FAST IMPLEMENTATION OF A $\ell_1 - \ell_1$ REGULARIZED SPARSE REPRESENTATIONS ALGORITHM.

Jean-Jacques FUCHS

IRISA/Université de Rennes I
Campus de Beaulieu - 35042 Rennes Cedex - France
fuchs@irisa.fr

ABSTRACT

When seeking a sparse representation of a signal on a redundant basis, one replaces generally the quest for the true sparsest model by an ℓ_1 minimization and solves thus a linear program. In the presence of noise one further replaces the exact reconstruction constraint by an approximate one. The ℓ_2-norm is generally chosen to measure the reconstruction error because of its link with Gaussian noise and the stability and simplicity of the ensuing algorithms, but the ℓ_1-norm may be preferred in some cases when the noise has heavier tails or in the presence of outliers.

We propose to replace the usual $\ell_2 - \ell_1$ regularized criterion by a $\ell_1 - \ell_1$ regularized criterion and show how to construct a fast dedicated optimization algorithm that solves this criterion in a finite number of steps. Since quite often even these fast optimal programs are considered to be too time consuming, we further develop an ad hoc sub-optimal algorithm that could be called the ℓ_1-matching pursuit algorithm.

Index Terms— Sparse representations, optimization, matching pursuit, continuation methods, ℓ_1-norm.

1. INTRODUCTION

There is currently a huge interest in sparse representations which is a technique that consists in decomposing a signal into a small number of components chosen from a user-designed over-complete set of vectors. It is mostly used to obtain a simple approximate model of a complex signal for denoising [1], compression or coding purposes [2, 3] in audio or video signal processing. Theoretical investigations tend to extend its applicability to a variety of new domains, as for instance, compressed sensing or compressed sampling, in which one investigates the possibility to sample a signal at a rate much lower that the Nyquist rate with a controlled loss in information [4, 5, 6].

More formally, given an observation $b \in R^n$ one seeks a sparse representation of b, in terms of the columns a_k of a $n \times m$ matrix A, with $m \gg n$. Provided A is full row-rank matrix, there are an infinity of representations x such that $b = Ax$ and to select a sparse one, one solves the linear program:

$$\min_x \|x\|_1 \quad \text{subject to} \quad Ax = b,$$

where $\|x\|_k$ denotes the ℓ_k norm of a vector x, $\|x\|_k = [\sum_1^m |x_j|^k]^{1/k}$ for $k \geq 1$. Since an approximate reconstruction may be sufficient and even preferable, it makes sense to replace the linear program by

$$\min_x \|x\|_1 \quad \text{subject to} \quad \|Ax - b\|_p \leq \rho_p,$$

with ρ_p a tolerance to be defined. Quite generally, and especially so in signal and image processing, p is taken equal to 2, i.e., the Euclidean norm is chosen as the error metric. For $p = 2$, the above criterion is equivalent to the quadratic program [7, 8]

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + h\|x\|_1, \quad h > 0. \tag{1}$$

that has also been considered in many earlier papers in many different areas. The quadratic programming routines that solve (1) exactly are quite time consuming and dedicated fast variants [9, 10, 11] have been developed. In practice even these "fast" algorithms are too time consuming in many applications and suboptimal algorithms such as the often re-discovered matching pursuit algorithm [12] that selects sequentially the components, are preferred and improved upon [13].

In the sequel we propose to replace the ℓ_2-norm in (1) by the ℓ_1-norm

$$\min_x \|Ax - b\|_1 + h\|x\|_1, \qquad h > 0. \tag{2}$$

This is a realistic criterion if the additive noise on the observation vector b follows a Laplace or double-exponential distribution or, in a Bayesian context and in a different setting, when the prior density on x is Laplace and one seeks the maximum a posteriori estimate of x. Following the lines in [9, 10, 11], we will develop an iterative algorithm that solves (2) exactly as well as an ad hoc matching-pursuit-like algorithm that tends to solve (2) approximatively.

2. THE CRITERION

2.1. Preliminary remarks

Let A be a (n,m) full row-rank matrix with columns a_j, we consider the optimization problem (2)

$$\min_x \|Ax - b\|_1 + h\|x\|_1, \qquad h > 0,$$

with $\|x\|_1 = \sum_i |x_i|$, the ℓ_1-norm of x. This is a convex program that can be transformed into a linear program. Since this kind of transformation yields quite interesting information about the structure of the optimum let us present it. By introducing slack variables, the optimization problem (2) becomes

$$\begin{aligned} &\min\ \mathbf{1}^T(r^+ + r^-) + h\mathbf{1}^T(x^+ + x^-) \qquad \text{under} \\ &A(x^+ - x^-) - r^+ + r^- = b \quad \text{and } x^+,\ x^-,\ r^+,\ r^- \geq 0, \end{aligned}$$

with $\mathbf{1}$ a vector of ones of adequate dimension. This is now a linear program in standard form with $2m$+$2n$ variables and n equality constraints. From basic results in linear programming theory it then follows that there is a basic feasible optimal solution, i.e., there is a optimum (x, r) that has generically n non-zero components. If, say, $p < n$ of these non-zero unknowns belong to x, the remaining $n - p$

non-zero components are in r and this in turn tells us that, at the optimum, $r = Ax - b$ has p zero components, i.e. Ax is equal to b for p indexes, and this is achieved using p non-zero components in x.

2.2. Optimality conditions

In order to be able to characterize easily the conditions satisfied by the optimum of (2), we introduce $\partial f(x)$ the sub-differential of a convex function f at a point x, it is a set of vectors called the sub-gradients of f at x. For $f(x) = \|x\|_1$ one has

$$\partial \|x\|_1 = \{u | u_i = \text{sign}(x_i) \text{ if } x_i \neq 0 \text{ and } |u_i| \leq 1 \text{ else}\} \quad (3)$$

where x_i is the i-th component of x. Note that if f is differentiable at x then $\partial f(x)$ reduces to the gradient.

Since (2) is a convex program, the first order optimality conditions (zeroing the sub-differential) are necessary and sufficient conditions for optimality and one thus gets

Lemma 1. The optimum of (2) is x iff

$$\exists u \in \partial\|x\|_1, w \in \partial\|r\|_1, \quad \text{such that} \quad A^T w + hu = 0\,, \quad (4)$$

with r=$Ax - b$, the residual or reconstruction error vector. △

While (4) does not allow to find the optimal x, it will nevertheless allow us to derive an iterative algorithm that solves (2) rigorously in a finite number of steps.

2.3. Some specific notations

Let us introduce some notations that will allow us to exploit the information contained in (4). As above, we denote p the number of non-zero components in the optimal x, we already known that the corresponding residual vector $r = Ax - b$ has then p zero components.

We partition the optimal x into $\bar{x}$ its p-nonzero components and $\bar{\bar{x}}$ its zero components. We partition accordingly $\diamond$ the sub-gradient u into $\bar{u}$ and $\bar{\bar{u}}$ and $\diamond$ the (columns in the) matrix A into $\bar{A}$ and $\bar{\bar{A}}$. One has then, for instance, $Ax = \bar{A}\bar{x}$. From (3), it follows that $\bar{u}$=sign($\bar{x}$) and $\|\bar{\bar{u}}\|_\infty \leq 1$.

We further partition the optimal r into $\underline{r}$ its n-p nonzero components and $\underline{\underline{r}}$ its p zero components. We similarly partition $w \in \partial\|r\|_1$ into $\underline{w}$ which is equal to sign($\underline{r}$) and $\underline{\underline{w}}$ which satisfies $\|\underline{\underline{w}}\|_\infty \leq 1$.

Note that while the partition of x induces a partition of the matrix A into $\bar{A}$ and $\bar{\bar{A}}$ in term of its columns, the partition of r induces a partition of A into $\underline{A}$ and $\underline{\underline{A}}$ in terms of its rows.

If both partitions are active the matrix A will thus be partitioned into four blocks with, e.g., $\underline{\underline{\bar{A}}}$ of dimension (p, p).

3. OPTIMIZATION ALGORITHM

Due to the presence of u and w, which belong to sets (see (3)), the relation in (4) is far from defining the optimal x. It nevertheless carries a lot of information, that is helpful if one is interested in the way the optimal x varies locally with h. More precisely, if the optimum is known for a given value of h, the relation (4) defines how the optimum varies in the neighborhood of this h and also carries enough information to precisely locate the boundaries of the neighborhood, i.e. the interval in h, on which the optima can be extended.

As a matter of fact, it is then also possible to cross these boundary, i.e., to propagate the optima to the next interval. This is the idea that is used to develop the algorithm.

3.1. Introduction

Assume we have the quadruple $x,\ u,\ r,\ w$, that satisfies the optimality condition (4) for a given h, we will extend the quadruple within an *interval* in h. To do so, we introduce the two partitions described above into (4) and the boundaries of the *interval* will be the values of h for which these partitions cease to be valid. From the beginning, we thus know that the following four sub-vectors are invariant in the current *interval* : $\bar{\bar{x}},\ \bar{u}\ ,\underline{r}$ and $\underline{w}$.

The optimality condition (4) $A^T w + hu = 0$ becomes first (column-block partition of A)

$$\bar{A}^T w + h\bar{u} = 0 \quad \text{and} \quad \bar{\bar{A}}^T w + h\bar{\bar{u}} = 0$$

which in turn can be further detailed (row-block partition) to yield

$$\underline{A}^T \underline{w} + \underline{\underline{A}}^T \underline{\underline{w}} + h\bar{u} = 0 \quad \text{and} \quad \bar{\bar{A}}^T \underline{w} + \underline{\underline{\bar{A}}}^T \underline{\underline{w}} + h\bar{\bar{u}} = 0. \quad (5)$$

Though the optimal x does not appear as such in these conditions, it can be recovered from u and w and their associated partitions. One has $\bar{x} = \underline{\underline{\bar{A}}}^{-1}\underline{b}$, and this actually tells us, since its expression is independent of h, that the optimal x is invariant within the *interval*.

Taking a close look at the first relation in (5), one observes that it consists of p equations that actually define the p-dimensional sub-vector $\underline{\underline{w}}$ in terms of h and the other sub-vectors $\underline{w}$ and $\bar{u}$ that are known and constant within the current interval, since $\underline{w} = \text{sign}(\underline{r})$ and $\bar{u} = \text{sign}(\bar{x})$. Provided the square order p matrix $\underline{\underline{\bar{A}}}$ is invertible, which we will assume, the first condition in (5) can be rewritten

$$\underline{\underline{w}}(h) = -h\underline{\underline{\bar{A}}}^{-T}\bar{u} - \underline{\underline{\bar{A}}}^{-T}\ \underline{\bar{A}}^T \underline{w}. \quad (6)$$

This relation, which is of the form $\underline{\underline{w}}(h) = hW_1 + W_2$, tells us how the optimal $\underline{\underline{w}}$ which satisfies $\|\underline{\underline{w}}\|_\infty \leq 1$, varies as a function of h.

Introducing (6) into the second condition in (5) similarly defines the way $\bar{\bar{u}}$ varies as a function of h, one gets an expression of the form

$$h\bar{\bar{u}}(h) = U_1 + hU_2. \quad (7)$$

These two relations are valid as long as $\underline{w}$ and $\bar{\bar{u}}$, that both depend upon h, remain smaller than one in ℓ_∞-norm, see (3). The boundaries of the current interval in h are thus the values of h for which $\underline{\underline{w}}$ or $\bar{\bar{u}}$ attain 1 in the ℓ_∞-norm. More precisely as h increases, there is either a component of $\underline{\underline{w}}$ or of $\bar{\bar{u}}$ that becomes equal to ± 1 first, and the corresponding value of h defines the upper bound of the interval and similarly for the lower bound as h decreases. In summary, within a given interval, among the eight sub-vectors of the optimal quadruple $x,\ u,\ r,\ w$, only two vary with h, namely $\bar{\bar{u}}$ and $\underline{\underline{w}}$, four are constant by definition $\bar{\bar{x}},\ \bar{u}\ ,\underline{r}$ and $\underline{w}$ and the remaining two $\bar{x}$ and $\underline{r}$ are constant by necessity.

3.2. Development

The idea of the $iterative$ algorithm is thus to start with h large for which the optimum is at zero and follow the optimum $x(h)$ for diminishing h. The number of nonzero components in $x(h)$ essentially increases as h diminishes and never exceeds n. We will define boundary values h^k for $k = 0,\ 1,\ 2, ..$ with $h^k < h^{k-1}$ which are such that within an interval $[h^k,\ h^{k-1}[$ the number and signs of the nonzero components in the optimal $x(h)$ and associated residual $r(h)$ remain constant. Within each such interval we will get the analytical expression of the optima. If one seeks the optimum of (2) for a given h one stops the algorithm as soon as the h of interest lies

within the current interval. The number of steps (intervals) required is unknown a priori and increases with diminishing h and increasing number of nonzero components in the optimal x. We have already seen how to get the boundary values, it remains to indicate how to start and how to cross such a boundary.

3.2.1. The initial step

Let us check that for h large, the optimum is at the origin. We verify that $x = 0$, $r = -b$ together with the associated u and w satisfy (4) for $h \geq h^0 = \max_j |a_j^T \text{sign}(b)|$. If $x = 0$ then $u = \bar{\bar{u}}$, $A = \bar{\bar{A}}$, $r = -b$ and, assuming for simplicity that b has no zero component, $w = \underline{w} = -\text{sign}(b)$. It follows then from the second relation in (5) that $\bar{\bar{u}}(h) = (1/h)\ A^T \text{sign}(b)$ which is admissible as long as $\|\bar{\bar{u}}(h)\|_\infty \leq 1$, i.e., $h \geq h^0 = \max_j |a_j^T \text{sign}(b)|$ which is thus the first boundary value.

The index of the first component of the optimal x to become non-zero is $j_1 = \arg\max_j |a_j^T \text{sign}(b)|$ and its sign is $\bar{u} = u_{j_1} = -\text{sign}\ (a_{j_1}^T \text{sign}(b))$.

As h becomes slightly smaller than h^0, x_{j_1} jumps to a non zero value with its sign given by $\bar{u} = u_{j_1}$. This is the value that makes a first component in $r = Ax - b$ become zero, i.e.,

$$|x_{j_1}| = \min_{\{i \mid \frac{b_i u_{j_1}}{a_{i,j_1}} > 0\}} \frac{b_i u_{j_1}}{a_{i,j_1}}. \tag{8}$$

This is the only way to keep (4) satisfied for h slightly smaller than h_0. We have thus defined how to cross the boundary h^0 and how to further characterize all the quantities appearing in (5), i.e., (6, 7) within this first h-interval whose lower bound is not yet known.

3.2.2. The standard step

At the beginning of a standard step, one knows the optimal x and all the other quantities in the current interval whose lower bound is however not known. The first task is thus to find this next boundary value, we already indicated how to get it. A boundary is hit either as - event 1 - a component in $\bar{\bar{u}}$ (7), or as - event 2 - a component in $\underline{\underline{w}}$ (6), becomes equal to ± 1. It is thus an easy task to find this boundary. In case the h of interest lies in the current interval, the procedure stops and the optimal x is the constant x valid for the whole interval, otherwise to fulfill the current step we need to define how to cross this boundary.

For event 1, we denote j_k the index of the component of x that becomes non zero and denote u_{j_k} its sign that is already known. We modify the x induced partition of A and u accordingly. Clearly $\bar{\bar{x}}$ looses a component which enters $\bar{x}$, while $\bar{u}$ gains a component equal to u_{j_k}. We are now in a transition where $\underline{\bar{A}}$ is no longer square and has dimension, say, $(p \times p + 1)$ and we need to make another change to recover a state in which the optimality conditions will be satisfied over the next interval. To do so we have to make $\underline{\bar{A}}$ square again. One can either (add a line) activate a new constraint, i.e., use the new unknown x_{j_k} to zero a component in $\underline{r}$ as we did at the end of the initial step above or (remove a column) one can zero an unknown, i.e., take advantage of the entrance of x_{j_k} to remove a previously non-zero unknown while keeping the same constraints active. There are $n = (n - p) + p$ potential cases to consider.

For event 2, we denote i_k the index of the component of the residual vector r that becomes non-zero and w_{i_k} its sign. This means that constraint i_k which was active becomes inactive and we now modify the r induced partition in A and w. A line is removed from $\underline{\underline{A}}$ and added to $\underline{A}$, again this matrix is no longer square and has dimension $(p - 1 \times p)$. To complete the transition phase we are in, one can either zero one of the p unknowns in $\bar{x}$ or activate a new constraint among the $n - p$ previously inactive constraints in $\underline{A}$. There are again n cases to test at the most.

In both events, since there is only one optimal decision, one stops the research as soon as the case under test satisfies the conditions in (4) within the next interval.

3.2.3. Relations to previous works

Several recent papers have proposed similar path-following methods for solving (1), [9, 10, 11]. All these methods are related to continuation techniques, which have also been studied in the optimization literature [14]. When the solution is sparse, i.e. when the (unknown) optimum has just a few non-zero components, they are indeed very fast but their computational complexity increases more than linearly in the number of non zero components in the optimum. To our knowledge, however, no such algorithms have been proposed for the criterion (2) except for some preliminary remarks in [15]. Note also that these algorithms are fully different from those presented in [16] and quite recently in [17] where the same criterion is considered and a connexion between the classical CLEAN algorithm [18] and ℓ_1-denoising is proposed.

4. ℓ_1 MATCHING PURSUIT

Having defined an exact way to solve (2), we now propose a approximate but much faster way to solve a similar problem.

Matching pursuit (MP) is an ad hoc way to get rapidly a more or less sparse representation of a vector b as a linear combination of a small number of columns a_j (assumed to be of unit Euclidean norm) of the matrix A. For completeness we remember the basic MP algorithm.

At each step, the MP algorithm fits to the current residual (reconstruction error), say r_{k-1} the most correlated vector of the redundant basis:

$$j_k = \arg\max_j |a_j^T r_{k-1}|,$$

it then subtracts from r_{k-1} a weighted version of this vector. The weight minimizes the norm of the new residual vector r_k

$$x_k = \min_x \|r_{k-1} - a_{j_k} x\|_2^2$$

and one gets

$$r_k = (I - a_{j_k} a_{j_k}^T) r_{k-1}.$$

One starts with $r_0 = b$ and stops when the Euclidean norm of the residual is smaller than a user-fixed threshold. This method is clearly ℓ_2-based, this appears in the choice of the correlation, the value of the optimal weight and the stopping criterion.

In order to develop an ℓ_1-based matching pursuit algorithm, one replaces the Euclidean $\|x\|_2^2 = x^T x$ by the ℓ_1-norm $\|x\|_1 = \sum |x_i| = x^T \text{ sign } x$, i.e., in some sense one replaces the usual scalar product $x^T y$ by a pseudo scalar product $x^T \text{sign } y$. The choice of the most correlated vector becomes then quite naturally

$$j_k = \arg\max_j |a_j^T \text{sign } r_{k-1}|$$

and the resulting vector a_{j_k} is indeed the most efficient column-vector in A if the objective is the (local)rate of variation of the ℓ_1-norm of r_{k-1} but is not (necessarily) the one that yields the largest decrease of the cost function. To find the column a_j that leads to the largest possible decrease one would have to solve an optimization problem for each column (see (9) below) and keep the best but this is too time-consuming for an approximate method as this one. Note that while for the basic (ℓ_2-norm) MP, the most efficient column locally is also the one that yields the largest decrease, this is not the case in the ℓ_1-norm case.

In case r_{k-1} has one or several zero components, the most efficient locally or equivalently the most correlated vector is given by, using the notations of a r_{k-1}-induced partition :

$$j_k = \arg\max_j \{ |\underline{a}_j^T \text{sign } \underline{r}_{(k-1)}| - \|\underline{a}_{\underline{=}j}\|_1 \} .$$

The resulting choice is however only valid if the value of the associated maximum is positive. Otherwise, there is no way to diminish $\|r_{k-1}\|_1$ using just one column, i.e. the optimum attainable with this ad hoc global scheme is reached.

Once the next column entering the selection, say a_{j_k} , is identified, the choice of the optimal weight, say x_{j_k}, to be assigned to it is non unique. One can choose the $smallest$ weight x that makes a first component in $r_k = r_{k-1} - x a_{j_k}$ zero, arguing that for larger weights the rate of decrease will be smaller and another vector in the basis possibly more efficient. The computation is quite simple but it appears in practice that this choice does not work. It introduces cycling (e.g. a same couple of vectors is used in turn a large number of times with infinitesimal steps before proceeding to another vector) and that the resulting algorithm converges extremely slowly.

We thus propose to take $x_{j_k} = \arg\min_x \|r_{k-1} - a_{j_k} x\|_1$. This optimization problem has no analytical solution but can be solved quite easily by inspecting all the values of x that make a component zero in r_k. The algorithm is

$$\min_{i \in I_k} \|r_{k-1} - a_{j_k} \text{ sign } (\underline{a}_{j_k}^T \text{sign } \underline{r}_{(k-1)}) \, |x_i| \, \|_1 \qquad (9)$$

with

$$|x_i| = \frac{\underline{r}_{(k-1)}(i)}{\underline{a}_{j_k}(i) \text{sign } (\underline{a}_{j_k}^T \text{sign } \underline{r}_{(k-1)})}$$

and

$$I_k = \{ \, i \mid \frac{\underline{r}_{(k-1)}(i)}{\underline{a}_{j_k}(i) \text{sign } (\underline{a}_{j_k}^T \text{sign } \underline{r}_{(k-1)})} > 0 \}.$$

The resulting global scheme works in general, i.e. proceeds until the current residual vector r_k has ℓ_1-norm below the fixed threshold. There are some cases where it fails to converge, it stops because no single column a_j allows to further diminish $\|r_k\|_1$. Further investigations are under progress to evaluate the global properties of this ℓ_1 matching pursuit scheme with stopping criterion on $\|r_k\|_1$ and to define an alternative in case the threshold is not attained.

5. CONCLUDING REMARKS

We have considered the following ℓ_1- ℓ_1 penalized criterion

$$\min_x \|Ax - b\|_1 + h\|x\|_1, \qquad h > 0.$$

We have shown how to construct a fast algorithm that minimizes it by actually following its optimum for decreasing values of h. and stopping the procedure when the optimal h is attained. As h decreases, one observes that, roughly speaking, the number of non-zero components in the optimal x increases. If the number of non-zero component in the optimum is small, which is generally the case in sparse representations applications, the algorithm we propose is much faster than linear programming routines that can be applied to this criterion. If, in case of a very large number of basis vectors, even faster algorithms are needed one must resort to approximate solutions and we also propose an ad hoc ℓ_1-matching pursuit algorithm.

So far we have applied this criterion to image denoising and coding and in both cases the results are promising. We also plan to investigate its applicability in decoding linear codes. Linear programming and sparse representations approaches have already been considered in this context [19, 20].

6. REFERENCES

[1] A. Chakrabarti and F. Hirakawa, "Effective separation of sparse and non-sparse image features for denoising," In *Proceedings ICASSP*, 857–860, Apr. 2008.

[2] M. Bertalmio et al., "Simultaneous structure and texture image impainting," *IEEE Trans. on Image Processing*, 12, 882–889, 2003.

[3] M. Zibulewsky and B. Pearlmutter, "Blind source separation by sparse decomposition on a signal dictionary," *Neural Computation* , 13, 863–882, 2001.

[4] E. Candes, J. Romberg and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE T. on I.T.*, 52, 489-509, 2006.

[5] D. Donoho, "Compressed sensing," *IEEE Trans. on I.T.*, 52, 1289-1306, 2006.

[6] E. Candes and T. Tao, "Near optimal signal recovery from random projections: Universal coding strategies ," *IEEE Trans. on I.T.*, 52, 5406-5425, 2006.

[7] S. Chen, D. Donoho and M. Saunders, "Atomic Decomposition by Basis Pursuit," *SIAM J. on Scientific Comput.*, 20, 1, 33-61, 1999.

[8] J.J. Fuchs. "On the application of the global matched filter to DOA estimation with uniform circular arrays," *IEEE-T-SP*, 49, p. 702–709, Apr. 2001.

[9] M. Osborne, B. Presnell and B. Turlach. "On the LASSO and its dual," *J. Comput. Graph. Statist.*, 9, 3319–337, 2000.

[10] B. Efron, T. Hastie, I. Johnstone and R. Tibshirani, "Least angle regression," *Annals of Statistics*, 32, pp. 407–499, Apr. 2004.

[11] S. Maria and J.J. Fuchs, "Application of the Global Matched Filter to STAP data: an efficient algorithmic approach," In *Proceedings ICASSP*, Toulouse, May 2006.

[12] S. Mallat and Z. Zhang, "Matching Pursuit with time-frequency dictionaries," *IEEE Trans. on S.P.*, 41, 3397-3415, 1993.

[13] S. Krstulovic and R. Gribonval, "MPTK: Matching Pursuit made tractable" In *Proceedings ICASSP*, Toulouse, May 2006.

[14] E. Allgower and K. Georg, "Continuation and path following," *Acta Numerica*, 2, 31-64, 1993.

[15] J.J. Fuchs, "Some further results on the recovery algorithms," In *Proceedings of SPARS'05*, Rennes, France, Nov. 2005.

[16] A. Alliney and S. Ruzinsky "An algorithm for the minimzation of mixed ℓ_1 and ℓ_2 norms with application to bayesian estimation," *IEEE Trans. on S.P.*, 42, 618-627, 1994.

[17] V. Solo, "A modified CLEAN algorithm does ℓ_1-denoising," In *Proceedings ICASSP*, 3665-3668, Apr. 2008.

[18] J.A. Hogbom, "Aperture synthesis with a nonregular distribution of interferometer baselines," In *Astron. Astro-phys. Suppl.*, 15, 417-426, 1974.

[19] E. Candes and T. Tao, "Decoding by linear programming," *IEEE-T-IT*, 51, 4203-4215, 2005.

[20] J. Feldman, M.J. Wainwright and D.R. Karger, "Using linear programming to decode binary linear codes," *IEEE-T-IT*, 51, 954-972, 2005.