Gradient Pursuits

Thomas Blumensath and Mike E. Davies

Gradient Pursuits

Thomas Blumensath, *Member, IEEE,* Mike E. Davies, *Member, IEEE*

***Abstract*— Sparse signal approximations have become a fundamental tool in signal processing with wide ranging applications from source separation to signal acquisition. The ever growing number of possible applications and in particular the ever growing problem sizes throw up new challenges in terms of computational strategies and the development of fast and efficient algorithms has become paramount.**

Recently, very fast algorithms have been developed to solve convex optimisation problems that are often used to approximate the sparse approximation problem, however, it has also been shown, that in certain circumstances, greedy strategies, such as Orthogonal Matching Pursuit, can have better performance than the convex methods.

In this paper we therefore concentrate on improving greedy strategies and develop algorithms that approximate Orthogonal Matching Pursuit, but which have computational requirements more akin to Matching Pursuit. We discuss three different directional optimisation schemes based on the gradient, the conjugate gradient and an approximation to the conjugate gradient respectively. We show that the conjugate gradient update leads to a novel implementation of Orthogonal Matching Pursuit, while the gradient based approach as well as the approximate conjugate gradient methods both lead to fast algorithms, with the approximate conjugate gradient method being superior to the gradient method.

***Index Terms*— Sparse Representations/Approximations, Matching Pursuit, Orthogonal Matching Pursuit, Gradient Optimisation, Conjugate Gradient Optimisation.**

I. Introduction

A sparse signal expansion is a signal model that uses a linear combination of a small number of elementary waveforms selected from a large collection to represent or approximate a signal. Such expansions are of increasing interest in signal processing with applications ranging from source coding [1] to de-noising [2], source separation [3] and signal acquisition [4].

Let $\mathbf{x} \in \mathbb{R}^M$ be a known vector and $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$ a matrix with $M < N$. We will refer to $\mathbf{\Phi}$ as the dictionary and call the column vectors ϕ_i of $\mathbf{\Phi}$ atoms. The problem addressed in this paper is to find a vector $\mathbf{y}$ satisfying the relationship:

$$\mathbf{x} = \mathbf{\Phi}\mathbf{y} + \epsilon. \tag{1}$$

If we allow for a non-zero error ϵ we talk about a signal *approximation*, while for zero ϵ we have an exact signal *representation*.

Because $M < N$, there are an infinite number of $\mathbf{y}$ satisfying the above equation. It is therefore common to search for a vector $\mathbf{y}$ optimising a certain sparsity measure. For example, it is common to look for a vector $\mathbf{y}$ with the smallest number of non-zero elements. The problem of finding such a $\mathbf{y}$ is, however, NP-hard [5], [6]. Therefore, different sub-optimal strategies are used in practise. Commonly used strategies are generally based on convex relaxation, non-convex (often gradient based) local optimisation or greedy search strategies. Convex relaxation is used in algorithms such as Basis Pursuit and Basis Pursuit De-Noising [7], the lasso and Least Angle Regression (LARS) [8]. Recently fast algorithms solving the lasso convex problem have been suggested in [9] and [10]. Non-convex local optimisation procedures include the Focal Underdetermined System Solver FOCUSS [11]. In this paper we are interested in greedy methods, the most important of which are Matching Pursuit (MP) [12], Orthogonal Matching Pursuit (OMP) [13] and Orthogonal Least Squares (OLS)[1] [14].

MP is an algorithm often used for practical applications and there are now very efficient $\mathcal{O}(N \log N)$ (for each iteration) implementations [15], [16] whenever $\mathbf{\Phi}$ is the union of dictionaries for which fast transforms are available. For general dictionaries MP is $\mathcal{O}(NM)$ (per iteration). On the other hand, OMP has superior performance. Current implementations, however, are more demanding both in terms of computation time and memory requirement.

In this paper we combine an MP type algorithm with directional optimisation to derive 'Directional Pursuit' algorithms. These new algorithms use a similar greedy element selection as MP and OMP, however, the costly orthogonal projection is (approximately) done using directional optimisation. We propose two update directions which can be calculated efficiently such that the algorithms have the same memory requirements and computational complexity as MP. A third update direction is based on the calculation of a conjugate gradient, which leads to a novel implementation of OMP with computational requirements similar to currently used methods based on QR factorisation.

A. Paper Overview

The main part of this paper starts with a review of MP and OMP in section II. Based on these two algorithms, we develop the general Directional Pursuit framework in section III. Particular directions are then suggested in the following three sections, starting with the gradient direction in section IV, followed by the conjugate gradient in section V and an approximate conjugate gradient in section VI. Section VII takes a closer look at the computational requirements of the proposed algorithms while section VIII gives theoretic bounds

The authors are with IDCOM & Joint Research Institute for Signal and Image Processing, Edinburgh University, King's Buildings, Mayfield Road, Edinburgh EH9 3JL, UK (Tel.: +44(0)131 6505659, Fax.: +44(0)131 6506554, e-mail: thomas.blumensath@ed.ac.uk, mike.davies@ed.ac.uk).

This research was supported by EPSRC grant D000246/1. MED acknowledges support of his position from the Scottish Funding Council and their support of the Joint Research Institute with the Heriot-Watt University as a component part of the Edinburgh Research Partnership.

[1]Note that OLS is different from OMP as it uses a different criterion to select new elements.

on the convergence of the gradient based algorithm. The paper concludes with a range of experiments presented in section IX. In particular, we analyse the approximation performance of the algorithms and the ability to exactly recover the underlying sparse structure in subsections IX-A and IX-B respectively. This is followed by two experiments that highlight the applicability of the methods to two different problems, audio de-noising in subsection IX-C and compressive sampling of Magnetic Resonance Imaging in subsection IX-D.

B. Notation

Before delving into the thick of the topic, let us define some additional notation. We will throughout use the set Γ^n to be the set containing the indices of the elements selected up to and including iteration n. Using this index set as a subscript, the matrix $\mathbf{\Phi}_{\Gamma^n}$ will then be a sub-matrix of $\mathbf{\Phi}$ containing only those columns of $\mathbf{\Phi}$ with indices in Γ^n. The same convention is used for vectors. For example, $\mathbf{y}_{\Gamma^n}$ is a sub-vector of $\mathbf{y}$ containing only those elements of $\mathbf{y}$ with indices in Γ^n. In general, the superscript in the subscript of $\mathbf{y}_{\Gamma^n}$ reminds us that we are in iteration n, on occasion, however, we resort to using superscripts (e.g. $\mathbf{y}^n$) to label the iteration. The gram matrix $\mathbf{G}_{\Gamma^n} = \mathbf{\Phi}_{\Gamma^n}^T\mathbf{\Phi}_{\Gamma^n}$ will also be used frequently. In general, lower case bold face characters represent vectors while upper case bold characters are used for matrices.

II. Matching pursuit and Orthogonal Matching Pursuit

The algorithms in this paper approximate a vector $\mathbf{x}$ iteratively. In iteration n we calculate an approximation using

$$\hat{\mathbf{x}}^n = \mathbf{\Phi}_{\Gamma^n}\mathbf{y}_{\Gamma_n}, \tag{2}$$

and calculate the approximation error as:

$$\mathbf{r}^n = \mathbf{x} - \hat{\mathbf{x}}^n. \tag{3}$$

In each iteration, the approximation error is then used to determine a new element to be selected from $\mathbf{\Phi}$ in order to find a better approximation.

One of the simplest such algorithm is possibly Matching Pursuit (MP) [12]. New elements are selected based on the inner product between the current residual $\mathbf{r}^n$ and the columns in $\mathbf{\Phi}$ after which a single element of $\mathbf{y}$ is updated. MP is summarised as follows:

1) Initialise $\mathbf{r}^0 = \mathbf{x}, \mathbf{y}^0 = 0$
2) for $n = 1; n := n + 1$ till stopping criterion is met
   a) $\nabla^n = \mathbf{\Phi}^T\mathbf{r}^n$
   b) $i^n = \arg_i \max|\nabla_i|$
   c) $y_{i^n}^n = y_{i^n}^{n-1} + \nabla_{i^n}$
   d) $\mathbf{r}^n = \mathbf{r}^{n-1} - \phi_{i^n}\nabla_{i^n}$
3) Output: $\mathbf{r}^n, \mathbf{y}^n$

MP requires the evaluation of matrix multiplications involving $\mathbf{\Phi}^T$. If $\mathbf{\Phi}$ is a union of dictionaries for which fast transforms exist, then these matrix operations can be computed efficiently. Another trick used in MP [12] is to compute the inner products between the residual and the dictionary elements recursively. This can be done using:

$$\nabla_i^{n+1} = \mathbf{r}^{n+1^T}\phi_i = \nabla_i^n - \nabla_{i^n}^n\phi_{i^n}^T\phi_i, \tag{4}$$

where ϕ_{i^n} is the last selected element. This result is a direct consequence of the MP error recursion:

$$\mathbf{r}^{n+1} = \mathbf{r}^n - g_{i^n}^n\phi_{i^n}. \tag{5}$$

This approach is of advantage whenever the inner products $\langle\phi_{i^n}, \phi_i\rangle$ between the dictionary elements can either be store or efficiently computed and, crucially, whenever these inner products are predominantly zero.

In Orthogonal Matching Pursuit [13], [17] $\mathbf{y}$ is updated in each iteration by projecting $\mathbf{x}$ orthogonally onto all selected atoms. OMP therefore finds the best signal approximation possible with these atoms. This algorithm is:

1) Initialise $\mathbf{r}^0 = \mathbf{x}, \mathbf{y}^0 = 0, \Gamma^0 = \oslash$
2) for $n = 1; n := n + 1$ till stopping criterion is met
   a) $\nabla^n = \mathbf{\Phi}^T\mathbf{r}^n$
   b) $i^n = \arg_{i\notin\Gamma^{n-1}} \max|\nabla_i|$
   c) $\Gamma^n = \Gamma^{n-1} \cup i^n$
   d) $\mathbf{y}^n = \mathbf{\Phi}_{\Gamma^n}^\dagger\mathbf{x}$
   e) $\mathbf{r}^n = \mathbf{x} - \mathbf{\Phi}\mathbf{y}^n$
3) Output: $\mathbf{r}^n, \mathbf{y}^n$

Here the dagger † indicates the Moore-Penrose pseudo-inverse. Note that this inverse should never be calculated explicitly and more efficient implementations of OMP based on QR factorisation [17] or Cholesky factorisation are available. The main drawback of these approaches is that they require additional storage, with the storage requirements for the QR based method being larger than that for the Cholesky based approach. The QR based method is, however, faster than the Cholesky based method. The QR based method requires the storage of an orthogonal matrix with dimensions $M \times n_{max}$ and an upper triangular matrix of dimension $n_{max} \times n_{max}$, where n_{max} is the overall number of iterations of the algorithm. The Cholesky method only stores a $n_{max} \times n_{max}$ triangular matrix but requires the solution to three inverse problems per iteration involving this matrix.

III. The Directional Pursuit Framework

In iteration n the problem solved by Orthogonal Matching Pursuit (OMP) is to minimise the quadratic cost-function (in n unknowns):

$$\|\mathbf{x} - \mathbf{\Phi}_{\Gamma^n}\mathbf{y}_{\Gamma^n}\|_2^2. \tag{6}$$

Instead of updating y_{i^n} by adding $\nabla_{i^n}^n$ as in Matching Pursuit (MP), we propose a directional update:

$$\mathbf{y}_{\Gamma^n}^n = \mathbf{y}_{\Gamma^n}^{n-1} + a^n\mathbf{d}_{\Gamma^n}, \tag{7}$$

where $\mathbf{d}_{\Gamma^n}$ is an update direction. The optimal direction in each iteration would be the direction that minimises the squared error between the signal and the approximation based on the selected subset of elements. However, as we discuss below, evaluating this direction can be computationally demanding. In this paper we propose the use of three different

directions, the gradient, a conjugate gradient and an approximated conjugate gradient. These are described in more details in the next three sections.

Once an update direction has been calculated, the step-size a^n can be determined. Because of the quadratic cost function, a close form solution to this exists and it is easy to show [18, pp. 521] that the optimum step size is:

$$a^n = \mathbf{r}^{n^T}\mathbf{c}^n / (\mathbf{c}^{n^T}\mathbf{c}^n), \tag{8}$$

where $\mathbf{c}^n$ is the vector $\mathbf{c}^n = \mathbf{\Phi}_{\Gamma^n}\mathbf{d}^n$.

In general, a single directional update does not guarantee convergence to the minimum of equation (6) (one could therefore also term the method "nearly" Orthogonal Matching Pursuit) and an additional generalisation of the proposed approach would be to use several directional update steps before selecting a new element.

This leads to the question of how and how often to select new elements from the dictionary. As in MP and OMP, this selection is based on the inner product between the residual and the dictionary elements. Because all previously chosen elements are updated in each iteration, it would be possible to restrict the selection of new elements to those elements not selected previously. However, whenever the selection criterion suggest that an element should be selected again, this indicates that the residual is 'far' from orthogonal to the selected elements (a requirement which would be met by the optimum of equation (6)).

The theoretical considerations discussed in section VIII as well as experimental results, some of which can be found in section IX, show that it is beneficial to always select the element with the largest inner product[2]. We therefore allow the selection step to select elements more than once, i.e. we do not force the selection step to select a *new* element in each iteration, thereby letting the algorithm automatically decide how many directional update steps are required for each new element.

The Directional Pursuit family of algorithms can then be summarised as follows:

1) Initialise: $\mathbf{r}^0 = \mathbf{x}, \mathbf{y}^0 = 0, \Gamma^0 = \oslash$
2) for $n = 1; n := n + 1$ till stopping criterion is met
   a) $\nabla^n = \mathbf{\Phi}^T\mathbf{r}^n$
   b) $i^n = \arg_i \max |\nabla_i|$
   c) if $i^n \notin \Gamma^{n-1}$: $\Gamma^n = \Gamma^{n-1} \cup i^n$, else $\Gamma^n = \Gamma^{n-1}$
   d) calculate update direction $\mathbf{d}_{\Gamma^n}$
   e) $\mathbf{c}^n = \mathbf{\Phi}_{\Gamma^n}\mathbf{d}_{\Gamma^n}$
   f) $a^n = \mathbf{r}^{n^T}\mathbf{c}^n / (\mathbf{c}^{n^T}\mathbf{c}^n)$
   g) $\mathbf{y}^n_{\Gamma^n} := \mathbf{y}^{n-1}_{\Gamma^n} + a^n\mathbf{d}_{\Gamma^n}$
   h) $\mathbf{r}^n = \mathbf{r}^{n-1} - a^n\mathbf{c}^n$
3) Output: $\mathbf{r}^n, \mathbf{y}^n$

IV. Gradient Pursuit

One of the simplest update directions that springs to mind is the gradient evaluated at the current coefficient value. This leads to an algorithm we will call *Gradient Pursuit* (GP). The gradient of equation (6) is:

$$\nabla_{\Gamma^n} = \mathbf{\Phi}^T_{\Gamma^n}(x - \mathbf{\Phi}_{\Gamma^n}\mathbf{y}^{n-1}_{\Gamma^n}). \tag{9}$$

Looking at this gradient it is important to realise, that this gradient is exactly the vector ∇^n restricted to the elements in Γ^n, which has already been calculated in step 2.a) of the Matching Pursuit (MP) algorithm, i.e. MP calculates this gradient in each step, so that the use of this gradient comes for free in a directional pursuit approach. The only additional cost compared to MP is then the evaluation of the step-size. (Note, however, that when updating more than one element in $\mathbf{y}^n$, the recursion in equation (4) becomes less efficient.)

V. Conjugate Gradient Pursuit

Another popular directional optimisation algorithm is the conjugate gradient method [18, Section 10.2], [20, Section 16.4], which is a well known optimisation procedure that is guaranteed to solve quadratic optimisation problems in as many steps as the dimension of the problem. The conjugate gradient algorithm can be summarised as follows. We denote the cost function to be minimised by $\frac{1}{2}\mathbf{y}^T\mathbf{G}\mathbf{y} - \mathbf{b}^T\mathbf{y}$ (which is equivalent to solving $\mathbf{G}\mathbf{y} = \mathbf{b}$ over $\mathbf{y}$). The conjugate gradient method uses directional updates that are $\mathbf{G}$-conjugate to the previously chosen directions. A set of vectors $\{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n\}$ is $\mathbf{G}$-conjugate if

$$\mathbf{d}^{n^T}\mathbf{G}\mathbf{d}^k = 0 \tag{10}$$

for all $k \neq n$. More details can be found in for example [18, Section 10.2] and [20, Section 16.4].

The same idea can be used in the directional Pursuit framework, where we now want to calculate an update direction that is $\mathbf{G}_{\Gamma^n}$ conjugate to all previously used update directions. Here $\mathbf{G}_{\Gamma^n} = \mathbf{\Phi}^T_{\Gamma^n}\mathbf{\Phi}_{\Gamma^n}$. The cost function is now

$$\|\mathbf{x} - \mathbf{\Phi}_{\Gamma^n}\mathbf{y}_{\Gamma^n}\|_2^2, \tag{11}$$

where the dimension n of the cost function changes whenever a new element is selected. Let $\mathbf{d}^k_{\Gamma^n}$ be the k^{th} conjugate gradient. The subscript reminds us that this vector is $|\Gamma^n|$ dimensional. We update $\mathbf{y}_{\Gamma^n}$ (which is an $|\Gamma^n|$-dimensional sub-vector of $\mathbf{y}$) in direction $\mathbf{d}^n_{\Gamma^n}$, which is equivalent to updating $\mathbf{y}$ using a directional vector $\mathbf{d}^n$ of dimension N, with all elements zero apart from the element indexed by Γ^n. Therefore, we can think of all previous update directions $\mathbf{d}^k_{\Gamma^k}$ (note the superscript in the subscript) as higher dimensional vectors $\mathbf{d}^k_{\Gamma^n}$, where the elements associated to the 'new' dimensions are set to zero.

To derive the algorithm, we recall the conjugate gradient Theorem [20, Theorem 16.2], which we give here using the notation introduced above:

Theorem 1: [20, Theorem 16.2] Let $[\mathbf{d}^1_{\Gamma^n}, \mathbf{d}^2_{\Gamma^n}, \ldots, \mathbf{d}^n_{\Gamma^n}]$ be any set of non-zero $\mathbf{G}_{\Gamma^n}$-conjugate vectors, then the solution to the problem $\mathbf{G}_{\Gamma^n}\mathbf{y}_{\Gamma^n} = \mathbf{\Phi}^T_{\Gamma^n}\mathbf{x}$ is

$$\mathbf{y}^n_{\Gamma^n} = \sum_{k=1}^{n} a^k\mathbf{d}^k_{\Gamma^n}, \tag{12}$$

[2]This for example ensures that the algorithm belongs to the class of *General MP* algorithms defined in [19](see IX).

with step-sizes:

$$a^k = \frac{\mathbf{r}^{k^T}\mathbf{\Phi}_{\Gamma^k}\mathbf{d}^k_{\Gamma^n}}{\mathbf{d}^{k^T}_{\Gamma^n}\mathbf{G}_{\Gamma^k}\mathbf{d}^k_{\Gamma^n}}, \tag{13}$$

where

$$\mathbf{r}^k = \mathbf{x} - \mathbf{\Phi}_{\Gamma^{k-1}}\left(\sum_{i=1}^{k-1} a^i \mathbf{d}^i_{\Gamma^n}\right). \tag{14}$$

The importance of this theorem lies in the fact that the step-size a^k only depends on the current residual error and the current conjugate gradient. This means that $\mathbf{y}^n_{\Gamma^n}$ can be approximated iteratively by calculating a new direction and step-size in each iteration.

The other important aspect of this theorem is that it guarantees an optimal solution in N iterations. In a directional pursuit framework, the dimensions can change from one iteration to the next. In the first iteration we have a single trivial direction and step-size. In general, if the $n-1$ previously used update directions are $\mathbf{G}_{\Gamma^n}$ conjugate, then, by the above theorem, we only require one additional conjugate gradient update to exactly solve the n dimensional problem.

Assume that the $n-1$ previously used update directions are $\mathbf{G}_{\Gamma^{n-1}}$-conjugate. We want to show that they are also $\mathbf{G}_{\Gamma^n}$-conjugate (note the different subscripts!). Using the matrix $\mathbf{D}^{n-1}_{\Gamma^{n-1}}$ to denote the matrix containing all conjugate update directions from iteration $n-1$ and the matrix $\mathbf{D}^{n-1}_{\Gamma^n}$ to be the same matrix but with an additional row of zeros at the bottom. From the definition of $\mathbf{G}_{\Gamma^n}$-conjugacy we require ${\mathbf{D}^{n-1}_{\Gamma^n}}^T\mathbf{G}_{\Gamma^n}\mathbf{D}^{n-1}_{\Gamma^n} = \mathbf{B}$, where $\mathbf{B}$ is a diagonal matrix. Because the last row of $\mathbf{D}^{n-1}_{\Gamma^n}$ contains only zeros, the last row and column of $\mathbf{G}_{\Gamma^n}$ are multiplied by zeros, which implies that

$$\mathbf{B} = {\mathbf{D}^{n-1}_{\Gamma^{n-1}}}^T\mathbf{G}_{\Gamma^{n-1}}\mathbf{D}^{n-1}_{\Gamma^{n-1}} = {\mathbf{D}^{n-1}_{\Gamma^n}}^T\mathbf{G}_{\Gamma^n}\mathbf{D}^{n-1}_{\Gamma^n}. \tag{15}$$

The main question is now how to calculate a new conjugate gradient. We require the new direction to be $\mathbf{G}_{\Gamma^n}$-conjugate. Therefore, the new direction has to satisfy

$${\mathbf{D}^{n-1}_{\Gamma^n}}^T\mathbf{G}_{\Gamma^n}\mathbf{d}^n_{\Gamma^n} = \mathbf{0} \tag{16}$$

We write each new direction as a combination of all previously chosen directions and the current gradient ∇_{Γ^n} [18, Section 10.2]

$$\mathbf{d}^n_{\Gamma^n} = b_0\nabla_{\Gamma^n} + \mathbf{D}^{n-1}_{\Gamma^n}\mathbf{b}. \tag{17}$$

Without loss of generality we can set $b_0 = 1$. Pre-multiplying by $\mathbf{D}^{n-1}_{\Gamma^n}\mathbf{G}_{\Gamma^n}$ and using the $\mathbf{G}_{\Gamma^n}$-conjugacy then leads to the $n-1$ constraints:

$${\mathbf{D}^{n-1}_{\Gamma^n}}^T\mathbf{G}_{\Gamma^n}(\nabla_{\Gamma^n} + \mathbf{D}^{n-1}_{\Gamma^n}\mathbf{b}) = \mathbf{0} \tag{18}$$

from which we can write

$$\mathbf{b} = -({\mathbf{D}^{n-1}_{\Gamma^n}}^T\mathbf{G}_{\Gamma^n}\mathbf{D}^{n-1}_{\Gamma^n})^{-1}({\mathbf{D}^{n-1}_{\Gamma^n}}^T\mathbf{G}_{\Gamma^n}\nabla_{\Gamma^k}). \tag{19}$$

Again using $\mathbf{G}_{\Gamma^n}$-conjugacy we find that $\mathbf{D}^{n-1}_{\Gamma^n}\mathbf{G}_{\Gamma^n}\mathbf{D}^{n-1}_{\Gamma^n}$ is diagonal so that the conjugate gradient can be calculated without matrix inversion.

Note that in the standard conjugate gradient algorithm [18, Section 10.2], [20, Section 16.4] each new update direction can be calculated as a combination of the current gradient and the *single* previous update direction alone, i.e. in the standard conjugate gradient algorithm, all but the last conjugate update direction turn out to be $\mathbf{G}$ conjugate to the current gradient such that $\mathbf{b}$ in equation (19) has only a single non-zero element. Unfortunately, in the context of OMP, the changing dimensionality destroys this property so that we have to take account of all previous update directions in each step.

For an efficient implementation it is worth noting that in the calculation of $\mathbf{b}$ the product $\mathbf{D}^{n-1}_{\Gamma^n}\mathbf{G}_{\Gamma^n}$ can be updated recursively by adding a single new row and column in each iteration. Note also that $(\mathbf{D}^{n-2}_{\Gamma^{n-1}}\mathbf{G}_{\Gamma^{n-1}}\mathbf{D}^{n-2}_{\Gamma^{n-1}})^{-1}$ and $(\mathbf{D}^{n-1}_{\Gamma^n}\mathbf{G}_{\Gamma^n}\mathbf{D}^{n-1}_{\Gamma^n})^{-1}$ are equal apart from a single additional value added in each iteration. This value is $({\mathbf{c}^{n-1}}^T\mathbf{c}^{n-1})$ used in step 2.f) of iteration $n-1$ and does not have to be recalculated.

It is important to realise that the algorithm derived here is different form an implementation of OMP in which a *full* conjugate gradient solver is used for *each* newly selected element. Instead, the proposed method only uses a single directional update step for each new element. The most similar method to the proposed algorithm is probably the implementation of OMP proposed in [13], which also uses a directional update. However, the method in [13] uses matrix inversions, which have to be updated iteratively and which can make the approach less stable.

Another approach to OMP is based on QR factorisation. The selected dictionary $\mathbf{\Phi}_{\Gamma^n}$ is decomposed into $\mathbf{\Phi}_{\Gamma^n} = \mathbf{Q}_{\Gamma^n}\mathbf{R}_{\Gamma^n}$ where in each iteration new elements are added to $\mathbf{Q}_{\Gamma^{n-1}}$ and $\mathbf{R}_{\Gamma^{n-1}}$. The algorithm then does not need to evaluate $\mathbf{y}_{\Gamma^n}$ in each iteration, instead, $\mathbf{r}^n = \mathbf{r}^{n-1} - (\mathbf{q}^T\mathbf{x})\mathbf{q}$, where $\mathbf{q}$ is the newly added column in $\mathbf{Q}_{\Gamma^n}$. Using $\mathbf{z}_{\Gamma^n} = \mathbf{Q}^T_{\Gamma^n}\mathbf{x}$, the final solution is $\mathbf{y}_{\Gamma^n} = \mathbf{R}^{-1}\mathbf{z}_{\Gamma^n}$, which can be solved efficiently by back-substitution.

Interestingly, the QR factorisation and the proposed conjugate gradient method show many similarities. In the n^{th} iteration the QR based approach calculates an estimate:

$$\hat{\mathbf{x}} = \mathbf{Q}_{\Gamma^n}\mathbf{R}_{\Gamma^n}\mathbf{y} = \mathbf{Q}_{\Gamma^n}\mathbf{z}_{\Gamma^n}, \tag{20}$$

whilst the conjugate gradient based approach calculates the approximation as:

$$\hat{\mathbf{x}} = \mathbf{\Phi}_{\Gamma^n}\mathbf{D}_{\Gamma^n}\mathbf{a}_{\Gamma^n} \tag{21}$$

where $\mathbf{a}_{\Gamma^n}$ is the vector containing the different update step sizes. Because of $\mathbf{G}_{\Gamma^n}$-conjugacy the matrix $\mathbf{D}^{n^T}_{\Gamma^n}\mathbf{\Phi}^T_{\Gamma^n}\mathbf{\Phi}_{\Gamma^n}\mathbf{D}^n_{\Gamma^n}$ must be diagonal. Also, it can be shown by induction that, with appropriate diagonal weighting matrix $\mathbf{W}$:

$$\mathbf{\Phi}_{\Gamma^n}\mathbf{D}^n_{\Gamma^n}\mathbf{W} = \mathbf{Q}_{\Gamma^n}. \tag{22}$$

This implies that,

$$\mathbf{W}^{-1}\mathbf{a}_{\Gamma^n} = \mathbf{z}_{\Gamma^n}. \tag{23}$$

The conjugate gradient approach therefore calculates a similar decomposition as the QR factorisation, the way this decomposition is represented is, however, slightly different.

VI. Approximate Conjugate Gradient Pursuit

We have just spent some effort in deriving an algorithm similar to Orthogonal Matching Pursuit (OMP) based on QR factorisation, not only in terms of the decomposition as shown above, but also in terms of the computational requirements. Apart from the additional insight this derivation has given us, it also allows us to derive a fast algorithm that approximates the conjugate gradient. This algorithm is then a compromise between MP and OMP. An approximate conjugate gradient update does not guarantee the OMP solution in each step, the benefit is, however, that the approximate conjugate gradient is much easier to calculate than the full conjugate gradient. Furthermore, the memory requirements are significantly reduced.

The conjugate gradient implementation derived above required the storage of all previous update directions. This storage requirement can be significantly reduced by calculating the new update direction such that it is $\mathbf{G}_{\Gamma^n}$-conjugate to only a limited number of previous directions. This leads to an *Approximate Conjugate Gradient Pursuit* (ACGP) algorithm. For notational simplicity, we here restrict the derivation and only considering a single update direction. The more general derivation follows similar arguments.

To enforce $\mathbf{G}_{\Gamma^n}$-conjugacy to the previous update direction we have

$$\mathbf{d}_{\Gamma^n}^n = b_0 \nabla_{\Gamma^n} + \mathbf{d}_{\Gamma^n}^{n-1} b_1. \tag{24}$$

We can again set $b_0 = 1$ and calculate b_1 using

$$\mathbf{d}_{\Gamma^n}^{n-1^T} \mathbf{G}_{\Gamma^n} (\nabla_{\Gamma^n} + \mathbf{d}_{\Gamma^n}^{n-1} b_1) = 0. \tag{25}$$

Therefore

$$b_1 = -(\mathbf{d}_{\Gamma^n}^{n-1^T} \mathbf{G}_{\Gamma^n} \mathbf{d}_{\Gamma^n}^{n-1})^{-1} (\mathbf{d}_{\Gamma^n}^{n-1^T} \mathbf{G}_{\Gamma^n} \nabla_{\Gamma^k}). \tag{26}$$

For an efficient implementation it is worth noting that

$$(\mathbf{d}_{\Gamma^n}^{n-1^T} \mathbf{G}_{\Gamma^n} \mathbf{d}_{\Gamma^n}^{n-1}) = (\mathbf{\Phi}_{\Gamma^n} \mathbf{d}_{\Gamma^n}^{n-1})^T (\mathbf{\Phi}_{\Gamma^n} \mathbf{d}_{\Gamma^n}^{n-1}) \tag{27}$$

where $\mathbf{\Phi}_{\Gamma^n} \mathbf{d}_{\Gamma^n}^{n-1} = \mathbf{c}^{n-1}$ has been evaluated in the previous iteration to determine the step size a^n and the same is true for the denominator, which is nothing else than the product $\mathbf{c}^{n-1^T} \mathbf{c}^{n-1}$ calculated in the previous iteration.

VII. Computational Complexity

The main computational bottel-neck in the algorithms discussed in this paper is the evaluation of matrix vector multiplications of the form $\mathbf{\Phi x}$ and $\mathbf{\Phi}^T \mathbf{y}$. In most practical situations, $\mathbf{\Phi}$ is chosen to be the union of dictionaries for which fast transforms exist such as Wavelet transforms or Fourier based transforms. This means that these multiplications can be computed efficiently [16]. Nevertheless, these matrix vector multiplications remain the limiting factor for most algorithms. As noted above, for Matching Pursuit (MP), the new gradient can often be updated locally as only a single element is updated in each iteration. In the table below, we make no use of this fact. All of the methods proposed in this paper update all selected elements at the same time making this strategy less effective. This is the price one has to pay for updating all elements.

TABLE I
Comparison of the methods in terms of computational requirements in iteration n.

Algo.	Multiplications of vector by $\mathbf{\Phi}$ (other)	Storage
MP	1	0
GP	2	0
ACGP	3	0
CGP a)	$n + 3(+1$ mult. by $\mathbf{D}\mathbf{\Phi}^T)$	$\mathbf{D}$
CGP b)	$3(+1$ by $\mathbf{D}$ and 1 by $\mathbf{D}^T\mathbf{G})$	$\mathbf{D}$, $\mathbf{D}^T\mathbf{G}$

To compare the different approaches we compare the required number of matrix vector multiplications and distinguish between those involving $\mathbf{\Phi}$ and those involving other matrices. The Gradient Pursuit (GP) method requires the additional evaluation of the step size, while the approximation to the conjugate gradient involves one further matrix vector multiplication to evaluate b_1. All of these mulitiplications involve $\mathbf{\Phi}$.

On the other hand, the Conjugate Gradient Pursuit algorithm requires additional storage and multiplications by more general matrices. Here two implementational strategies are available, one in which the matrix $\mathbf{D}^T\mathbf{G}$ is updated iteratively (which can be done efficiently) and one, in which the matrix is re-evaluated in each iteration. The former uses less computations but requires more storage.

We summarise the computational cost in terms of multiplications of vectors with the dictionary (and with general matrices) as well as the additional storage required in table I for Matching Pursuit (MP), Gradient Pursuit (GP), Approximate Conjugate Gradient Pursuit (ACGP) and the two implementations of Conjugate Gradient Pursuit (CGP).

VIII. Convergence

Note that the directional pursuit algorithms proposed here belong to the general family of *General MP* as defined in [19][3]. The theoretic results presented in [19] do therefore also hold for the directional pursuit algorithms.

We here derive an additional convergence result for Gradient Pursuit (GP).

Theorem 2: There exist a constant $k < 1$, which only depends on $\mathbf{\Phi}$, such that the residual calculated with GP decays as:

$$\|\mathbf{r}^n\|_2^2 \leq k \|\mathbf{r}^{n-1}\|_2^2 \tag{28}$$

Proof: Let us use $\mathbf{r}^n = \mathbf{r}^{n-1} - a^n \mathbf{\Phi}_{\Gamma^n} \mathbf{d}$, then it can be shown that

$$\|\mathbf{r}^n\|_2^2 = \|\mathbf{r}^{n-1}\|_2^2 - \frac{(\mathbf{r}^{n-1^T} \mathbf{\Phi}_{\Gamma^n} \mathbf{d})^2}{\|\mathbf{\Phi}_{\Gamma^n} \mathbf{d}\|_2^2}. \tag{29}$$

Using $\mathbf{d} = \mathbf{\Phi}_{\Gamma^n}^T \mathbf{r}^{n-1}$ we can bound

$$\begin{aligned} \frac{(\mathbf{r}^{n-1^T} \mathbf{\Phi}_{\Gamma^n} \mathbf{\Phi}_{\Gamma^n}^T \mathbf{r}^{n-1})^2}{\|\mathbf{\Phi}_{\Gamma^n} \mathbf{\Phi}_{\Gamma^n}^T \mathbf{r}^{n-1}\|_2^2} &\geq \frac{\|\mathbf{\Phi}_{\Gamma^n}^T \mathbf{r}^{n-1}\|_2^4}{\|\mathbf{\Phi}_{\Gamma^n}\|_2^2 \|\mathbf{\Phi}_{\Gamma^n}^T \mathbf{r}^{n-1}\|_2^2} \\ &\geq \frac{\|\mathbf{\Phi}_{\Gamma^n}^T \mathbf{r}^{n-1}\|_2^2}{\|\mathbf{\Phi}_{\Gamma^n}\|_2^2} \geq \frac{\|\mathbf{\Phi}_{\Gamma^n}^T \mathbf{r}^{n-1}\|_\infty^2}{\|\mathbf{\Phi}_{\Gamma^n}\|_2^2} \end{aligned} \tag{30}$$

[3]Given that the algorithms selects the element with the largest inner product in each iteration

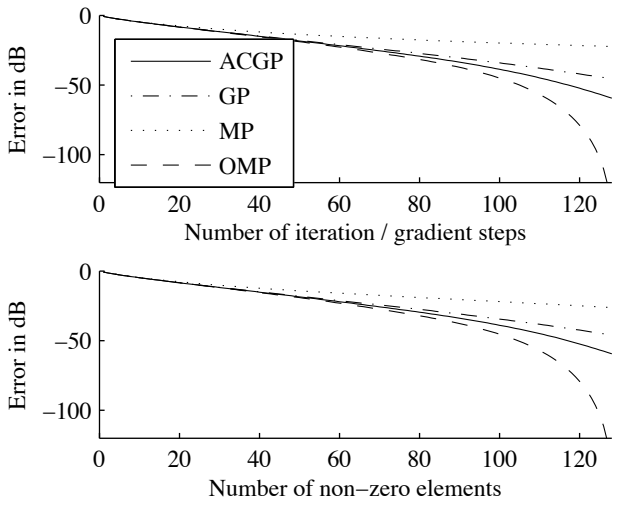


Fig. 1. Comparision between Matching Pursuit (dotted), Orthogonal Matching Pursuit (dashed), Gradient Pursuit (dash-dotted) and Approximate Conjugate Gradient Pursuit (solid) for a mildly sparse signal. Data averaged over 1 000 randomly generated signals (see text).

By theorem 9.10 in [15, pp. 422], there exist a $\omega > 0$ such that $\|\mathbf{\Phi}^T\mathbf{x}\|_\infty^2 > \omega\|\mathbf{x}\|_2^2$, for all $\mathbf{x}$. Due to the selection procedure, $\|\mathbf{\Phi}^T\mathbf{x}\|_\infty^2 = \|\mathbf{\Phi}_{\Gamma^n}^T\mathbf{x}\|_\infty^2$. Gathering this all together we see that the theorem holds for $k = (1 - \frac{\omega}{\|\mathbf{\Phi}\|_2^2})$, where $\|\mathbf{\Phi}\|_2^2 \geq \|\mathbf{\Phi}_{\Gamma^n}\|_2^2$ is the squared norm of the full dictionary. ■

Note that for Matching Pursuit (and similarly for Orthogonal Matching Pursuit), a corresponding result holds where $k = (1 - \omega)$[15, section 9.5.2], which in general would suggest a faster decay, however, the numerical studies below show that GP outperforms MP in general. Note also, that the direction $\mathbf{d}$ that minimises the expression in equation (29) is exactly the direction that would give the OMP solution.

Unfortunately, we have currently no convergence proof for ACGP, however, a slight modification of the proposed algorithms can be shown to converge in a finite number of steps. This approach would use a gradient step (equation (9)) whenever a new element is selected, while the update direction in equation (24) is used, whenever no new element is added to the selected subset. Empirical evidence for this method shows that the actual performance is close to that of GP and we therefore do not pursue this method further here.

IX. Experimental evaluation

A. Signal Approximation Performance

We first evaluate the algorithms on a toy example and compare the performance to Matching Pursuit and Orthogonal Matching Pursuit. We generated 1 000 dictionaries of size 128×256, drawing elements ϕ_i uniformly from the unit sphere. From each dictionary we selected 64 elements at random and multiplied these with unit variance zero mean Gaussian coefficients to generate 1 000 test signals. We then run Matching Pursuit (MP), Orthogonal Matching Pursuit (OMP), Approximate Conjugate Gradient Pursuit (ACGP) and Gradient Pursuit (GP) on each example.

The averaged results are shown in figure 1 where we plot the approximation error in dB against the iteration (top panel) as well as against the number of non-zero coefficients selected (lower panel). The results are virtually identical with the exception of the results for MP, which often selected elements repeatedly and therefore had an average error of 4dB higher after 128 iterations as compared to the average error after it had selected 128 different elements (which took on average 179 iterations). From these results it can be seen that the GP algorithm is closer to the performance of OMP than to MP. Overall we see that by incorporating a gradient step into the pursuit framework, significant improvements can be made over MP. Using ACGP can be seen to offer additional benefits.

MP	0.02
GP	0.02
ACGP	0.03
OMP-QR	0.12

TABLE II
Average time in seconds each algorithm took to calculate the results.

We repeated a similar experiment in which we used a fixed number of gradient steps in each iteration of GP (not shown). Unsurprisingly, such an increase gives results that get closer to the performance of OMP. It is, however, noteworthy that ACGP with a single directional update step was found to outperform GP, even if this method uses two gradient steps per iteration.

The average computation times for the results are shown in table II. All results are for our Matlab implementation[4] of the methods run on a Apple Macintosh G5 Quad 2.5GHz.

The above experiments were conducted with a signal generated using half as many non-zero coefficients than the signal dimension. As shown in the next subsection, in this regime, the algorithms are not able to exactly recover the exact non-zero coefficients. We therefore repeat the experiment reported above, but this time, the signals were generated using only 12 non-zero coefficients. As shown in the next section, for such highly sparse signals, all algorithms are able to recover the correct non-zero elements with high probability.

The averaged results are shown in figure 2, where we again plot the SNR value in dB against the iteration count, i.e. against the number of gradient steps (top panel) as well as against the number of selected elements (bottom panel).

The performance in the lower panel shows that in this experiment the algorithms perform comparable. They all select the correct atoms most of the time, so that they all find an exact signal representation. They also do not differ significantly in their approximation behaviour. A slight exception is MP, which occasionally select more than 12 non-zero elements. This accounts for the slightly worse performance observed. The results in the top panel reveal that ACGP and GP require more iterations in general to select the 12 non-zero elements, using roughly 20 and 25 steps respectively. Obviously, OMP selects a new element in each iteration and finds the correct representation after 12 iterations, while MP shows its typical

[4]The software will be made available through the first authors web-page.

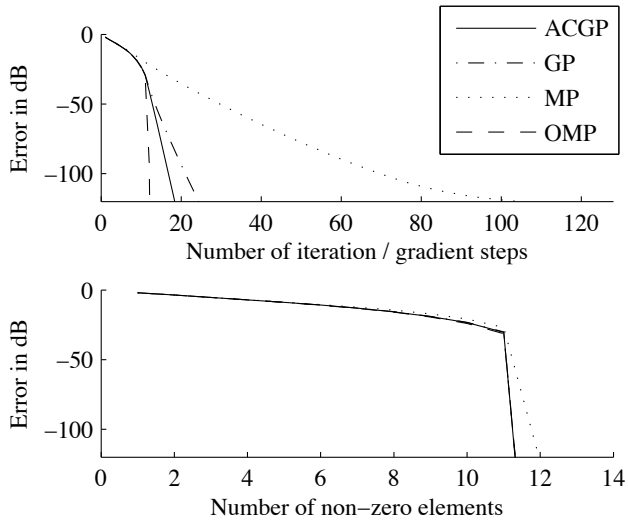


Fig. 2. Comparision between Matching Pursuit (dotted), Orthogonal Matching Pursuit (dashed), Gradient Pursuit (dash-dotted) Approximated Conjugate Gradient Pursuit (solid) for a highly sparse signal. Data averaged over 1 000 randomly generated signals (see text).

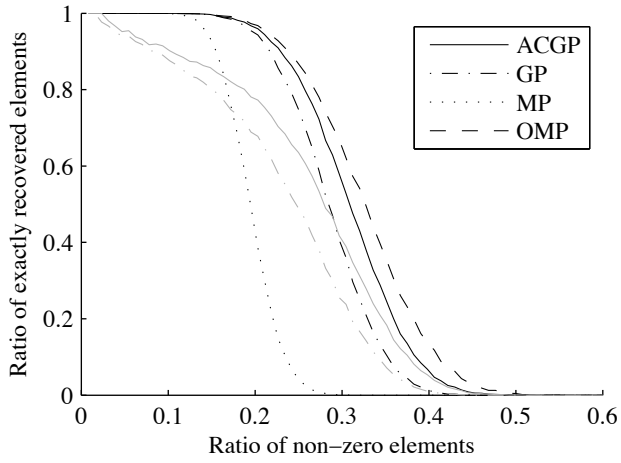


Fig. 3. Comparison of the algorithms in terms of exactly recovering the original coefficients. The grey lines are the implementation of the algorithms in which a new element is chosen in each iteration, while the black lines allow elements to be chosen repeatedly. Results averaged over 1000 runs.

exponential error decay, which theoretically would mean that the algorithm would never reach the exact result. (We stopped MP once it attained a certain approximation accuracy.)

B. Exact Recovery Performance

For highly sparse signals and certain dictionaries, it is known that the Pursuit type algorithms are guaranteed to exactly recover the elements used to generate the signal [19]. These bounds are worst case bounds and the same bounds hold for all algorithms given in this paper. We here analyse the average performance of the methods in terms of exact recovery of the elements used to generate the signal.

The signals were generated as in the previous experiment, however, we varied the number of elements used to generate the signals over a larger range. The results (averaged over 10 000 runs) are shown in figure 3. To support our argument for a repeated selection of atoms, we here show the results for two different implementations of GP and ACGP, one in which we select a new element in each iteration (shown with grey lines) and one in which we allowed the algorithm to select atoms repeatedly (black lines). All algorithms were stopped after they had selected exactly the number of elements used to generate the signal.

We see that OMP outperforms the other algorithms in terms of retrieving the true signal elements, however, ACGP and GP are not far off, when we allow elements to be selected repeatedly. Forcing GP and ACGP to select a new element in each iteration is seen to have detrimental effects for the performance in terms of exact signal recovery.

C. Audio Example

In order to demonstrate the applicability of the proposed methods to larger problems that cannot be solved with current OMP implementations, we used MP, GP and ACGP on a ca. 2.5 second long excerpt from a jazz trio audio recording (mono, 44 100kHz sampling frequency). As a dictionary we used a four times overcomplete Modified Discrete Cosine Transform (MDCT) dictionary, using Tukey windows of length 4096 and 512 with 75% overlap. We also added i.i.d. Gaussian noise to the signal before analysis, so that the analysed signal had a signal to noise ratio (SNR) of 20dB.

We run MP, GP and ACGP (both with a single gradient step), using a fast implementation of the MDCT, to retrieved as many coefficients as the dimension of the signal (110 592 samples)[5].

In figure 4 we compare the approximation performance and the de-noising performance of the methods. The lower panel shows the detail in the box in the top panel. We here show the performance in terms of signal approximation (the error between the signal estimate and the noisy signal) and in terms of de-noising performance (the error between the signal estimate and the noiseless signal). We found that the performance of ACGP was virtually identical to GP in this example. Also, looking at the decay of the error in the top panel for the signal approximation performance of GP, we see that the error goes virtually to zero (150dB) once we select as many elements as the signal dimension. This suggests that the algorithm gives nearly the exact signal projection, explaining why ACGP did not offer any advantages in this example.

We see that GP offers a much better signal approximation for the same number of used elements. For example, using 10% of the elements GP offers 0.75 dB better performance than MP, while for 25% the performance benefit is 1.7 dB. For de-noising the maximum SNR were, 22.2 dB for GP and 21.9 dB for the MP algorithm. These were achieved using 7.8% and

[5]Note that we do not use OMP as an implementation of OMP based on QR factorisation would require the storage of a square $110\ 592 \times 110\ 592$ matrix as well as the storage of an upper triangular matrix of the same size. For 64 bit floating point arithmetic, this would require 50 451 628 032 bytes or roughly 47 gigabytes of storage! An implementation based on Cholesky factorisation requires less storage ('only' an upper triangular matrix), but the required solutions to the inverse problems is far to costly for upper triangular matrixes of this size.

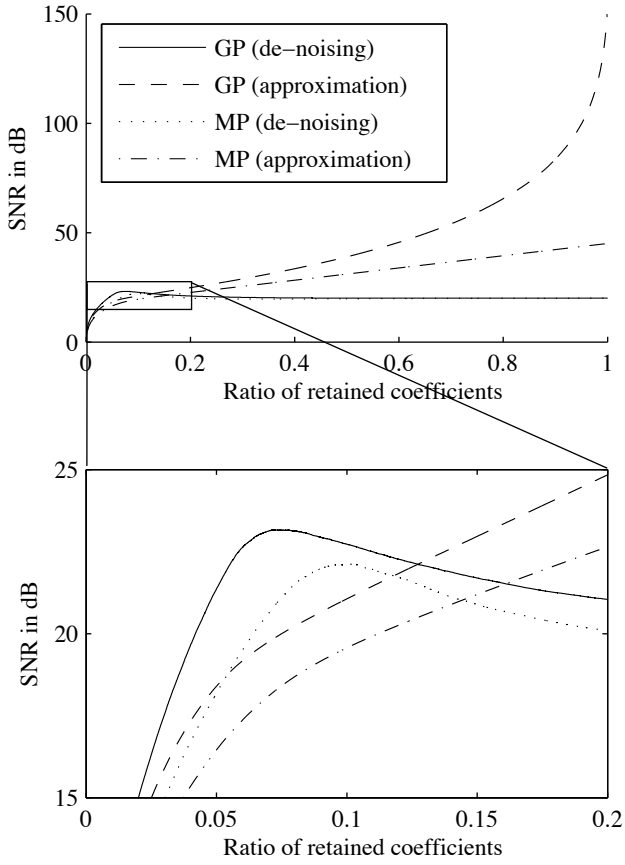


Fig. 4. Approximation and de-noising performance of Matching Pursuit and Gradient Pursuit.

9.3% of the selected elements respectively. Therefore GP does not only require less elements to approximate or de-noise the signal, it also shows a slightly better de-noising performance.

D. Compressive Sampling

An application that crucially depends on the algorithm being able to estimate the correct elements is compressive sampling. Compressive sampling is an emerging paradigm that exploits the sparsity of a signal in some transform domain in order to reduce the number of samples required for signal acquisition [4]. In the language of this paper, assume a signal $\mathbf{x}$ has a sparse representation $\mathbf{\Phi y}$. In several applications, it is often not possible to measure all values of $\mathbf{x}$, instead, it is possible to acquire a small number of linear measurements of $\mathbf{x}$ of the form $\mathbf{z} = \mathbf{Mx}$, where $\mathbf{M}$ is a measuring matrix and where the dimension of $\mathbf{z}$ is less than that of $\mathbf{x}$. The problem is then to estimate $\mathbf{x}$ given only $\mathbf{z}$, $\mathbf{M}$ and $\mathbf{\Phi}$. If $\mathbf{M}$ and $\mathbf{\Phi}$ have certain properties and if $\mathbf{y}$ is sparse enough, then it is possible to recover $\mathbf{x}$ [4] by finding a sparse representation $\mathbf{y}$ such that $\mathbf{z} = \mathbf{M\Phi y}$.

One particularly promising application domain of compressive sampling is Magnetic Resonance Imaging (MRI) [21] and we take our example from this area. In particular, we here use the same MRI example as presented [4], which uses the Logan-Shepp phantom. The Logan-Shepp phantom is shown in the top left panel in figure 5.

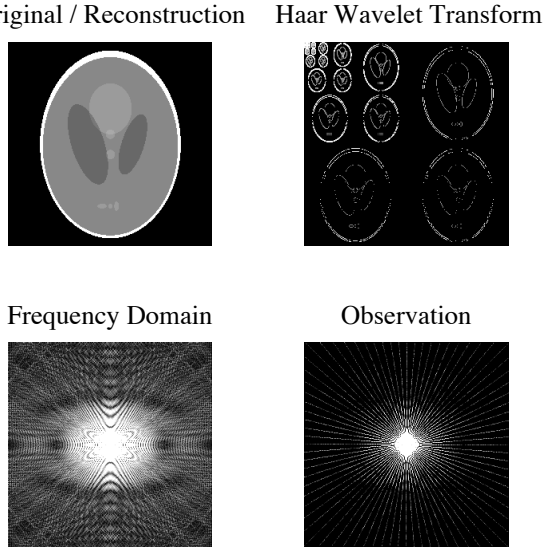


Fig. 5. Magnetic Resonance Imaging (MRI) Example. Original phantom image (top left), Fourier domain representation (bottom left), observation of 15% of the frequency coefficients sampled along 42 radial lines (bottom right) and sparse representation in Haar wavelet domain (top right).

The physical process of acquiring MRI images is equivalent of taking one dimensional slices from the 2 dimension Fourier domain of the image under investigation. The magnitude of the 2-D Fourier transform is shown in the bottom left panel of figure 5. The measuring matrix M here only takes a small subset of these slices as shown in the bottom right panel in figure 5.

In order to reconstruct the original phantom image, we utilise the fact that the image has a sparse representation in the Haar wavelet transform. The Haar wavelet coefficients are shown in the top right panel of figure 5, where we plot the logarithm of the absolute of these coefficients. For this particular image of size 256×256, it was observed that the original image is well approximated (over 300 dB peak signal to noise ratio) using only 4000 of the wavelet coefficients.

We compare the performance of OMP, MP, GP and ACGP using between 30 and 52 radial lines from the 2 dimensional Fourier domain as the measurements. We here used the implementations of the algorithms in which we allowed the algorithms to select elements repeatedly. All algorithms were run until they had selected 4000 different elements. After the selection of 4000 elements with the different algorithms, we additionally calculated the vectors $\mathbf{y}_{\Gamma^n}$ that minimised equation (6) for the particular elements selected with each of the methods.

For comparison, we also used the Gradient Projection for Sparse Reconstruction (GPSR) algorithm proposed in [10] (labelled L1 in figure 6), which solved the problem $\|\mathbf{x} - \mathbf{\Phi y}\|_2^2 + \lambda\|\mathbf{y}\|_1$. We set the parameter λ for each condition such that the algorithm recovered approximately 4000 non-zero elements.

The results are shown in figure 6, where we show the results after the projection onto the selected subset. The solid lines are the results calculated with ACGP, the dashed lines are

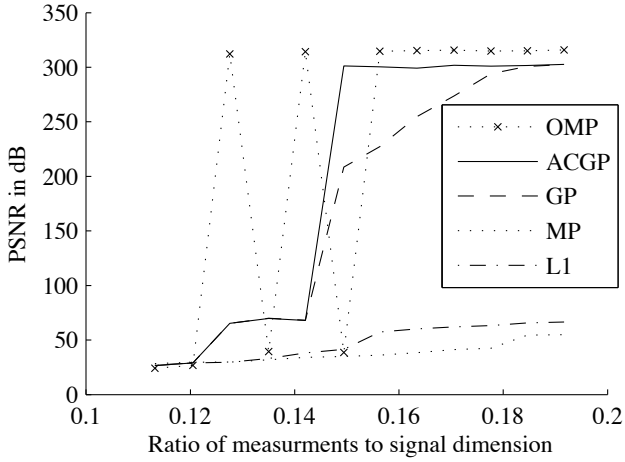
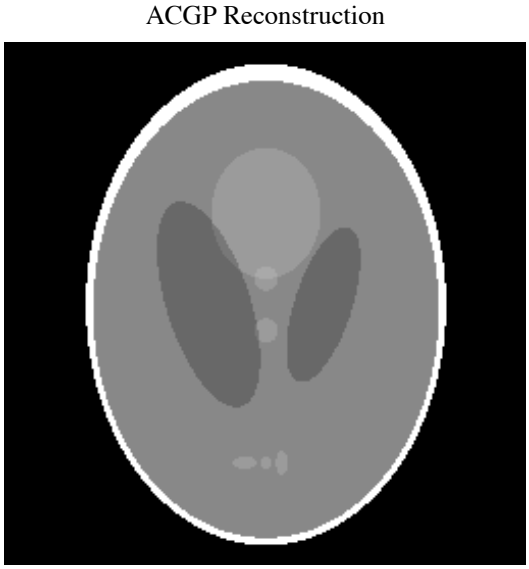


Fig. 6. Comparison between the different algorithms for different numbers of observations in the compressive sampling example. The solid line shows the results for the approximate conjugate gradient method, the dashed line shows the results for Gradient Pursuit and the dotted line shows the performance of Matching Pursuit. Shown are the Peak Signal to Noise Ratio (PSNR) after orthogonal projection onto the selected elements.

the results for GP and the dotted lines are the results for MP. The abscissa here is the ratio of the dimension of the measurement $\mathbf{z}$ to the dimension of the signal $\mathbf{x}$. The ordinate shows the Peak Signal to Noise Ratio (PSNR) in dB. The results for GP and ACGP before projection were nearly the same as those shown here for ratios of measurement to signal dimension above 0.15, while they were several dB less for the other ratios. For MP as well as for GPSR, the results always were several dB better after projection. Running OMP, it is interesting to note that for certain problem sizes the algorithm is able to exactly recover the significant coefficients, while for slightly larger ones it might fail. A visual comparison of the reconstruction for the experiment in which the observation dimension was 15% of the signal dimension is given in figure 7 for the result found with ACGP and the L1 method.

From the results it is clear that MP fails to recover the original signal over the range of measurement dimensions used, while ACGP recovers the original signal for a measurement to signal dimension ratio above 15%, where it furthermore also returns a result close to the projection onto the selected elements. It should be noted that the best PSNR value achievable with 4000 wavelet coefficients was over 300dB for this example and ACGP was here able to nearly recover these elements above a measurement to signal dimension ratio above 15%. GP was found to also perform well, however, we found that this algorithm required more observations to exactly recover the signal. Interestingly, the GPSR algorithm, which solves a constraint quadratic optimisation problem did not seem to be able to recover the correct elements and the performance was found to be worse than that of GP and ACGP. However, GPSR outperformed MP. Note that the better performance of OMP as compared to convex optimisation of the problem was shown for certain problems in [22] (for other problems the performance of OMP can be worse).

All the greedy algorithms were run until they had selected 4000 different elements. Interestingly, MP and OMP were found to be slower in this example than the other two algorithms. The OMP algorithm used here was based on Cholesky factorisation and we used the implementation available in the SparseLab toolbox (http://sparselab.stanford.edu/). MP took roughly twice as long as ACGP and three times as long as GP. This can be explained by looking at the total number of iterations, i.e. at the number of times the algorithms selected elements repeatedly. ACGP as well as GP only selected a small number of elements repeatedly and the total number of iterations was not much more than 4000, MP on the other hand was found to require around 12 000 iterations to select 4000



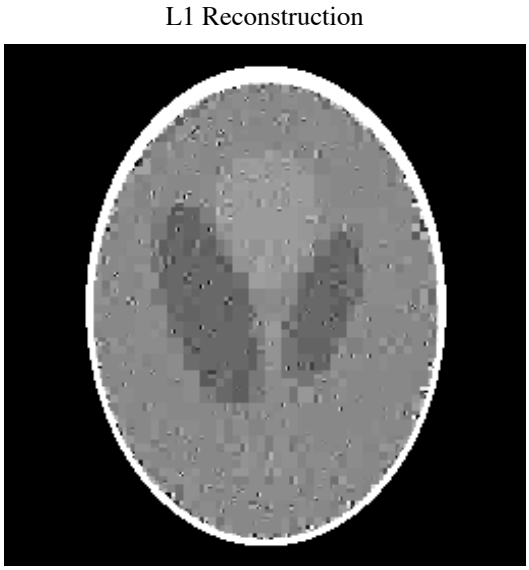


Fig. 7. Magnetic Resonance Imaging (MRI) Example. Comparison between the reconstructed images using the ACGP algorithm and the l_1 algorithm for the experiment in which the observation dimension was 15% of the signal dimension.

TABLE III
COMPARISON BETWEEN THE APPROXIMATE COMPUTATION TIME OF SEVERAL ALGORITHMS RUN ON THE MIR EXAMPLE. TIMES ARE NORMALISED TO THE TIME OF GP.

GP	1.0
ACGP	1.5
MP	3.4
OMP-Chol	3.2
GPSR-PBB	0.4
TIM	0.4
LARS	3.4
Homotopy	3.7

different elements, i.e. MP on average selected each element three times. The GPSR algorithm was found to be very fast and took a fifth of the time of MP to converge, however, the calculated coefficients are far from the minimum least squares solution for the selected coefficients and the required orthogonal projection, which we calculated using a conjugate gradient algorithm, took roughly the same computation time than the GPSR itself. The results also show that GPSR is not able to exactly recover the correct non-zero elements in the example used here. A more thorough study and comparison of the performance of l_1-regularised optimisation methods and OMP can be found in [22].

A comparison between the computational times of the different algorithms is given in table III[6]. The times shown include the times for the calculation of the orthogonal projection onto the selected subset, which was here done using a conjugate gradient algorithm. We also state the computation times for other state of the art algorithms available. In particular we also use the LARS algorithm and the Homotopy method as described in [8], both implemented in the SparseLab toolbox (http://sparselab.stanford.edu/). We also used the Truncated Interior-Point Method for l_1-regularised least squares (which we will call TIM) from [9] an implementation of which is available at (http://www.stanford.edu/∼boyd/l1_ls/). This method is very similar to the GPSR-PBB method in [10], which is available for download from (http://www.lx.it.pt/∼mtf/GPSR/). To facilitate comparison we have here normalised the computation times by dividing them all by the time of GP (which took around 30 minutes on a Apple Macintosh G5 Quad 2.5GHz computer). Note that the results for GPSR-PBB, the homotopy method as well as TIM (which all solve the same convex optimisation problem) where comparable to the results shown above for GPSR-PBB. LARS (which only approximately solves the convex optimisation problem) was found to perform marginally worse.

X. DISCUSSION AND CONCLUSION

Sparse representations are used in many areas of signal processing and efficient algorithms are required to solve many real world problems. In this paper we have introduced a novel extension to greedy Matching Pursuit type algorithms based on directional optimisation. This framework allows different directions to be chosen and we have here discussed three possibilities, the gradient, the conjugate gradient and an approximation to the conjugate gradient. While the conjugate gradient solves the Orthogonal Matching Pursuit (OMP) algorithm exactly, the evaluation of this direction has the same computational complexity as previous implementations of OMP, such as the approach based on QR factorisation. The gradient as well as the approximate conjugate gradient are much easier to calculate, with the gradient being available in Matching Pursuit (MP) for free. The resulting algorithms therefore have the same order of computational complexity as MP.

For many applications, OMP can outperform convex optimisation methods. For large problems, in which the number of non-zero elements is of the order of several thousands or more, the computational requirements and storage demands of currently available implementations of OMP can easily become too large, and faster alternatives are required. In this paper we have suggested two such alternatives, the Gradient Pursuit (GP) algorithm and the Approximate Conjugate Gradient Pursuit (ACGP) method.

Experimental results show that both, GP as well as ACGP outperform MP and often get a performance close to OMP, with ACGP often exhibiting a better performance than GP. In the de-noising example, the performance of GP was comparable to ACGP and the results suggested that the method gives results close to OMP. This is probably due to the particular example used in which the selected elements only influence a small part of the signal. A newly selected element does then not influence the optimal solution for most of the other coefficients.

Greedy strategies often select a single element at a time and therefore require at least as many iteration as the number of non-zero elements to be selected.This could be overcome by selecting more than one element at a time, for example by using a thresholding procedure as recently proposed for the Stagewise Orthogonal Matching Pursuit (StOMP) algorithm [23].

The only draw back of the suggested approach when compared to MP is that in MP it is often possible to update the inner products locally, whenever the Gram matrix is sparse. As the algorithms suggested in this paper update all previously selected elements, such an approach is not directly applicable. Nevertheless, as shown in the experiments, the methods are applicable to larger problems, for which traditional implementations of OMP are not feasible.

[6]Note that we here do not use equation (4) in our MP implementation, which would make MP substantially faster [16].

ACKNOWLEDGMENT

Some of the ideas that led to this paper first arose in a discussion of the second author with Rémi Gribonval whom we like to thank for the inspiration.

REFERENCES

[1] V. K. Goyal, M. Vetterli, and N. T. Thao, "Quantized overcomplete expansions in R^N: Analysis, synthesis and algorithms," *IEEE Transactions on Information Theory*, vol. 44, pp. 16–31, Jan. 1998.

[2] C. Fevotte and S. Godsill, "Sparse linear regression in unions of bases via bayesian variable selection," *IEEE Signal Processing Letters*, vol. 13, no. 7, pp. 441–444, 2006.

[3] M. Davies and N. Mitianoudis, "A simple mixture model for sparse overcomplete ICA," *IEE Proc.-Vision, Image and Signal Processing*, vol. 151, pp. 35–43, August 2004.

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

[5] G. Davis, *Adaptive Nonlinear Approximations*. PhD thesis, New York University, 1994.

[6] B. K. Nataratjan, "Sparse approximat solutions to linear systems," *SIAM Journal on Computing*, vol. 24, pp. 227–234, Apr 1995.

[7] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal of Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[8] B. Efron, T. Hastie, I. Johnston, and R. Tibshirani, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[9] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "A method for large-scale l_1-regularised least squares problems with applications in signal processing and statistics.," *submitted*, 2007.

[10] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems," *submitted manuscript*, 2007.

[11] J. F. Murray and K. Kreutz-Delgado, "An improved FOCUSS-based learning algorithm for solving sparse linear inverse problems," in *Conf. Record of the Thirty-Fifth Asilomar Conf. on Signals, Systems and Computers*, pp. 347–351, 2001.

[12] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[13] Y. C. Pati, R. Rezaifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in 27^th *Asilomar Conf. on Signals, Systems and Comput.*, Nov. 1993.

[14] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification.," *International Journal of Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[15] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[16] S. Krustulovic and R. Gribonval, "MPTK: Matching pursuit made tractable," in *Proc. Int. Conf. on Acoustic Speech and Signal Processing*, (Toulouse, France), May 2006.

[17] S. Mallat, G. Davis, and Z. Zhang, "Adaptive time-frequency decompositions," *SPIE Journal of Optical Engineering*, vol. 33, pp. 2183–2191, July 1994.

[18] G. H. Golub and F. Van Loan, *Matrix Computations*. Johns Hopkins University Press, 3rd ed., 1996.

[19] R. Gribonval and P. Vandergheynst, "On the exponential convergence of matching pursuits in quasi-incoherent dictionaries," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 255–261, 2006.

[20] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Addison Wesley, September 1999.

[21] M. Lustig, D. L. Donoho, and J. M. Pauly, "Sparse mri: The application of compressed sensing for rapid mr imaging," *submitted*, 2007.

[22] D. L. Donoho and Y. Tsaig, "Fast solutions of l_1-norm minimisation problems when the solution may be sparse," *manuscript*, 2006.

[23] D. Donoho, Y. Tsaig, I. Drori, and J. Starck, "Sparse solutions of underdetermined linear equations by stagewise orthogonal matching pursuit.," 2006.