algorithms

Article

Improved Convergence Speed of a DCD-Based Algorithm for Sparse Solutions

Zhi Quan * and Shuhua Lv

School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China; shuhualv@163.com
* Correspondence: iezquan@zzu.edu.cn

Received: 6 May 2020; Accepted: 26 May 2020; Published: 4 June 2020

Abstract: To solve a system of equations that needs few updates, such as sparse systems, the leading dichotomous coordinate descent (DCD) algorithm is better than the cyclic DCD algorithm because of its fast speed of convergence. In the case of sparse systems requiring a large number of updates, the cyclic DCD algorithm converges faster and has a lower error level than the leading DCD algorithm. However, the leading DCD algorithm has a faster convergence speed in the initial updates. In this paper, we propose a combination of leading and cyclic DCD iterations, the leading-cyclic DCD algorithm, to improve the convergence speed of the cyclic DCD algorithm. The proposed algorithm involves two steps. First, by properly selecting the number of updates of the solution vector used in the leading DCD algorithm, a solution is obtained from the leading DCD algorithm. Second, taking the output of the leading DCD algorithm as the initial values, an improved soft output is generated by the cyclic DCD algorithm with a large number of iterations. Numerical results demonstrate that when the solution sparsity γ is in the interval $[1/8, 6/8]$, the proposed leading-cyclic DCD algorithm outperforms both the existing cyclic and leading DCD algorithms for all iterations.

Keywords: dichotomous coordinate descent; leading DCD; cyclic DCD; sparse systems

1. Introduction

With the development of information technology, the number of participated devices and data transmission rate have substantially increased in recent years. Solving the problems in a wide range of signal processing applications is equivalent to getting the solution of a linear least squares (LS) problem [1]. These applications include adaptive antenna array applications [2], multi-user detection [3], multiple-input multiple-output (MIMO) detection [4], echo cancellation [5], equalization [6], and system identification [1,7–9]. If the channel information is known, zero forcing (ZF) algorithm and minimum mean-square error (MMSE) algorithm are popular to be used in these applications. They are simple to implement but require the operation of matrix inversion.

The complexity of matrix inversion requires $\mathcal{O}(U^3)$ arithmetic operations, where U is system size [10]. When the system size is large, the complexity of the matrix inversion is prohibitively high. As a result, there are some techniques proposed to solve systems of equations without inverting the matrix. Direct methods, for example Gaussian elimination, Cholesky factorization, and QR decomposition attain a complexity of $\mathcal{O}(U^3)$ [10,11]. Therefore, it is difficult for direct methods to be implemented in real-time signal processing and hardware applications. Using these methods to solve the linear equations of large sparse systems may be prohibitively expensive and are infeasible for practical applications. Iterative methods, for example, the conjugate gradient method and the steepest descent method could achieve fast convergence. In each iteration, they require $\mathcal{O}(U^2)$ complexity. Other iterative methods, for example, the Southwell's relaxation [12], Jacobi [10] and Gauss-Seidel [10], are coordinate descent approaches. At each iteration, they only require $\mathcal{O}(U)$ complexity but have a slower convergence speed. The number of iterations performed decides the computational complexity of these techniques. Iterative methods can solve problems with sparse structures more efficiently compared with direct methods [13]. Moreover, iterative methods have the potential to achieve a good initial guess for the expected results, which can reduce the number of iterations required to obtain a solution [14]. However, some iterative methods include the numerical operations of multiplying or dividing, which are difficult for hardware implementation.

Prior research work investigates the dichotomous coordinate descent (DCD) algorithms that use a reduced number of operations to solve systems of equations. DCD-based algorithms are well known in many applications because the DCD algorithm avoids the operations of multiplying and dividing, which are expensive to implement. In [15–17], by applying the DCD iterations, the numerical complexity of affine projection algorithms for active noise control is reduced. In [18], DCD iterations solve the recursive least squares (RLS) matrix inversion problem by using bit-shift. The DCD algorithm [14,19] is a non-stationary iterative method based on stationary CD techniques. Only $\mathcal{O}(U)$ additions or $\mathcal{O}(1)$ additions are required in each iteration [11]. The DCD algorithm, therefore, is a good choice for real-time hardware implementations. The DCD algorithm variants, cyclic DCD and leading DCD algorithm, were presented in [20]. When scenarios with sparse true solutions are considered, such as multi-path channel estimation and detection of a multi-user system with several unknown active users, the leading DCD algorithm converges faster than the cyclic DCD algorithm. The cyclic DCD algorithm is ideal for solving systems of equations that require a large number of iterations because it converges faster and has a lower error level than the leading DCD algorithm. However, the leading DCD algorithm has a faster convergence rate than the cyclic DCD algorithm in the initial iterations if the solution is not very sparse.

In this paper, we considered a combination of leading and cyclic DCD iterations and proposed a leading-cyclic DCD algorithm for obtaining a sparse solution in systems requiring a large number of updates. The idea is that the leading DCD algorithm uses a small number of iterations to obtain the initial input for the cyclic DCD algorithm. The number of iterations used in the leading DCD algorithm has been thoroughly investigated under different system matrix conditions and solution sparsities. In the proposed algorithm, a range of the number of iterations is determined that will produce an optimal number of updates for the leading DCD algorithm. The results show that the proposed leading-cyclic DCD algorithm improves the convergence speed of the cyclic DCD algorithm and lowers the steady-state level.

Notation: Boldface uppercase and boldface lowercase letters represent matrices and vectors (e.g., $\mathbf{R}$ and $\mathbf{\Gamma}$). Elements of matrix and vector are denoted as $R_{p,n}$ and Γ_n, respectively. The n-th column of $\mathbf{R}$ is denoted as $\mathbf{R}_{:,n}$. $(\cdot)^T$ denotes a matrix transpose, and $(\cdot)^H$ denotes the Hermitian transpose.

2. Preliminaries

We start by introducing the system model used in this work. We then discuss the DCD algorithms used to solve the linear model.

2.1. System Model

The system linear model can be modeled as

$$\mathbf{Z} = \mathbf{A}\mathbf{x}, \tag{1}$$

where $\mathbf{A}$ is a $Q \times U$ matrix, $\mathbf{Z}$ is a $Q \times 1$ received vector, and $\mathbf{x}$ is a $U \times 1$ unknown vector. We assume that $U < Q$ and the matrix and vectors are real-valued. In addition, K ($K < U$) elements of vector $\mathbf{x}$ are non-zero. For example, $\mathbf{x}$ is a sparse vector, and the index set of non-zero values is not given. There are many applications to solve problems with sparse true solutions. The multi-path communication channels, concerning the uncertainty delay interval, several multi-path components can be very small. The maximum likelihood (ML) estimation is usually applied to solve normal equations [21], to achieve the sparse true solutions $\mathbf{x}_0$ in sparse multi-path channels. Multi-user detection can be considered to be another example when the number of active users K is less than the expected number of users U [22]. Whether the matrix $\mathbf{R} = \mathbf{A}^H\mathbf{A}$ can be pre-computed determines the different computational complexity of sparse solution estimators. Calculation of the matrix $\mathbf{R}$ is complex for the algorithm. The pre-computed matrix $\mathbf{R}$ leads to the low complexity of the algorithm. In this work, we assume that matrix $\mathbf{R}$ is known. The vector of the signal is calculated as:

$$\mathbf{x} = \mathbf{R}^{-1}\boldsymbol{\beta}, \tag{2}$$

where $\boldsymbol{\beta} = \mathbf{A}^T\mathbf{Z}$. It costs $\mathcal{O}(U^3)$ complexity for computing the matrix inversion directly. The matrix inversion computing becomes prohibitively high when the system size U increases. To avoid such a high computational complexity, we can solve the normal Equation (1) by minimizing the quadratic function:

$$J(\mathbf{x}) = \|\boldsymbol{\beta} - \mathbf{R}\mathbf{x}\|_2^2 \tag{3}$$

To solve (3), an iterative descent search method can be applied as a possible choice. The descent search method updates the solution vector $\hat{\mathbf{x}}^{(i)}$ via $\hat{\mathbf{x}}^{(i)} = \hat{\mathbf{x}}^{(i-1)} + \sigma^{(i)}\boldsymbol{\Lambda}^{(i)}$ at each iteration i, where $\boldsymbol{\Lambda}^{(i)}$ is the update direction of the solution. The direction is designed to be non-orthogonal to the residual vector $\boldsymbol{\Gamma}$ ($\boldsymbol{\Gamma}^{(i)} = \boldsymbol{\beta} - \mathbf{R}\hat{\mathbf{x}}^{(i)}$). When current $\hat{\mathbf{x}}^{(i)}$ is not the exact point, i.e., $J(\hat{\mathbf{x}}^{(i+1)}) < J(\hat{\mathbf{x}}^{(i)})$, it shows that the descent search method converges to the original problem solution with descent objective values in the convex area.

2.2. DCD Algorithms

The DCD algorithm is one of many iterative techniques for solving the linear LS problem [19]. There are M_b bits to represent the elements of the solution vector. The amplitude of the element is limited to $[-\zeta, \zeta]$. The iterative search begins with the most significant bits of the solution $\mathbf{x}$ and ends with the least significant bits updated. The execution is controlled by the step size σ ($\sigma > 0$). σ starts at $\sigma = \zeta$ and decreases as $\sigma \rightarrow \sigma/2$ for less significant bits. DCD-based techniques usually use bit-shift to replace the multiplication/division operations, which is attractive for hardware implementation. The variants of the DCD algorithm: cyclic DCD algorithm and leading DCD algorithm, have different properties and been applied to different applications. These algorithms are described below.

2.2.1. Cyclic DCD Algorithm

The cyclic DCD algorithm [20], which is described in Table 1, updates a solution vector $\mathbf{x}$ in cyclic order ($n = 1, 2, \ldots, U$). In each iteration, when an element of the solution vector is updated, we call it a "successful" update. When σ updates, the algorithm executes the successful updates until the elements in the residual vector $\boldsymbol{\Gamma}_{\mathbf{cyclic}}$ are too small to meet the condition in step 3. Then, σ is updated in step 1. The number of successful iterations p and the number of bits M_b are the essential parameters of the algorithm. They determine the main computational complexity of the cyclic DCD algorithm. The maximum number of successful iterations, $N_{u_{cyclic}}$, is predetermined. It can be used as a stopping condition for the algorithm. For a given $N_{u_{cyclic}}$ and M_b, the computational complexity of the cyclic DCD algorithm is restricted to an upper limit $U(2N_{u_{cyclic}} + M_b - 1) + N_{u_{cyclic}}$ additions. However, the cyclic order update is not an efficient choice for solving a system of equations that needs a few updates.

Table 1. Cyclic DCD algorithm.

Step	Input $\boldsymbol{\beta}$, $\mathbf{R}$, M_b, ζ, $N_{u_{cyclic}}$; Output: $\mathbf{x}$, $\mathbf{\Gamma_{cyclic}}$	Addition
	Initialization: $\mathbf{x} = 0$, $\mathbf{\Gamma_{cyclic}} = \boldsymbol{\beta}$, $\sigma = \zeta$, $p = 0$	
	for $m = 1, \ldots, M_b$	
1	$\sigma = \sigma/2$	
2	g = 0	
	for $n = 1, \ldots, U$	
3	if $\lvert\Gamma_{cyclic_n}\rvert > (\sigma/2)R_{n,n}$	
4	$x_n = x_n + \text{sign}(\Gamma_{cyclic_n})\sigma$	1
5	$\mathbf{\Gamma_{cyclic}} = \mathbf{\Gamma_{cyclic}} - \text{sign}(\Gamma_{cyclic_n})\sigma\mathbf{R}_{:,n}$	U
6	$p = p + 1$, g = 1	
7	if $p = N_{u_{cyclic}}$, execution stops	
8	if g = 1, repeat step 2	
	end for	
complexity	$\leq U(2N_{u_{cyclic}} + M_b - 1) + N_{u_{cyclic}}$ adds	

2.2.2. Leading DCD Algorithm

A good method of index selection may speed up the convergence. The leading DCD algorithm [20] is briefly described in Table 2. The index n in the leading DCD algorithm is chosen via $n = \arg\max_{j=1,2,\cdots U}\{\lvert\Gamma_{leading_j}\rvert\}$. The ($n$-th) element in $\mathbf{x}$ corresponds to the (n-th) element in the residual vector $\mathbf{\Gamma_{leading}}$ that has the largest absolute value. Given several iterations $N_{u_{leading}}$, the computational complexity of the leading DCD algorithm is restricted to an upper limit $(2U+1)N_{u_{leading}} + M_b$ additions.

Table 2. Leading DCD algorithm.

Step	Input $\boldsymbol{\beta}$, $\mathbf{R}$, M_b, ζ, $N_{u_{leading}}$; output: $\mathbf{x}$, $\mathbf{\Gamma_{leading}}$	Addition
	Initialization: $\mathbf{x} = 0$, $\mathbf{\Gamma_{leading}} = \boldsymbol{\beta}$, $\sigma = \zeta$, $m = 1$	
	for $p = 1, \ldots, N_{u_{leading}}$	
1	$n = \arg\max_{j=1,\ldots,U}\{\lvert\Gamma_{leading_j}\rvert\}$, goto step 4	$U - 1$
2	$m = m + 1$, $\sigma = \sigma/2$	
3	if $m > M_b$, execution stops	
4	if $\lvert\Gamma_{leading_n}\rvert \leq (\sigma/2)R_{n,n}$, goto step 2	1
5	$x_n = x_n + \text{sign}(\Gamma_{leading_n})\sigma$	1
6	$\mathbf{\Gamma_{leading}} = \mathbf{\Gamma_{leading}} - \text{sign}(\Gamma_{leading_n})\sigma\mathbf{R}_{:,n}$	U
	end for	
complexity	$\leq (2U+1)N_{u_{leading}} + M_b$ adds	

2.2.3. Complexity Discussion

Table 1 shows that when $N_{u_{cyclic}}$ is much larger than M_b, then the computational complexity of the cyclic DCD is dominated by $2UN_{u_{cyclic}}$ additions. When the $N_{u_{cyclic}}$ value is small, then the computational complexity is upper bounded by the term UM_b.

2.2.4. Numerical Results

In this section, numerical results show that cyclic and leading DCD algorithms deal with system matrices $\mathbf{R}$ with different condition numbers. We consider real-valued systems here. The condition number of the matrix $\mathbf{R}$ is attained by modifying the ratio between Q and U. Usually when U is closer to Q, the matrix condition number is higher. For example, performing a highly loaded multi-user detection is equivalent to solving a linear equation $\boldsymbol{\beta} = \mathbf{R}\mathbf{x}$, where $\boldsymbol{\beta}$ is the output vector of the matched filter. $\mathbf{x}$ is a randomly generated $U \times 1$ transmitted data vector whose elements are uniformly distributed on $[-1, +1]$.

Using the DCD algorithm to solve (1), an estimated solution $\hat{\mathbf{x}}$ is obtained. We determine the misalignment between $\hat{\mathbf{x}}$ and $\mathbf{x}$ as

$$\epsilon = \frac{[\hat{\mathbf{x}} - \mathbf{x}]^T [\hat{\mathbf{x}} - \mathbf{x}]}{\mathbf{x}^T \mathbf{x}}. \tag{4}$$

The misalignments ϵ are averaged by 100 simulation trials and plotted (in decibels) against the number of updates.

Figure 1 depicts the misalignments of the cyclic and leading DCD algorithms. A system matrix $\mathbf{R}$ is designed with small condition numbers in the interval $[2, 5]$. It is seen that the cyclic DCD algorithm has a slightly slower convergence than the leading DCD algorithm.

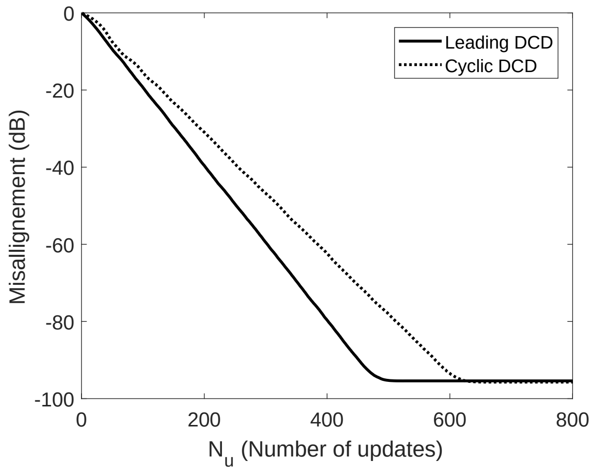

Figure 1. Misalignments of the DCD algorithms in the system with small condition numbers in the range of [2, 5]; $U = 64$, $M_b = 15$, $Q = 512$.

Figure 2 depicts the misalignments of the cyclic and leading DCD algorithms in the system with high condition numbers in the range of $[300, 400]$. It shows that the cyclic DCD algorithm has a faster convergence than the leading DCD algorithm. Also, the convergence speed of the two DCD algorithms is slower than that in Figure 1.

When we consider a scenario with a sparse true solution, the number of updates N_u is expected to be reduced. Figures 3 and 4 display misalignments of the DCD algorithms in the case of a system matrix with high condition numbers. It is seen that the solution is sparser, the convergence of the DCD algorithm is faster.

From the results in Figures 3 and 4, we can see that in the case of $K = 16$, compared to the non-sparse system, the number of iterations required by the leading DCD algorithm to attain a misalignment of -50 dB is approximately reduced by a factor of 20. However, for the cyclic DCD algorithm, the number of iterations reduction is not significant. The number of updates required in the leading DCD algorithm is approximately reduced by a factor of 30 in the case of $K = 8$. For the cyclic DCD algorithm, the number of iterations is approximately reduced by a factor of 1.5. Therefore, the leading DCD algorithm is preferable for highly sparse scenarios because it could achieve a significant reduction in the number of updates. When $K = 32$, we can see that the cyclic DCD algorithm has a lower steady-state misalignment; however, the leading DCD algorithm has a faster convergence speed in the initial iterations. When $K = 64$ (non-sparse system), the cyclic DCD algorithm has faster convergence and lower steady-state misalignment than the leading DCD algorithm.

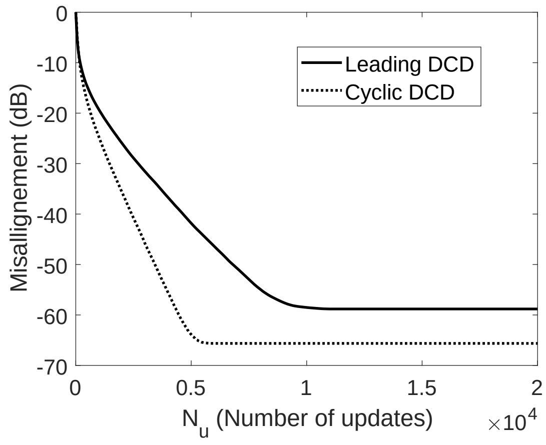

Figure 2. Misalignments of the DCD algorithms in the system with high condition numbers in the range of [300, 400]; $U = 64$, $M_b = 15$, $Q = 75$.

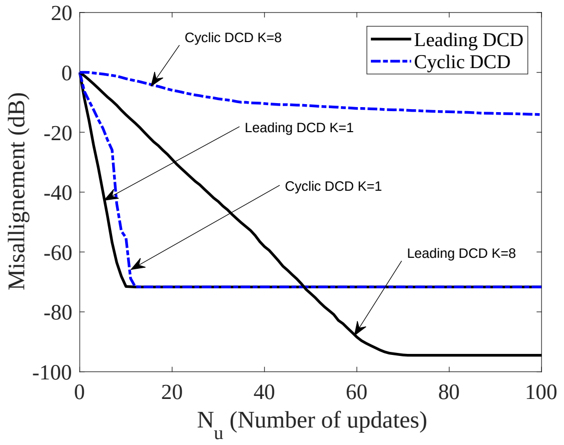

Figure 3. Misalignments of the DCD algorithms in the system with high condition numbers in the range of $[300, 400]$ and sparse solutions; $U = 64$, $M_b = 15$, $Q = 75$.

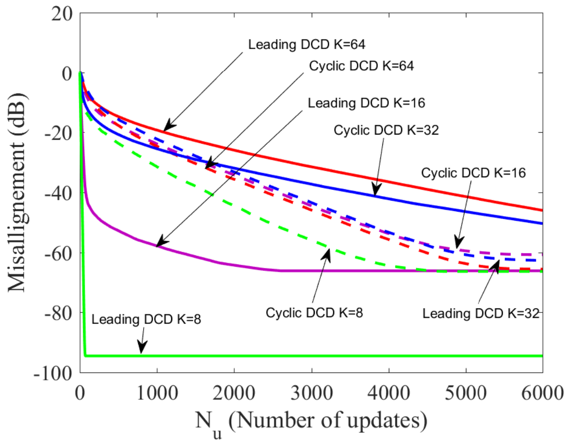

Figure 4. Misalignments of the DCD algorithms in the system with the high condition numbers in the range of $[300, 400]$ and sparse solutions; $U = 64$, $M_b = 15$, $Q = 75$.

3. Proposed Leading-Cyclic DCD Algorithm

From the results above, we can see that the cyclic DCD algorithm is more suitable for solving a non-sparse system with a large condition number. However, it is noted that when the percentage of non-zero elements of the solution is approximately 50%, the leading DCD algorithm provides a faster convergence in the initial updates. To further improve the convergence when the solution is not highly sparse, we consider combining the leading and cyclic DCD algorithms and propose the leading-cyclic DCD algorithm shown in Table 3.

Table 3. Leading-cyclic DCD algorithm.

Step	Input β, $\mathbf{R}$, M_b, ζ, N_u, $N_{u_{leading}}$; Output: $\mathbf{x}$, $\mathbf{\Gamma}$	Addition
	Initialization: $\mathbf{x} = 0$, $\mathbf{\Gamma} = \boldsymbol{\beta}$, $\sigma = \zeta$, $m_{leading} = 1$	
	for $p = 1, \ldots, N_{u_{leading}}$	
1	$\quad n = \arg\max_{j=1,\ldots,U}\{\lvert\Gamma_j\rvert\} \Rightarrow$ goto step 4	$U-1$
2	$\quad m_{leading} = m_{leading} + 1, \sigma = \sigma/2$	
3	$\quad$ if $m_{leading} > M_b$, execution stops	
4	$\quad$ if $\lvert\Gamma_n\rvert \le (\sigma/2)R_{n,n}$, goto step 2	1
5	$\quad x_n = x_n + \text{sign}(\Gamma_n)\sigma$	1
6	$\quad \mathbf{\Gamma} = \mathbf{\Gamma} - \text{sign}(\Gamma_n)\sigma\mathbf{R}_{:,n}$	
7	end for	
8	for $m = m_{leading} - 1, \ldots, M_b$	
9	$\quad \sigma = \sigma/2$	
10	$\quad$ g = 0	
11	$\quad$ for $n = 1, \ldots, U$	
12	$\qquad$ if $\lvert\Gamma_n\rvert > (\sigma/2)R_{n,n}$	1
13	$\qquad\quad x_n = x_n + \text{sign}(\Gamma_n)\sigma$	1
14	$\qquad\quad \mathbf{\Gamma} = \mathbf{\Gamma} - \text{sign}(\Gamma_n)\sigma\mathbf{R}_{:,n}$	U
15	$\qquad\quad p = p + 1$, g = 1	
16	$\qquad\quad$ if $p = N_u$, execution stops	
17	$\quad$ if g = 1, repeat step 10	
	end for	
complexity	$\le (2U+1)N_u + (1-U)m_{leading}$ $+(M_b - 1)U$ adds	

The proposed algorithm involves two steps. In the first step, a data vector is obtained using the leading DCD algorithm with a considerably smaller number of iterations $N_{u_{leading}}$ than in the standard version of the leading DCD algorithm. In the second step, an improved soft output is generated by applying the cyclic DCD algorithm with a large number of iterations N_u. The updated vectors of $\mathbf{x}$ and $\mathbf{\Gamma}$ and the step size of σ obtained from the leading DCD algorithm are the initial values for the cyclic DCD algorithm. There are two factors that determine the worst-case computational complexity of the leading-cyclic DCD algorithm. The cyclic DCD algorithm starts at $m_{leading} - 1$ bit for updating an element of the solution. The upper computational complexity of the cyclic DCD algorithm is calculated as follows. For the m-th bit, $m = m_{leading} - 1, 2, 3, \cdots, M_b - 1$ has $(N_u - N_{u_{leading}})$ successful iterations. That is, executing the initial $(M_b - m_{leading})$ bits does not include a successful iteration and thus requires $(M_b - m_{leading})U$ additions. Among the U iterations $(n = 1, 2, \cdots, U)$, if there is only one successful iteration, then calculating the last bit $(m = M_b)$ requires U additions for the comparison and

$(U+1)$ additions for updating the residual vector $\mathbf{\Gamma}$ and elements x_n. In total, $(N_u - N_{u_{leading}})$ successful iterations require $N_u - N_{u_{leading}}(2U+1)$ additions. The upper bound of computational complexity of the cyclic DCD algorithm is $\{U[2(N_u - N_{u_{leading}}) + (M_b - m_{leading}) - 1] + (N_u - N_{u_{leading}})\}$ additions. The worst-case scenario for the complexity of the leading DCD algorithm occurs when the algorithm uses all $N_{u_{leading}}$ updates. By using $N_{u_{leading}}$ iterations, the computational complexity of the leading DCD algorithm is $\{(2U+1)N_{u_{leading}} + m_{leading}\}$ additions. Therefore, the computational complexity of the leading-cyclic DCD algorithm has an upper limit of $\{(2U+1)N_u + (1-U)m_{leading} + (M_b - 1)U\}$ real-valued additions. In a practical situation, in each pass there should be several successful updates, and the average complexity is close to $2UN_u$. $N_{u_{cyclic}}$ is approximately equal to N_u; therefore, the average complexity of the leading-cyclic DCD algorithm is close to that of the cyclic DCD algorithm.

The selection of $N_{u_{leading}}$ varies based on the system matrix condition and the solution sparsity. The leading-cyclic algorithm is evaluated by using different $N_{u_{leading}}$ values in the range of $[U(1-\gamma), Q+U]$. Among these values of $N_{u_{leading}}$, the one that allows the proposed algorithm to achieve the fastest convergence to the steady-state level is chosen as the optimal $N_{u_{leading}}$. Figure 5 shows the optimal $N_{u_{leading}}$ in the leading-cyclic DCD algorithm for matrices with different sparsities γ.

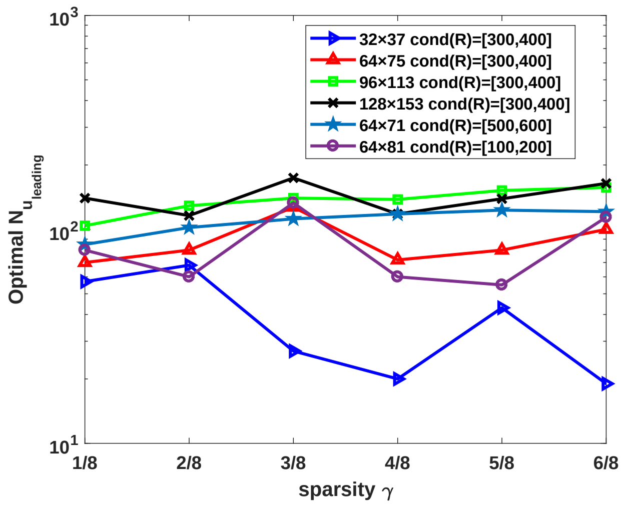

Figure 5. Optimal $N_{u_{leading}}$ vs. sparsity.

The misalignments compared to the number of successful iterations for large system matrices with condition numbers ($\geq$100) and sparse solutions are shown in Figures 6–11. By comparing the results from Figures 6 and 7, we can see that the lower the condition number of the system matrix is, the faster the convergence speed and the lower the steady-state level of these DCD algorithms. From these figures, we can see that when $\gamma \leq 1/8$, the leading DCD algorithm has a significantly faster convergence speed and a lower steady-state misalignment than the cyclic DCD algorithm. The leading-cyclic DCD algorithm has approximately the same steady-state level as the leading DCD algorithm. As γ increases, the cyclic DCD algorithm gradually outperforms the leading DCD algorithm. The leading-cyclic DCD algorithm with a given $N_{u_{leading}}$ (shown in Figure 5) has an increased convergence speed and a lower steady-state misalignment compared to the leading DCD algorithm. When $\gamma = 2/8$ increases to $\gamma = 6/8$, the leading-cyclic DCD algorithm exhibits an insignificantly increased convergence speed compared to the cyclic DCD algorithm.

Figures 8–11 show the misalignments of the DCD algorithms with conditional numbers of the system matrix in the interval [300, 400]. We can see that when $\xi = Q/U$ increases, the convergence speed of the DCD algorithms decreases.

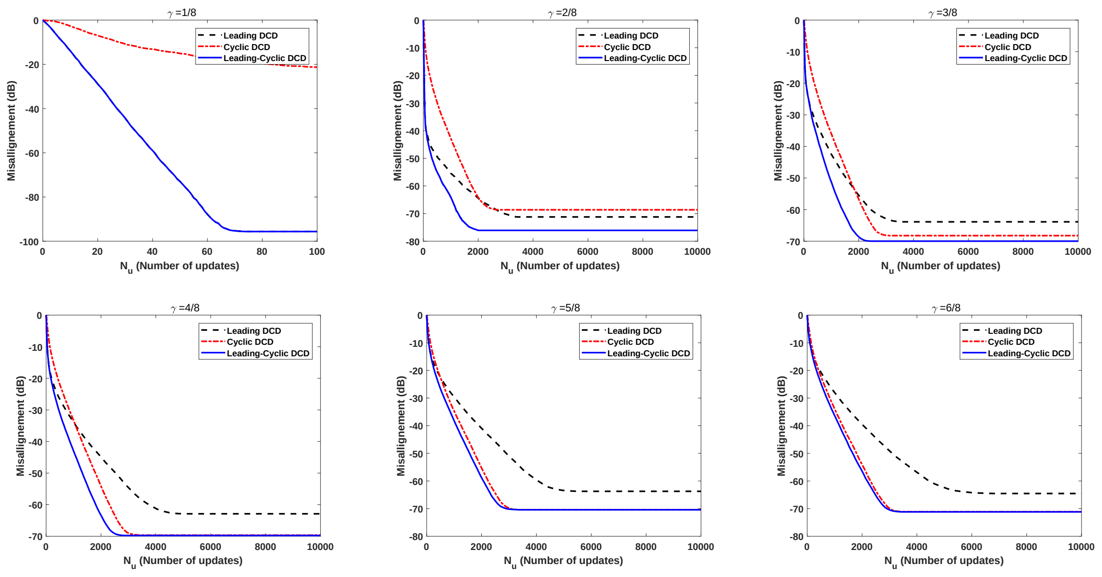

Figure 6. Misalignments of the DCD algorithms for different sparse solutions with the condition numbers in the range of [100, 200] and sparse solutions; $U = 64$, $M_b = 15$, $Q = 81$, $\xi = 1.2656$.

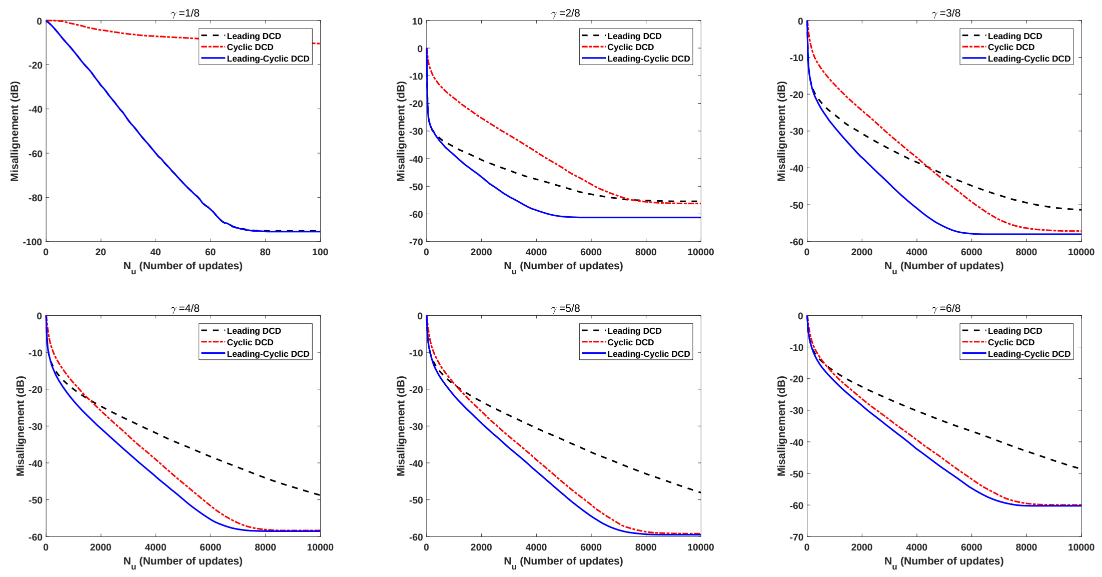

Figure 7. Misalignments of the DCD algorithms for different sparse solutions with the condition numbers in the range of [500, 600]; $U = 64$, $M_b = 15$, $Q = 71$, $\xi = 1.109$.

According to the numerical results above, we can see that the leading-cyclic DCD algorithm is an effective approach for solving a sparse system with a large number of updates. For example, in code division multiple access (CDMA)with a spreading factor Q, there are approximately half of the users are active in the U user system. The columns of $\mathbf{A}$ are represented as spreading sequences. The matrix $\mathbf{R}$ is represented as a correlation matrix of the spreading sequence.

Figure 12 shows the misalignments of the RLS algorithm and DCD$_{leading-cyclic}$-RLS (forgetting factor $\lambda = 0.96$, the length of the filter $U = 16$, and $\gamma = 1/2$). When system noise energy increases by 10 times between 1000 iterations and 1100 iterations, the DCD$_{leading-cyclic}$-RLS maintains a low error rate. The proposed algorithm-based RLS provides a lower error level and better tracking performance than the RLS algorithm.

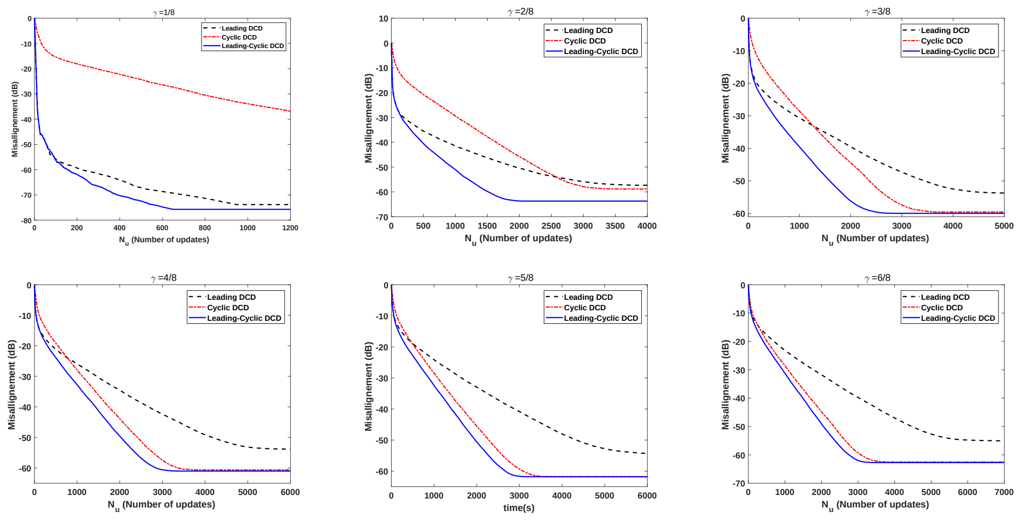

Figure 8. Misalignments of the DCD algorithms for different sparse solutions with the condition numbers in the range of $[300, 400]$; $U = 32$, $M_b = 15$, $Q = 37$, $\xi = 1.156$.

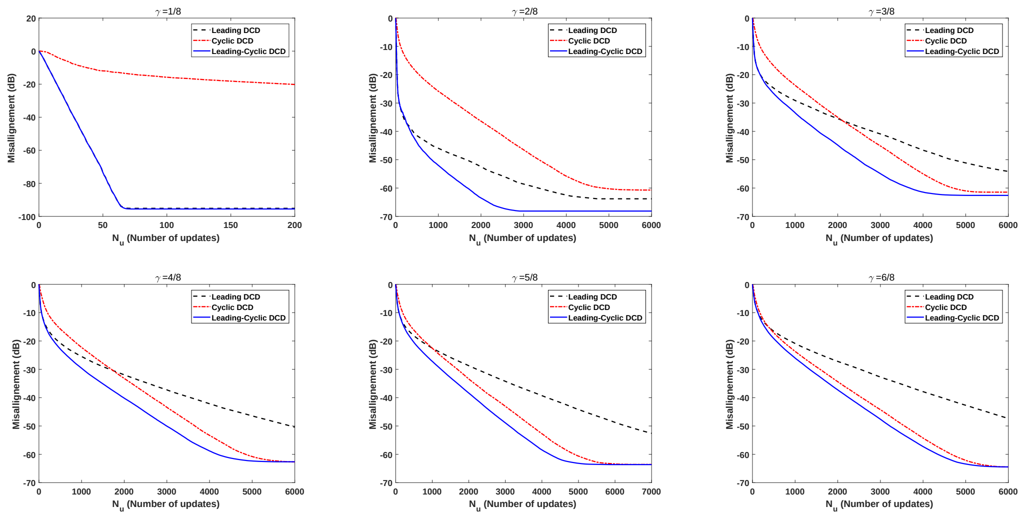

Figure 9. Misalignments of the DCD algorithms for different sparse solutions with the condition numbers in the range of $[300, 400]$; $U = 64$, $M_b = 15$, $Q = 75$, $\xi = 1.172$.

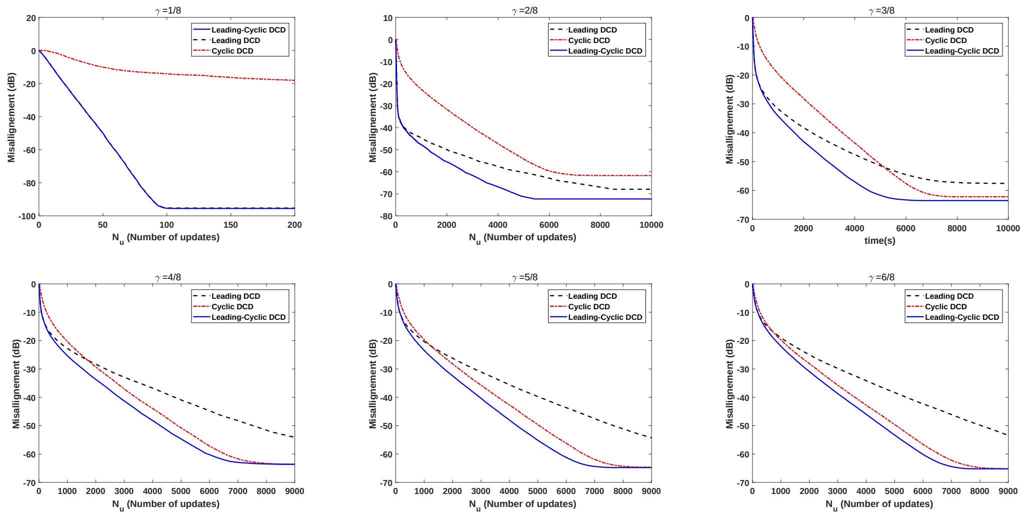

Figure 10. Misalignments of the DCD algorithms for different sparse solutions with the condition numbers in the range of $[300, 400]$; $U = 96$, $M_b = 15$, $Q = 113$, $\xi = 1.177$.

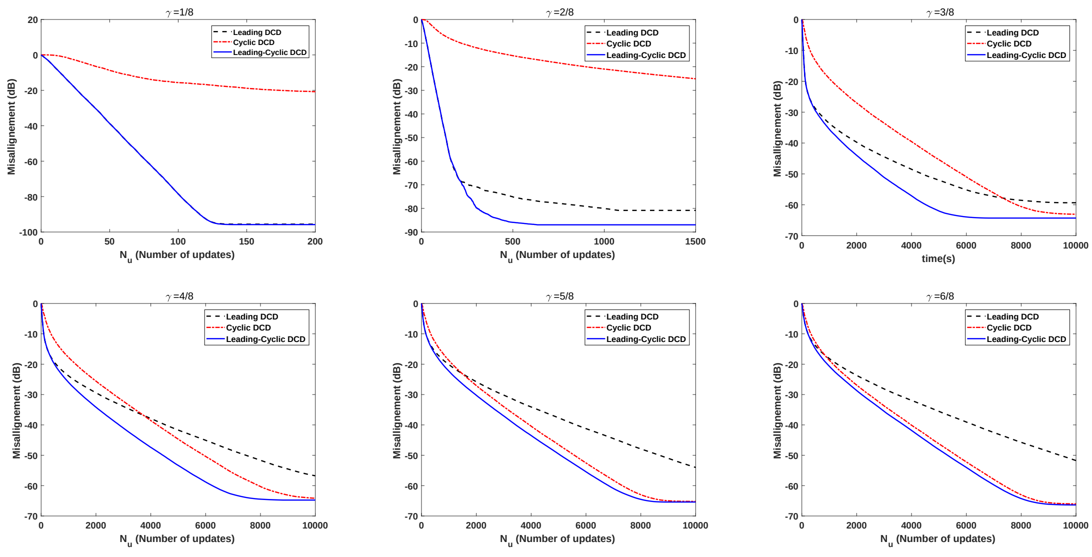

Figure 11. Misalignments of the DCD algorithms for different sparse solutions with the condition numbers in the range of $[300, 400]$; $U = 128$, $M_b = 15$, $Q = 153$, $\xi = 1.195$.

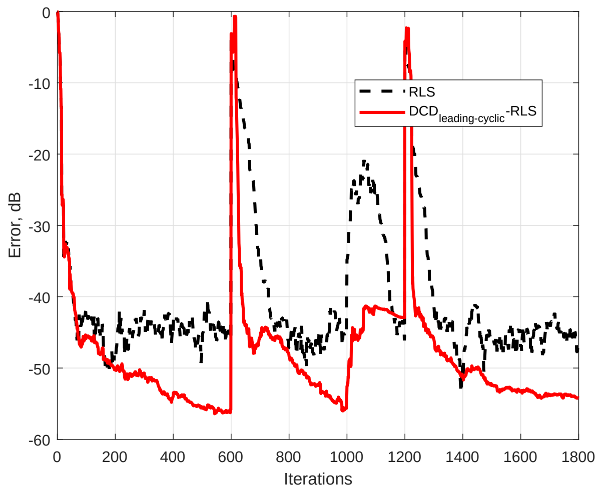

Figure 12. Misalignments of the DCD$_{leading-cyclic}$-RLS and the RLS algorithm when there is an abrupt change in the unknown system; $\lambda = 0.96$, $U = 16$, $\gamma = 1/2$.

4. Discussion

When γ increases, the convergence speed of the proposed algorithm (e.g., at the misalignment -50 dB) decreases. The optimal $N_{u_{leading}}$ required for solving a system of equations varies depending on the system size, the solution sparseness and the condition number of the system matrix. In this work, we choose the value that allows the proposed algorithm to achieve the fastest convergence to the steady-state level as the optimal $N_{u_{leading}}$. The results show that the optimal $N_{u_{leading}}$ varies around Q and approximately remains stable with different γ (except for in the case of 32×37). The ratio between Q and U in the case of 32×37 is lower than those in the other cases with the same condition number range, and the leading DCD algorithm might contribute fewer iterations in the proposed algorithm to achieving the steady state when γ increases. Investigation of the general theory of induction for choosing the proper $N_{u_{leading}}$ in the combined algorithm is left for future work.

5. Conclusions

In this paper, we have demonstrated that the leading DCD algorithm and the cyclic DCD algorithm are useful in different applications. In particular, we consider a scenario in which the sparsity of the true solution is $\gamma \in [1/8, 6/8]$. We propose a leading-cyclic DCD algorithm that solves normal equations with sparse solutions. The results show that the proposed leading-cyclic DCD algorithm has an optimal $N_{u_{leading}}$ value in the range of $[U(1-\gamma), Q+U]$ and exhibits improved convergence compared to the cyclic DCD algorithm and the leading DCD algorithm. In addition, within the same range of the system matrix condition number, the larger Q/U is, the slower the convergence of the DCD algorithms.

Author Contributions: Conceptualization, Z.Q. and S.L.; methodology, Z.Q. and S.L.; software, Z.Q. and S.L.; validation, Z.Q. and S.L.; formal analysis, Z.Q. and S.L.; investigation, Z.Q. and S.L.; resources, Z.Q. and S.L.; data curation, Z.Q. and S.L.; writing—original draft preparation, Z.Q. and S.L.; writing–review and editing, Z.Q. and S.L.; visualization, Z.Q. and S.L.; supervision, Z.Q.; project administration, Z.Q.; funding acquisition, Z.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This work was founded by the National Natural Science Foundation of China by Grant No. U1604160 and the China Scholarship Council.

Acknowledgments: The authors would like to thank Jie Liu for his comments and the two anonymous reviewers for their valuable suggestions.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Haykin, S. *Adaptive Filter Theory*, 4th ed.; Prentice Hall, Inc.: Upper Saddle River, NJ, USA, 2001.
2. Dakulagi, V.; Bakhar, M. Advances in Smart Antenna Systems for Wireless Communication. *Wirel. Pers. Commun.* **2000**, *110*, 931–957. [CrossRef]
3. Verdu, S. *Multiuser Detection*; Cambridge University Press: Cambridge, UK, 1998.
4. Albreem, M.A.; Juntti, M.; Shahabuddin, S. Massive MIMO detection techniques: A survey. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3109–3132. [CrossRef]
5. Messini, M.; Djendi, M. A new adaptive filtering algorithm for stereophonic acoustic echo cancellation. *Appl. Acoust.* **2019**, *146*, 345–354. [CrossRef]
6. Proakis, J.; Manolakis, D. *Digital Signal Processing: Principles, Algorithms, and Applications*, 2nd ed.; Macmillan: New York, NY, USA, 1992.
7. Wen, H.; Yang, S.; Hong, Y.; Luo, H.A. Partial Update Adaptive Algorithm for Sparse System Identification. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 240–255. [CrossRef]
8. Yazdanpanah, H.; Diniz, P.S.R.; Lima, M.V.S. Feature Adaptive Filtering: Exploiting Hidden Sparsity. *IEEE Trans. Circuits Syst. Regul. Pap.* Available online: http://www02.smt.ufrj.br/~diniz/papers/ri99.pdf (accessed on 3 June 2020). [CrossRef]
9. Liu, J.; Grant, S.L. Proportionate Adaptive Filtering for Block-Sparse System Identification. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2016**, *24*, 623–630. [CrossRef]
10. Golub, G.H.; Van Loan, C.F. *Matrix Computations*, 3rd ed.; The Johns Hopkins University Press: Baltimore, MD, USA, 1996.
11. Liu, J. DCD Algorithm: Architectures, FPGA Implementations and Applications. Ph.D. Thesis, University of York, York, UK, 2008.
12. Cryer, C.W. The solution of a quadratic programming problem using systematic overrelaxation. *SIAM J. Control* **1971**, *9*, 385–392. [CrossRef]
13. Watkins, D.J. *Fundamentals of Matrix Computations*; Wiley: Hoboken, NJ, USA, 2002.
14. Quan, Z.; Tian, T. DCD-based branch and bound detector with reduced complexity for MIMO systems. *IEICE Trans. Commun.* **2018**, *E101-B*, 2230–2238. [CrossRef]
15. Albu, F. Efficient multichannel filtered-x affine projection algorithm for active noise control. *Electron. Lett.* **2006**, *42*, 421–423. [CrossRef]
16. Albu, F.; Bouchard, M.; Zakharov, Y. Pseudo Affine Projection Algorithms for Multichannel Active Noise Control. *IEEE Trans. Audio Speech Lang. Process.* **2007**, *15*, 1044–1052. [CrossRef]

17. Zakharov, Y. Low complexity implementation of the affine projection algorithm. *IEEE Signal Process. Lett.* **2008**, *15*, 557–560. [CrossRef]
18. Zakharov, Y.; White, G.; Liu, J. Low complexity RLS algorithms using dichotomous coordinate descent iterations. *IEEE Trans. Signal Process.* **2008**, *56*, 3150–3161. [CrossRef]
19. Zakharov, Y.V.; Tozer, T.C. Multiplication-free iterative algorithm for LS problem. *Electron. Lett.* **2004**, *40*, 567–569. [CrossRef]
20. Liu, J.; Zakharov, Y.V.; Weaver, B. Architecture and FPGA design of dichotomous coordinate descent algorithms. *IEEE Trans. Circuits Syst. Regul. Pap.* **2009**, *56*, 2425–2438.
21. Cotter, S.F.; Rao, B.D. Sparse channel estimation via matching pursuit with application to equalization. *IEEE Trans. Commun.* **2002**, *50*, 374–377. [CrossRef]
22. Wu, W.C.; Chen, K.C. Identification of active users in synchronous CDMA multiuser detection. *IEEE J. Sel. Areas Commun.* **1998**, *16*, 1723–1735.