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 increasing problem sizes now addressed lead to 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 improvements to greedy strategies are proposed and algorithms are developed that approximate Orthogonal Matching Pursuit with computational requirements more akin to Matching Pursuit. Three different directional optimisation schemes based on the gradient, the conjugate gradient and an approximation to the conjugate gradient are discussed respectively. It is shown 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 approximations to Orthogonal Matching Pursuit, 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 in general [5], [6]. Therefore, different sub-optimal strategies are used in practise. Commonly used strategies are often 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 problem have been suggested in [9] and [10]. Non-convex local optimisation procedures include the Focal Underdetermined System Solver FOCUSS [11] and Bayesian approaches such as the Relevance Vector Machine, also known as Sparse Bayesian Learning [12] [13] or Monte Carlo based approaches such as those in [14], [15], [16], [17] and [18]. In this paper we are interested in greedy methods, the most important of which are Matching Pursuit (MP) [19], Orthogonal Matching Pursuit (OMP) [20] and Orthogonal Least Squares (OLS) [21], also often known as ORMP, OOMP or, in the regression literature, as forward selection.

MP is an algorithm often used for practical applications and there are now very efficient $\mathcal{O}(N \log M)$ (for each iteration) implementations [22], [23] 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 that 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. This 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 subsections, starting with the gradient direction in subsection III-A, followed by the conjugate gradient in subsection III-B and an approximate conjugate gradient in subsection

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.

III-C. Section IV takes a closer look at the computational requirements of the proposed algorithms and compares these to MP and two different OMP implementations. Section V gives theoretic bounds on the convergence of the gradient based algorithm. The paper concludes with a range of experiments presented in section VI. The first experiment explores how the proposed approaches compare with MP and OMP, both, in terms of approximation performance (subsection VI-A) as well as in their ability to exactly recover the underlying sparse structure (subsection VI-B). This is followed by an experiment that highlights the applicability of the methods to an audio de-noising example where OMP is not feasible (subsection VI-C). The final example in subsection VI-D analyses the performance of the methods for compressed sensing and contrasts the performance to other approaches.

B. Notation

Γ^n will denote a 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 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. Individual elements from a vector will be in standard type face with a subscript. For example $\mathbf{g}$ will be used to refer to a gradient vector with g_i denoting the i^{th} element of this vector. Inner products between vectors will often be written using angled brackets, e.g. $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$.

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 algorithms of this form is possibly Matching Pursuit (MP) [19]. A new element is selected based on the inner product between the current residual $\mathbf{r}^n$ and the columns in $\mathbf{\Phi}$ and the corresponding element of $\mathbf{y}$ is updated. MP is summarised as follows

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

Note the slight abuse of notation in the expression $\arg_i \max |g_i^n|$, which in general could return a set of indices. In this case one of the elements would have to be chosen.

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 [19] is to compute the inner products between the residual and the dictionary elements recursively. This can be done using

$$g_i^{n+1} = \left\langle \mathbf{r}^{n+1}, \phi_i \right\rangle = g_i^n - g_{i^n}^n \left\langle \phi_{i^n}, \phi_i \right\rangle, \tag{4}$$

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 stored or efficiently computed and, crucially, whenever these inner products are predominantly zero.

In Orthogonal Matching Pursuit [20], [24] the approximation $\mathbf{y}$ is updated in each iteration by projecting $\mathbf{x}$ orthogonally onto all selected atoms. OMP therefore finds the optimum (in terms of squared error) signal approximation achievable with the selected atoms. This algorithm is

1) Initialise $\mathbf{r}^0 = \mathbf{x}, \mathbf{y}^0 = \mathbf{0}, \Gamma^0 = \emptyset$
2) for $n = 1; n := n + 1$ till stopping criterion is met
   a) $\mathbf{g}^n = \mathbf{\Phi}^T \mathbf{r}^{n-1}$
   b) $i^n = \arg_i \max |g_i^n|$
   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 [24] or Cholesky factorisation [25] are available. The main drawback of these approaches is that they require additional storage, as discussed in detail in section IV. This storage requirement can become an issue for large problems in which $\mathbf{\Phi}$ cannot be stored[1]. The aim of this paper is therefore to develop fast approximate OMP algorithms that require less storage.

III. The Directional Pursuit Framework

In iteration n the problem solved by Orthogonal Matching Pursuit (OMP) is the minimisation over $\mathbf{y}_{\Gamma^n}$ of the quadratic cost-function (in n unknowns)

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

[1]If $\mathbf{\Phi}$ could be stored explicitly as a matrix, then one could most likely also store a QR or Cholesky factorisation of a subset of its columns.

Instead of updating y_{i^n} by adding $g_{i^n}^n$ 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. Different directions $\mathbf{d}_{\Gamma^n}$ can be chosen and three different possibilities will be discussed below.

Once an update direction has been calculated, the step-size a^n can be determined explicitly. As shown in [26, pp. 521], the optimum step size[2] is

$$a^n = \frac{\langle \mathbf{r}^n, \mathbf{c}^n \rangle}{\|\mathbf{c}^n\|_2^2}, \tag{8}$$

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 expression (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.

In MP and OMP, the selection of new elements is based on the inner product between the residual and the dictionary elements. In OMP, elements are not selected repeatedly, because due to the orthogonal projection used, the residual $\mathbf{r}^n$ is always orthogonal to all previously selected elements. If directional optimisation is used to approximate the orthogonal projection, this orthogonality is no longer guaranteed. Because all previously chosen elements are updated in each iteration, it would nevertheless be possible to restrict the selection of new elements to those elements not selected previously. However, a large inner product between $\mathbf{r}^n$ and any atom selected in a previous step indicates that the residual is 'far' from orthogonal to the selected elements (as would be required by the residual in exact OMP). Furthermore, if the algorithm always selects the element with the largest inner product, irrespective of whether this element has previously been selected or not, then the algorithm belongs to the family of *General MP* as defined in [27]. The theoretic results presented in [27] would then also hold for the directional pursuit algorithms. For example, the following theorem from [27] would hold for all algorithms in this paper.

Theorem 1: (Theorem 1 from [27]) Let Γ be an index set and let $\mathbf{x} = \mathbf{\Phi}_\Gamma \mathbf{y}_\Gamma$ for some $\mathbf{y}_\Gamma$. If $\sup_{i\notin\Gamma} \|(\mathbf{\Phi}_\Gamma)^\dagger \phi_i\|_1 \leq 1$, then a general MP algorithm picks up an atom from the set Γ in each step.

These theoretical considerations, as well as experimental results, some of which can be found in section VI, show that it is beneficial to always select the element with the largest inner product. 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.

[2]The step size is optimum in terms of OMP, i.e. in terms of finding the minimum squared error solution in the update direction. Other algorithms, such as LARS can also be understood in terms of directional optimisation. As these algorithms work with a different cost function, the optimal step size in these algorithms is different to the one used here.

The Directional Pursuit family of algorithms can be summarised as follows

1) Initialise $\mathbf{r}^0 = \mathbf{x}, \mathbf{y}^0 = \mathbf{0}, \Gamma^0 = \emptyset$
2) for $n = 1; n := n + 1$ till stopping criterion is met
   a) $\mathbf{g}^n = \mathbf{\Phi}^T \mathbf{r}^{n-1}$
   b) $i^n = \arg_i \max |g_i^n|$
   c) $\Gamma^n = \Gamma^{n-1} \cup i^n$
   d) calculate update direction $\mathbf{d}_{\Gamma^n}$
   e) $\mathbf{c}^n = \mathbf{\Phi}_{\Gamma^n}\mathbf{d}_{\Gamma^n}$
   f) $a^n = \frac{\langle \mathbf{r}^n, \mathbf{c}^n \rangle}{\|\mathbf{c}^n\|_2^2}$
   g) $\mathbf{y}_{\Gamma^n}^n := \mathbf{y}_{\Gamma^n}^{n-1} + a^n \mathbf{d}_{\Gamma^n}$
   h) $\mathbf{r}^n = \mathbf{r}^{n-1} - a^n \mathbf{c}^n$
3) Output $\mathbf{r}^n, \mathbf{y}^n$

Different choices for the update direction are feasible and the gradient $\mathbf{g}^n$ discussed in subsection III-A is the obvious choice. However, the optimal choice of $\mathbf{d}_{\Gamma^n}$ would find the minimum of (6) in a single step. It turns out that this optimal direction is a conjugate direction as discussed in subsection III-B and can be evaluated explicitly. Unfortunately, the evaluation of this direction requires computational resources similar to an OMP implementation based on QR factorisation. In subsection III-C we therefore propose the use of an approximation to the conjugate direction which can be evaluated more efficiently.

A. Gradient Pursuit

The gradient of expression 6 with respect to $\mathbf{y}$ is

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

Using this gradient as the update direction gives a directional pursuit algorithm we call *Gradient Pursuit* (GP). It is important to realise that this gradient is exactly the vector $\mathbf{g}^n$ 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 iteration, so that the use of this gradient comes at no additional computational cost. 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.

B. Conjugate Gradient Pursuit

Another popular directional optimisation algorithm is the conjugate gradient method [26, Section 10.2], [28, 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[3]. 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 $\mathbf{G}\mathbf{y} = \mathbf{b}$ for $\mathbf{y}$). 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

$$\mathbf{d}_n^T \mathbf{G} \mathbf{d}_k = 0 \tag{10}$$

[3]At least for infinite precision arithmetic. See for example [26] for a detailed discussion regarding stability issues.

for all $k \neq n$. More details can be found in for example [26, Section 10.2] and [28, 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 $\mathbf{G}_{\Gamma^n} = \mathbf{\Phi}_{\Gamma^n}^T \mathbf{\Phi}_{\Gamma^n}$. 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 $\mathbf{d}_{\Gamma^n}^k$ be the k^{th} conjugate direction. 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 $\mathbf{d}_{\Gamma^n}^n$, 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 $\mathbf{d}_{\Gamma^k}^k$ (note the superscript in the subscript) as higher dimensional vectors $\mathbf{d}_{\Gamma^n}^k$, where the elements associated with the 'new' dimensions are set to zero.

To derive the algorithm, we recall the conjugate gradient Theorem [28, Theorem 16.2], which we give here using the notation introduced above.

Theorem 2: [28, Theorem 16.2] Let $[\mathbf{d}_{\Gamma^n}^1, \mathbf{d}_{\Gamma^n}^2, \ldots, \mathbf{d}_{\Gamma^n}^n]$ be any set of non-zero $\mathbf{G}_{\Gamma^n}$-conjugate vectors, then the solution to the problem $\mathbf{G}_{\Gamma^n} \mathbf{y}_{\Gamma^n} = \mathbf{\Phi}_{\Gamma^n}^T \mathbf{x}$ is

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

with step-sizes

$$a^k = \frac{\left\langle \mathbf{r}^k, \mathbf{\Phi}_{\Gamma^k} \mathbf{d}_{\Gamma^n}^k \right\rangle}{\|\mathbf{\Phi}_{\Gamma^k} \mathbf{d}_{\Gamma^n}^k\|_2^2}, \tag{13}$$

where

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

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 direction. This means that $\mathbf{y}_{\Gamma^n}^n$ 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 direction 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 $\mathbf{D}_{\Gamma^{n-1}}^{n-1}$ to denote the matrix containing all conjugate update directions from iteration $n-1$ and the matrix $\mathbf{D}_{\Gamma^n}^{n-1}$ 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 $(\mathbf{D}_{\Gamma^n}^{n-1})^T \mathbf{G}_{\Gamma^n} \mathbf{D}_{\Gamma^n}^{n-1} = \mathbf{B}$, where $\mathbf{B}$ is a diagonal matrix. Because the last row of $\mathbf{D}_{\Gamma^n}^{n-1}$ contains only zeros, the last row and column of $\mathbf{G}_{\Gamma^n}$ are multiplied by zeros, which implies that

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

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

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

We write each new direction as a combination of all previously chosen directions and the current gradient $\mathbf{g}_{\Gamma^n}$ [26, Section 10.2]

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

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

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

from which we can write

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

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

Note that in the standard conjugate gradient algorithm [26, Section 10.2], [28, 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 $\mathbf{D}_{\Gamma^n}^{n-1} \mathbf{G}_{\Gamma^n}$ can be updated recursively by adding a single new row and column in each iteration. Note also that $((\mathbf{D}_{\Gamma^{n-1}}^{n-2})^T \mathbf{G}_{\Gamma^{n-1}} \mathbf{D}_{\Gamma^{n-1}}^{n-2})^{-1}$ and $((\mathbf{D}_{\Gamma^n}^{n-1})^T \mathbf{G}_{\Gamma^n} \mathbf{D}_{\Gamma^n}^{n-1})^{-1}$ are equal apart from a single additional value added in each iteration. This value is $\|\mathbf{c}^{n-1}\|_2^2$ 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 [20], which also uses a directional update. However, the method in [20] 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, $\mathbf{r}^n = \mathbf{r}^{n-1} - \langle \mathbf{q}, \mathbf{x} \rangle \mathbf{q}$, where $\mathbf{q}$ is the

newly added column in $\mathbf{Q}_{\Gamma^n}$. Using $\mathbf{z}_{\Gamma^n} = \mathbf{Q}_{\Gamma^n}^T \mathbf{x}$, the solution becomes $\mathbf{y}_{\Gamma^n} = \mathbf{R}^{-1}\mathbf{z}_{\Gamma^n}$, 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 $(\mathbf{D}_{\Gamma^n}^n)^T\mathbf{\Phi}_{\Gamma^n}^T\mathbf{\Phi}_{\Gamma^n}\mathbf{D}_{\Gamma^n}^n$ must be diagonal. Also, it can be shown by induction that, with appropriate diagonal weighting matrix $\mathbf{W}$

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

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.

C. Approximate Conjugate Gradient Pursuit

In the previous section we derived a novel implementation of Orthogonal Matching Pursuit (OMP), which is similar to the QR factorisation based method, both in terms of computational cost and storage requirements. This new insight is the basis for the development of a fast algorithm derived in this section. Instead of calculating the exact conjugate direction, we propose to us an approximately conjugate direction. This suboptimal direction will then only approximate OMP. The benefit of this approach are, however, that the approximate conjugate gradient is much easier to calculate than the full conjugate gradient and that 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 derive the method using a single update direction. The more general derivation follows similar arguments.

The approximate conjugate direction is now a combination of the current gradient and the previous direction

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

We can again set $b_0 = 1$ and enforce $\mathbf{G}_{\Gamma^n}$-conjugacy to the previous update direction

$$\left\langle (\mathbf{G}_{\Gamma^n}\mathbf{d}_{\Gamma^n}^{n-1}), (\mathbf{g}_{\Gamma^n}^n + b_1\mathbf{d}_{\Gamma^n}^{n-1})\right\rangle = 0, \tag{25}$$

from which we can calculate b_1

$$b_1 = -\frac{\left\langle (\mathbf{\Phi}_{\Gamma^n}\mathbf{d}_{\Gamma^n}^{n-1}), (\mathbf{\Phi}_{\Gamma^n}\mathbf{g}_{\Gamma^n}^n)\right\rangle}{\|\mathbf{\Phi}_{\Gamma^n}\mathbf{d}_{\Gamma^n}^{n-1}\|_2^2}. \tag{26}$$

For an efficient implementation it is worth noting that $\mathbf{\Phi}_{\Gamma^n}\mathbf{d}_{\Gamma^n}^{n-1} = \mathbf{c}^{n-1}$ has been evaluated in the previous iteration to determine the step size a^{n-1} and the same is true for the denominator, which is nothing else than $\|\mathbf{c}^{n-1}\|_2^2$ calculated in the previous iteration.

IV. Computational Complexity

In this paper we have proposed three algorithms to solve or approximate OMP. When using the exact conjugate direction in the proposed framework, the algorithm is a novel OMP implementation. The above analysis shows that the computational complexity and the storage requirements of this method are similar to an OMP algorithm based on QR factorisation. The other two algorithms can have computational benefits compared to exact OMP implementations.

The computational requirements of each algorithm depend on several parameters and an informed algorithm choice requires a detailed study and comparison of the methods, both, in terms of floating point operations (flops) and in terms of storage requirements. We therefore summarise the required resources for GP and ACGP and contrast them to two efficient implementations of OMP, namely a QR factorisation based implementation and a Cholesky factorisation based approach. For completeness, we also look at an MP implementation.

For most large scale problems, for which the proposed algorithms are of particular interest, the matrix $\mathbf{\Phi}$ cannot be stored explicitly in general and structure in this matrix has to be exploited. This often leads to efficient implementations for matrix vector products of the form $\mathbf{\Phi}\mathbf{y}$ and $\mathbf{\Phi}^T\mathbf{x}$. For example these operations can often be based on fast Fourier transforms or fast wavelet transform algorithms. The exact computation cost then depends on the exact dictionary used. In the following discussion the cost of these operations is therefore listed not in flops, but is specified only in terms of *dictionary-vector* multiplications. For example, fast Fourier based transforms often allow these products to be calculated in $\mathcal{O}(N\log(M))$, whilst the worst case complexity for totally unstructured dictionaries would be $\mathcal{O}(MN)$ flops. But even if fast transforms are used, it should be noted that these matrix vector products often remain the most costly operations in the proposed algorithms. Efficient implementations of these products are therefore paramount in practical situations.

In the following discussion we look at the computational requirements for each algorithm when selecting the n^{th} element. It is worth bearing in mind that $N \geq M$ and that for OMP $n \leq M$.

A. Resources required by all algorithms

All algorithms require the computation of the gradient $\mathbf{g} = \mathbf{\Phi}^T\mathbf{r}$, which costs one dictionary vector multiplication and the search for the largest element in $|\mathbf{g}|$, which can be done in N operations. In addition to the storage of $\mathbf{\Phi}$, all algorithms require the storage of $\mathbf{r}$, $\mathbf{y}_\Gamma$, Γ and $\mathbf{g}$, which are vectors of length M, n, n and N respectively.

TABLE I

COMPARISON OF THE METHODS IN TERMS OF COMPUTATIONAL REQUIREMENTS IN ONE ITERATION.

Algo.	Computation Cost (flops)	Storage Cost (floating point numbers)
MP	$M + [\mathbf{\Phi} + N]$	$[\mathbf{\Phi} + M + 2n + N]$
OMP QR	$2Mn + 3M + [\mathbf{\Phi} + N]$	$Mn + 0.5n(n+1) + [\mathbf{\Phi} + M + 2n + N]$
OMP Cholesky	$2\mathbf{\Phi} + 3n^2 + 2M + [\mathbf{\Phi} + N]$	$0.5n(n+1) + [\mathbf{\Phi} + M + 2n + N]$
GP	$1\mathbf{\Phi} + n + 3M + [\mathbf{\Phi} + N]$	$M + [\mathbf{\Phi} + M + 2n + N]$
ACGP	$2\mathbf{\Phi} + 2n + 4M + [\mathbf{\Phi} + N]$	$M + [\mathbf{\Phi} + M + 2n + N]$

B. MP

Apart from the requirements shared by all algorithms, Matching Pursuit only requires the updating of the residual, which costs M flops.

C. QR factorisation base OMP

The QR factorisation based OMP algorithm stores and updates a QR factorisation of the sub-dictionary $\mathbf{\Phi}_{\Gamma^n}$ in each iteration. The additional storage requirements for this factorisation is one triangular matrix $\mathbf{R}_{\Gamma^n}$ with $n(n+1)/2$ elements and an orthogonal matrix $\mathbf{Q}_{\Gamma^n}$ with Mn elements. Note that even if $\mathbf{\Phi}$ is structured and can be stored efficiently, the QR factorisation is not guaranteed to have any structure.

The additional computational cost for this method is $2Mn + 3M$ flops required to update the QR factorisation in each iteration and M flops to update the residual. As this method does not calculate $\mathbf{y}$ in each iteration, $\mathbf{y}$ has to be calculated after the algorithm has been stopped using back substitution, which costs n_{max}^2 flops, where n_{max} is the total number of elements selected by the method.

D. Cholesky factorisation base OMP

The Cholesky factorisation stores and updates a Cholesky factorisation of the Gramm matrix $\mathbf{G}_{\Gamma^n}$. This requires the storage of a triangular matrix with $n(n+1)/2$ elements.

In each iteration, updating the Cholesky factorisation requires one dictionary vector multiplication, one back substitution (n^2 flops) and an additional $2M$ flops. Calculating the non-zero coefficients $\mathbf{y}$ requires two further back substitutions to be used at the cost of $2n^2$ flops. Finally, updating the error requires one further dictionary vector multiplication.

E. GP

The gradient pursuit algorithm require the additional storage of a vector of length M.

The computation of the step size costs an additional dictionary vector multiplication plus $2M$ flops. Updating $\mathbf{y}$ costs n flops and updating the residual can be done using M flops.

F. ACGP

The approximate conjugate gradient algorithm also require the storage of a vector of length M.

The computation of the update direction now requires one further dictionary vector multiplication and $M + n$ additional flops. The step size is again calculated using one dictionary vector multiplication plus $2M$ flops. Updating $\mathbf{y}$ costs n flops and updating the residual can be done using M flops. Apart from the additional computational cost to evaluate the update direction, the cost is the same as for the gradient pursuit algorithm.

G. Empirical analysis

To illustrate the above discussion we compared the speed of Matlab implementations of the different algorithms (the implementations are available on the first author's web-page) run on a Macintosh G5 (2.5GHz Quad) computer. We compared a randomly generated unstructured dictionary and a structured dictionary that was the union of a discrete Fourier transform and a Dirac basis. Both dictionaries had twice as many columns as rows. We then varied the dimension M of $\mathbf{x}$ up to the maximum size for which all algorithms could extract $M/2$ non-zero elements. We also estimated the time the methods took to extract $M/10$ non-zero elements. From the results in figure 1 it is clear that GP and ACGP are always faster than the OMP implementations. This difference becomes larger if either more elements are extracted or if fast dictionary implementations are available.

H. Summary

The computational requirements for the different strategies are summarised in table I, where the computation costs and storage requirements are given for one iteration. Again, n is the number of elements selected in that iteration. Here $\mathbf{\Phi}$ refers to the storage of the dictionary or the computation of one application of the dictionary or its transpose to a single vector. Costs common to all algorithms are collected together and given in square brackets. Note that typically $N \geq M \geq n$.

For unstructured dictionaries and if M is small enough, the QR factorisation based approach is often faster than the Cholesky factorisation based approach, however, for larger problems, the storage requirements become too large and another approach must be adopted. The Cholesky factorisation based approach requires much less storage than the QR factorisation based method, however, it requires two additional dictionary vector multiplications, which for unstructured dictionaries can be costly. However, the Cholesky factorisation approach can be faster if structured dictionaries are used and if the number of non-zero elements is small, while for larger numbers of extracted non-zero elements and for unstructured dictionaries, the QR factorisation based method is faster. Both GP and ACGP are limited by the speed with which the dictionary vector product can be evaluated. GP requires two such products while ACGP requires three. Both of these methods are generally faster than OMP methods[4] and require

[4]even though for unstructured dictionaries the differences can be small

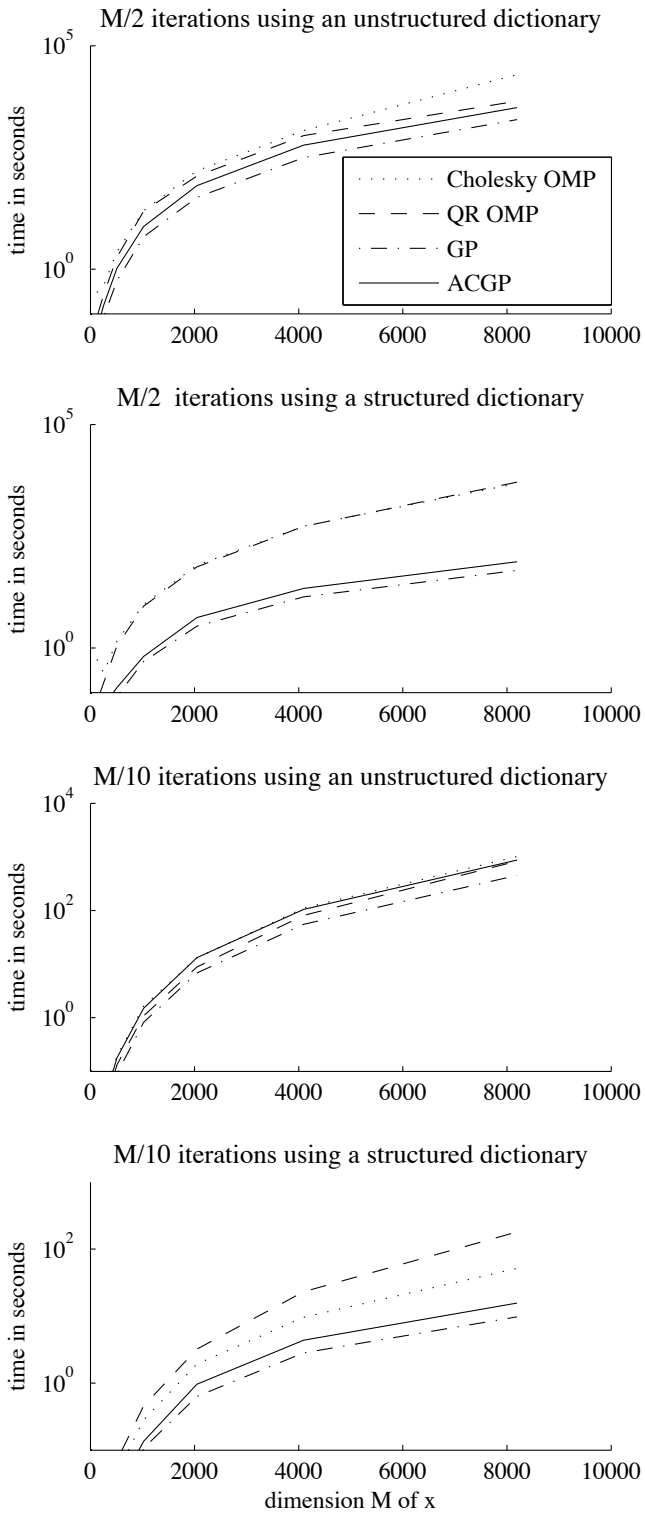


Fig. 1. Comparison between Cholesky based OMP (dotted), QR based OMP (dashed), Gradient Pursuit (dash-dotted) and Approximate Conjugate Gradient Pursuit (solid) for unstructured dictionaries (first and third panels) and for a dictionary that is the union of a Dirac basis and an DFT (second and fourth panel). Time in seconds is shown against dictionary size. All dictionaries where $M \times 2M$. The algorithms were used to extract $\frac{M}{2}$ (top two panels) and $\frac{M}{10}$ (lower two panels) non-zero elements. Whilst GP and ACGP are generally faster than OMP, these differences are larger if either more elements are extracted or if fast dictionary implementations are available.

less storage, so that they are applicable to much larger problem sizes, where OMP is infeasible due to storage demands.

V. Convergence

An important aspect of approximation algorithms is their convergence and we here derive a convergence result for Gradient Pursuit (GP).

Theorem 3: There exist a constant $c < 1$, which only depends on $\mathbf{\Phi}$, such that the residual calculated with GP decays as

$$\|\mathbf{r}^n\|_2^2 \leq c\|\mathbf{r}^{n-1}\|_2^2 \tag{27}$$

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

$$\|\mathbf{r}^n\|_2^2 = \|\mathbf{r}^{n-1}\|_2^2 - \frac{\left\langle \mathbf{r}^{n-1}, \mathbf{\Phi}_{\Gamma^n}\mathbf{d}\right\rangle^2}{\|\mathbf{\Phi}_{\Gamma^n}\mathbf{d}\|_2^2}. \tag{28}$$

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

$$\begin{aligned}\frac{\|\mathbf{\Phi}_{\Gamma^n}^T\mathbf{r}^{n-1}\|_2^4}{\|\mathbf{\Phi}_{\Gamma^n}\mathbf{\Phi}_{\Gamma^n}^T\mathbf{r}^{n-1}\|_2^2} &\geq \frac{\|\mathbf{\Phi}_{\Gamma^n}^T\mathbf{r}^{n-1}\|_2^4}{\|\mathbf{\Phi}_{\Gamma^n}\|_2^2\|\mathbf{\Phi}_{\Gamma^n}^T\mathbf{r}^{n-1}\|_2^2} \\ &\geq \frac{\|\mathbf{\Phi}_{\Gamma^n}^T\mathbf{r}^{n-1}\|_2^2}{\|\mathbf{\Phi}_{\Gamma^n}\|_2^2} \geq \frac{\|\mathbf{\Phi}_{\Gamma^n}^T\mathbf{r}^{n-1}\|_\infty^2}{\|\mathbf{\Phi}_{\Gamma^n}\|_2^2}\end{aligned} \tag{29}$$

By theorem 9.10 in [22, 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 $c = (1 - \frac{\omega}{\|\mathbf{\Phi}\|_2^2})$, where $\|\mathbf{\Phi}\|_2^2 \geq \|\mathbf{\Phi}_{\Gamma^n}\|_2^2$ is the squared Euclidean operator norm of the full dictionary. ■

Note that for Matching Pursuit[5], a corresponding result holds where $c = (1 - \omega)$[22, section 9.5.2], which in general would suggest a faster decay, however, the numerical studies below show that GP outperforms MP in general.

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.

VI. Experimental Evaluation

The proposed algorithms only approximate OMP and should therefore be compared to exact OMP solutions. To do this, we study the average performance of the algorithms on relatively small problems with randomly generated dictionaries. In subsection VI-A we look at two regimes, mildly sparse signals in which the algorithms are not able to exactly recover the correct sparse signal and highly sparse signals in which the algorithms are able to recover the correct atoms. The transition between

[5]The direction $\mathbf{d}$ that minimises the expression in equation (28) is exactly the direction that would give the OMP solution so that the same bound holds for OMP. However in finite dimensions, OMP is guaranteed to converge in a finite number of steps. For the infinite dimensional setting, and for functions with certain smoothness properties, convergence results for OMP have been derived in [29].

these two regimes and the performance of the approximate algorithms during this transition is then studied in more detail in subsection VI-B.

The main advantage of the proposed approximate methods is that they require less storage than fast OMP implementations. They can therefore be applied to large scale problems, where OMP is infeasible. The last two subsections of this section therefore demonstrate the applicability of the proposed approaches to two larger problems. The first example given in subsection VI-C is an audio de-noising problem and the second example in subsection VI-D looks at compressed sensing of MRI images. The problem size of the first of these examples is so large that exact OMP methods cannot be applied to the problem, while the second example is of the maximum size for which the Cholesky factorisation based OMP approach is still feasible. The last example therefore allows us to compare different methods, both in terms of performance and in terms of computational speed.

A. Signal Approximation Performance

To compare the approximate OMP algorithms of this paper to Matching Pursuit and Orthogonal Matching Pursuit, we use a simple toy problem. 1 000 dictionaries of size 128×256 were generated with elements ϕ_i drawn uniformly from the unit sphere. From each dictionary 64 elements were selected at random and multiplied with unit variance zero mean Gaussian coefficients to generate 1 000 different signals.

The averaged results are shown in figure 2 where we plot the approximation error, both in dB (top panels) and on a linear scale (lower panels) against the iteration (left panel) as well as against the number of non-zero coefficients selected (right panels). From these results it is clear that the approximate algorithms perform nearly as well as OMP, while MP performs markedly worse. Therefore, the simple incorporation of a gradient step into matching pursuit significantly improved the performance, whilst using the approximate conjugate gradient offered even larger benefits.

We repeated a similar experiment in which we used several gradient steps in each iteration of GP (not shown). Unsurprisingly, such an increase gives results that get even 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 above experiments were conducted with a signal generated using $M/2$ non-zero elements. As shown in the next subsection, in this regime, the algorithms are not able to exactly recover the exact non-zero coefficients. The experiment was therefore repeated, 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 of this experiment are shown in figure 3, where we again plot the SNR value in both, dB (top panels) and on a linear scale (lower panels) against the iteration count, i.e. against the number of gradient steps (left panels) as well as against the number of selected elements (right panels).

In this scenario, all algorithms were able to recover the correct elements and the error decreased to zero after 12 elements were selected. Looking at the error at each iteration, it can be seen that whilst OMP only required 12 iterations, both, GP and ACGP only required slightly more iterations for the error to drop below 120 dB. Matching Pursuit on the other hand required nearly 100 iterations. Looking at the error plotted on a linear scale, it can be seen that during the first iterations GP and ACGP did not deviate markedly from OMP.

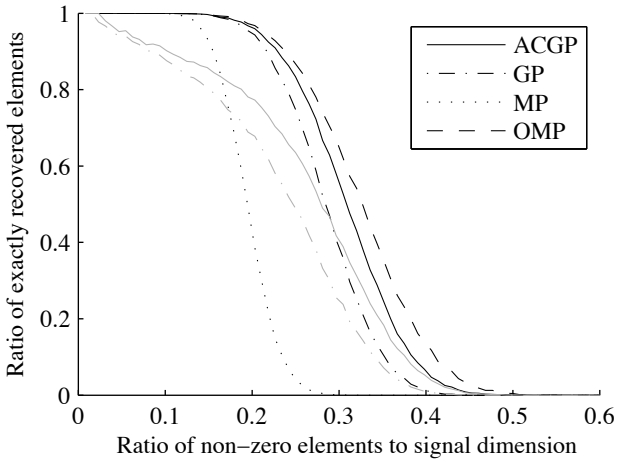


Fig. 4. Comparison of the algorithms in terms of exactly recovering the original coefficients. The ordinate shows the fraction of runs in which the algorithms exactly recovered the index set Γ used to generate the data while the abscissa shows the ratio of the size of Γ to the dimension of $\mathbf{x}$. 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 10 000 runs.

B. Exact Recovery Performance

For highly sparse signals and certain dictionaries, it is known that greedy pursuit type algorithms are guaranteed to exactly recover the elements used to generate the signal [27]. These bounds are worst case bounds and the same bounds hold for all algorithms given in this paper. We therefore 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 4. 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.

OMP again outperforms the other algorithms in terms of retrieving the true signal elements, however, ACGP and GP perform relatively well 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. Matching Pursuit did perform significantly worse than the other methods, for example, matching pursuit requires that less than 20 elements

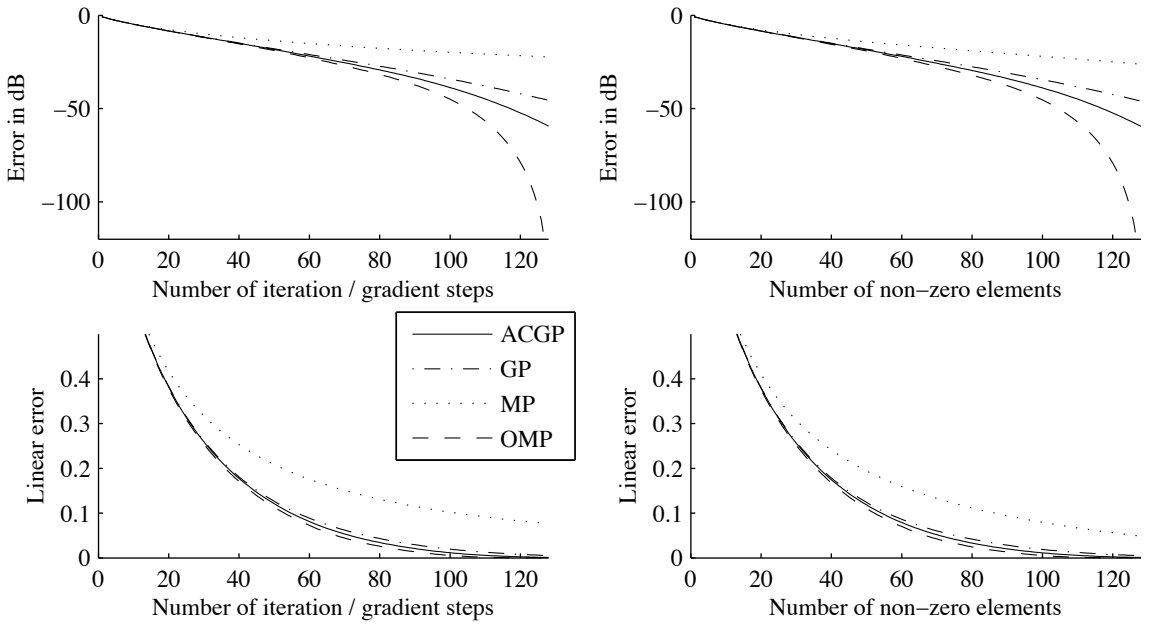


Fig. 2. Comparison between Matching Pursuit (dotted), Orthogonal Matching Pursuit (dashed), Gradient Pursuit (dash-dotted) and Approximate Conjugate Gradient Pursuit (solid) for a signal generated using half as many non-zero coefficients as the dimension of the signal $\mathbf{x}$. The left plots show the error (in dB (top) and on a linear scale (bottom)) plotted against the number of iterations whilst the right panels show the error plotted against the number of selected elements. All results have been averaged over 1 000 runs with dictionary elements drawn uniformly form the unit sphere and non-zero coefficients drawn from a Gaussian distribution.

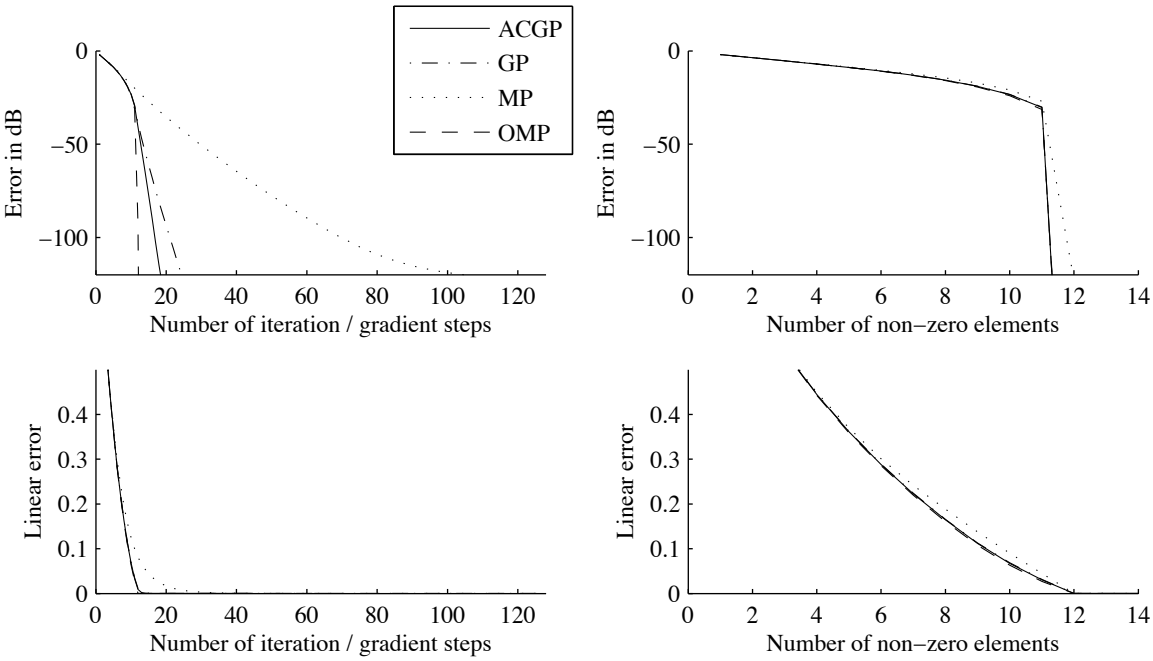


Fig. 3. Comparison between Matching Pursuit (dotted), Orthogonal Matching Pursuit (dashed), Gradient Pursuit (dash-dotted) and Approximate Conjugate Gradient Pursuit (solid) for a signal generated using 12 non-zero coefficients, which is less than a tenth of the dimension of the signal $\mathbf{x}$. The left plots show the error (in dB (top) and on a linear scale (bottom)) plotted against the number of iterations whilst the right panels show the error plotted against the number of selected elements. All results have been averaged over 1 000 runs with dictionary elements drawn uniformly form the unit sphere and non-zero coefficients drawn from a Gaussian distribution.

are non-zero to be able to recover these elements in 90% of cases, whilst our ACGP algorithm can recover up to 28 non-zero elements in 90% of cases, which is an increase of 40%.

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 use an audio de-noising example. We here used MP, GP and ACGP on a 2.5 second long excerpt from a jazz trio audio recording (mono, 44 100 kHz sampling frequency) to which we added i.i.d. Gaussian noise, so that the analysed signal had a signal to noise ratio (SNR) of 20dB. As a dictionary we used a four times overcomplete Modified Discrete Cosine Transform (MDCT), using Tukey windows of length 4096 and 512 with 75% overlap.

We run MP, GP and ACGP (both with a single gradient step), using a fast implementation of the MDCT. We here monitored the performance after each iteration until the methods had retrieved as many coefficients as the signal dimension (110 592 samples)[6].

In audio de-noising, the typical performance measure would be in terms of how close the signal approximation gets to the original *noiseless* signal. This is shown in figure 5 with the solid and dotted lines. In addition, we also show how well the algorithms approximate the *noisy* signal (dashed and dash-dotted lines). The lower panel shows the detail in the region in which the best de-noising performance was observed. We found that the performance of ACGP was virtually identical to GP in this example and we here only show the latter. From 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 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. Compressed Sensing

Whilst the previous experiment demonstrated the performance gains available over matching pursuits in problems where exact OMP algorithms are infeasible, this section studies a large problem that is still solvable with exact OMP. Also, contrasting the previous experiment where the emphasis was on approximating $\mathbf{x}$, we here look at a compressed sensing application in which it is crucial that the algorithm is able to closely approximate the sparse coefficients $\mathbf{y}$. Compressed sensing 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]. The use of OMP to solve the recovery problem in compressed sensing was proposed in [30]. In the language of this paper, assume a signal $\mathbf{z}$ has a sparse representation $\mathbf{z} = \mathbf{\Psi y}$ for some orthogonal transform $\mathbf{\Psi}$. In several applications, it is often not possible to measure all values of $\mathbf{z}$, instead, it is possible to acquire a small number of linear measurements of $\mathbf{z}$ of the form $\mathbf{x} = \mathbf{M z}$, where $\mathbf{M}$ is a measuring matrix and where the dimension of $\mathbf{x}$ is less than that of $\mathbf{z}$. The problem is then to estimate $\mathbf{z}$ given only $\mathbf{x}$, $\mathbf{M}$ and $\mathbf{\Psi}$. If $\mathbf{M}$ and $\mathbf{\Psi}$ have certain properties and if $\mathbf{y}$ is sparse enough, then it is possible to recover $\mathbf{z}$ [4] by finding a sparse representation $\mathbf{y}$ such that $\mathbf{x} = \mathbf{M\Psi y} = \mathbf{\Phi y}$, where the product $\mathbf{\Phi}$ of the sparsity basis $\mathbf{\Psi}$ and the measurement operator $\mathbf{M}$ now takes on the function of the "dictionary".

One particularly promising application domain of com-

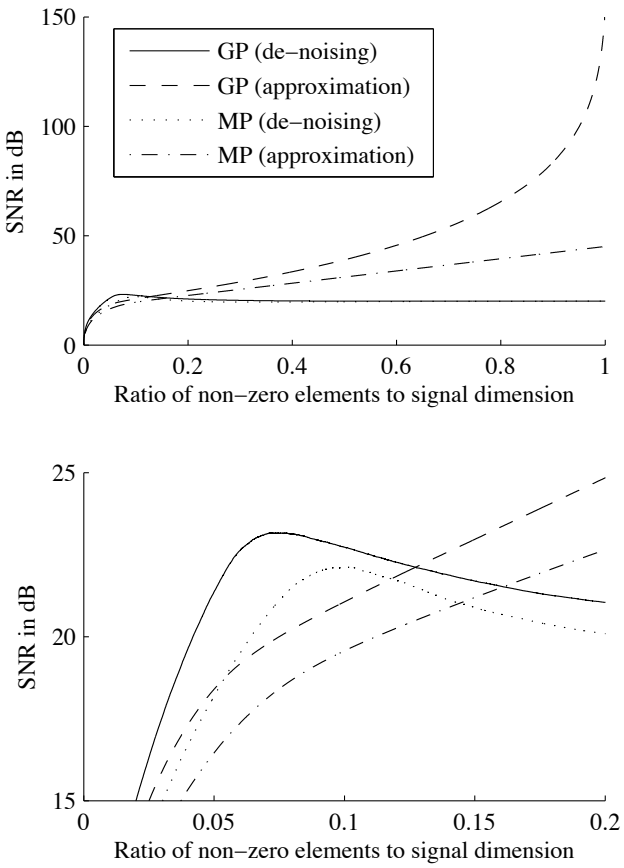


Fig. 5. Comparison of Gradient Pursuit (GP) and Matching Pursuit (MP) in terms of approximating a noisy audio signal (dashed and dash-dotted lines) and in terms of de-noising the noisy audio signal (solid and dotted lines). The lower panel magnifies the area in which the de-noising performance is optimal. The SNR ratio is either the ratio of the signal energy to the error between the approximation and the *noisy* signal (dashed and dash-dotted lines) or the ratio of the signal to the error between the approximation and the *noiseless* signal (solid and dotted lines). This is plotted against the number of non-zero elements selected, which is shown as a ratio to the signal length.

[6]For this example 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 too costly for upper triangular matrixes of this size.

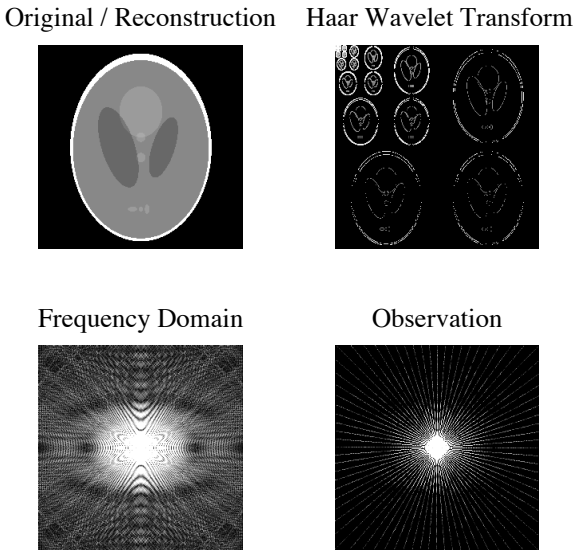


Fig. 6. Magnetic Resonance Imaging (MRI) example. Original phantom image (top left), Fourier domain representation (bottom left), observation of 15% of the frequency coefficients sampled along 42 radial lines (bottom right) and sparse representation in Haar wavelet domain (top right).

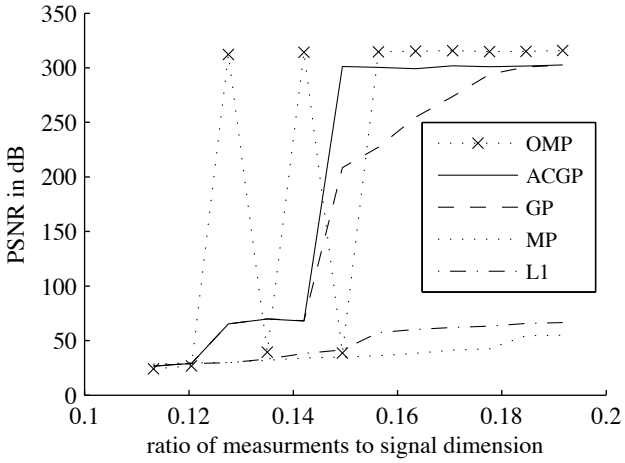


Fig. 7. Comparison between the different algorithms for different numbers of observations in the compressed sensing example. The solid line shows the Peak Signal To Noise Ratio PSNR for the approximate conjugate gradient method, the dashed line shows the PSNR for Gradient Pursuit and the dotted line shows the PSNR for Matching Pursuit. Shown are the PSNR values after orthogonal projection onto the selected elements.

pressed sensing is Magnetic Resonance Imaging (MRI) [31] and we take our example from this area. In particular, we here use the same MRI example as presented in [4][7], which uses the Logan-Shepp phantom. The Logan-Shepp phantom is shown in the top left panel in figure 6.

The physical process of acquiring MRI images is equivalent to taking one dimensional slices from the 2 dimension Fourier domain of the image under investigation. The magnitude of the 2-D Fourier transform of the Logan-Shep phantom is shown in the bottom left panel of figure 6. The measuring matrix $\mathbf{M}$ here only takes a small subset of these slices as shown in the bottom right panel in figure 6.

In order to reconstruct the original phantom image, we utilise the fact that the image has a sparse representation in the Haar wavelet transform, i.e. the sparsity basis $\mathbf{\Psi}$ is a two dimensional discreet Haar transform. The Haar wavelet coefficients are shown in the top right panel of figure 6, 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 again allowed the algorithms to select elements repeatedly and stopped once 4000 different elements had been selected. We then used a conjugate gradient algorithm to calculated the solution vectors $\mathbf{y}_{\Gamma^n}$ that minimised (6) for the particular elements selected with each of the methods.

For comparison, we also used several algorithms that solve the ℓ_1 problem $\|\mathbf{x} - \mathbf{\Phi}\mathbf{y}\|_2^2 + \lambda\|\mathbf{y}\|_1$. We used for example the Gradient Projection for Sparse Reconstruction (GPSR) algorithm proposed in [10] available for download from (http://www.lx.it.pt/∼mtf/GPSR/) and the Truncated Interior-Point Method for l_1-regularised least squares (which we will call TIM) from [9] available at (http://www.stanford.edu/∼boyd/l1_ls/). For both of these methods we set the parameter λ for each condition such that the algorithm recovered approximately 4000 non-zero elements[8]. We also tried the homotopy method and the LARS algorithm discussed in [8] and available in the SparseLab toolbox (http://sparselab.stanford.edu/). We stopped these algorithms after they had selected 4000 elements. All methods gave results comparable to those labelled $L1$ in figure 7, where the Peak Signal to Noise Ratios are plotted after the orthogonal projection onto the selected subset.

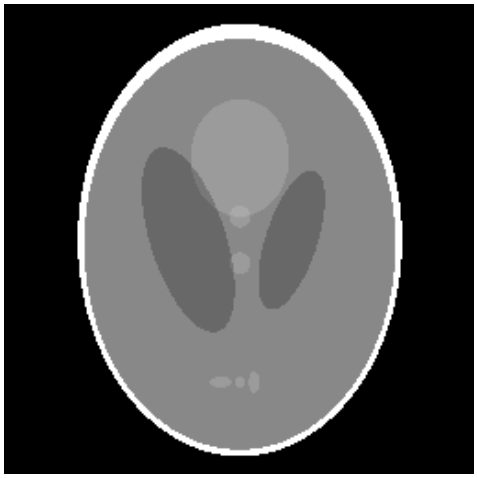
The results for GP and ACGP before projection were close to those shown here for ratios of measurement to signal dimension above 0.15 suggesting that these algorithms returned a good approximation to the orthogonal projection whenever they were able to recover the correct index set. For MP and the ℓ_1 methods, the results always improved by several dB 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. This seems to be due to the fact that for different observation dimensions, completely different Fourier coefficients were selected and that for certain problems the chosen coefficients were not informative enough, even though more coefficients were available. A visual comparison of the reconstruction for the experiment in which the observation dimension was 15% of the signal dimension is given in figure 8 for the result found with ACGP and the ℓ_1 method.

In this example, MP fails to recover the original signal

[7]It is important to note that we here use a Haar wavelet basis in all experiments and not a total variation based constraint as used for example in [4].

[8]We here found the appropriate value for λ by running the methods repeatedly, each time using a different values for λ.

ACGP Reconstruction



L1 Reconstruction

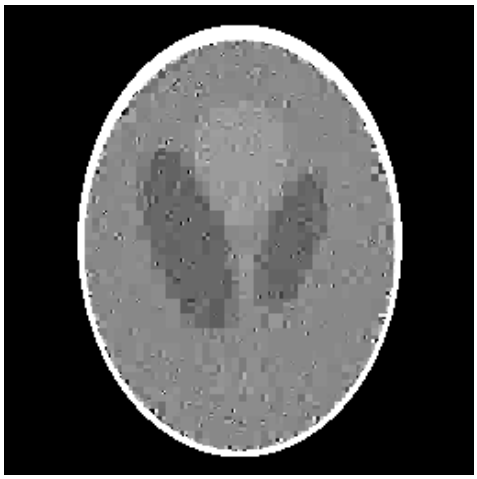


Fig. 8. Magnetic Resonance Imaging (MRI) Example. Comparison between the reconstructed images using the ACGP algorithm and the l_1 algorithm for the experiment in which the observation dimension was 15% of the signal dimension.

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 orthogonal projection onto the selected elements. GP also performed well, however, GP required more observations to exactly recover the signal. Note also that the superior performance of OMP over the ℓ_1 method observed here has previously been observed in [32] for certain problems. For a more thorough study and comparison of the performance of l_1-regularised optimisation methods and OMP we refer to [32].

All the greedy algorithms were run until they had selected 4000 different elements and both, MP and OMP were found to be slower in this example than GP and ACGP. The OMP algorithm used here was based on Cholesky factorisation and

TABLE II
COMPARISON BETWEEN THE APPROXIMATE COMPUTATION TIME OF SEVERAL ALGORITHMS RUN ON THE MRI 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

we used the implementation available in the SparseLab toolbox (http://sparselab.stanford.edu/). Even though each iteration of MP is faster than a single iteration of GP and ACGP, 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 used by the algorithms. 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 different elements. 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 as the GPSR algorithm itself.

A comparison between the computation times of the different algorithms is given in table II[9]. The times shown also include the times for the calculation of the orthogonal projection onto the selected subset. 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).

VII. 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 from the first step of Matching Pursuit (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

[9]Note that we here do not use equation (4) in our MP implementation, which would make MP substantially faster [23].

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 [33] or by using a regularisation approach as that used in the Regularised Orthogonal Matching Pursuit algorithm (ROMP) [34].

The only drawback 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.

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. We also like to thank the anonymous reviewers for their insightful comments that helped to significantly strengthen this paper.

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. Candès, 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. Natarajan, "Sparse approximate 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," *submitte*, 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] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[13] D. Wipf and B. D. Rao, "Sparse Bayesian learning for basis selection," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2153–2164, 2004.

[14] R. E. McCulloch and E. I. George, "Approaches for Bayesian variable selection," *Statistica Sinica*, vol. 7, no. 2, pp. 339–374, 1997.

[15] R. E. McCulloch and E. I. George, "Variable selection via Gibbs sampling," *Journal of the American Statistical Association*, pp. 881–889., September 1993.

[16] J. Geweke, "Variable selection and model comparison in regression," in *Bayesian Statistics 5*. (J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, eds.), Oxford University Press, 1996.

[17] B. A. Olshausen and K. Millman, "Learning sparse codes with a mixture-of-gaussians prior," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 841–847, 2000.

[18] P. J. Wolfe and S. J. Godsil, "Bayesian modelling of time-frequency coefficients for audio signal enhancement," in *Advances in Neural Information Processing Systems (NIPS)* (S. T. S. Becker and K. Obermayer, eds.), (Cambridge, MA), 2003.

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

[20] 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.

[21] 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.

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

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

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

[25] S. F. Cotter, J. Adler, B. Rao, and K. Kreutz-Delgado, "Forward sequential algorithms for best basis selection," in *IEE Proc. Vision Image and Signal Processing*, pp. 235–244, 1999.

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

[27] 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.

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

[29] R. A. DeVore and V. N. Temlyakov, "Some remarks on greedy algorithms," *Advances in Computational Mathematics*, vol. 5, pp. 173–187, 1996.

[30] J. A. Tropp and A. C. Gilbert, "Signal recovery from partial information via orthogonal matching pursuit," *submitted,*, 2006.

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

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

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

[34] D. Needell and R. Vershynin, "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit," *submitted*, 2007.