FAμST: speeding up linear transforms for tractable inverse problems

Luc Le Magoarou, Rémi Gribonval, Alexandre Gramfort

FAμST: SPEEDING UP LINEAR TRANSFORMS FOR TRACTABLE INVERSE PROBLEMS

Luc Le Magoarou[*], *Rémi Gribonval*[*] *and Alexandre Gramfort*[†]

[*]Inria, Rennes - Bretagne Atlantique, France
[†]Institut Mines-Telecom, Telecom ParisTech, CNRS LTCI, Paris, France

ABSTRACT

In this paper, we propose a technique to factorize any matrix into multiple sparse factors. The resulting factorization, called Flexible Approximate MUlti-layer Sparse Transform (FAμST), yields reduced multiplication costs by the matrix and its adjoint. Such a desirable property can be used to speed up iterative algorithms commonly used to solve high dimensional linear inverse problems. The proposed approach is first motivated, introduced and related to prior art. The compromise between computational efficiency and data fidelity is then investigated, and finally the relevance of the approach is demonstrated on a problem of brain source localization using simulated magnetoencephalography (MEG) signals.

***Index Terms*—** Inverse problems, Deconvolution, Matrix factorization, Fast algorithms, Brain source localization

1. INTRODUCTION

We consider the classical linear inverse problem setting where parameters $\boldsymbol{\gamma} \in \mathbb{R}^n$ are to be estimated from data $\mathbf{y} \in \mathbb{R}^m$ assuming a linear model with a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$:

$$\mathbf{y} \approx \mathbf{X}\boldsymbol{\gamma}. \tag{1}$$

The model is often further constrained by assuming that the data dimension (or number of measurements) is way smaller than the number of parameters: $m \ll n$. This configuration results in under-determined linear systems, for which the sparsest solution is often sought. Iterative algorithms designed to solve such problems involve many multiplications by the operator $\mathbf{X}$ and its transpose $\mathbf{X}^T$. Their computational cost [1] is thus dominated by such multiplications. In high dimension ($1 \ll m \ll n$), such operations that cost $\mathcal{O}(mn)$ flops for dense matrices can be impractical.

However, there exist dense matrices for which such products can be done way more efficiently. Indeed, popular transforms (Fourier, wavelets, DCT...) are associated with fast algorithms allowing to perform matrix/vector products in $\mathcal{O}(n \log n)$ or $\mathcal{O}(n)$ flops (these transform matrices are square, $m = n$). Such fast algorithms can be seen as M consecutive multiplications of the input vector by sparse matrices $\mathbf{S}_j$ (see, e.g., [2, Appendix A]), indicating that the transform matrices take the form[1]: $\mathbf{T} = \prod_{j=1}^{M} \mathbf{S}_j$. This *multi-layer sparse* structure is what allows for efficient algorithms.

Inspired by this structure, the objective of this paper is to approximate arbitrary dense matrices $\mathbf{X}$ such as those arising in inverse problems by multi-layer sparse forms:

$$\mathbf{X} \approx \prod_{j=1}^{M} \mathbf{S}_j. \tag{2}$$

Provided the factors $\mathbf{S}_j$ are sparse enough, the resulting Flexible Approximate MUlti-layer Sparse Transform (FAμST) allows to compute approximate products with $\mathbf{X}$ or $\mathbf{X}^T$ much more quickly. A FAμST can thus be used in any algorithm for solving the inverse problem in a computationally efficient way. The relative complexity of a FAμST with respect to the dense form somehow quantifies the computational gains:

Definition 1.1. *The Relative Complexity (abbreviated RC) is the ratio between the total number of non-zero entries in the FAμST and the number of non-zero entries of $\mathbf{X}$:*

$$RC := \frac{\sum_{j=1}^{M} \|\mathbf{S}_j\|_0}{\|\mathbf{X}\|_0}, \tag{3}$$

where $\|\cdot\|_0$ counts the total number of non-zero entries of its argument. The Relative Complexity Gain (RCG) is simply the inverse of the Relative Complexity ($RCG = 1/RC$).

The condition for the FAμST to be beneficial in terms of complexity compared to the dense form then writes mathematically: RC $\ll 1$ or equivalently RCG $\gg 1$. There is a natural tradeoff with the fidelity to the original operator which can be measured through the relative operator norm error:

$$\text{RE} = \frac{\left\|\mathbf{X} - \prod_{j=1}^{M} \mathbf{S}_j\right\|_2}{\|\mathbf{X}\|_2}, \tag{4}$$

where $\|\cdot\|_2$ is the operator norm.

We proposed in a recent paper [3] an algorithm that hierarchically factorizes an arbitrary input matrix $\mathbf{X}$ of interest into M sparse matrices, using the Proximal Alternating Linearized Minimization (PALM) method of [4]. Our contribution in this paper is to demonstrate the potential of this algorithm to speed up linear inverse problems, with controlled impact on the quality of the retrieved solutions.

This work was supported in part by the European Research Council, PLEASE project (ERC-StG- 2011-277906).

[1]The product being taken from right to left: $\prod_{i=1}^{N} \mathbf{A}_i = \mathbf{A}_N \cdots \mathbf{A}_1$

Related work. Other approaches to speed up sparse solvers in the context of linear inverse problems have been recently proposed in [5–7]. The principle is to adapt weights of a computational network corresponding to a fixed number of iterations of a recovery algorithm. Sparse solutions are then obtained in a fixed number of computations. In the present paper, instead of fixing the number of iterations, the strategy is to lower the cost of these iterations. These two ideas are complementary since they act on different components of the recovery algorithms, and could actually be combined together. The proposed factorization method can also be seen as a generalization of certain constrained dictionary learning methods imposing structures such as double-sparsity [8] or product of sparse convolutions [9], see [3] for a literature review.

2. ALGORITHM

The idea of the algorithm we recently introduced [3] is to factorize $\mathbf{X}$ into two factors first, and then iteratively factorize the leftmost factors into two factors, in order to increase the number of factors until the desired number M is attained (the factorization could also be carried out from the right just by transposing the input matrix). Each factorization is constrained in a way that imposes sparse factors. A global optimization step is inserted between each factorization into two factors in order to keep a relation with the input matrix $\mathbf{X}$. A simplified high level overview of the algorithm is given in Algorithm 1, where the constraint sets $\mathcal{E}_i$ and $\tilde{\mathcal{E}}_i$ impose sparsity, and are to be chosen carefully in order to obtain a good approximation of $\mathbf{X}$. For more details about the algorithm (implementation details, convergence conditions), see [3] or our technical report [2]. Practical examples of constraint sets will be considered in the next section.

Algorithm 1 Hierarchical factorization

Input: Operator $\mathbf{X}$; desired number of factors M; constraint sets $\mathcal{E}_i$ and $\tilde{\mathcal{E}}_i$, $i \in \{1 \dots M-1\}$.

1: $\mathbf{T}_0 \leftarrow \mathbf{X}$
2: **for** $i = 1$ to $M-1$ **do**
3: Factorize the residual $\mathbf{T}_{i-1}$ into 2 factors: $\mathbf{T}_{i-1} \approx \mathbf{T}_i \mathbf{S}_i$ with $\mathbf{T}_i \in \tilde{\mathcal{E}}_i$ and $\mathbf{S}_i \in \mathcal{E}_i$
4: Update $\mathbf{T}_i$ and $\mathbf{S}_j$, $j \in \{1 \dots i\}$ to fit $\mathbf{X}$: $\mathbf{X} \approx \mathbf{T}_i \prod_{j=1}^{i} \mathbf{S}_j$ with $\mathbf{T}_i \in \tilde{\mathcal{E}}_i$ and $\mathbf{S}_j \in \mathcal{E}_j, \forall j$
5: **end for**
6: $\mathbf{S}_M \leftarrow \mathbf{T}_{M-1}$

Output: The estimated factorization: $\{\mathbf{S}_j\}_{j=1}^{M}$.

3. EXPERIMENTS

In the present work, we explore the use of FAμST in the context of functional brain imaging using magnetoencephalography (MEG) and electroencephalography (EEG) signals. Source imaging with MEG and EEG delivers insights into the active brain at a millisecond time scale in a non-invasive way. To achieve this, one needs to solve the bioelectromagnetic inverse problem. It is a high dimensional ill-posed regression problem requiring proper regularization. As it is natural to assume that a limited set of brain foci are active during a cognitive task, sparse focal source configurations are commonly promoted using convex sparse priors [10, 11]. The bottleneck in the optimization algorithms are the dot products with the forward matrix $\mathbf{X}$ and its transpose. This matrix is computed using the MNE software [12] implementing a BEM.

3.1. Factorization compromise

The objective of this first set of experiments is to see what tradeoffs between relative complexity and accuracy are achievable. To this end, an MEG gain matrix $\mathbf{X} \in \mathbb{R}^{204 \times 8193}$ ($m = 204$ and $n = 8193$) was factorized into M sparse factors using Algorithm 1 with different settings. The way to set the constraint sets $\mathcal{E}_i$ and $\tilde{\mathcal{E}}_i$ in order to make complexity savings is evoked in [3]. Indeed, denoting $\mathbf{L}_i = \prod_{j=i+1}^{M} \mathbf{S}_j$, a simple calculation shows that if we constrain each $\mathbf{S}_j$ to have $\mathcal{O}(h)$ non-zero entries per row, then $\mathbf{L}_i$ will not have more than $\mathcal{O}(h^{M-(i+1)})$ non-zero entries per row. This suggests to decrease exponentially the number of non-zero entries in $\mathbf{T}_i$ with i in Algorithm 1.

Settings. The rightmost factor $\mathbf{S}_1$ was set of the same dimension than $\mathbf{X}$, but with k-sparse columns, which corresponds to the constraint set $\mathcal{E}_1 = \{\mathbf{A} \in \mathbb{R}^{204 \times 8193}, \|\mathbf{a}_l\|_0 \leq k, \|\mathbf{A}\|_F = 1\}$. The other factors $\mathbf{S}_j$, $j \in \{2, \dots, M\}$ were set square, with a global sparsity constraint controlled by the parameter s, and the residual at each step $\mathbf{T}_i$, $i \in \{1, \dots, M-1\}$ was also set square, with a global sparsity constraint imposing a number of non-zero entries exponentially decreasing with i, controlled by the parameters ρ and P, and normalized columns for $i = 1$. This corresponds to the constraint sets $\tilde{\mathcal{E}}_1 = \{\mathbf{A} \in \mathbb{R}^{204 \times 204}, \|\mathbf{a}_l\|_2 = 1\}$, and $\mathcal{E}_i = \{\mathbf{A} \in \mathbb{R}^{204 \times 204}, \|\mathbf{A}\|_0 \leq sm, \|\mathbf{A}\|_F = 1\}$, $\tilde{\mathcal{E}}_i = \{\mathbf{A} \in \mathbb{R}^{204 \times 204}, \|\mathbf{A}\|_0 \leq P\rho^{i-1}, \|\mathbf{A}\|_F = 1\}$ for $i \in \{2, \dots, M-1\}$. The controlling parameters are set to:

- Number of factors: $M \in \{2, \dots, 10\}$.
- Sparsity in the first factor: $k \in \{5, 10, 20, 30\}$.
- Sparsity in the other factors: $s \in \{2, 4, 8\}$.
- Rate of decrease of the residual sparsity: $\rho = 0.8$.

The parameter P controlling the sparsity in the residual was found to have only limited influence, and was set to $204^2 \times 1.4$. Other values for ρ were tested, leading to slightly different complexity/accuracy tradeoffs not shown here. The factorization setting is summarized in Figure 1, where the sparsity of each factor is explicitly given.

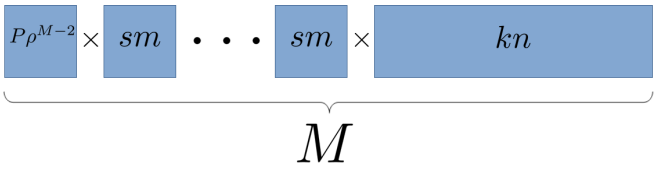

Fig. 1: Factorization setting used. Each factor is represented with its total sparsity.

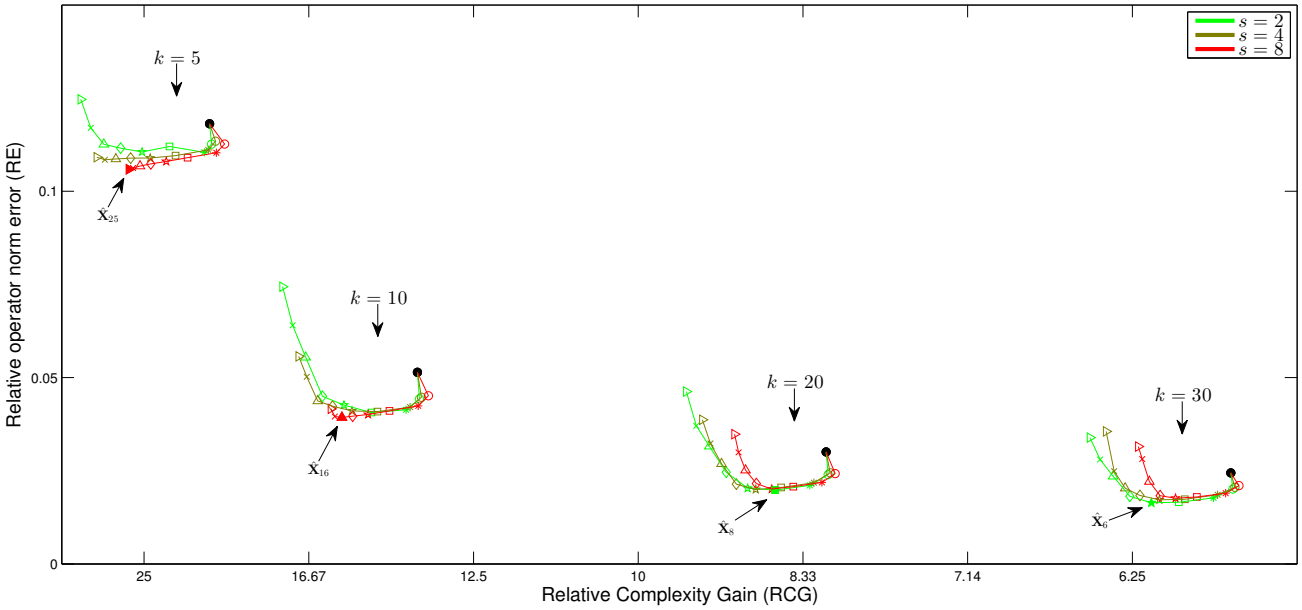

Fig. 2: Results of the factorization of a 204×8193 MEG matrix. The shape of the symbols denotes the number of factors M ($\bullet : M = 2$, $\circ : M = 3$, $* : M = 4$, $\square : M = 5$, $\star : M = 6$, $\diamond : M = 7$, $\triangle : M = 8$, $\times : M = 9$, $\triangleright : M = 10$), and the color the value of the parameter s.

Results. Factorizations were computed for 127 parameter settings. The computation time for each factorization was around $(M-1)\times 10$ minutes on a regular laptop. Figure 2 displays the tradeoff between speed (the RCG measure (3)) and approximation error (the RE measure (4)) of each obtained FAμST. We observe that:

- The overall relative complexity of the obtained factorization is essentially controlled by the parameter k. This seems natural, since k controls the sparsity of the rightmost factor which is way larger than the other ones.
- The tradeoff between complexity and approximation for a given k is mainly driven by the number of factors M: higher values of M lead to lower relative complexities, but a too large M leads to a higher relative error. Taking $M = 2$ (black dots) never yields the best compromise, hence the relevance of *multi-layer* sparse approximations.
- For a fixed k, one can distinguish nearby tradeoff curves corresponding to different sparsity levels s of the intermediate factors. The parameter s actually controls the horizontal spacing between two consecutive points on the same curve: a higher s allows to take a higher M without increasing the error, but in turn leads to a higher relative complexity for a given number M of factors.

In summary, one can distinguish as expected a tradeoff between relative complexity and approximation error. The configuration exhibiting the lowest relative error for each value of k is highlighted on Figure 2, this gives $\hat{\mathbf{X}}_{25}$, $\hat{\mathbf{X}}_{16}$, $\hat{\mathbf{X}}_{8}$, $\hat{\mathbf{X}}_{6}$, where the subscript indicates the achieved RCG (rounded to the closest integer). For example, $\hat{\mathbf{X}}_{6}$ can multiply vectors with 6 times less flops than $\mathbf{X}$ (saving 84% of computation), and $\hat{\mathbf{X}}_{25}$ can multiply vectors with 25 times less flops than $\mathbf{X}$ (96% savings). These four matrices will next be used to solve an inverse problem and compared to results obtained with $\mathbf{X}$.

3.2. Source localization

Let us now assess the FAμST approximations of the MEG gain matrix $\mathbf{X} \in \mathbb{R}^{204\times 8193}$ regarding a specific application, namely a brain source localization problem on simulated data. For this experiment, two brain sources chosen uniformly at random were activated with gaussian random weights, giving a 2-sparse vector $\boldsymbol{\gamma} \in \mathbb{R}^{8193}$, whose support encodes the localization of the sources. The experiment then amounts to solving the inverse problem to get $\hat{\boldsymbol{\gamma}}$ from the measurements $\mathbf{y} = \mathbf{X}\boldsymbol{\gamma} \in \mathbb{R}^{204}$, using either $\mathbf{X}$ or a FAμST $\hat{\mathbf{X}}$ during the recovery process. We used Orthogonal Matching Pursuit (OMP) [13] (choosing 2 atoms) as recovery method. Note that two other recovery methods (ℓ_1-regularized least squares ($\mathrm{l_1ls}$) [14] and Iterative Hard Thresholding (IHT) [15]) were tested, with qualitatively similar results. We present here only the results for OMP for the sake of conciseness. The matrices used for the recovery are the actual matrix $\mathbf{X}$ and its FAμST approximations $\hat{\mathbf{X}}_{25}$, $\hat{\mathbf{X}}_{16}$, $\hat{\mathbf{X}}_{8}$ and $\hat{\mathbf{X}}_{6}$ just described. The expected computational gain of using a FAμST instead of the true dense operator is of the order of RCG, since the computational cost of OMP is dominated by products with $\mathbf{X}^T$.

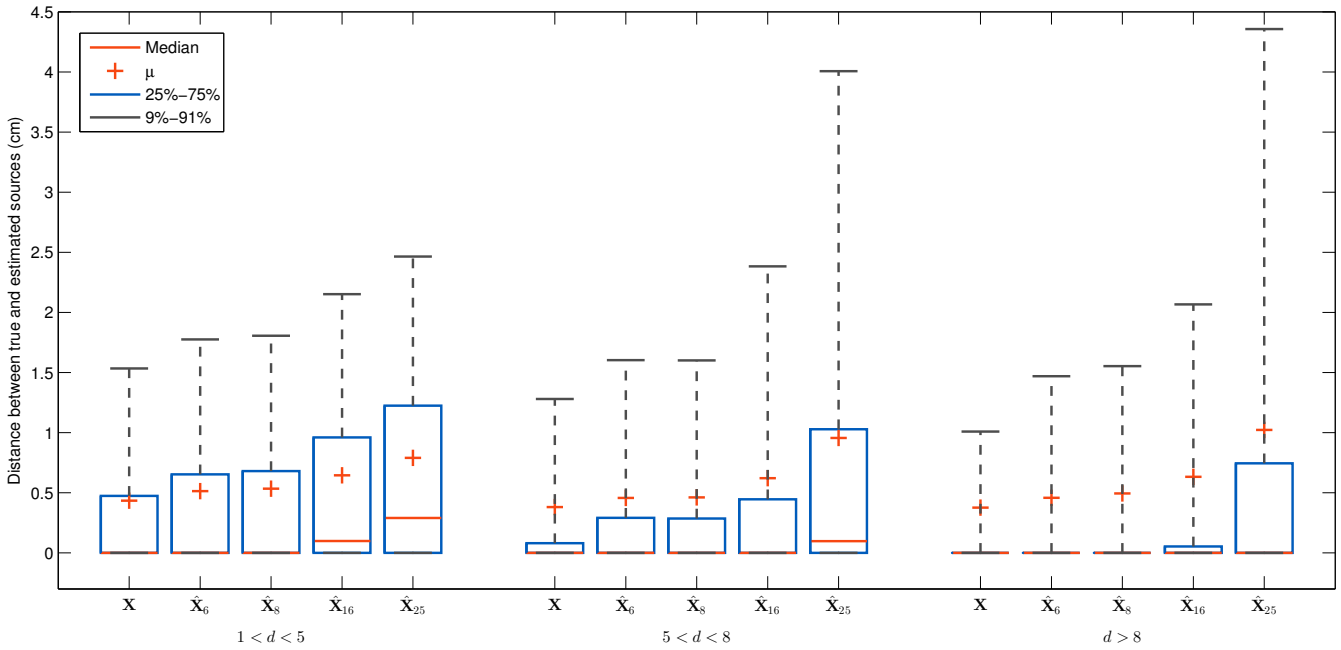

Fig. 3: Source localization performance, measured in distance between actual and retrieved source, obtained with different distances between actual sources and matrices.

Three configurations in term of distance d (in centimeters) between the sources were considered. For each configuration, 500 vectors $\mathbf{y} = \mathbf{X}\boldsymbol{\gamma}$ were generated, and OMP was run using each matrix. The distance between each actual source and the closest retrieved source was measured. Figure 3 displays the statistics of this distance for all scenarios:

- As expected, localization is better when the sources are more separated, independently of the choice of matrix.
- Most importantly, the performance is almost as good when using FAμSTs $\hat{\mathbf{X}}_6$ and $\hat{\mathbf{X}}_8$ than when using the actual matrix $\mathbf{X}$, although the FAμSTs are way more computationally efficient (6 and 8 times less computations). For example, in the case of well separated sources ($d > 8$), the FAμSTs allow to retrieve exactly the sought sources more than 75% of the time, which is almost as good as when using the actual matrix $\mathbf{X}$. The performance with the two other FAμSTs $\hat{\mathbf{X}}_{16}$ and $\hat{\mathbf{X}}_{25}$ is a bit poorer, but they are even more computationally efficient matrix (16 and 25 times less computations). For example, in the case of well separated sources ($d > 8$), they allow to retrieve exactly the sought sources more than 50% of the time. These observations indicate that there is a compromise between computational efficiency and localization performance, and that FAμSTs can be used in an inverse problem framework without a large precision loss.

3.3. Runtime vs Relative Complexity

In the experiments performed in the two previous subsections, computational efficiency is predicted through the Relative Complexity (RC) defined in (3). However, a Relative Complexity Gain (RCG) of, e.g., 10, does not necessarily mean that the matrix/vector product will be 10 times quicker using the multi-layer sparse approximation instead of the dense form. Indeed, the runtime using an interpreted language like matlab is not exactly proportional to the number of flops.

In order to illustrate this phenomenon, we performed an experiment to compare the actual runtime to the expected one considering RCG. For this experiment, a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, with $n = 2^M$ was generated as the product of M sparse matrices: $\mathbf{A} = \prod_{j=1}^{M} \mathbf{S}_j$, each $\mathbf{S}_j$ being square, full-rank and having exactly 2 non-zero entries per row and per column. This configuration gives RCG $= \frac{n}{2M}$. A gaussian random vector $\mathbf{z} \in \mathbb{R}^n$ was then generated, and the runtime needed to multiply $\mathbf{z}$ by $\mathbf{A}$ was measured using the dense form and the FAμST. This experiment was repeated for different values of $M \in \{1, \ldots, 14\}$, 1000 times for each dimension, and the results are given in Figure 4.

Several comments are in order:

- The FAμSTs are quicker than dense forms in high dimension ($n \geq 512$) but not in low dimension (top plot), although RCG > 1 (for example, RCG ≈ 9 for $n = 128$).
- The empirical runtime gain $\widehat{\text{RCG}}$ (bottom plot) does not fully achieve the predicted gain (RCG), but becomes closer to it as the dimension grows, indicating the existence of higher constant computational costs in the multi-layer sparse implementation used here. A rule of thumb seems to be $\widehat{\text{RCG}} \approx \text{RCG}/5$ for large matrix dimensions.

This can be explained by the very optimized implementation of the matrix product in matlab (built on LAPACK and BLAS), compared to the rudimentary home made implementation of the multi-layer sparse matrix product used here. A more careful low-level implementation of FAμST is expected to yield a better fit between $\widehat{\text{RCG}}$ and RCG. It could also speed up the factorization process itself.

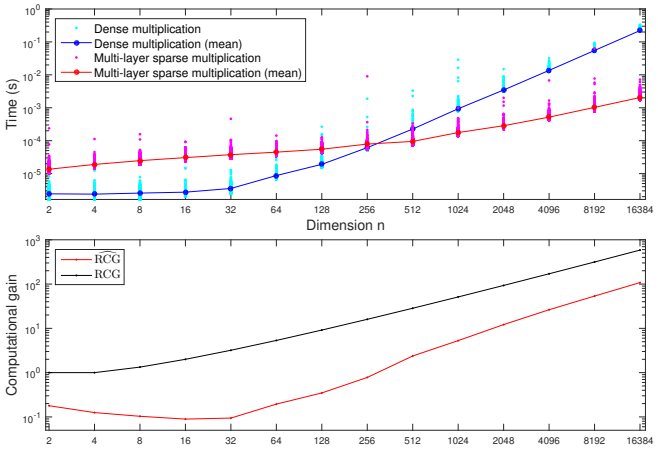

Fig. 4: Vector multiplication runtime comparison, averaged over 1000 repetitions. The bottom plot indicates for each dimension RCG and the empirical runtime gain $\widehat{\text{RCG}}$.

4. CONCLUSION

In this paper, we proposed to approximate matrices arising in linear inverse problems by Flexible Approximate MUlti-layer Sparse Transforms (FAμST) in order to accelerate resolution algorithms. A hierarchical factorization algorithm based on non-convex optimization techniques was given. The benefit of the approach was experimentally assessed on an MEG gain matrix. The first set of experiments allowed to exhibit the existence of a flexible tradeoff between computational complexity and fidelity to the original matrix. Second, the usability of FAμSTs was assessed on a brain source localization task, with up to 25 times speedup with controlled accuracy. Finally, a discussion on the discrepancy between relative complexity of multi-layer sparse approximations and actual runtime was presented, pointing out the need for an optimized implementation for multi-layer sparse matrices.

In the future, one interesting development would be to combine the ideas of this paper (accelerating matrix/vector product) with those of the previous papers aimed at accelerating the resolution of inverse problems (considering the recovery algorithm as a computational network). This could lead to even more computationally efficient recovery algorithms for sparsity-regularized inverse problems.

REFERENCES

[1] J.A. Tropp and S.J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proc. IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[2] L. Le Magoarou and R. Gribonval, "Learning computationally efficient dictionaries and their implementation as fast transforms," *CoRR*, vol. abs/1406.5388, 2014.

[3] L. Le Magoarou and R. Gribonval, "Chasing butterflies: In search of efficient dictionaries," 2015 IEEE Int. Conf. on Acoustics, Speech and Signal Processing.

[4] J. Bolte, S. Sabach, and M. Teboulle, "Proximal Alternating Linearized Minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, pp. 1–36, 2013.

[5] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Ann. Int. Conf. on Machine Learning*, 2010, pp. 399–406.

[6] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Learning efficient sparse and low rank models," *CoRR*, vol. abs/1212.3631, 2012.

[7] A. Makhzani and B. Frey, "k-sparse autoencoders," *CoRR*, vol. abs/1312.5663, 2013.

[8] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: learning sparse dictionaries for sparse signal approximation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1553–1564, 2010.

[9] O. Chabiron, F. Malgouyres, J.-Y. Tourneret, and N. Dobigeon, "Toward fast transform learning," *International Journal of Computer Vision*, pp. 1–22, 2014.

[10] S. Haufe, V. V. Nikulin, A. Ziehe, K.-R. Müller, and G. Nolte, "Combining sparsity and rotational invariance in EEG/MEG source reconstruction," *NeuroImage*, vol. 42, no. 2, pp. 726–738, 2008.

[11] A. Gramfort, M. Kowalski, and M. S. Hämäläinen, "Mixed-norm estimates for the M/EEG inverse problem using accelerated gradient methods," *Phys. Med. Biol.*, vol. 57, no. 7, pp. 1937–1961, 2012.

[12] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, and M. S. Hämäläinen, "MNE software for processing MEG and EEG data," *NeuroImage*, vol. 86, no. 0, pp. 446 – 460, 2014.

[13] J.A. Tropp and A.C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[14] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale l1-regularized least squares," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 606–617, 2007.

[15] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *J. Fourier Anal. Appl.*, vol. 14, no. 5-6, pp. 629–654, 2008.