# Orthogonal matching pursuit using dichotomous coordinate descent iterations

Yuriy Zakharov[†], *Senior Member*, *IEEE*, and Vítor Nascimento[††], *Senior Member*, *IEEE*

*Abstract*— **Greedy algorithms are a family of computationally efficient optimization techniques for solving the sparse representation problem. Matching pursuit (MP) and orthogonal MP (OMP) are popular greedy algorithms; MP possesses the lowest complexity whereas OMP provides better performance. In this paper, we modify OMP using dichotomous coordinate descent iterations and arrive at an algorithm that has performance close to that of OMP and complexity even lower than that of MP.**

## I. Introduction

Sparse recovery techniques find many applications in signal processing. Real-time implementation of such techniques, particularly on FPGAs, is an important area for research [1]–[4]. The matching pursuit (MP) and orthogonal MP (OMP) algorithms are considered as the most suitable candidates for such implementation [1]–[3], [5]. MP is computationally efficient, but its performance is inferior to that of OMP, which is more complicated. The higher OMP complexity is due to a sequence of least squares (LS) problems solved in greedy iterations. In [6], it was proposed to use line search methods [7] to reduce the complexity; specifically, methods based on gradient descent or conjugate gradient iterations were considered. However, the complexity of such modifications of the OMP algorithm is still high.

In this paper, we propose a modification to OMP based on the line search with coordinate descent (CD) iterations. The motivation is that for an LS problem of size $k$, the complexity of the above mentioned methods is $\mathcal{O}(k^2)$ operations per iteration, whereas the complexity of a CD iteration is $\mathcal{O}(k)$. It has been previously recognized that the CD search has an inherent property of being low complexity if the solution is sparse [8]–[10]. CD iterations usually correspond to an exact line search [7], i.e. the step size at each iteration is chosen to minimize the cost function. In [8], [11], it was demonstrated that dichotomous CD (DCD) iterations, though belonging to inexact line search methods, provide an even lower complexity solution to sparse problems with similar convergence rate to that of the exact CD search. The features that differentiate DCD from CD iterations are: (a) no multiplication and division operations, which is highly beneficial in real-time systems [11], and (b) existence of a large proportion of *no-update* iterations possessing especially low complexity. Our modification to OMP possesses a performance close to that of OMP, whereas it requires fewer operations than even the MP algorithm. Moreover, most of the operations in the proposed algorithm are additions, which makes it especially attractive for real-time implementation, e.g. on FPGAs.

[†]Department of Electronics, Univ. of York, UK, yz1@ohm.york.ac.uk. [††]Dept. of Electronic Systems Eng. Univ. of São Paulo, Brazil, vitor@lps.usp.br. The work of Y. V. Zakharov was partly supported by FAPESP grant 2011/06994-2. The work of V. H. Nascimento was partly supported by CNPq and FAPESP grants 303921/2010-2 and 2011/06994-2.

TABLE I
MP algorithm

| Step | Equation |
|---|---|
| | Initialization: $\mathbf{c} = \mathbf{A}^H\mathbf{y}, \mathbf{R} = \mathbf{A}^H\mathbf{A}, \mathbf{x} = \mathbf{0}, I = \varnothing$ |
| 1 | Repeat until termination conditions met: |
| 2 | $\quad q = \arg\max_n \lvert c_n \rvert^2$ |
| 3 | $\quad x_q \leftarrow x_q + c_q, \quad I \leftarrow I \cup q$ |
| 4 | $\quad \mathbf{c} \leftarrow \mathbf{c} - c_q\mathbf{R}^{(q)}$ |

*Notations:* We use capital and small bold fonts to denote matrices and vectors, respectively; e.g. $\mathbf{A}$ is a matrix and $\mathbf{x}$ a vector. Elements of the matrix and vector are denoted as $A_{n,p}$ and $x_n$, respectively. We use $I$ to denote a support (indexes of non-zero elements); the cardinality of $I$ is denoted as $|I|$. We also denote: $\mathbf{A}^{(q)}$ the $q$th column of $\mathbf{A}$; $\mathbf{A}^H$ Hermitian transpose of $\mathbf{A}$; $\mathbf{A}_I$ a matrix obtained from $\mathbf{A}$ keeping only columns corresponding to support $I$; $\mathbf{R}_{I,I}$ a $|I| \times |I|$ matrix obtained from $\mathbf{R}$ extracting elements from rows and columns with indexes in the support $I$; $\mathbf{x}_I$ the subset of $\mathbf{x}$ that contains non-zero entries from $\mathbf{x}$ corresponding to the support $I$; $\Re\{\cdot\}$ and $\Im\{\cdot\}$ are the real and imaginary part of a complex number, respectively; $tr[\cdot]$ the trace operator.

## II. Signal model, MP, OMP and GP algorithms

We deal with the complex-valued linear model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n} \tag{1}$$

where $\mathbf{A} \in \mathbb{C}^{M\times N}$ is the measurement matrix, $\mathbf{n}$, $\mathbf{y} \in \mathbb{C}^{M\times 1}$, and $\mathbf{x} \in \mathbb{C}^{N\times 1}$ are the noise, observation, and unknown vectors, respectively. We are interested in the case $M < N$. It is assumed that only $K < M$ elements of $\mathbf{x}$ are non-zeros, i.e. the vector $\mathbf{x}$ is sparse, and the support is unknown.

Applications of sparse recovery algorithms differ in the possibility of precomputing the matrix $\mathbf{R} = \mathbf{A}^H\mathbf{A}$. If $\mathbf{R}$ cannot be precomputed, the complexity of algorithms presented below will be dominated by the on-line computation of $\mathbf{R}$ or its submatrixes, and thus all the algorithms will have comparable complexity. In other applications, $\mathbf{R}$ can be precomputed or updated in real-time with low complexity as in adaptive filters [12], [13]. Here we are interested in applications where $\mathbf{R}$ is available.

TABLE II
OMP ALGORITHM

| Step | Equation |
|---|---|
| | Initialization: $\mathbf{c} = \mathbf{A}^H\mathbf{y}, \mathbf{x} = \mathbf{0}, \mathbf{R} = \mathbf{A}^H\mathbf{A}, I = \varnothing$ |
| 1 | Repeat until termination conditions met: |
| 2 | $q = \arg\max_n \lvert c_n\rvert^2, \quad I \leftarrow I \cup q$ |
| 3 | Solve $\mathbf{R}_{I,I}\mathbf{x}_I = \mathbf{f}_I$, where $\mathbf{R}_{I,I} = \mathbf{A}_I^H\mathbf{A}_I$ and $\mathbf{f}_I = \mathbf{A}_I^H\mathbf{y}$ |
| 4 | $\mathbf{c} \leftarrow \mathbf{c} - \mathbf{R}_I\mathbf{x}_I$ |

*1) MP algorithm:* MP uses greedy iterations as shown in Table I. At each iteration, one element in $\mathbf{x}$ is selected for updating, based on the maximum magnitude of elements of the residual vector $\mathbf{c}$. The MP complexity in terms of real-valued operations is then given by $P_{\text{MP}} \approx 8MN + 12NL + 4L_g^3$. Note that for simplicity we count one addition, one multiplication, one comparison or one division as a single operation. This is more adequate for DSPs, which perform multiplications in a single clock cycle. The complexity of FPGA implementations, however, ~~also~~ depends on the type of operation; e.g. additions and comparisons are significantly less complicated than multiplications or divisions. Since our proposed algorithm is mostly based on additions and comparisons, this complexity measure will in a sense be the worst-case complexity for it.

The first term of the MP complexity is for computing $\mathbf{c} = \mathbf{A}^H\mathbf{y}$ at the initialization stage. The second term is the complexity of steps 2 and 4, where $L$ is the number of greedy iterations. The third term is the complexity of a debiasing stage (described below) after the support of size $L_g$ is found. Note that usually $L$ is larger than $L_g$ since in MP the same element can be selected multiple times and the final support identification for the debiasing, described by (3) below, can further remove some elements from the support.

We are interested in applications where the objective is a low mean squared error (MSE), e.g. such as estimation of multipath sparse channels [14]–[17]. Therefore, after a sparse recovery algorithm terminates and the final support is identified, a debiasing on the support should be done. This stage computes the minimum MSE (MMSE) estimate of the vector $\mathbf{x}$ using the finally estimated support $I$ as a solution to the equation:

$$(\mathbf{R}_{I,I} + \eta\mathbf{I})\mathbf{x}_I = \mathbf{A}_I\mathbf{y} \qquad (2)$$

where $\eta = \sigma^2|I|/tr[\mathbf{R}_{I,I}]$, $\sigma^2$ is the noise variance and $\mathbf{I}$ is the identity matrix. The final support $I$ is identified as a set of elements in the solution $\mathbf{x}$ found by the greedy algorithm, satisfying the condition

$$|x_k| > \mu \max_n\{|x_n|\} \qquad (3)$$

where $\mu$ $(0 \le \mu < 1)$ is a predefined parameter.

In the MP algorithm, it is assumed that $\mathbf{A}$ is normalized so that $R_{q,q} = 1$, $q = 1, \ldots, N$. If this is not the case, $c_q$ at steps 2 and 3 should be replaced by $c_q/R_{q,q}$.

Different stopping criteria can be incorporated in MP and other algorithms. Here, we will use a limit $L_{\max}$ to the number of greedy iterations and a lower limit $c_{\min}$ to the maximum magnitude of the residual vector $\mathbf{c}$: $\max_n |c_n| < c_{\min}$.

*2) OMP algorithm:* MP does not provide high recovery performance when the percentage of non-zero elements increases. In this case, the OMP algorithm presented in Table II can be used, which at the $k$th greedy iteration solves a $k$-size LS problem (step 3); this results in an increase in complexity compared to MP. Complexity of the OMP algorithm is given by $P_{\text{OMP}} \approx 8MN + 4NL + P_{\text{LS}} + 4NL(L+1) + 4L_g^3$. The first term is for computing $\mathbf{c} = \mathbf{A}^H\mathbf{y}$. The second term is the complexity of selecting a new element into the support (step 2). The third term is for solving the LS problems (step 3). The LS problem can be solved using a direct approach that would result in a complexity of $\mathcal{O}(L_g^4)$. This can be reduced using the Cholesky factorization as suggested in [18]. A smaller complexity can be achieved using a recursive inversion of $\mathbf{R}_{I,I}$; in this case, we have $P_{\text{LS}} \approx 4L_g^3$. The fourth term is for updating the residual $\mathbf{c}$ at step 4. The last term is the complexity of the debiasing after the support is fixed.

*3) GP algorithm:* The GP algorithm uses a number $(N_u)$ of gradient descent iterations per greedy iteration to solve the LS problem in the OMP [6]. If $\mathbf{R}$ is unavailable, the GP complexity is dominated by computation of matrix-vector products (such as $\mathbf{A}^H\mathbf{y}$) [6] at every greedy iteration. Thus, for the complex-valued case, the GP complexity is $P_{\text{GP}} \approx 8MNL$ real-valued operations. The complexity can be reduced if $\mathbf{R}$ is available [6]. In this case, the main contributions into the complexity are from initialization of $\mathbf{c}$ similarly to that in Table I ($8MN$ operations), computation of the step size and update of $\mathbf{c}$ in every gradient descent iteration ($4NN_uL^2$ operations) and debiasing ($4L_g^3$ operations), thus resulting in the total complexity $P_{\text{GP}} \approx 8MN + 4NN_uL^2 + 4L_g^3$.

## III. OMP-DCD ALGORITHM

For solving the LS problem, i.e. minimizing the cost function $J(\mathbf{x}) = ||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2$ on $I$ at the $k$th greedy iteration, we can use line search methods, e.g. conjugate-gradient, gradient-descent or other iterations [6]. The complexity of such iterations is typically $\mathcal{O}(k^2)$ operations. CD iterations is the simplest line search method, with a complexity of $\mathcal{O}(k)$. This happens since a CD direction vector has only one non-zero entry, thus leading to significant reduction in the complexity of matrix-vector multiplications required by the line search.

Suppose that the current solution to the LS problem is given by a vector $\mathbf{x}$. Updating the single $t$-th element $(t \in I)$ of the vector as $\tilde{x}_t = x_t + \alpha$ results in a new solution $\tilde{\mathbf{x}}$, and the decrement of the cost function $\Delta J = J(\tilde{\mathbf{x}}) - J(\mathbf{x})$ is given by $\Delta J = |\alpha|^2 R_{t,t} - 2\Re\{\alpha^* c_t\} = |\alpha|^2 R_{t,t} - 2|\alpha|\Re\{e^{-j\arg(\alpha)}c_t\}$, where we use the representation: $\alpha = |\alpha|e^{j\arg(\alpha)}$. The minimum of $\Delta J$ over $\arg(\alpha)$ is achieved at $\arg(\alpha) = \arg(c_t)$. In this case, $\Delta J = |\alpha|^2 R_{t,t} - 2|\alpha||c_t|$, and this function is minimized if $|\alpha| = |c_t|/R_{t,t}$. Thus, for a fixed $t$, $\alpha = c_t/R_{t,t}$ provides the largest decrement of the cost function; in this case, $\Delta J = -|c_t|^2/R_{t,t}$. If $\mathbf{A}$ is normalized so that $R_{t,t} = 1$, we have $\alpha = c_t$ and $\Delta J = -|c_t|^2$, and therefore it is useful to chose $t$ maximizing $|c_t|$.

It is seen that the MP algorithm is a modification to the OMP algorithm where, for solving the LS problems in greedy iterations, the CD line search is used with one CD iteration

TABLE III
OMP-DCD ALGORITHM

| Step | Equation |
|---|---|
| | Initialization: $\mathbf{c} = \mathbf{A}^H\mathbf{y}, \mathbf{R} = \mathbf{A}^H\mathbf{A}, I = \varnothing$ |
| 1 | Repeat until termination conditions met: |
| 2 | $q = \arg\max_n \lvert c_n\rvert^2,\; I \leftarrow I \cup q,\; \delta = H$ |
| 3 | for $m = 1$ to $M_b$ repeat: |
| 4 | $\delta = \delta/2,\; \boldsymbol{\alpha} = [\delta,\, -\delta,\, j\delta,\, -j\delta]$, Flag = 0 |
| 5 | for $n = 1$ to $\lvert I\rvert$ repeat: $p = I(n)$ |
| 6 | for $t = 1$ to 4 repeat: |
| 7 | if $\Re\{\alpha_t c_p^*\} > R_{p,p}\delta^2/2$ do: |
| 8 | $x_p \leftarrow x_p + \alpha_t,\; \mathbf{c} \leftarrow \mathbf{c} - \alpha_t \mathbf{R}^{(p)}$, Flag = 1 |
| 9 | if Flag = 1, go to step 5 |

per greedy iteration ($N_u = 1$). With an increase in $N_u$, the performance of the algorithm would approach the performance of the OMP algorithm as, with $N_u \to \infty$, the CD line search solves precisely the LS problem; the latter is also true for DCD iterations. Although CD iterations have low complexity comparing to other line search techniques, the complexity can be further reduced when using DCD iterations.

With DCD iterations, elements of $\mathbf{x}$ are fixed-point numbers with $M_b$ bits within an amplitude interval $[-H, H]$. In DCD iterations, first the most significant bits and then less significant bits of the solution are updated. This is controlled by a step-size $\delta > 0$ that starts with $\delta = H$ and then is reduced as $\delta \leftarrow \delta/2$ for a less significant bit. The OMP-DCD algorithm is presented in Table III. For the complex-valued DCD search, four directions of possible updates of $x_t$ on the complex plane are to be considered, defined by the vector $\boldsymbol{\alpha} = [\delta, -\delta, j\delta, -j\delta]$, where $j = \sqrt{-1}$. Note that the step size values need not be recalculated for each DCD iteration (as required in exact line search methods), they are predefined. Avoiding this computation significantly reduces the complexity. With a choice of $H$ as a power-of-two number, the algorithm is well suited to implementation on hardware platforms, e.g. FPGAs, as all its multiplications and divisions are replaced by bit-shifts. Moreover, there are many exceptionally simple DCD iterations (*no-update* or *unsuccessful* iterations), requiring only one comparison (step 7), when the solution is not updated. In the OMP-DCD algorithm, we use $N_u$ to define the maximum number of *successful* DCD iterations; it should be chosen as $N_u \ll N$ as, due to the accurate initialization of the LS solution within every greedy iteration, a few DCD iterations are enough to achieve high accuracy.

Complexity of the OMP-DCD algorithm is given by $P_{\text{OMP-DCD}} \approx 8MN + 4NL + 2C_uN + C_i + 2N_{\text{deb}}L$. The first term is for computing $\mathbf{c}$ at the initialization stage. The second term is for selection of elements in the support (step 2). The term $2C_uN$ is for updating $\mathbf{c}$ in the total number $C_u$ of successful DCD iterations (step 8); each update requires only $2N$ real-valued additions, as multiplications by $\alpha$ are bit-shift operations. The next term takes into account $C_i$ (total number of DCD iterations) tests at step 7 to decide if the DCD iteration is successful. The debiasing (last term) is now done by using extra $N_{\text{deb}}$ DCD iterations at the finally fixed support.

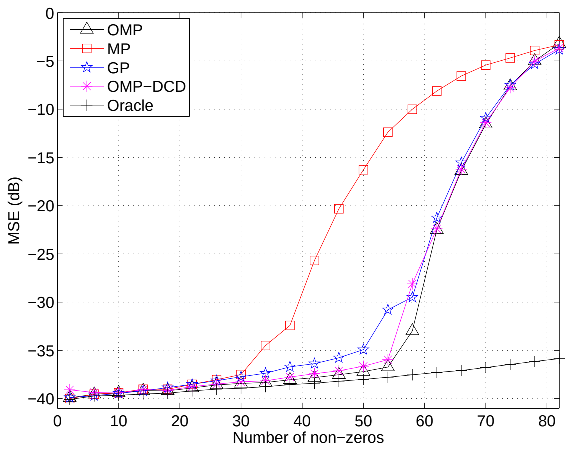

Fig. 1. MSE performance of the sparse recovery algorithms.

Note that the OMP-DCD algorithm can deal with arbitrary (not only normalized) matrices $\mathbf{A}$, whereas the MP algorithm as described above requires the normalization. This is not a high difference in complexity in terms of the total number of operations. However, the normalization involves a division operation, which is difficult for implementation on real-time design platforms (such as FPGAs). The OMP-DCD algorithm does not involve division operations. Moreover, DCD iterations, which comprise the largest part of the OMP-DCD complexity, are multiplication-free and division-free.

## IV. Numerical results

For simulation, we use matrices $\mathbf{A}$ of size $128 \times 256$ with entries generated as follows. In each simulation trial, we generate a binary vector of length $N + M$ with independent random entries $\pm 1/\sqrt{M}$ each with probability $1/2$; this vector may represent a pilot signal in a communication system. The $n$-th column of $\mathbf{A}$ is then obtained from the pilot signal by taking its entries from $n$ to $n + M - 1$; the columns of $\mathbf{A}$ have unit energy. Positions of the $K$ non-zero elements of $\mathbf{x}$ are chosen randomly, the non-zero elements are generated as independent complex-valued Gaussian zero-mean random numbers of unit variance and then $\mathbf{x}$ is normalized to the energy $K$. The noise vector $\mathbf{n}$ contains complex-valued random Gaussian entries of variance $\sigma^2$. For each fixed $K$, we run 1000 simulation trials and average the MSEs obtained in the trials. In all the algorithms, we set the maximum number of greedy iterations $L_{\text{max}} = 100$, the threshold for debiasing $\mu = 0.035$, and $c_{\text{min}}$ in each trial is computed as $c_{\text{min}} = 0.02 \max_n \lvert c_n\rvert$ at the first greedy iteration. In the OMP-DCD algorithm, we set $N_u = 32$, and, for the debiasing, we use extra $N_{\text{deb}} = 512$ DCD iterations. We also set $M_b = 6$ and $H = 4$ for DCD iterations within the greedy iterations and $M_b = 13$ and $H = 4$ in the debiasing stage. On the plots below, we also show the MSE performance of an *oracle* algorithm that, in each simulation trial, knows the true support; it is used for comparison as the lower bound of MSE performance. We also present simulation results for the gradient pursuit (GP) algorithm, which is the simplest line search algorithm proposed in [6].

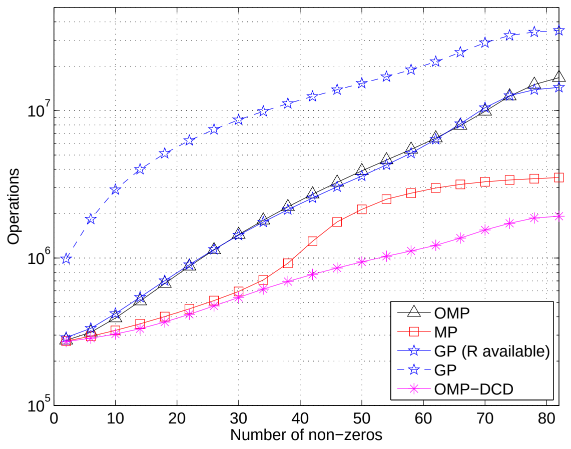

Fig. 2. Complexity of the sparse recovery algorithms in terms of real-valued operations.

Fig.1 shows the MSE performance and Fig.2 shows the complexity of the algorithms in terms of real-valued operations. It is seen that OMP, GP, and OMP-DCD algorithms significantly outperform the MP algorithm in the number of non-zero elements that can be recovered. The MSE performance of the OMP-DCD algorithm is close to that of OMP; it is also close to the oracle MSE up to the number of non-zeros $K = 54$, which is as high as about 40% of the number of measurements $M = 128$.

The OMP-DCD complexity is lower than that of the other algorithms, including the MP complexity, for all values of $K$. For example, for $K = 50$, where MP produces an MSE of -16 dB, the OMP-DCD complexity is about 2 times lower than that of MP and 3 times lower than that of OMP. At $K = 70$, with an MSE performance of approximately -11 dB, still acceptable for channel estimates in some communications scenarios, the OMP-DCD algorithm requires about 6 times less operations than OMP. What, in addition, is very important is the fact that most of the operations in the OMP-DCD algorithm are additions, that are much easier for real-time implementation than multiplications. E.g., for $K = 50$, almost 85% of operations in the OMP-DCD algorithm are additions.

From Fig. 2, it is seen that, if $\mathbf{R}$ is unavailable, the GP complexity is significantly higher than that of the other algorithms. However, even if $\mathbf{R}$ is available and only one gradient descent iteration is used ($N_u = 1$), the GP complexity is close to the OMP (implemented using the recursive inversion) complexity and significantly higher than the OMP-DCD complexity. Note that although with $N_u = 1$ the MSE performance of the GP algorithm is close to the OMP performance, it is still inferior to the OMP-DCD performance.

At low $K$, the complexity of all the algorithms is dominated by the term $8MN$ for computing the residual vector $\mathbf{c} = \mathbf{A}^H\mathbf{y}$ at the initialization stage. In some applications, this can be efficiently computed with a much lower complexity; e.g. this is the case in transversal adaptive filtering where efficient recursive procedures exist for such computations [12], [13]. In this case, the majority of computations in the OMP-DCD algorithm becomes multiplication-free. E.g., for $K = 50$, almost 96% of operations in the OMP-DCD algorithm are additions, and only 4% are for the squaring operations at step 2 in Table III.

## V. Conclusions

In this paper, we proposed a modification to orthogonal matching pursuit. The purpose of the modification is to reduce the complexity whereas having an MSE performance close to that of OMP. We have shown by simulation that the proposed OMP-DCD algorithm has an MSE performance close to that of OMP and complexity lower than the MP complexity. Besides, most operations in the OMP-DCD algorithm are additions, thus it is well suited to real-time implementation, e.g. on FPGA platforms.

## References

[1] Y. Meng, A. Brown, R. Iltis, T. Sherwood, H. Lee, and R. Kastner, "MP core: algorithm and design techniques for efficient channel estimation in wireless applications," in *Proc. 42nd Design Automation Conf.*, June 2005, pp. 297–302.

[2] B. Benson, A. Irturk, J. Cho, and R. Kastner, "Survey of hardware platforms for an energy efficient implementation of matching pursuits algorithm for shallow water networks," in *Proc. 3rd ASM Int. Workshop on Underwater Networks*, 2008, pp. 83–86.

[3] P. Maechler, P. Greisen, N. Felber, and A. Burg, "Matching Pursuit: Evaluation and implementation for LTE channel estimation," in *Proceedings of IEEE Int. Symp. Circuits and Systems (ISCAS)*, 2010, pp. 589–592.

[4] J. Lu, H. Zhang, and H. Meng, "Novel hardware architecture of sparse recovery based on FPGAs," in *2nd IEEE Int. Conf. on Signal Processing Systems (ICSPS)*, 2010, pp. V1–302–V1–306.

[5] D. Yang, H. Li, G. D. Peterson, and A. Fathy, "Compressed sensing based UWB receiver: Hardware compressing and FPGA reconstruction," in *43rd Annual Conference on Information Sciences and Systems, CISS 2009*, 2009, pp. 198–201.

[6] T. Blumensath and M. E. Davies, "Gradient pursuits," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2370–2382, 2008.

[7] G. H. Golub and C. F. Van Loan, *Matrix computations*, The Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[8] Y. V. Zakharov and T. C. Tozer, "Multiplication-free iterative algorithm for LS problem," *Electonics Letters*, vol. 40, no. 9, pp. 567–569, 2004.

[9] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, "Pathwise coordinate optimization," *The Annals of Applied Statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[10] T. T. Wu and K. Lange, "Coordinate descent algorithms for Lasso penalized regression," *The Annals of Applied Statistics*, vol. 2, no. 1, pp. 224–244, 2008.

[11] J. Liu, Y. V. Zakharov, and B. Weaver, "Architecture and FPGA design of dichotomous coordinate descent algorithms," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 56, no. 11, pp. 2425–2438, 2009.

[12] Y. Zakharov, G. White, and J. Liu, "Low complexity RLS algorithms using dichotomous coordinate descent iterations," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3150–3161, July 2008.

[13] D. Angelosante, J. A. Bazerque, and G. B. Giannakis, "Online adaptive estimation of sparse signals: Where RLS meets the $l_1$-norm," *IEEE Transactions on Signal Processing*, vol. 58, no. 7, pp. 3436–3447, 2010.

[14] S. F. Cotter and B. D. Rao, "Sparse channel estimation via matching pursuit with application to equalization," *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 374–377, 2002.

[15] G. Z. Karabulut and A. Yongacoglu, "Sparse channel estimation using orthogonal matching pursuit algorithm," in *IEEE 60th Vehicular Technology Conference, VTC2004-Fall*, 2004, vol. 6, pp. 3880–3884.

[16] C. R. Berger, Z. Wang, J. Huang, and S. Zhou, "Application of compressive sensing to sparse channel estimation," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 164–174, 2010.

[17] C. Qi, X. Wang, and L. Wu, "Underwater acoustic channel estimation based on sparse recovery algorithms," *IET Signal Processing*, vol. 5, no. 8, pp. 739–747, 2011.

[18] D. L. Donoho, I. Drori, Y. Tsaig, and J. L. Starck, *Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit*, Department of Statistics, Stanford University, 2006.