, 2000.

[8] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. New York: Academic Press, 1981.