Viterbi and RLS decoding for deterministic blind symbol estimation in DS-CDMA wireless communication

Geert Leus*, Marc Moonen

Katholieke Universiteit Leuven - ESAT, Kardinaal Mercierlaan 94, 3001 Heverlee, Belgium

Abstract

Adaptive deterministic blind symbol estimation algorithms for direct-sequence code-division multiple-access (DS-CDMA) wireless communication are presented. We consider a subspace and an equalizer-based approach as well as the relation between both approaches. Both Viterbi and recursive least-squares (RLS) decoding schemes are studied. Simulation results show that these adaptive processing algorithms perform better than the block processing algorithm of Liu and Xu (Proceedings of the ICASSP’96, Atlanta, GA, May 1996), especially for time-varying channels. © 2000 Elsevier Science B.V. All rights reserved.

Zusammenfassung


Résumé

Nous présentons des algorithmes d’estimation aveugle de symboles déterministes adaptatifs pour les communications sans fil de type accès multiple, division de code, séquence directe (DS-CDMA). Nous considérons une approche de type sous-espace et une de type égaliseur ainsi que la relation entre celles-ci. Des méthodes de décodage de Viterbi et moindres carrés récursifs sont étudiées. Les résultats de simulation montrent que ces algorithmes de traitement adaptatif ont de meilleures performances que l’algorithme de traitement par bloc de Liu et Xu (Proceedings of the ICASSP’96, Atlanta, GA, May 1996), spécialement pour les canaux variant dans le temps. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: DS-CDMA communication; Deterministic blind equalization; Adaptive processing; Wireless communication

*Corresponding author. Tel.: 32-16-32-1924; fax: 32-16-32-1970.
E-mail addresses: geert.leus@esat.kuleuven.ac.be (G. Leus), marc.moonen@esat.kuleuven.ac.be (M. Moonen).
Nomenclature

$(\cdot)^\dagger$ conjugate
$(\cdot)^\mathsf{T}$ transpose
$(\cdot)^\mathsf{H}$ Hermitian transpose
$(\cdot)^\dagger$ pseudoinverse
$\| \cdot \|$ Frobenius norm
$\text{fc}\{ \cdot \}$ flipping of columns
$\text{fr}\{ \cdot \}$ flipping of rows
$X(r_1 : r_2, c_1 : c_2)$ submatrix of $X$ containing the rows $r_1$ to $r_2$ and columns $c_1$ to $c_2$; if $r_1$ and $r_2$ ($c_1$ and $c_2$) are omitted, all rows (columns) are considered; if $r_1 : r_2 (c_1 : c_2)$ is replaced by $r (c)$, only the $r$th row ($c$th column) is considered
$x(i_1 : i_2)$ subvector of $x$ containing the elements $i_1$ to $i_2$; if $i_1 : i_2$ is replaced by $i$, only the $i$th element is considered
$0$ zero (row or column) vector
$1_i$ unit (row or column) vector with the $i$th element equal to 1
$\mathbf{0}$ zero matrix
$I$ identity matrix
$\sharp$ don't care object (number, vector or matrix)
$O^{\dagger}\ast$ a square upper triangular matrix with don’t care elements
$\#Y$ number of elements in the set $Y$
$\text{proj}_{Y}\{ \cdot \}$ projection on the set $Y$
$|x|$ modulus of $x$
$\lfloor x \rfloor$ largest integer smaller or equal to $x$ ($x$ real)
$\lceil x \rceil$ smallest integer larger or equal to $x$ ($x$ real).

1. Introduction

DS-CDMA is an attractive allocation technique that allows users to be simultaneously active over the total available bandwidth. In a DS-CDMA system each user's data symbol sequence is spread using a distinct code sequence which allows the receiver to distinguish between the data from different users. This approach offers simple transmitter structures at the expense of complex receiver structures, for the receiver has to suppress both multi-user interference (MUI) and inter-chip interference (ICI). Especially when we consider uplink communication, i.e., communication from different users to a base station, or peer-to-peer communication, i.e., direct communication from one user to another user, the receiver design gets very complicated. Every transmitted data symbol sequence then is distorted by a different channel and arrives at the receiver at a different time instant (asynchronous communication). Conventional receivers based on matched filter techniques treat both ICI and MUI as noise [23,16]. Hence, the allowable interference is severely limited.

Recent research on DS-CDMA systems focuses on blind receiver design, which means that no training signal has to be transmitted (hence, there is no transmission overhead). Blind receivers can be stochastic or deterministic. When block processing is performed on a short block of received samples (this is necessary for time-varying channels), deterministic schemes generally have a better performance than stochastic schemes, which is our motivation for studying deterministic schemes. The most popular deterministic blind receivers for DS-CDMA systems make use of deterministic blind channel estimation [2,11,17,18,24]. All these
methods are subspace-based. For some classical works on subspace-based methods for deterministic blind channel estimation in time-division multiple-access (TDMA) systems we refer the interested reader to [15,1]. Other deterministic blind receivers for DS-CDMA systems make use of deterministic blind equalizer estimation, i.e., direct estimation of a linear equalizer [10] (related approach in [12]). We consider deterministic blind receivers for DS-CDMA systems based on deterministic blind symbol estimation, i.e., direct estimation of the transmitted data symbols. To the best of our knowledge, such receivers have up till now not been studied in literature. Moreover, we focus on adaptive processing instead of block processing (we will study block processing only as a basis for adaptive processing). This makes it possible to handle time-varying channels. All previously mentioned schemes [2,11,17,18,24,10] only consider block processing.

Note that for TDMA systems with co-channel users there do exist a number of deterministic blind receivers based on deterministic blind symbol estimation, all of which are subspace-based. For block processing we refer to [9,19]. Adaptive processing is discussed in [21], where all users are decoded using a common Viterbi scheme. By introducing decision feedback ideas this common Viterbi scheme can be split into separate user-specific reduced Viterbi schemes. These can even be replaced by separate user-specific RLS schemes, decreasing the total computational complexity at the cost of a slightly decreased performance. However, such algorithms based on decision feedback ideas rely on a good initialization step, e.g., a computationally demanding initialization based on block processing [9,19]. In [7], an extreme case of decision feedback is illustrated, leading to one-state reduced Viterbi schemes and one-dimensional RLS schemes, but the approach can be generalized to less extreme cases of decision feedback. A related class of algorithms makes use of specific coding schemes (different from the coding used in DS-CDMA systems). In [22], block coding is used to split the common Viterbi scheme of [21] into separate user-specific Viterbi schemes, while in [8], convolutional coding is used to improve the one-state reduced Viterbi schemes of [7]. Like the algorithms of [7], the algorithm of [8] can also be generalized to less extreme cases of decision feedback. Note that all the algorithms mentioned in this paragraph are subspace-based.

In this paper, we consider two approaches, a subspace-based approach, which uses a subspace preprocessing step, and an equalizer-based approach, which uses a linear equalizer (no subspace preprocessing step). We will also show how both approaches are related. Adaptive processing leads to separate user-specific Viterbi and RLS schemes. Block processing is studied as a basis for adaptive processing. The developed deterministic blind symbol estimation techniques are based on a DS-CDMA system with multiple outputs, which can be obtained by spatial and/or temporal oversampling.

In Section 2, the data model is introduced. Then deterministic blind symbol estimation techniques are discussed. In Section 3, we focus on block processing that serves as a basis for adaptive processing. Then block-based (Section 4) and exponentially weighted (Section 5) adaptive processing is discussed. The computational complexity of these methods is also discussed. In Section 6, we show some simulation results of the presented adaptive processing algorithms and compare them with the results of the block processing algorithm of [10]. We end with some conclusions in Section 7.

2. Data model

Consider a DS-CDMA system, where \( J \) users are transmitting to a base station (uplink communication: \( J \) signals of interest) or to another user (peer-to-peer communication: 1 signal of interest). Assume that the data symbol sequence \( s_j[k] \) for the \( j \)th user \( (j = 1, 2, \ldots, J) \), with symbols in some finite alphabet \( \Omega \), is spread by a factor of \( N \) using the length-\( \rho N \) \( (\rho \gg 1) \) code sequence \( c_j[n] \) \( (c_j[n] \neq 0, \text{ for } n = 0, 1, \ldots, \rho N - 1, \text{ and} \)

\[^1\text{Details on the Viterbi algorithm can be found in [5].}\]

\[^2\text{Details on the reduced Viterbi algorithm can be found in [4].}\]
This chip sequence $x_j[n]$ is then transmitted at a rate of $N/T$ (the chip rate), where $T$ is the data symbol period. Assume that our DS-CDMA system has $M$ outputs, that are sampled at chip rate. These $M$ outputs can be obtained by spatial and/or temporal oversampling (see Section 6 for more details). Further, let $h_j^{(m)}[n]$ denote the discrete-time channel impulse response from the $j$th user to the $m$th output. The received sequence at the $m$th output then is

$$y_j^{(m)}[n] = \sum_{j=1}^{J} \sum_{n'=-\infty}^{+\infty} h_j^{(m)}[n']x_j[n-n'] + e_j^{(m)}[n],$$

where $e_j^{(m)}[n]$ is additive noise at the $m$th output. Stacking the received samples from the $M$ outputs:

$$y[n] = \begin{bmatrix} y[j]^{(1)}[n] & y[j]^{(2)}[n] & \cdots & y[j]^{(M)}[n] \end{bmatrix}^T,$$

we obtain

$$y[n] = \sum_{j=1}^{J} \sum_{n'=-\infty}^{+\infty} h_j[n']x_j[n-n'] + e[n],$$

where $e[n]$ is similarly defined as $y[n]$ and $h_j[n]$ is the discrete-time $M \times 1$ vector channel impulse response for the $j$th user, given by

$$h_j[n] = \begin{bmatrix} h_j^{(1)}[n] & h_j^{(2)}[n] & \cdots & h_j^{(M)}[n] \end{bmatrix}^T.$$

We assume that $h_j[n]$ is an FIR vector filter of order $L_j$ with delay index $\delta_j$ ($h_j[n] \neq 0$, for $n = \delta_j$ and $n = \delta_j + L_j$, and $h_j[n] = 0$, for $n < \delta_j$ and $n > \delta_j + L_j$).

The matrix that plays a central role in the next sections is the following $(Q + 1)M \times wN$ output matrix:

$$Y_n = \begin{bmatrix} y[n] & y[n+1] & \cdots & y[n+wN-1] \\
\vdots & \vdots & \cdots & \vdots \\
y[n+Q] & y[n+Q+1] & \cdots & y[n+Q+wN-1] \end{bmatrix},$$

where $Q$ and $w$ are design parameters that will be discussed in the next sections ($Q + 1$ is sometimes called the 'smoothing' factor). This output matrix can be written as

$$Y_n = \sum_{j=1}^{J} H_j X_{j,n} + E_n,$$

where $E_n$ is similarly defined as $Y_n$, $H_j$ is the $(Q + 1)M \times (L_j + Q + 1)$ channel matrix for the $j$th user, given by

$$H_j = \begin{bmatrix} h_j[\delta_j + L_j] & \cdots & h_j[\delta_j] & 0 & \cdots & 0 \\
0 & h_j[\delta_j + L_j] & \cdots & h_j[\delta_j] & 0 & \cdots & 0 \\
0 & \cdots & 0 & h_j[\delta_j + L_j] & \cdots & h_j[\delta_j] \end{bmatrix},$$

and $X_{j,n}$ is the following $(L_j + Q + 1) \times wN$ input matrix for the $j$th user:

$$X_{j,n} = \begin{bmatrix} x_{j,n,\delta_j - L_j} & \cdots & x_{j,n,\delta_j + Q} \end{bmatrix}^T,$$
with
\[ x_{j,n} = [x_j[n] \ x_j[n+1] \ \ldots \ x_j[n+wN-1]]. \] (4)

Note that (2) can also be written as
\[ Y_n = H X_n + E_n, \] (5)
where, if we define \( L = \sum_{j=1}^J L_j \), \( H \) is the \((Q+1)M \times (L+(Q+1)J)\) channel matrix, given by
\[ H = [H_1 \ \ldots \ H_J], \]
and \( X_n \) is the following \((L+(Q+1)J) \times wN\) input matrix:
\[ X_n = [X_{1,n}^T \ \ldots \ X_{J,n}^T]^T. \] (6)

The algorithms we propose are based on a single user of interest. For uplink communication this means we have to run \( J \) parallel algorithms at the receiver, while for peer-to-peer communication this means we only have to run one algorithm at the receiver. In the rest of this work, we assume that the user of interest is the user \( j = 1 \) and that \( \delta_1 \) is known at the receiver. We further assume w.l.o.g. that \( \delta_1 = 0 \).

3. Block processing

In this section, we focus on block processing algorithms, which serve as a basis for the adaptive processing algorithms to be derived in the next sections. We consider a block of \( w \) data symbols starting at a fixed time index \( k \). Hence, we focus on
\[ s_{1,k} = [s_1[k] \ s_1[k+1] \ \ldots \ s_1[k+w-1]]. \] (7)
Using (1) and (4), it is clear that this vector is ‘contained’ in \( x_{1,k,N} \):
\[ x_{1,k,N} = [s_1[k]c_1[k] \ \ldots \ s_1[k+w-1]c_1[k+w-1]] = s_{1,k}C_{1,k}, \] (8)
where \( c_1[k] \) is the code vector used to spread the data symbol \( s_1[k] \) (spreading factor \( N \)):
\[ c_1[k] = [c_1[(k \mod \rho)N] \ c_1[(k \mod \rho)N+1] \ \ldots \ c_1[(k \mod \rho)N+N-1]], \]
and \( C_{1,k} \) is the \( w \times wN \) code matrix, defined as
\[ C_{1,k} = \begin{bmatrix} c_1[k] \\ \vdots \\ c_1[k+w-1] \end{bmatrix}. \] (9)

The vector \( x_{1,k,N} \) is a row of every input matrix from the set \( \{X_{kN+a}^{L_1} \}_{a=-Q} \) (see (3), (4) and (6)) and is therefore ‘contained’ in every output matrix from the set \( \{Y_{kN+a}^{L_1-1} \}_{a=-Q} \) (see (5)). The problem addressed here is to compute \( s_{1,k} \) from a subset of \( A \) output matrices from the above set, based on the knowledge of the long code sequence \( c_1[k] \) of the user of interest \( j = 1 \). The set of \( A \) output matrices we use is denoted as \( \{Y_{kN+a}^{L_1-1} \}_{a=-A_1} \), with \(-Q \leq A_1 \leq A_2 \leq L_1 (A = A_2 - A_1 + 1)\). To solve this problem we introduce the following assumptions:

(A1) The channel matrix \( H \) has full column rank \( L + (Q+1)J \).
(A2) Every input matrix from the set \( \{X_{kN+a}^{L_1} \}_{a=-A_1} \) has full row rank \( L + (Q+1)J \).

These assumptions require that \((Q+1)M \geq L + (Q+1)J\) and \( wN \geq L + (Q+1)J \). We also introduce a third assumption, which will be used further on. Note that this assumption depends on the space in which we search for \( s_{1,k} \), which we will call the data search space \( \Gamma \):
In this work unique always means unique up to a possible complex factor.

\[(A3)\] For any \(s_1, k\) in \(I\), linearly independent of \(s_{1, k}\), there exists an input matrix \(X_{kN + a}\), with \(A_1 \leq a \leq A_2\), for which

\[
\begin{bmatrix}
    x'_{1, kN} \\
    X_{kN + a}
\end{bmatrix},
\]

with \(x'_{1, kN} = s'_{1, k}C_{1, k}\), has full row rank \(L + (Q + 1)J + 1\).

This assumption requires that \(wN > L + (Q + 1)J\), irrespective of \(I\). Note that the data search space \(I\) is determined by the constraint we impose on the data symbols, e.g., when we impose the finite alphabet constraint on the data symbols we have \(I = \Omega^w\) and when we do not impose any constraints on the data symbols we have \(I = \mathbb{C}^w\), where \(\mathbb{C}\) represents the set of all complex numbers.

We will consider a subspace-based approach and an equalizer-based approach, both leading to two different block processing problems. We further show how both approaches are related and end with some remarks. For a discussion on the choice of \(A_1, A_2\) and \(Q\) we refer to Section 6.

3.1. Subspace-based approach

The problem is solved here using a subspace preprocessing step. We assume that assumptions (A1) and (A2) are satisfied. The calculation of the singular-value decomposition (SVD) [6] of \(Y_{kN + a}\) leads to

\[
Y_{kN + a} = U_{kN + a} \Sigma_{kN + a} V_{kN + a}^H,
\]

where \(\Sigma_{kN + a}\) is a diagonal matrix (diagonal elements in descending order) of the same size as \(Y_{kN + a}\) and \(U_{kN + a}\) and \(V_{kN + a}\) are square unitary matrices. We define \(V_{kN + a}^n\) as the first \(L + (Q + 1)J\) columns of \(V_{kN + a}\) and \(V_{kN + a}^n\) as the last \(wN - L - (Q + 1)J\) columns of \(V_{kN + a}\). We initially ignore additive noise for the sake of clarity. Because of (A1) and (A2), the columns of \(V_{kN + a}^n\) form an orthonormal basis of the right null space of \(X_{kN + a}\):

\[
X_{kN + a} V_{kN + a}^n = 0.
\]

Since \(x_{1, kN}\) is a row of \(X_{kN + a}\), we get

\[
x_{1, kN} V_{kN + a}^n = 0. \tag{10}
\]

The known long code information that is in \(x_{1, kN}\) (see (8)) then allows us to write (10) as

\[
s_{1, k} \underbrace{C_{1, k} V_{kN + a}^n}_{w \times (wN - L - (Q + 1)J)} = 0.
\]

This can be derived for \(a = A_1, A_1 + 1, \ldots, A_2\). All these results can then be combined, leading to

\[
s_{1, k} C_{1, k} \begin{bmatrix} V_{kN + A_1}^n & \cdots & V_{kN + A_2}^n \end{bmatrix} = s_{1, k} \begin{bmatrix} C_{1, k} & \cdots & C_{1, k} \end{bmatrix} \begin{bmatrix} V_{kN + A_1}^n & O & O \\
O & \ddots & O \\
O & O & V_{kN + A_2}^n \end{bmatrix} \]

\[
\triangleq s_{1, k} \tilde{C}_{1, k} \tilde{V}_{k}^n = 0. \tag{11}
\]

The next theorem deals with the uniqueness\(^3\) of the solution of (11) in the data search space \(I\). For a proof see Appendix A.

\(^3\)In this work unique always means unique up to a possible complex factor.
Theorem 1. Under assumptions (A1) and (A2), we can state that (11) has a unique solution $s_{1,k}$ in $\Gamma$, if and only if (A3) is satisfied.

When we do not consider any constraints on the data symbols ($\Gamma = \mathbb{C}^w$), Theorem 1 can also be formulated as follows. The proof is trivial.

Theorem 2. Under assumptions (A1) and (A2), we can state that (11) has a unique solution $s_{1,k}$, if and only if $\hat{\mathbf{C}}_{1,k} \hat{\mathbf{P}}_k$ has rank $w - 1$.

So, under assumptions (A1) and (A2), this rank condition is equivalent to assumption (A3) for $\Gamma = \mathbb{C}^w$ and requires

$$A(wN - L - (Q + 1)J) \geq w - 1.$$  \hspace{1cm} (12)

Let us now assume that additive noise is present. In this case we consider the following minimization problem:

$$\min_{s_{1,k}} \{ ||s_{1,k} \hat{\mathbf{C}}_{1,k} \hat{\mathbf{P}}_k||^2 \}. \hspace{1cm} (13)$$

To avoid the all-zero solution some non-triviality constraint is imposed on the data symbols, which then determines $\Gamma$. We elaborate on this constraint in the section dealing with adaptive processing.

This block processing step for the subspace-based approach is related to the block processing steps in [21,7], which have some strong connections with the algorithms in [9,19]. However, our method uses long code sequences to split the multi-user problem that appears in the block processing steps of [21,7] into separate single-user problems, which are much easier to solve.

3.2. Equalizer-based approach

The problem is solved here using a linear equalizer (no subspace preprocessing step). Like before, we assume that assumptions (A1) and (A2) are satisfied. Like before, we initially ignore additive noise for the sake of clarity. Because of (A1) and (A2), there exists a linear equalizer $f_{1,kN+a}$, for which

$$f_{1,kN+a}Y_{kN+a} = x_{1,kN}, \hspace{1cm} (14)$$

and this linear equalizer $f_{1,kN+a}$ is a zero-forcing linear equalizer, i.e., $f_{1,kN+a} \mathcal{H} = 1_{L_2 + 1-a}$, with $(Q + 1)(M - J) - L$ degrees of freedom. The known long code information that is in $x_{1,kN}$ (see (8)) then allows us to write (14) as

$$[f_{1,kN+a} | s_{1,k}] \begin{bmatrix} Y_{kN+a} \\ - \mathbf{C}_{1,k} \end{bmatrix} = 0.$$  \hspace{1cm} (15)

This can be derived for $a = A_1, A_1 + 1, \ldots, A_2$. All these results can then be combined, leading to

$$[f_{1,kN+A_1}, \ldots, f_{1,kN+A_2} | s_{1,k}] \begin{bmatrix} Y_{kN+A_1} \\ \mathbf{O} \hspace{1cm} \mathbf{O} \\ \mathbf{O} \hspace{1cm} \ldots \hspace{1cm} \mathbf{O} \\ \mathbf{O} \hspace{1cm} \mathbf{O} \hspace{1cm} Y_{kN+A_2} \\ - \mathbf{C}_{1,k} \hspace{1cm} \ldots \hspace{1cm} - \mathbf{C}_{1,k} \end{bmatrix} \triangleq [\tilde{f}_{1,k} | s_{1,k}] \begin{bmatrix} \tilde{f}_k \\ - \mathbf{C}_{1,k} \end{bmatrix} = 0. \hspace{1cm} (15)$$
The next theorem deals with the uniqueness of the solution for $s_{1,k}$ of (15) in the data search space $\Gamma$. For a proof see Appendix B.

**Theorem 3.** Under assumptions (A1) and (A2), we can state that (15) has a unique solution $s_{1,k}$ in $\Gamma$ for the data symbols, if and only if (A3) is satisfied.

When we do not consider any constraints on the data symbols ($\Gamma = \mathbb{C}^w$), Theorem 1 can also be formulated as follows. The proof is trivial.

**Theorem 4.** Under assumptions (A1) and (A2), we can state that (15) has a unique solution $s_{1,k}$ for the data symbols, if and only if

\[
\begin{bmatrix}
\tilde{y}_k \\
-\tilde{c}_{1,k}
\end{bmatrix}
\]

has rank $A(L + (Q + 1)J) + w - 1$.

So, under assumptions (A1) and (A2), this rank condition is equivalent to assumption (A3) for $\Gamma = \mathbb{C}^w$ and requires

\[
A(wN - L - (Q + 1)J) \geq w - 1. \tag{16}
\]

Let us now, like before, assume that additive noise is present. In this case we consider the following minimization problem:

\[
\min_{\tilde{f}_{1,k}, s_{1,k}} \left\{ \left\| \begin{bmatrix} \tilde{f}_{1,k} \\ s_{1,k} \end{bmatrix} \begin{bmatrix} \tilde{y}_k \\ -\tilde{c}_{1,k} \end{bmatrix} \right\|^2 \right\}. \tag{17}
\]

To avoid the all-zero solution some non-triviality constraint is imposed. Here, we only impose a constraint on the data symbols, which then determines $\Gamma$. We elaborate on this constraint in the section dealing with adaptive processing.

This block processing step for the equalizer-based approach is related to the algorithm of [10]. However, our method focuses directly on the data symbols without an explicit computation of the linear equalizer, which can lead to simple adaptive processing (see Sections 4.2.2 and 5.1.1). Moreover, our method is capable of handling multiple delays ($A > 1$), which can improve performance.

### 3.3. Equivalences

There is a clear equivalence between Theorems 1 and 3, which implies the equivalence between the rank condition in Theorem 2 and the rank condition in Theorem 4. This was already indicated by the correspondence between the necessary conditions (12) and (16).

The relation between the subspace-based approach and the equalizer-based approach becomes clear after rewriting the problems (13) and (17), keeping in mind that for (17) we do not impose any constraints on the linear equalizer. Rewriting (13) (see also (11)) leads to

\[
\min_{s_{1,k}} \left\{ \left\| \begin{bmatrix} O & O \\ O & \cdots & O \\ O & O & P_{\text{eff}}^{e_n,e_{n+1}} \end{bmatrix} s_{1,k} \tilde{c}_{1,k} \right\|^2 \right\}, \tag{18}
\]
where $P_{V_k N + a}$ is the projection matrix on the row space of $V_{k N + a}^H$:

\[
P_{V_k N + a} = V_{k N + a} V_{k N + a}^H.
\]

Rewriting (17) (see also (15)), keeping in mind that we do not impose any constraints on the linear equalizer (i.e., we can simply eliminate $f_{1,k}$), leads to

\[
\min_{s_{1,k}} \left\{ \left[ \begin{array}{cccc}
P_{V_k N + a} & O & O \\
O & \ddots & O \\
O & O & P_{Y_k N + a}^\perp
\end{array} \right] \left[ \begin{array}{c}
s_{1,k} C_{1,k} \\
c_{1,k}
\end{array} \right]^2 \right\},
\]

(19)

where $P_{Y_k N + a}^\perp$ is the projection matrix on the space orthogonal to the row space of $Y_{k N + a}$:

\[
P_{Y_k N + a}^\perp = I_{w N \times w N} - Y_{k N + a}^\dagger Y_{k N + a}.
\]

When no additive noise is present and assumptions (A1) and (A2) hold, both problems are exactly the same, which explains the equivalences we found.

**Remark.**

(1) The subspace-based approach requires the knowledge of $L = \sum_{l=1}^L L_l$. When assumptions (A1) and (A2) hold, this can be derived from the dimension of the column space of the channel matrix $\mathcal{H}$ or equivalently from the dimension of the row space of an input matrix $X_{k N + a}$, with $A_1 \leq a \leq A_2$, which both are $L + (Q + 1) J$. When no additive noise is present, this dimension is equal to the rank of $Y_{k N + a}$. When additive noise is present, this dimension has to be estimated by deciding how many singular values of $Y_{k N + a}$ are above the noise level. Note that $L$ is user-independent, which can be exploited if the processing of all users is centralized, e.g., for uplink communication but not for peer-to-peer communication. Although we assume the exact knowledge of $L$ in this work, the proposed methods are quite robust against an overestimation of $L$ (see [20], where related methods are discussed).

(2) For the equalizer-based approach, we could also impose a constraint on the linear equalizer, loosing of course the relation between (18) and (19). Here, for the sake of simplicity, we will not impose any constraints on the linear equalizer.

4. **Block-based adaptive processing**

In this section, block-based adaptive processing is introduced. We derive block-based adaptive processing problems for the subspace-based approach and the equalizer-based approach. In this context we could, as a first attempt, view the block processing problems (13) and (17) as problems at time index $k$, where $k$ is not considered fixed. The block-based adaptive processing problems presented here are different in that they take into account the shift structure of the solutions for the data symbols of successive block processing problems. Translating this into mathematics, it means that at time index $k$ we will express

\[
[s_{1,0} \hat{C}_{1,0} \ s_{1,1} \hat{C}_{1,1} \ \cdots \ s_{1,k} \hat{C}_{1,k}],
\]

as a function of the following vector of data symbols:

\[
s_{1,0}^{(k+w)} = [s_{1}[0] \ s_{1}[1] \ \cdots \ s_{1}[k + w - 1]],
\]

(20)

where the superscript indicates the number of data symbols in the vector and the second subscript indicates the starting time index ($s_{1,k}$, see (7), can thus also be represented by $s_{1,k}^{(w)}$). This can be done as follows:

\[
[s_{1,0} \hat{C}_{1,0} \ s_{1,1} \hat{C}_{1,1} \ \cdots \ s_{1,k} \hat{C}_{1,k}] = s_{1,0}^{(k+w)} \hat{C}_{1,0}.
\]
where $\mathcal{C}_{1, k}$ is the $(k + w) \times (k + 1) w N$ matrix, given by

$$
\mathcal{C}_{1, k} = \begin{bmatrix}
\tilde{C}_{1,0} & 0 & 0 \\
0 & \tilde{C}_{1,1} & \vdots \\
\vdots & \vdots & \ddots & 0 \\
0 & 0 & \tilde{C}_{1, k}
\end{bmatrix}.
$$

(21)

For both the subspace-based approach and the equalizer-based approach we consider two types of constraints on the data symbols (a finite alphabet constraint and a linear constraint) and present efficient recursive algorithms to solve the adaptive problem under these constraints.

4.1. Subspace-based approach

We consider the following minimization problem at time index $k$:

$$
\min_{s_{1,0}^{(k+w)}} \|s_{1,0}^{(k+w)} \mathcal{C}_{1, k} \mathcal{V}_{nk}\|^2,
$$

(22)

where $s_{1,0}^{(k+w)}$ is defined in (20), $\mathcal{C}_{1, k}$ is defined in (21) and $\mathcal{V}_{nk}$ is the block diagonal matrix, given by

$$
\mathcal{V}_{nk} = \begin{bmatrix}
\mathcal{V}_0^n & & \\
& \mathcal{V}_1^n & \\
& & \ddots & \\
& & & \mathcal{V}_k^n
\end{bmatrix}.
$$

To avoid the all-zero solution some non-triviality constraint is imposed on the data symbols. Two constraints are investigated: a finite alphabet constraint and a linear constraint. Note that the size of the minimization problem grows with $k$. This problem will be dealt with.

4.1.1. Finite alphabet constraint

The first method we discuss uses the finite alphabet constraint

$$
\mathcal{s}_{1,0}^{(k+w)} \in \Omega^{k+w}.
$$

(23)

So, the constrained minimization problem at time index $k$ considered here is defined by (22) and (23). This problem can also be formulated using a discrete-time finite-state Markov process that is based on a state diagram where every state represents an element of $\Omega^{w-1}$ and where the $\# \Omega^w$ state transitions at time index $k$ (every state has $\# \Omega$ possible next states) are governed by the cost function

$$
\|s \tilde{C}_{1, k} \mathcal{V}_{nk}\|^2.
$$

with $s \in \Omega^w$. The problem is then equivalent to finding the path in the state diagram with the minimal total cost, from time index 0 up to time index $k$. This search can be carried out recursively by the Viterbi algorithm [5]. Note that, while the size of the minimization problem (22) grows with $k$, the computational complexity per data symbol period is independent of $k$. 
This method is related to the algorithm in [21]. However, our method uses long code sequences to split the common Viterbi scheme in [21] into separate user-specific Viterbi schemes.

4.1.2. Linear constraint

The second method uses the linear constraint

\[ s_{1,0}^{(k+w)}[1_1, \ldots, 1_{k-l}] = \hat{s}_{1,0}^{(k-l)} (l > -w), \tag{24} \]

where \( \hat{s}_{1,0}^{(k-l)} \) is defined using the notation of (20), with \( \hat{s}_1[k-l] \) representing the ‘hard’ estimate of \( s_1[k-l] \) made at time index \( k \). This means we only have \( w+l \) unknowns (combined in \( s_{1,k-l}^{(w+l)} \)), where \( l \) is a design parameter to change the number of unknowns, once \( w \) is fixed. So, the constrained minimization problem at time index \( k \) considered here is defined by (22) and (24). This problem can also be formulated as a least-squares (LS) problem at time index \( k \):

\[ \min_{s_{1,k-l}^{(w+l)}} \left( : , k-l+1 : k+w \right) s_{1,k-l}^{(w+l)} \text{LS} \left( : , 1 : k-l \right) \hat{s}_{1,0}^{(k-l)} = \min_{s_{1,k-l}^{(w+l)}} \left( : , k-l+1 : k+w \right) s_{1,k-l}^{(w+l)} \text{LS} \left( : , 1 : k-l \right) \hat{s}_{1,0}^{(k-l)}, \tag{25} \]

which can then be solved recursively by an RLS algorithm based on QR-updating with an R-matrix of fixed size \((w+l) \times (w+l)\), as we will show next.

First rewrite (25) as

\[ \min_{s_{1,k-l}^{(w+l)}} \left( : , k-l+1 : k+w \right) s_{1,k-l}^{(w+l)} \text{LS} \left( : , 1 : k-l \right) \hat{s}_{1,0}^{(k-l)} = b_{1,k}. \tag{26} \]

The reason for this modified representation will become clear later. To solve (26) we can make use of the QR decomposition (QRD) [6]. The QRD of \([\min_{s_{1,k-l}^{(w+l)}} \left( : , k-l+1 : k+w \right) s_{1,k-l}^{(w+l)} \text{LS} \left( : , 1 : k-l \right) \hat{s}_{1,0}^{(k-l)} \] focusing on all columns but the last, is given by

\[ \left[ \min_{s_{1,k-l}^{(w+l)}} \left( : , k-l+1 : k+w \right) s_{1,k-l}^{(w+l)} \text{LS} \left( : , 1 : k-l \right) \hat{s}_{1,0}^{(k-l)} \right] \right| b_{1,k} = \left[ Q_k \mid \star \right] \begin{bmatrix} R_k & z_k \\ O & \star \end{bmatrix}, \tag{27} \]

where \( Q_k, R_k \) is the QRD of \([\min_{s_{1,k-l}^{(w+l)}} \left( : , k-l+1 : k+w \right) s_{1,k-l}^{(w+l)} \text{LS} \left( : , 1 : k-l \right) \hat{s}_{1,0}^{(k-l)} \] and \([Q_k \mid \star \]) is unitary. We assume \([\min_{s_{1,k-l}^{(w+l)}} \left( : , k-l+1 : k+w \right) s_{1,k-l}^{(w+l)} \text{LS} \left( : , 1 : k-l \right) \hat{s}_{1,0}^{(k-l)} \] has full column rank, for which we need

\[ A(wN-L-(Q+1)J) \geq 1. \tag{28} \]

The solution of (26) then satisfies

\[ R_k \text{fr} \left( s_{1,k-l}^{(w+l)} \right) = z_k, \tag{29} \]

which can be solved through backsubstitution.

Assume now that \( R_k \) and \( z_k \) are known. The aim is then to find an efficient updating rule to update \( R_k \) and \( z_k \) into \( R_{k+1} \) and \( z_{k+1} \). For this, we first have to find \( \hat{s}_1[k-l] \) (to update \( s_{1,0}^{(k-l)} \) into \( \hat{s}_{1,0}^{(k-l)} \)). As mentioned, \( \hat{s}_1[k-l] \) represents the ‘hard’ estimate of \( s_1[k-l] \) made at time index \( k \), i.e., made by solving (29) for \( s_1[k-l] \) and projecting the solution onto the finite alphabet \( \Omega \). Since \( s_1[k-l] \) is the last element of \( \text{fr} \left( s_{1,k-l}^{(w+l)} \right) \), only the first step of the backsubstitution scheme to solve (29) has to be executed in order to find \( \hat{s}_1[k-l] \) (parallel implementation possible). This is the reason why we used the representation of (26). So,
\( \hat{s}_1[k - l] \) is obtained as
\[
\hat{s}_1[k - l] = \text{proj}_\Omega \left\{ \frac{z_k(w + l)}{R_k(w, l, w + l)} \right\}.
\]

Once we have found \( \hat{s}_1[k - l] \) we are ready to update \( R_k \) and \( z_k \) in a second step.

First we consider \( \{w \#l \} \). From (27) we can then derive
\[
\left[ \gamma_{k+1}^{\downarrow} \Phi_G \left\{ \tilde{\mathcal{G}}_{1,k+1}^{\top} \right\} b_{1,k+1} \right] = \begin{bmatrix}
0 \\
\gamma_{k}^{\downarrow} \Phi_G \left\{ \tilde{\mathcal{G}}_{1,k}^{\top} \left( \cdot, 2 : l + w \right) \right\} b_{1,k} - \gamma_{k}^{\downarrow} \Phi_G \left\{ \tilde{\mathcal{G}}_{1,k}^{\top} \left( \cdot, 1 \right) \hat{s}_1[k - l] \right\} \\
\tilde{V}_{k+1}^{\downarrow} \Phi_G \left\{ \tilde{\mathcal{C}}_{1,k+1}^{\top} \right\} O \times l \\
0
\end{bmatrix}
\]
\[
= \begin{bmatrix}
Q_\star \\
O
\end{bmatrix} \begin{bmatrix}
R_k(\cdot, 1 : l + w - 1) \\
O \\
\tilde{V}_{k+1}^{\downarrow} \Phi_G \left\{ \tilde{\mathcal{C}}_{1,k+1}^{\top} \right\} O \times l \\
0
\end{bmatrix}
\]
\[
\tilde{R} = \begin{bmatrix}
Q \times O \\
O \times O
\end{bmatrix}
\]
\[
= \begin{bmatrix}
\tilde{P} \times \tilde{R} \\
\tilde{P} \times \tilde{z}_k \\
\tilde{P} \times 0
\end{bmatrix} = \begin{bmatrix}
R \\
O \\
O \\
O
\end{bmatrix},
\]

where \( \tilde{P} \times O \) is a unitary transformation matrix that triangularizes \( \tilde{R} \), except for the last column (see Section 4.3 for more details). Note that this triangularization can be performed by transforming only the first and third blockrows of \( \tilde{R} \). If we now compare (30) with
\[
\left[ \gamma_{k+1}^{\downarrow} \Phi_G \left\{ \tilde{\mathcal{G}}_{1,k+1}^{\top} \right\} b_{1,k+1} \right] = \begin{bmatrix}
Q_{k+1} \times \tilde{z}_{k+1} \\
O \times \tilde{z}_{k+1}
\end{bmatrix},
\]

we see that the matrices \( R \) and \( z \) in (30) are the matrices \( R_{k+1} \) and \( z_{k+1} \) sought. Keeping in mind that \( \tilde{P} \times O \) only transforms the first and third blockrow of \( \tilde{R} \), the updating formula for \( R_k \) and \( z_k \) is given by
\[
\begin{bmatrix}
R_{k+1} \\
O \times \tilde{z}_{k+1}
\end{bmatrix} \leftarrow \frac{P \times \begin{bmatrix}
0 \\
R_k(\cdot, 1 : w + l - 1) \\
\tilde{V}_{k+1}^{\downarrow} \Phi_G \left\{ \tilde{\mathcal{C}}_{1,k+1}^{\top} \right\} O \times l \\
0
\end{bmatrix}}{0 \\
R_k(\cdot, w + l) \hat{s}_1[k - l] - R_k(\cdot, w + l) \hat{s}_1[k - l]}.
\]

where \( P \) is obtained by removing the part of \( \tilde{P} \) corresponding to the second blockrow of \( \tilde{R} \).

When we consider \( \{w \leq l < 0\}(0 < w + l < w) \), the updating formula for \( R_k \) and \( z_k \) is derived in a similar fashion:
\[
\begin{bmatrix}
R_{k+1} \\
O \times \tilde{z}_{k+1}
\end{bmatrix} \leftarrow \frac{P \times \begin{bmatrix}
0 \\
R_k(\cdot, 1 : w + l - 1) \\
\tilde{V}_{k+1}^{\downarrow} \Phi_G \left\{ \tilde{\mathcal{C}}_{1,k+1}^{\top} \right\} O \times l \\
0
\end{bmatrix}}{0 \\
R_k(\cdot, w + l) \hat{s}_1[k - l] - R_k(\cdot, w + l) \hat{s}_1[k - l]}.
\]
Note that, while the size of the minimization problem (22) grows with \( k \), the computational complexity per data symbol period is independent of \( k \).

This method is related to the algorithm in [7] with the one-dimensional RLS schemes. However, our method uses long code sequences to improve the performance of the one-dimensional RLS schemes in [7]. Moreover, we consider a more general approach, where we can choose the dimension of the RLS schemes through \( l \), once \( w \) is fixed (one-dimensional RLS schemes correspond to \( l = -w + 1 \)).

4.2. Equalizer-based approach

We consider the following minimization problem at time index \( k \):

\[
\min_{\mathbf{f}_{1,0:k}} \min \left\{ \left\| \mathbf{f}_{1,0:k} \left[ s_{1,0}^{(k + w)} \right] \left[ \mathbf{Y}_{1,0} \right]^{-1} \mathbf{C}_{1,0} \right\|^2 \right\},
\]

(31)

where \( s_{1,0}^{(k + w)} \) is defined in (20), \( \mathbf{C}_{1,0} \) is defined in (21), \( \mathbf{f}_{1,0:k} \) is a collection of linear equalizers, given by \( \mathbf{f}_{1,0:k} = [\mathbf{f}_{1,0} \cdots \mathbf{f}_{1,l}] \)

and \( \mathbf{Y}_{1,0} \) is the block diagonal matrix, given by

\[
\mathbf{Y}_{1,0} = \begin{bmatrix}
\mathbf{Y}_0 & \mathbf{Y}_1 & \cdots & \mathbf{Y}_k
\end{bmatrix}
\]

To avoid the all-zero solution some non-triviality constraint is imposed. Here, we only impose a constraint on the data symbols. Two constraints are investigated: a finite alphabet constraint and a linear constraint. Note that the size of the minimization problem grows with \( k \). Like before, this problem will be dealt with.

4.2.1. Finite alphabet constraint

The first method we discuss uses the finite alphabet constraint

\[
s_{1,0}^{(k + w)} \in \Omega^{k + w}.
\]

(32)

So, the constrained minimization problem at time index \( k \) considered here is defined by (31) and (32). Like before, this problem can also be formulated using a discrete-time finite-state Markov process that is based on a state diagram where every state represents an element of \( \Omega^{w-1} \) and where the \( \# \Omega^w \) state transitions at time index \( k \) (every state has \( \# \Omega \) possible next states) are governed by the cost function

\[
\min_{\mathbf{f}_{1,k}} \left\{ \left\| \mathbf{f}_{1,k} \left[ s \right] \left[ \mathbf{Y}_k \mathbf{C}_{1,k} \right] \right\|^2 \right\},
\]

with \( s \in \Omega^w \). This means that for every time index \( k \) we have to solve \( \# \Omega^w \) LS problems to find the costs associated with all possible state transitions. The problem is then equivalent to finding the path in the state diagram with the minimal total cost, from time index 0 up to time index \( k \). Like before, this search can be carried out recursively by the Viterbi algorithm [5]. Note that, while the size of the minimization problem (31) grows with \( k \), the computational complexity per symbol period is independent of \( k \).

4.2.2. Linear constraint

The second method uses the linear constraint

\[
s_{1,0}^{(k + w)} \left[ \mathbf{I}_1 \cdots \mathbf{I}_{k-1} \right] = \mathbf{s}_{1,0}^{(k-w)} \quad (l > w),
\]

(33)
where $\hat{s}_1^{k-1:l}$ is defined using the notation of (20), with $\hat{s}_1[k-l]$ representing the ‘hard’ estimate of $s_1[k-l]$ made at time index $k$. Like before, this means we only have $w+l$ unknowns (combined in $s_1^{w+l}$), where $l$ is a design parameter to change the number of unknowns, once $w$ is fixed. So, the constrained minimization problem at time index $k$ considered here is defined by (31) and (33). This problem can also be formulated as an LS problem at time index $k$:

$$
\begin{bmatrix} \mathbf{y}_k^T - \mathbf{c}_1^T \mathbf{1}_k \end{bmatrix} \left[ \begin{bmatrix} \mathbf{f}_1^T \mathbf{1}_k \end{bmatrix} \right] \approx b_{1,k},
$$

which can then be solved recursively by an RLS algorithm. When we only focus on the solution for $s_1^{w+l}$ this RLS algorithm can be implemented using QR-updating with an R-matrix of fixed size $(d+w+l) \times (d+w+l)$ $(d = A(Q+1)M)$, as we will show next.

First rewrite (34) as

$$
\begin{bmatrix} \mathbf{y}_k^T - \mathbf{c}_1^T \mathbf{1}_k \end{bmatrix} \left[ \begin{bmatrix} \mathbf{f}_1^T \mathbf{1}_k \end{bmatrix} \right] \approx b_{1,k}.
$$

Like before, the reason for this modified representation will become clear later. To solve (35) we can make use of the QRD [6]. The QRD of $[\mathbf{y}_k^T - \mathbf{c}_1^T \mathbf{1}_k] | b_{1,k}$, focusing on all columns but the last, is given by

$$
[\mathbf{y}_k^T - \mathbf{c}_1^T \mathbf{1}_k | b_{1,k}] = [Q_k^{(1)} | Q_k^{(2)}] \star \left[ \begin{bmatrix} R_k^{(1,1)} & R_k^{(1,2)} & z_k^{(1)} \\ O & R_k^{(2,2)} & z_k^{(2)} \\ O & O & \ast \end{bmatrix} \right] = [Q_k | \ast ] \left[ \begin{bmatrix} R_k & z_k \\ O & \ast \end{bmatrix} \right],
$$

where $Q_k R_k$ is the QRD of $[\mathbf{y}_k^T - \mathbf{c}_1^T \mathbf{1}_k]$ and $[Q_k | \ast ]$ is unitary. We assume $[\mathbf{y}_k^T - \mathbf{c}_1^T \mathbf{1}_k]$ has full column rank, for which we need

$$
AwN \geq d + 1 = A(Q+1)M + 1.
$$

The solution for $s_1^{w+l}$ of (35) then satisfies

$$
R_k^{(2,2)} \mathbf{f}_1^T s_1^{w+l} = z_k^{(2)},
$$

which can be solved through backsubstitution.

Assume now that $R_k^{(2,2)}$ and $z_k^{(2)}$ are known. The aim is then to find an efficient updating rule to update $R_k^{(2,2)}$ and $z_k^{(2)}$ into $R_k^{(2,2)+1}$ and $z_k^{(2)+1}$. Note that this updating can be performed without the knowledge of $R_k^{(1,1)}$, $R_k^{(1,2)}$ and $z_k^{(1)}$. For this, we first have to find $\hat{s}_1[k-l]$ (to update $\hat{s}_1^{k-1:l}$), as mentioned, $\hat{s}_1[k-l]$ represents the ‘hard’ estimate of $s_1[k-l]$ made at time index $k$, i.e., made by solving (38) for $s_1[k-l]$ and projecting the solution onto the finite alphabet $Ω$. Since $s_1[k-l]$ is the last element of $\mathbf{f}_1^T s_1^{w+l}$, only the first step of the backsubstitution scheme to solve (38) has to be executed in order to find $\hat{s}_1[k-l]$ (parallel implementation possible). Like before, this is the reason why we used the representation of (35). So, $\hat{s}_1[k-l]$ is obtained as

$$
\hat{s}_1[k-l] = \text{proj}_Ω \left\{ \frac{z_k^{(2)}(w+l)}{R_k^{(2,2)}(w+l, w+l)} \right\}.
$$

Once we have found $\hat{s}_1[k-l]$ we are ready to update $R_k^{(2,2)}$ and $z_k^{(2)}$ in a second step.
First we consider \( l \geq 0 \) \((w + l \geq w)\). From (36) we can then derive

\[
\begin{bmatrix}
\mathcal{Y}_k^T \mid - f(\mathcal{C}_{k+1}^{T}) \mid b_{k+1}
\end{bmatrix}
= \begin{bmatrix}
\mathcal{Y}_k^T \mid O \mid 0 \mid - f(\mathcal{C}_{k+1}^{T} ; 2 : l + w) \mid b_{k+1} \mid \tilde{C}_{k+1} \mid \tilde{s}_1[k - l]
\end{bmatrix}
= \begin{bmatrix}
Q_k^{(1)} \star Q_k^{(2)} \star O
\end{bmatrix}
\]

where \( P_l \) is a unitary transformation matrix that triangularizes \( \tilde{R} \), except for the last column (see Section 4.3 for more details). Note that this triangularization can be performed by transforming only the second, third and fifth blockrows of \( \tilde{R} \). If we now compare (39) with

\[
\begin{darray}{cccc}
R_k^{(1,1)} & 0 & R_k^{(1,2)} ; 1 : l + w - 1) & \tilde{z}_k^{(1)} - R_k^{(1,2)} ; 1 : l + w \tilde{s}_1[k - l]
\hline
0 & O_{d 	imes d} & 0 & O
\hline
O & O & 0 & R_k^{(2,2)} ; 1 : l + w - 1) & \tilde{z}_k^{(2)} - R_k^{(2,2)} ; 1 : l + w \tilde{s}_1[k - l]
\hline
0 & O & 0 & O
\hline
0 & \tilde{Y}_{k+1}^T & - f(\mathcal{C}_{k+1}^{T}) \mid O_{\star \times l} & 0
\end{darray}
\]

we see that the matrices \( R_k^{(2,2)} \) and \( \tilde{z}_k^{(2)} \) in (39) are the matrices \( R_k^{(2,2)} \) and \( \tilde{z}_k^{(2)} \) sought. Keeping in mind that \( \tilde{P}_l \) only transforms the second, third and fifth blockrows of \( \tilde{R} \), the updating formula for \( R_k^{(2,2)} \) and \( \tilde{z}_k^{(2)} \) is given
by

\[
\begin{bmatrix}
O & \vdots & \star & \star \\
O & R_{k+1}^{(2,2)} & \theta_{k+1}^{(2)} \\
O & O & \star & \star \\
\end{bmatrix}
\leftarrow \mathbf{p}^H
\begin{bmatrix}
\mathbf{O}_{d \times d} & \mathbf{O} & \mathbf{0} \\
\mathbf{O} & R_k^{(2,2)}(:,1:w+l-1) & \mathbf{z}_k^{(2)} - R_k^{(2,2)}(:,w+l)\tilde{s}_i[k-l] \\
\mathbf{0}_{\cdot \times l} & -\mathbf{fc}\{\mathbf{C}_{1,k+1}^T\} & \mathbf{O}_{\cdot \times l} \\
\end{bmatrix}
\]

where \(\mathbf{P}\) is obtained by removing the part of \(\tilde{\mathbf{P}}\) corresponding to the first and fourth blockrows of \(\tilde{\mathbf{R}}\).

When we consider \(-w < l < 0\) (0 < w + l < w), the updating formula for \(R_k^{(2,2)}\) and \(z_k^{(2)}\) is derived in a similar fashion:

\[
\begin{bmatrix}
O & \vdots & \star & \star \\
O & R_{k+1}^{(2,2)} & \theta_{k+1}^{(2)} \\
O & O & \star & \star \\
\end{bmatrix}
\leftarrow \mathbf{p}^H
\begin{bmatrix}
\mathbf{O}_{d \times d} & \mathbf{O} & \mathbf{0} \\
\mathbf{O} & R_k^{(2,2)}(:,1:w+l-1) & \mathbf{z}_k^{(2)} - R_k^{(2,2)}(:,w+l)\tilde{s}_i[k-l] \\
\mathbf{0}_{\cdot \times l} & -\mathbf{fc}\{\mathbf{C}_{1,k+1}^T\} & \mathbf{O}_{\cdot \times l} \\
\end{bmatrix}
\]

Note that, while the size of the minimization problem (31) grows with \(k\), the computational complexity per data symbol period is independent of \(k\).

4.3. Computational complexity

In this section, we will only focus on the computational complexity of the adaptive block processing algorithms with the linear constraint. We assume that the data as well as the code sequences are complex.

First of all, we focus on the computational complexity of the QR-updating at time index \(k\). The matrix we want to triangularize, except for the last column, consists of an upper triangular matrix of size \(n_{\text{triang}} \times (n_{\text{triang}} + 1)\) extended with an \(n_{\text{rows}} \times (n_{\text{triang}} + 1)\) matrix at the bottom (this is the case for both the subspace-based and the equalizer-based approach). This means that all the elements of this \(n_{\text{rows}} \times (n_{\text{triang}} + 1)\) matrix have to be zeroed, except for the last column, without changing the upper triangular structure of the \(n_{\text{triang}} \times (n_{\text{triang}} + 1)\) upper triangular matrix. We recommend to do this column after column using Householder transformations [6]. Suppose that the \(i - 1\) first columns of the \(n_{\text{rows}} \times (n_{\text{triang}} + 1)\) matrix have already been set to zero and now the \(i\)th column has to be set to zero. This can be done by first computing a Householder vector of dimension \(n_{\text{rows}} + 1\) using the \(i\)th column of the \(n_{\text{rows}} \times (n_{\text{triang}} + 1)\) matrix and the \(i\)th diagonal element of the \(n_{\text{triang}} \times (n_{\text{triang}} + 1)\) upper triangular matrix. This operation involves \(6(n_{\text{rows}} + 1)\) flops. Then we perform the corresponding Householder transformation on the \(n_{\text{rows}} \times (n_{\text{triang}} + 1)\) matrix and the \(i\)th row of the \(n_{\text{triang}} \times (n_{\text{triang}} + 1)\) upper triangular matrix. This transformation will zero the \(i\)th column of the \(n_{\text{rows}} \times (n_{\text{triang}} + 1)\) matrix without changing the upper triangular structure of the \(n_{\text{triang}} \times (n_{\text{triang}} + 1)\) upper triangular matrix (note that the previous \(i - 1\) columns of the \(n_{\text{rows}} \times (n_{\text{triang}} + 1)\) matrix remain zero). Since the transformation only has to be performed on the last \(n_{\text{triang}} + 1 - i\) columns, this operation involves \(16(n_{\text{rows}} + 1)(n_{\text{triang}} + 1 - i) + 4(n_{\text{rows}} + 1) + 2(n_{\text{triang}} + 1 - i)\) flops. Knowing that for the subspace-based approach we have \(n_{\text{rows}} = A(wN - L - (Q + 1)J)\) and \(n_{\text{triang}} = w + l\) and that for the equalizer-based approach we have \(n_{\text{rows}} = A w N\) and \(n_{\text{triang}} = d + w + 1\) (\(d = A(Q + 1)M\)), the number of flops for the QR-updating at time index \(k\) can easily be calculated. It is clear that the QR-updating is less complex for the subspace-based approach than for the equalizer-based approach.

However, the QR-updating for the subspace-based approach at time index \(k\) requires the computation of the matrix product \(\mathbf{C}_{1,k} \mathbf{V}_{(k)N}^\ast \mathbf{P}_k^\ast\). This matrix product can be constructed based on the set of matrix products \(
\{\mathbf{C}_{1,k} \mathbf{V}_{(k)N}^\ast + a\}_{a=A_1}
\)
where \(\mathbf{C}_{1,k}\) has size \(w \times w N\) and \(\mathbf{V}_{(k)N}^\ast + a\) has size \(w N \times (w N - L - (Q + 1)J)\). Taking into
account the sparse structure of $C_{1,k}$ (see (9)), the computation of these matrix products results in $AwN(wN - L - (Q + 1)J)$ flops. Of course, we also need to derive the set of matrices $\{I_{(k)N}^n+a_{1a=A_1}\}$. This can be done in two ways. A first way starts with computing the SVD of $Y_n$ for every $n$, satisfying $A_1 \leq n - N\lfloor n/N \rfloor \leq A_2$ or $A_1 \leq n - N\lceil n/N \rceil \leq A_2$, from scratch, e.g., using the algorithm in [3]. From the resulting subspace decompositions we then consider the decompositions corresponding to $\{n = (k)N + a\}_{a=A_1}^{A_2}$. This method results in the following number of flops (only third-order terms in $wN$ and $(Q + 1)M$ are retained):

$$\min\{A,N\}(16(wN)^2(Q + 1)M + 12wN((Q + 1)M)^2).$$ (40)

A second way starts with computing recursively the SVD of $Y_n$ for every $n$, e.g., using a modified version of the algorithm in [14] that computes the SVD of $Y_n$ for every $n$ based on a QR up- and downdating step followed by $n_{it}$ Jacobi iterations. From the resulting subspace decompositions we then consider the decompositions corresponding to $\{n = (k)N + a\}_{a=A_1}^{A_2}$. This method results in the following number of flops (only second-order terms in $wN$ and $(Q + 1)M$ are retained):

$$N(18(wN)^2 + (29 + 36n_{it}))((Q + 1)M)^2 + (24 + 18n_{it})wN(Q + 1)M).$$ (41)

Note that the subspace decomposition of $Y_n$ is user-independent, which can be exploited if the processing of all users is centralized, e.g., for uplink communication but not for peer-to-peer communication.

Let us now consider a DS-CDMA system with the following parameters (see Section 6 for more details): $J = 5$, $M = 8$, $N = 6$, $L_1 = 1$ and $L = 7$. In Fig. 1 we show the computational complexity of the block SVD (40) and the adaptive SVD (41) (we take $n_{it} = 1$ for the adaptive SVD) as a function of $w$ for $Q = 2$ and $A = 1, 2$ (see Section 6 for why we take these specific values for $Q$ and $A$). We clearly see that for this particular example the computational complexity of the adaptive SVD, which is independent of $A$, is smaller than the computational complexity of the block SVD for $A = 1$ as well as for $A = 2$. Taking $l = 0$, Fig. 2 shows the total computational complexity of the subspace-based approach (using the adaptive SVD with $n_{it} = 1$) and the equalizer-based approach as a function of $w$ for $Q = 2$ and $A = 1, 2$. We see that for $A = 1$ the equalizer-based approach is less complex than the subspace-based approach, while for $A = 2$ it is the other

![Fig. 1. Computational complexity of the block SVD and the adaptive SVD.](image-url)
way around. Notice that for the subspace-based approach the most significant part of the computational complexity comes from the calculation of the adaptive SVD.

Remark. (1) In Section 3 dealing with block processing, we stated that, under assumptions (A1) and (A2), assumption (A3) (depending on the data search space $\Gamma$) is a necessary and sufficient condition for a unique solution in $\Gamma$ for the data symbols. However, the block-based adaptive processing algorithms are found to be fairly robust with respect to this condition (we still need $wN > L + (Q + 1)J$ though). This can intuitively be understood as follows. If the condition is not satisfied at some time index $k$, a spurious solution may arise at this time index but when this does not happen very often these solutions will not survive.

(2) Until now we did not specify the initialization or termination of the algorithms. For the initialization we only assume the knowledge of the first transmitted data symbol. If we take $k = 0$ as the first time index (like in the previous sections), we position this data symbol at time index $k = w - 2$, in order to have the best possible initialization for the algorithm (before time index $k = w - 2$ zeros are transmitted). The termination makes use of the fact that after transmitting a burst of $B$ data symbols zeros are transmitted. So, we consider $\alpha_1[k]$ non-zero for $k = w - 2, w - 1, \ldots, B + w - 3$ (for the other users we consider just the same). The description of the initialization and termination for the proposed algorithms is then rather straightforward.

(3) Although we have developed the algorithms for time-invariant channels, the algorithms can also be used for time-varying channels. In this case it is clear that $w$ should not be chosen larger than the coherence time of the time-varying channels.

5. Exponentially weighted adaptive processing

In this section exponentially weighted adaptive processing is introduced. We derive an exponentially weighted adaptive processing problem for the equalizer-based approach (for the subspace-based approach this is not useful). We only consider a linear constraint on the data symbols and present an efficient recursive algorithm to solve the adaptive problem under this constraint (the finite alphabet constraint does not lead to an efficient recursive implementation).
5.1. Equalizer-based approach

We consider the following minimization problem at time index \( k \) (compare with (31)):

\[
\min_{\bar{f}_{1,k}, s^{(k+1)}} \left\{ \left\| \left[ \bar{f}_{1,k} \left[ s^{(k+1)} \right] \right] \begin{bmatrix} \mathcal{Y}_k \\ - \sigma_{1,k} \end{bmatrix} \right\|^2 \right\},
\]

(42)

where \( s^{(k+1)} \) is defined using the notation of (7), \( \sigma_{1,k} \) is given by

\[
\sigma_{1,k} = \begin{bmatrix} \lambda^k \bar{c}_{1,0} \\ \vdots \\ \lambda^{k_{1,1}} \bar{c}_{1,k} \end{bmatrix}
\]

and \( \mathcal{Y}_k \) is defined as

\[
\mathcal{Y}_k = [\lambda^k \bar{y}_0 \quad \cdots \quad \lambda^{k_{1,1}} \bar{y}_{k_{1,1}}].
\]

Here, \( \bar{y}_k \) and \( \bar{c}_{1,k} \) are defined like before, but now for \( w = 1 \). Note that here only one linear equalizer \( \bar{f}_{1,k} \) is used, common to all subproblems (with \( w = 1 \)), while in Section 4.2 all subproblems (with arbitrary \( w \)) use different linear equalizers (combined in \( \bar{f}_{1,0:k} \)). The parameter \( \lambda \leq 1 \) is called the forget factor. To avoid the all-zero solution some non-triviality constraint is imposed. Here, we only impose a constraint on the data symbols. Only a linear constraint is investigated. Note that the size of the minimization problem grows with \( k \). This problem will be dealt with.

5.1.1. Linear constraint

The method we discuss uses the linear constraint

\[
s^{(k+1)} = \begin{bmatrix} 1 \cdots 1 \end{bmatrix} \left[ s^{(k-1)} \right], \quad (l \geq 0),
\]

(43)

where \( s^{(k-1)} \) is defined using the notation of (20), with \( \hat{s}_1[k-l] \) representing the ‘hard’ estimate of \( s_1[k-l] \) made at time index \( k \). Now, this means we only have \( l + 1 \) unknowns (combined in \( s^{(k-1)} \)), where \( l \) is a design parameter to change the number of unknowns. So, the constrained minimization problem at time index \( k \) considered here is defined by (42) and (43). This problem can also be formulated as an LS problem at time index \( k \):

\[
\left[ \mathcal{Y}_k^T \sigma_{1,k}^T \right] \begin{bmatrix} \bar{f}_{1,k}^T \\ \hat{s}_1^{(l+1)} \end{bmatrix} = \begin{bmatrix} \sigma_{1,k}^T \sigma_{1,k}^T \end{bmatrix} \begin{bmatrix} \hat{s}_1^{(k-0)} \end{bmatrix},
\]

(44)

which can be solved recursively by an RLS algorithm based on QR-updating with an R-matrix of fixed size \((d + l + 1) \times (d + l + 1)\) (\( d = A(Q+1)M \)), as we will show next.
First rewrite (34) as
\[
\begin{bmatrix}
[\gamma^T_k - fc(\gamma^T_{1,k})]
\end{bmatrix}
\begin{bmatrix}
\tilde{f}^T_{1,k}
\end{bmatrix}
\leq b_{1,k}. 
\tag{45}
\]

To solve (45) we can make use of the QRD [6]. The QRD of \([\gamma^T_k - fc(\gamma^T_{1,k})] b_{1,k}\), focusing on all columns but the last, is given by
\[
\begin{bmatrix}
[\gamma^T_k - fc(\gamma^T_{1,k})]
\end{bmatrix}
\begin{bmatrix}
b_{1,k}
\end{bmatrix}
= \begin{bmatrix}
Q_k \| \star \\
0
\end{bmatrix}
\begin{bmatrix}
R_k \| z_k
\end{bmatrix},
\tag{46}
\]

where \(Q_k R_k\) is the QRD of \([\gamma^T_k - fc(\gamma^T_{1,k})]\) and \([Q_k \| \star]\) is unitary. We assume \([\gamma^T_k - fc(\gamma^T_{1,k})]\) has full column rank, for which we need
\[
(l + 2)AN \geq d + l + 1 = A(Q + 1)M + l + 1. 
\tag{47}
\]

Note that this is the full rank condition for the time index \(k = l + 1\), i.e., the time index for which \(s_1[1]\) is estimated. As we will see further on this is the first data symbol that has to be estimated. Further note that if \(AN > 1\) there always exist an \(l\) for which this condition is satisfied.

Using a reasoning similar as before, \(\hat{s}_1[k - l]\) is obtained as
\[
\hat{s}_1[k - l] = \text{proj}_{\alpha}
\begin{bmatrix}
z_k(d + l + 1)
\end{bmatrix}
\begin{bmatrix}
R_k(d + l + 1, d + l + 1)
\end{bmatrix},
\]

and we can derive the following updating formula for \(R_k\) and \(z_k\):
\[
\begin{bmatrix}
R_{k+1} \\
O
\end{bmatrix}
\begin{bmatrix}
z_{k+1}
\end{bmatrix}
= P^H
\begin{bmatrix}
\lambda\beta R_k(:, 1 : d) \\
\lambda R_k(:, d + 1 : d + l)
\end{bmatrix}
\begin{bmatrix}
0 \\
\tilde{Y}_{k+1}^T
\end{bmatrix}
\begin{bmatrix}
\lambda(z_k - R_k(:, d + l + 1)\hat{s}_1[k - l])
\end{bmatrix}.
\]

Note that, while the size of the minimization problem (31) grows with \(k\), the computational complexity per data symbol period is independent of \(k\).

The computational complexity of the QR-updating at time index \(k\) is the same as the one computed in Section 4.3 but now we should use \(n_{\text{rows}} = AN\) and \(n_{\text{triang}} = l + 1\). Like before, for the initialization we only assume the knowledge of the first transmitted data symbol. If we take \(k = 0\) as the first time index (like above), we now position this data symbol at time index \(k = 0\), in order to have the best possible initialization for the algorithm. Like before, the termination makes use of the fact that after transmitting a burst of \(B\) data symbols zeros are transmitted. So, now we consider \(s_1[k]\) non-zero for \(k = 0, 1, \ldots, B - 1\) (for the other users we consider just the same). The description of the initialization and termination of the proposed algorithm is then rather straightforward.

6. Simulation results

In this section, we consider a specific procedure to obtain a DS-CDMA system with \(M\) outputs, that are sampled at the chip rate (see Section 2). Assume there are \(M_r\) receiver antennas. Further assume that these antennas are sampled at \(P\) times the chip rate \(N/T\) and let \(\phi^{(m)}_{j}([P])\) denote the discrete-time channel impulse response from the \(j\)th user to the \(m\)th receiver antenna, including the physical channel and the transmitter
and receiver filters. To make this temporal oversampling by a factor of $P$ possible without any resolution problems, we assume that the transmitter and receiver filters have a bandwidth around $PN/T$. Moreover, before transmission we spread $x_j[n]$ by a factor of $P$ using the length-$P$ code sequence $d_j[p]$ ($d_j[p] \neq 0$, for $p = 0, 1, \ldots, P - 1$, and $d_j[p] = 0$, for $p < 0$ and $p \geq P$). Note that $d_j[p]$ is a short (or periodic) code sequence. We interpret $d_j[p]$ as an additional unknown transmitted filter. To conclude, we have a system with $M = M_rP$ outputs, that are sampled at the chip rate, and the discrete-time channel impulse responses from the $j$th user to these $M$ outputs, i.e., $\{h_j^{(m)}[n]\}_{m=1}^M$, are given by

$$h_j^{(m-1)p + p + 1}[n] = \sum_{p=0}^{P-1} d_j[p]g_j^{(m)}[nP + p - p'], \quad m_r = 1, \ldots, M_r \quad \text{and} \quad p = 0, 1, \ldots, P - 1.$$ 

We further introduce some more assumptions and definitions. We assume DBPSK modulation ($\Omega = \{1, -1\}$) and we assume that the $J$ data symbol sequences $\{s_j[k]\}_{j=1}^J$ are mutually independent and zero-mean white. We also assume that the $M$ additive noises $\{e^{(m)}[n]\}_{m=1}^M$ are mutually independent and zero-mean white complex circular Gaussian with variance $\sigma_e^2$. We use randomly generated BPSK code sequences, i.e., $c_j[n] = \pm 1/\sqrt{N}$, for $n = 0, 1, \ldots, \rho N - 1$ and $j = 1, 2, \ldots, J$, and $d_j[p] = \pm 1/\sqrt{P}$, for $p = 0, 1, \ldots, P - 1$ and $j = 1, 2, \ldots, J$. For the $M_r$ discrete-time channel impulse responses from the $j$th user to the $M_r$ receiver antennas, i.e., for $\{g_j^{(m)}[p]\}_{m=1}^{M_r}$, the taps are modeled as independent samples from the same zero-mean complex circular Gaussian distribution (Rayleigh fading). Adopting the notation

$$E_j = \sum_{m=1}^{M_r} \sum_{p=-\infty}^{+\infty} |g_j^{(m)}[p]|^2,$$

we then define the signal-to-noise ratio (SNR) and the near-far ratio (NFR) for the user of interest $j = 1$ at the input of the receiver as $\text{SNR} = E_1/(M_r \sigma_e^2)$ and $\text{NFR} = E_j/E_1$, where $E_j = E$ for $j \neq 1$.

We will now focus on a DS-CDMA system with $5$ users ($J = 5$) and $4$ receiver antennas ($M_r = 4$). We consider a burst of $B = 101$ data symbols for every user (100 bits of information for every user). We will discuss two related strategies, i.e., two choices for $N$ and $P$, for which $NP$ is the same. The first strategy is based on $N = 12$ and $P = 1$, i.e., we have a system with $M = M_rP = 4$ outputs (the system is overloaded: $M \leq J$) and exploit a spreading factor of $N = 12$ to isolate the desired user $j = 1$. The second strategy is based on $N = 6$ and $P = 2$, i.e., we have a system with $M = M_rP = 8$ outputs (the system is underloaded: $M > J$) and exploit a spreading factor of $N = 6$ to isolate the desired user $j = 1$. We assume that all the channels from the set $\{g_j^{(m)}[p]\}_{m=1}^{M_r}$ have the same order and delay index, namely order 2, for $j = 1, 2, \ldots, 5$, and delay index 0, 4, 7, 9 or 10, for $j = 1, j = 2, j = 3, j = 4$ or $j = 5$, respectively. This means that for the first strategy we have $\delta_1 = 0$, $L_1 = 2$ and $L = 10$, while for the second strategy we have $\delta_1 = 0$, $L_1 = 1$ and $L = 7$.

We will compare the developed adaptive algorithms with the block processing algorithm of [10], which directly estimates a linear equalizer. We consider a simple modification of [10] that is capable of handling multiple delays ($A > 1$). We use the first 20 data symbols to estimate the linear equalizer. To remove the large bias this linear equalizer introduces we scale it by doing an LS fit between the ‘soft’ and ‘hard’ estimates of the first 20 data symbols obtained by the linear equalizer.

---

4 We use DBPSK modulation instead of BPSK modulation for two reasons. First, DBPSK is insensitive to the sign ambiguity present in the ‘hard’ data symbol estimate sequence, which arises from the fact that we assume the knowledge of the first transmitted data symbol. Second, DBPSK is robust against sign reversals present in the ‘hard’ data symbol estimate sequence, which can occur for the developed adaptive algorithms when a large noise peak is present. However, these problems can also be resolved for BPSK modulation (the 3 dB loss related with DBPSK modulation is then avoided) by introducing some short known headers or words in the transmitted data symbol sequence [13].
Note that in this work we have always assumed that (A1) is satisfied. When (A1) is not satisfied, the subspace-based and equalizer-based approach lose their nice interpretation but they can still be applied. Assume now that (A1) is satisfied when $H$ has more rows than columns. First, for an overloaded system ($M \leq J$) the performance then saturates when $Q$ reaches $L_1$, since there exists no value of $Q$ for which (A1) is satisfied (see also [12]). Because the first strategy leads to an overloaded system with $L_1 = 2$, we will take $Q = 2$ for this first strategy. Since then $Q = L_1$, we only consider $A_1 = A_2 = 0$ ($A = 1$) for the first strategy. Second, for an underloaded system ($M > J$) the performance then saturates when $Q$ reaches the maximum of $L_1$ and $\lceil L/(M - J) \rceil - 1$, the latter being the smallest value of $Q$ for which (A1) is satisfied (see also [12]). Because the second strategy leads to an underloaded system with $L_1 = 1$ and $\lceil L/(M - J) \rceil - 1 = 2$, we will take $Q = 2$ for this second strategy as well. Since then $Q = L_1 + 1$, we consider $A_1 = A_2 = 0$ ($A = 1$), as well as $A_1 = -1$ and $A_2 = 0$ ($A = 2$) for the second strategy. Unless mentioned otherwise, we will always use $w = 10$ and $l = 0$ for the block RLS algorithms.

We first consider a set of time-invariant (TI) channels. Taking SNR = 20 dB and NFR = 0 dB, Fig. 3 shows the mean-square error (MSE) of the ‘soft’ differentially decoded data symbol estimates for the two different strategies using different algorithms. Taking SNR = 10 dB, Fig. 4 shows the BER as a function of the NFR for the two different strategies using different algorithms.

First of all we notice that the MSE of the block processing algorithm is smaller than the MSE of the corresponding subspace-based or equalizer-based block RLS algorithm. However, since the block RLS algorithms do not contain a bias removal step like the block processing algorithm, we should better take a look at the BER. Then we see that the BER of the block processing algorithm is larger. For the block RLS algorithms the most significant part of the bit errors occurs during convergence of the MSE. For the first strategy the performance of the subspace-based and equalizer-based approach is exactly the same. For the second strategy the performance of the subspace-based approach is a little bit better because of the noise reduction capability of the SVD. We also notice that the second strategy has a lower MSE and BER than the first strategy, for the block RLS algorithms as well as for the block processing algorithm. This means that the positive effect of increasing the number of outputs $M$ is stronger than the negative effect of decreasing the spreading factor $N$. We further see that for the second strategy, $A = 2$ always leads to a lower MSE and BER than $A = 1$.

We then consider a set of time-varying (TV) channels. We assume a Doppler frequency of 100 Hz (speed of 120 km/h for a carrier frequency of 900 MHz) and a data symbol rate of $1/T = 100$ kHz. Fig. 5 shows the
typical time variation of a channel tap $g_{j}^{(m)}[p]$. Introducing $E_{j}^{av}$ as the average of $E_{j}$ over the total data symbol burst, SNR and NFR are now defined as $\text{SNR} = E_{j}^{av} / (M\sigma_{j}^{2})$ and $\text{NFR} = E / E_{j}^{av}$, where $E_{j}^{av} = E$ for $j \neq 1$. Taking $\text{SNR} = 20$ dB and $\text{NFR} = 0$ dB, Fig. 6 shows the MSE of the ‘soft’ differentially decoded data symbol estimates for the two different strategies using different algorithms. Taking $\text{SNR} = 10$ dB, Fig. 7 shows the BER as a function of the NFR for the two different strategies using different algorithms.

First of all we see that the BER of the block processing algorithm in [10] has increased compared to the TI channels case. This is caused by the fact that the estimated linear equalizer now becomes less and less accurate as time evolves (the MSE increases as time evolves). The MSE and BER for the block RLS algorithms have remained more or less the same compared to the TI channels case.

Next we look at the influence of $l$ on the performance of the presented block RLS algorithms (we still keep $w = 10$). We consider the set of TV channels and take $\text{SNR} = 10$ dB and $\text{NFR} = 5$ dB. Fig. 8 shows the BER
as a function of $l$ for the second strategy using two different block RLS algorithms. We clearly see that for this example both algorithms have an optimal value of $l = 0$.

Finally, we compare the presented block RLS algorithms with the corresponding Viterbi algorithms and we also give some results for the exponential RLS algorithm. We consider the set of TV channels and take $\text{SNR} = 10 \text{ dB}$. Table 1 shows the BER as a function of the NFR for the second strategy using different adaptive algorithms (we perform 500 runs). We clearly see that the Viterbi algorithms perform better than the corresponding block RLS algorithms. However, their computational complexity is much larger. Like the performance of the block RLS algorithms, the performance of the exponential RLS algorithm is greatly improved by using $A = 2$ instead of $A = 1$. Finally the exponential RLS algorithm performs better than the corresponding block RLS algorithms.
Table 1
BER as a function of the NFR for the second spreading strategy using different adaptive algorithms

<table>
<thead>
<tr>
<th>Algorithm (M = 8, N = 6, Q = 2), NFR</th>
<th>0 dB</th>
<th>5 dB</th>
<th>10 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block RLS (subspace-based, w = 10, l = 0, A = 1)</td>
<td>1.0 × 10⁻⁴</td>
<td>2.0 × 10⁻³</td>
<td>2.0 × 10⁻²</td>
</tr>
<tr>
<td>Block RLS (subspace-based, w = 10, l = 0, A = 2)</td>
<td>2.0 × 10⁻⁵</td>
<td>4.2 × 10⁻⁴</td>
<td>2.2 × 10⁻³</td>
</tr>
<tr>
<td>Block RLS (equalizer-based, w = 10, l = 0, A = 1)</td>
<td>2.2 × 10⁻⁴</td>
<td>2.5 × 10⁻³</td>
<td>2.1 × 10⁻²</td>
</tr>
<tr>
<td>Block RLS (equalizer-based, w = 10, l = 0, A = 2)</td>
<td>4.0 × 10⁻⁵</td>
<td>4.8 × 10⁻⁴</td>
<td>3.3 × 10⁻³</td>
</tr>
<tr>
<td>Viterbi (subspace-based, w = 10, A = 1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Viterbi (equalizer-based, w = 10, A = 1)</td>
<td>0</td>
<td>2.4 × 10⁻⁴</td>
<td>4.1 × 10⁻³</td>
</tr>
<tr>
<td>exp. RLS (l = 9, 采暖 = 0.95, A = 1)</td>
<td>1.0 × 10⁻⁴</td>
<td>1.2 × 10⁻³</td>
<td>5.8 × 10⁻³</td>
</tr>
<tr>
<td>exp. RLS (l = 9, 采暖 = 0.95, A = 2)</td>
<td>2.0 × 10⁻⁵</td>
<td>1.2 × 10⁻⁴</td>
<td>5.6 × 10⁻⁴</td>
</tr>
</tbody>
</table>

Fig. 8. BER as a function of l for the second spreading strategy using two different block RLS algorithms (TV channels).

7. Conclusions

We have presented different types of adaptive deterministic blind symbol estimation algorithms for DS-CDMA wireless communication. The algorithms differ from most existing deterministic blind equalization schemes for DS-CDMA systems in that they directly estimate the transmitted data symbols and are fully adaptive. Simulation results were given to prove the validity of all methods and to show that the adaptive processing algorithms perform better than the block processing algorithm of [10], especially for time-varying channels.

Acknowledgements

Geert Leus is supported by the Fund for Scientific Research - Flanders (Belgium) (FWO). Marc Moonen is a Research Associate of the FWO. This research work was carried out at the ESAT laboratory of the Katholieke Universiteit Leuven, in the framework of the Concerted Action Project MIPS (‘Model-based
Appendix A. Proof of Theorem 1

Under assumptions (A1) and (A2) we know that \( s_{1,k} \) is a solution of (11). We now prove that assumption (A3) (depending on the data search space \( I \)) is a necessary and sufficient condition for \( s_{1,k} \) to be a unique solution in \( I \). We first prove that assumption (A3) is a sufficient condition. Suppose that, next to \( s_{1,k}, (11) \) has another solution

\[
\begin{bmatrix}
s'_{1,k}^1 & \ldots & s'_{1,k}^{\ell}
\end{bmatrix}
\]

in \( I \), linearly independent of \( s_{1,k} \). Then we can state that

\[
\begin{bmatrix}
x'_{1,kN} \\
x_{kN+a}
\end{bmatrix} Y_{kN+a} = 0 \quad \text{for } a = A_1, A_1 + 1, \ldots, A_2,
\]

where \( x'_{1,kN} = s'_{1,k} C_{1,k} \). Because for every \( a (a = A_1, A_1 + 1, \ldots, A_2) \), the columns of \( x_{kN+a}^H \) already span the left null space of \( V_{kN+a} \) (this is due to (A1) and (A2)), it is clear that there exists no matrix \( x_{kN+a} \), with \( A_1 \leq a \leq A_2 \), for which

\[
\begin{bmatrix}
x'_{1,kN} \\
x_{kN+a}
\end{bmatrix}
\]

has full row rank \( L + (Q + 1)J + 1 \). In a similar way, we can prove that assumption (A3) is necessary.

Appendix B. Proof of Theorem 3

Under assumptions (A1) and (A2) we know that \( s_{1,k} \) is a solution of (15) for the data symbols, with a corresponding set of solutions for the linear equalizers \( \{f_{1,kN+a}^{(i)}\}_{a=A_i}^{A_i+1} \). We now prove that assumption (A3) (depending on the data search space \( I \)) is a necessary and sufficient condition for \( s_{1,k} \) to be a unique solution in \( I \) for the data symbols. We first prove that assumption (A3) is a sufficient condition. Suppose that, next to \( s_{1,k}, (15) \) has another solution

\[
\begin{bmatrix}
s'_{1,k}^1 & \ldots & s'_{1,k}^{\ell}
\end{bmatrix}
\]

in \( I \), linearly independent of \( s_{1,k} \), with a corresponding set of solutions for the linear equalizers \( \{f'_{1,kN+a}^{(i)}\}_{a=A_i}^{A_i+1} \). Then we can state that

\[
f'_{1,kN+a} C_{1,k} = x'_{1,kN} \quad \text{for } a = A_1, A_1 + 1, \ldots, A_2,
\]

where \( x'_{1,kN} = s'_{1,k} C_{1,k} \). From this it is clear that there exists no matrix \( x_{kN+a} \), with \( A_1 \leq a \leq A_2 \), for which

\[
\begin{bmatrix}
x'_{1,kN} \\
x_{kN+a}
\end{bmatrix}
\]

has full row rank \( L + (Q + 1)J + 1 \). In a similar way, we can prove that assumption (A3) is necessary.
References