Efficient Matrix Computations in Wideband Communications

Patrick Dewilde
Delft University of Technology,
Lang Tong
Cornell University,
Alle-Jan van der Veen
Delft University of Technology.
Correspondence: p.dewilde@its.tudelft.nl

Abstract

Modern telecommunications put increasing demands on the efficient use of bandwidth in a channel. This is especially true in the upcoming wideband channels, where a large number of users are spread across the same large frequency band. There are two main techniques to achieve the spreading, one works 'in the time domain' and uses randomized codes to characterize users. It is called 'Code Division Multiple Access', and in the case of a wide band channel - WCDMA. The other works in the frequency domain. It is called OFDM and utilizes orthogonal sinusoidal waveforms as carriers. In this paper we concentrate on WCDMA and show how new methods from the field of time-varying system theory can be applied to achieve efficient algorithms for signal separation of individual users. In particular, the efficient computation of different kinds of pseudo-inverses of a matrix with a jagged block-structure appear to be the main algorithms needed and they can be achieved using a state-space representation of the original matrix and keeping the state-space representation throughout the computation.

1 WCDMA

WCDMA stands for 'Wideband Code Division Multiple Access'. It is one of the main methods for coding and modulation in modern telecommunication systems such as UMTS, wireless LAN and even the upcoming Ultra Wide Band modulation method. Interestingly enough, the coding and modulation techniques of WCDMA lead to the handling of (relatively) large matrices with a definite but uncommon structure. More specifically, a receiving station in these systems is obtaining a combination of signals that bear a time-varying coding and modulation characteristic, tainted furthermore by multiple path distortion and noise. The problem the receiver has to solve is to obtain an estimate of the original signals which is as close as possible to the original. The classical way of doing that is to use a so-called 'rake receiver' consisting of a number of 'fingers', each of which consists of a matched filter tuned to the code of one of the constituent signals. From the point of view of computational efficiency, this is a reasonable solution, because each matched filter has the same computational complexity as the original coding matrix (they are merely Hermitian transforms of the originals). However, from the point of view of optimal signal estimation, it is a poor solution for two reasons. First the method does not provide for good channel estimation leading to unpleasant (and not optimal) inter symbol interference, and second the matched

filters of the different fingers are not orthogonal to each other, with equally unpleasant inter channel interference as a result. However, it is possible to formulate the estimation problem in such a way that optimal solutions can be written down mathematically as the solution of a pseudo-inverse of the original code matrix. The key point, however, is that at a first glance the computation of a pseudo-inverse matrix does not appear to have the low complexity of the matched filters, rather, its implementation seems to require operations with full matrix complexity. The present paper develops the techniques needed to obtain exact optimal solutions with minimal computational complexity, in fact a computational complexity equal to the matched filter. This is achieved by exploiting the structural properties of the matrix which describes the coding operation and the subsequent channel distortion. It turns out that these techniques are in fact derived from new insights in time-varying system theory combined with a careful analysis that brings factor analysis into play.

The behaviour of a WCDMA channel from a signal processing point of view is provided by a so-called 'data model' which describes algebraically the relation between the original signal and what is actually received after the signal has been coded, modulated, transmitted through the channel and demodulated. The CDMA coding technique assigns a unique code to each user. This code is used to differentiate between users, each user has specific codes assigned to him and these codes are orthogonal to those assigned to other users. Each symbol to be transmitted is coded by that characteristic sequence (if the specific symbol is s and the code the time sequence $c(t)$, the transmitted chunk of data for that symbol is $sc(t)$, it is called a sequence of chips.) In principle the code assignments to different users is in a natural sense orthogonal to each other so that exact detection for each single user simply achieved by projection, but after the coded signal has been subjected to channel distortion, the orthogonality between distorted signals of different users is in general no longer valid and provisions have to be taken to improve the chances of good estimation. In WCDMA this is done by a combination of techniques. First, besides the useful signal, a known control signal is added (also in an orthogonal way) which will assist with channel estimation. Next, to increase randomness and enhance the individuality of each user, a so called 'long scrambling code' is superimposed on the already coded signal so that the actual user specific code varies in time while transmission is progressing. It is this scrambling code which is the source of the time-varying nature of the signal processing in the WCDMA channel.

The single-user original data $s(t)$ has the form

$$s(t) = \sum_k s[k]\phi(t - kT) \tag{1.1}$$

where $s[k]$ are the symbols to be transmitted (real, discrete valued) and ϕ is an otherwise irrelevant single symbol pulse shape, while the known control signal is also discrete but imaginary and given by

$$\overline{s}(t) = \sum_k \overline{s}[k]\phi(t - kT). \tag{1.2}$$

Each of these signals are subsequently coded with a so called short data, respect. control channelization code (the normal Code Division Multiple Access Code which renders the signals of different users separable) $c_d(t)$ and $c_c(t)$. The coding replaces each symbol with a sequence of so called 'chips'. This first coding stage is then followed by another one, in which sequences of chips corresponding to subsequent symbols are concatenated into a long

sequence, which is coded further with what is called a long scrambling code extending over a large number of symbols. This sequence is then the actual input data used by the modulator, which takes care of conditioning the overal signal so that it is ready for transmission (we omit the electrical engineering details). The modulated signal is then transmitted and demodulated after reception to result in a sequence of received data which is a distorted copy of the original chip sequences. In modulation theory one shows that all these operations between modulator and demodulator may be summarized by assuming that the chip sequence is passed through a linear, discrete time filter. Given the high speed of data transmission this filter may be assumed time invariant (but of course varying on a longer time scale) and hence represented by an unknown and user dependent impulse response $\{h[i]\}$. We make one further assumption about the channel distortion, namely that it is limited in time, extending over at most L chips, where L is (much) smaller than the length of the scrambling code. This commonly made assumption may seem to be arbitrary, it does not seem to be really essential but simplifies the algebra at least in appearance.

The data code, control code and scrambling code for a single user combine as $c[n] = c_s[n]c_d[n]$ and $\overline{c}[n] = c_s[n]c_c[n]$ respectively, the signals coming out of the channel and to be estimated become now

$$\begin{array}{rcl} y[n] & = & y_d[n] + y_c[n] \\ & = & \sum_{i=0}^{L} h[i] \left\{ c[n-i](\sum_k s[k]q[n-i-kG]) + \overline{c}[n-i](\sum_k \overline{s}[k]q[n-i-kG]) \right\} \end{array} \tag{1.3}$$

in which q is a square window of the length of one chip and G the total length of the message. The (straighforward but somewhat involved) vectorization of this expression leads to the following matrix expressions [4]:

$$\mathbf{y} = \mathbf{T}(\mathcal{C})(\mathbf{I}_M \otimes \mathbf{h})\mathbf{s} + \mathcal{T}(\overline{\mathcal{C}})(\mathbf{I}_M \otimes \mathbf{h})\overline{s} + \mathbf{w}. \tag{1.4}$$

In this expression $\mathbf{T}(\mathcal{C})$ has the form (G is the length of a data or control code)

$$\mathbf{T}(\mathcal{C}) = \begin{bmatrix} \mathbf{D}_1 & & & & \\ \mathbf{E}_2 & \mathbf{D}_2 & & & \\ & \mathbf{E}_3 & \ddots & & \\ & & \ddots & \ddots & \\ & & & \mathbf{E}_M & \mathbf{D}_M \end{bmatrix} \tag{1.5}$$

where

$$\begin{bmatrix} \mathbf{D}_k \\ \mathbf{E}_{k+1} \end{bmatrix} = \left[\begin{array}{ccc} c[kG+1] & & \\ c[kG+2] & \ddots & \\ \vdots & & c[kG+1] \\ \hline c[(k+1)G] & & c[kG+2] \\ & \ddots & \vdots \\ & & c[(k+1)G] \\ & \mathbf{0}_{(\mathbf{G}-\mathbf{L})\times(\mathbf{L}+\mathbf{1})} & \end{array}\right] \tag{1.6}$$

and

$$(\mathbf{I}_M \otimes \mathbf{h}) = \begin{bmatrix} h_1 & & & \\ \vdots & & & \\ h_L & & & \\ & h_1 & & \\ & \vdots & & \\ & h_L & & \\ & & \ddots & \\ & & & h_1 \\ & & & \vdots \\ & & & h_L \end{bmatrix} \tag{1.7}$$

and $\mathbf{w}$ is the overall noise. The important point in this expression is the separation in a product of matrices dependent solely on the code and a matrix solely dependent on the channel transfer function. We see that the received vector $\mathbf{y}$ is (except for noise) a linear combination of columns determined exclusively by the code. This crucial property will allow for separate estimation and identification of symbols and channel (in fact, once one can identify the symbols one does not need the channel information anymore, but it is interesting to know that it is no problem to estimate one and the other - but the method makes no use of any explicit knowledge of the transfer medium, it is a blind estimation method!) Finally, signals from different users (with of course different codes and different transfer characteristics) combine linearly to produce the overall received signal.

A proper representation of the transfer matrix is an essential stepping stone in the development of optimal estimation methods, but another crucial element appears, the complexity and efficiency of the receiver. The matrix representing the signal received from a single user is highly structured (it is a product of two matrices with a jagged band structure and is highly sparse). Producing the code at the transmitter's side is computationally very efficient, it is now crucial to keep the estimation algorithm equally efficient. The matrix operations needed to perform the separation of signals and channel on the one hand and then perform optimal estimation on the remainders has to be just as computationally efficient as the original coding operation - if possible. The main results presented in this paper are algorithms capable of doing just that, by systematic exploitation of the structure of the matrices involved. We concentrate on the optimal handling of the code matrix, and treat two cases of independent interest: (1) efficient annihilation whereby an efficient computation of the projection matrix on the orthogonal complement of the space spanned by columns of a matrix is desired and (2) efficient oblique projection where the projection matrix is not orthogonal but projects parallel to a given space spanned by columns in a matrix. It turns out that these algorithms are special cases of a more general efficient method to compute the pseudo-inverse of a matrix with state space structure, presented in [2]. The algorithms have independent interest in their own right, however.

2 Efficient annihilation

A first algorithm we wish to consider here is the efficient orthogonal projection on the complement of a space spanned by a set of columns of a matrix T with structure as described above. This operation is useful to separate data from code of a single user, or to 'project out' other users in a more general setting. It is an algorithm of general interest and a good introduction to more general methods. Suppose we are given a (data) matrix of the type (the previous section should provide sufficient motivation)

$$T = \begin{bmatrix} D_1 & & & & \\ E_2 & D_2 & & & \\ & E_3 & \ddots & & \\ & & \ddots & \ddots & \\ & & & E_M & D_M \end{bmatrix} \tag{2.8}$$

and we wish to find a maximal (efficient) projection matrix P such that $PT = 0$. By efficient we mean that the computational complexity for P should equal the computational complexity of T, and since T is a banded matrix this complexity is determined by the size of the band. The problem is solved through (generalized) inner-outer factorization, to be explained next. We need the notion of 'causal' or 'lower' operator, corresponding to a 'causal' computational flow. For details see [1]. In the present case, T is a (block-) causal operator, represented by a block-lower triangular matrix. In general, all block lower triangular matrices will be called causal, while their block upper counterparts are called 'upper' or anti-causal, because they correspond to computations running from high indices to low. We reserve the right to determine the dimensions of the blocks in a matrix, and hence also the dimensions of the blocks on the main diagonal. To be more precise, we assume that a block matrix maps a (possibly non-uniform) sequence of vectors indexed by m_k to a (possibly non-uniform) sequence of vectors index by n_k. Hence, if u is the input sequence of vectors, then m_k is the dimension of u_k, and if y is the output sequence then n_ℓ is the dimension of y_ℓ. The matrix that maps u to y then has blocks of dimension $n_\ell \times m_k$. Blocks are on the main diagonal when $\ell = k$. We allow dimensions to disappear completely, zero dimension $m_k = 0$ means that there is no vector present at position k in the sequence, not even the numerical value zero. We say that there is only a 'placeholder' (indicated here by just a dot '$\cdot$', in computer science textbooks one often uses a perp '$\perp$' for a placeholder). Block-matrices have markedly different properties than regular matrices, for example, the inverse of a lower block matrix (assuming that it exists) can be an upper block matrix.

We shall say that a lower matrix is 'inner' if it is unitary, and that it is 'right-outer' if it has a right lower inverse. An inner-outer factorization of T now consists in finding an inner $U = [U_1\ U_2]$ and a right-outer T_r such that $T = [U_1\ U_2] \begin{bmatrix} 0 \\ T_r \end{bmatrix}$. It turns out (1) that such a factorization is always possible (see [2]) and (2) that the decomposion $U = \begin{bmatrix} U_1 & U_2 \end{bmatrix}$ in which both U_1 and U_2 are isometric ($U_i^* U_i = I$ for $i = 1, 2$) is such that U_1 forms a basis for the nullspace of T corresponding to the dimensions of the zero in the right factor.

We have:

Proposition 1 *The maximal causal orthonormal projection which is such that* $PT = 0$ *is given by* $P = U_1 U_1^*$, *where* $T = [U_1\ U_2] \begin{bmatrix} 0 \\ T_r \end{bmatrix}$ *is a right inner-outer factorization for* T.

Proof

That P is an orthonormal projection follows directly from the fact that U_1 is an isometry, and from the fact that it is hermitian by definition. That $PT = 0$ is also immediate from the definition of P. Finally, that it is maximal follows from the outerness of T_r, assuming existence of the factorization (proven in the literature cited). □

Hence, the determination of an inner-outer factorization will yield the desired projection. In the literature ([1]) an attractive algorithm to determine this factorization has been derived for general inner-outer decompositions. It is a so-called 'square-root algorithm' - a generalization of the square root algorithm known for the Kalman filter and originally presented in [3]. In the present case, the algorithm has a simpler form which we shall derive next. Our goal will be to find the numerically most efficient representation for U_1.

3 The square root algorithm to determine the projection

We start out by determining a 'state space representation' for T. Since it is block-banded with only one subband, the representation will be especially simple. We assume a unidirectional (or 'causal') computational flow proceeding in stages. In general, at stage k, the causal system takes as input a vector x_k called the state which summarizes the data needed from previous computations and an input u_k, to produce the next state x_{k+1} (if applicable) and the output k. For $k \leq 0$ and $k > M$ there are no inputs nor outputs, so these vectors disappear, as explained earlier. The (linear) map $\begin{bmatrix} x_k \\ u_k \end{bmatrix} \mapsto \begin{bmatrix} x_{k+1} \\ y_k \end{bmatrix}$ is called a 'realization of T' and has a standard representation as

$$\mathbf{T}_k = \begin{bmatrix} A_k & B_k \\ C_k & D_k \end{bmatrix} \tag{3.9}$$

corresponding to the partitioning of inputs and outputs (it may happen that some of the matrices disappear for lack of corresponding vectors). In the present case, especially simple realization matrices exist, corresponding to the block-banded form of the matrix

$$\begin{aligned} &\mathbf{T}_1 = \begin{bmatrix} \vert & I \\ \vert & D_1 \end{bmatrix},\ \mathbf{T}_2 = \begin{bmatrix} 0 & I \\ E_2 & D_2 \end{bmatrix},\ \cdots, \\ &\mathbf{T}_{M-1} = \begin{bmatrix} 0 & I \\ E_{M-1} & D_{M-1} \end{bmatrix},\ \mathbf{T}_M = \begin{bmatrix} - & - \\ E_M & D_M \end{bmatrix}, \end{aligned} \tag{3.10}$$

where '|' indicates a column of zero dimension and '−' a row of zero dimension. To simplify somewhat further, we make the not unreasonable assumption that the columns of T are linearly independent - in fact, even more is true: all the blocks D_k and E_k may be assumed

to have linearly independent columns. If that is the case, then the realization given is minimal in the classical sense. The assumption is not needed, strictly speaking, but it fits well with the classical theory, where one starts with a minimal realization for T - and in any other case a prior reduction would be called for. There is a compact way to express the operator in terms of its realization. To do so we define some block diagonal operators based on the realization matrices: $A = \mathrm{diag}[A_k]$, $B = \mathrm{diag}[B_k]$, etc... and define the 'causal shift' (it is nothing but a re-indexing matrix, when Z is applied to a vector $[u_i]$, the same vector but now indexed as $[u_{i+1}]$ is obtained)

$$Z = \begin{bmatrix} - & & & & \\ I & 0 & & & \\ & I & \ddots & & \\ & & \ddots & 0 & \\ & & & I & | \end{bmatrix} \tag{3.11}$$

and obtain $T = D + C(I - ZA)^{-1}ZB$. In the particular case where T is double-banded, we have $A = 0$, and T has the representation $T = D + CZB$, where $D = \mathrm{diag}[D_1, \cdots, D_M]$, $B = \mathrm{diag}[I, I, \cdots, -]$ and $C = \mathrm{diag}[|, E_2, \cdots, E_M]$.

It is well known in linear time-varying system theory (for a comprehensive treatment see [1]) that an isometric causal operator U_i has an isometric realization, let it be given by $U_i = \delta_i + \gamma(I - Z\alpha)^{-1}Z\beta_i$, we must have

$$T = U_2 T_r = (\delta_2 + \gamma(I - Z\alpha)^{-1}Z\beta_2)(D_r + C_r ZB) \tag{3.12}$$

since it is also known that T_r inherits the A, B operator pair of T. U itself has a unitary realization given by $U = \delta + \gamma(I - Z\alpha)^{-1}Z\beta$ in which $\delta = [\delta_1\, \delta_2]$ and $\beta = [\beta_1\, \beta_2]$. The 'square root algorithm' to determine the unknown parameters $\{\alpha, \beta, \gamma, \delta\}$ and $\{C_r, D_r\}$ is now easily derived by inversion of U: $\begin{bmatrix} 0 \\ T_r \end{bmatrix} = U^* T$, we find:

$$\begin{aligned} &(\delta^* + \beta^* Z^* (I - \alpha^* Z^*)^{-1}\gamma^*)(D + CZB) = \\ &= \begin{bmatrix} 0 \\ D_r + C_r ZB \end{bmatrix} \end{aligned} \tag{3.13}$$

The critical term in the product is

$$\beta^* Z^* (I - \alpha^* Z^*)^{-1}\gamma^* CZB = \beta^* Z^* [I + (I - \alpha^* Z^*)^{-1}\alpha^* Z^*]\gamma^* CZB \tag{3.14}$$

Introducing the upward diagonal shift: $X^{(-1)} = Z^* X Z$ on block diagonal matrices X (keeping appropriate conformal dimensions!), we find

$$\beta^* Z^* (I - \alpha^* Z^*)^{(-1)}[\gamma^* D + \alpha^* (\gamma^* C)^{(-1)} B] + \delta^* D + \beta^* (\gamma^* C)^{-1} B + \delta^* CZB = \begin{bmatrix} 0 \\ D_r + C_r ZB \end{bmatrix}. \tag{3.15}$$

Utilizing minimality of the state realizations, we find

$$\begin{bmatrix} \gamma^* D + \alpha^* (\gamma^* C)^{(-1)} B &=& 0 \\ \delta_2^* D + \beta_2^* (\gamma^* C)^{(-1)} B &=& D_r \\ \delta_2^* C = C_r & & \end{bmatrix} \tag{3.16}$$

This is actually (except for the zero entries) the 'square root algorithm'. To put it in a more recognizable form, let $Y = \gamma^* C$, and making the index explicit, then these equations can be written

$$\begin{bmatrix} \alpha_k^* & \gamma_k^* \\ \beta_k^* & \delta_k^* \end{bmatrix} \begin{bmatrix} 0 & Y_{k+1} \\ C_k & D_k \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ Y_k & 0 \\ (C_r)_k & (D_r)_k \end{bmatrix}. \tag{3.17}$$

In accordance with the basic inner-outer factorization theory, we have the additional requirement that $\ker[\cdot Y] = 0$ and $\ker[\cdot D_r] = 0$, which makes the factorization essentially unique (up to unitary equivalence on the realization). The procedure amounts to a Q-L factorization on the middle matrix, where knowledge of Y_{k+1} is assumed, and all unknown quantities follow from the procedure. The recursion starts at M (see further) and goes backwards towards $k = 1$. Hence: the values in the last blocks E_M, D_M actually influence the form of the projection operator for earlier blocks, the square root recursion works in an anticausal way. This may appear undesirable, but it is a fact of life, for which additional measures and approximations are possible - see further. The column dimensions of the blocks on the right hand side are the same as the column dimensions on $[C\ D]$. The row dimensions of Y_k and $(C_r)_k$ $(D_r)_k$, and of the zero entries constituting the sought after U_1 follow from the procedure. In the case assumed here (full column rank), the left hand side will be lower triangular, and Y_k and $(D_r)_k$ will be lower triangular, square matrices, so their dimensions are also known a priori. Further simplifications are obtained by remarking that all the B_k matrices are actually unit matrices. The recursion starts with $Y_{M+1} = \cdot$. The first real step then has:

$$\begin{bmatrix} \alpha_M^* & \gamma_M^* \\ \beta_M^* & \delta_k^* \end{bmatrix} \begin{bmatrix} - & - \\ E_M & D_M \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ Y_M & 0 \\ (E_r)_M & (D_r)_M \end{bmatrix} \tag{3.18}$$

(notice that in case the columns of $\mathbf{T}_k$ are not linearly independent, the algorithm still produces a result, but not one in which the matrices Y_k and $(D_r)_k$ are necessarily square).

Example

$$\left[\begin{array}{c|c} 0 & * \\ * & * \\ * & * \end{array}\right] \mapsto \left[\begin{array}{c|c} * & 0 \\ * & * \\ * & * \end{array}\right] \mapsto \left[\begin{array}{c|c} * & 0 \\ * & 0 \\ * & * \end{array}\right] \mapsto \left[\begin{array}{c|c} 0 & 0 \\ * & 0 \\ \hline * & * \end{array}\right] \tag{3.19}$$

The square root algorithm results in a state space representations for all the relevant matrices: the projection matrices U_1 and U_2 as well as the outer factor T_r. Although we concentrated on the computation of U_1 here, it may be remarked that in the same token a Moore-Penrose pseudo-inverse for the matrix T has been obtained. The complexity to compute these state space representations is of the order of the complexity of the original state space representation, while the resulting state space structures for the projections is also again of that order.

4 Oblique projections

A criticism of the previous method which in practice may or may not give problems is that the resulting square root algorithm runs in the reverse direction than the original computational flow. This is the price one has to pay for orthogonality of the projection P - orthogonality corresponds to finding least squares solutions, and it turns out that even its first components are dependent on the last entries in the matrix (atlhough possibly in a very weak manner). If one is willing to stray from orthogonality, 'oblique projections' or non-orthogonal pseudo-inverses may be attractive, because they may be choosen so that they do not require reverse computations. Here we present a non-orthogonal oblique projection algorithm which can be used to separate data from code or multiple users as desired. For each of exposition we restrict ourselves to a double band case.

Let

$$A = [L\ M] \tag{4.20}$$

be a tall matrix whose columns are linearly independent. Then A has a left inverse:

$$A^\dagger = \begin{bmatrix} L_1 \\ M_1 \end{bmatrix} \tag{4.21}$$

such that

$$A^\dagger A = \begin{bmatrix} L_1 \\ M_1 \end{bmatrix} [L\ M] = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{4.22}$$

Suppose now that a vector x is a linear combination of the rows of A:

$$x = Lx_1 + Mx_2 \tag{4.23}$$

then

$$x_1 = L_1 x \tag{4.24}$$

since $L_1 L = I$ and $L_1 M = 0$. So the problem of finding the 'oblique' projection x_1 reduces to finding a (possibly not optimal) left pseudo-inverse to A. We specialize to the case where the matrix is block-banded with two bands (this would correspond to a case with two users utilizing similar coding methods):

$$[l\ M] = \left[\begin{array}{cccccc|cccccc} D_1 & & & & & & F_1 & & & & & \\ E_2 & D_2 & & & & & G_2 & F_2 & & & & \\ & \ddots & \ddots & & & & \ddots & \ddots & & & & \\ & & & & E_n & D_n & & & & & G_n & F_n \end{array}\right] \tag{4.25}$$

or, reselecting:

$$\begin{bmatrix} D_1 & F_1 & & & & & & & & & & \\ E_2 & G_2 & D_2 & F_2 & & & & & & & & \\ & \ddots & \ddots & \ddots & \ddots & & & & & & & \\ & & & \ddots & \ddots & \ddots & & & & & & \\ & & & & & & D_{n-1} & F_{n-1} & & & & \\ & & & & & & E_n & G_n & D_n & F_n \end{bmatrix} \tag{4.26}$$

which equals

$$[L\ M] = \left[\begin{array}{cccccc} I & 0 & & & & \\ & & I & 0 & & \\ & & & & \cdots & \\ & & & & I & 0 \\ \hline 0 & I & & & & \\ & & 0 & I & & \\ & & & & \cdots & \\ & & & & 0 & I \end{array}\right] := [L\ M]S \tag{4.27}$$

Hence the problem reduces to computing the left inverse of a double band matrix, which for convenience we now denote as

$$\left[\begin{array}{cccc} D_1 & & & \\ E_2 & D_2 & & \\ & \ddots & \ddots & \\ & & \ddots & D_{n-1} \\ & & & E_n \quad D_n \end{array}\right]^{\dagger} \tag{4.28}$$

where we have grouped corresponding columns in L and M after the indicated reordering. This now is a fairly trivial operation, if we may assume that all D_k have linearly independent columns, which we now do. Starting from an input canonical realization for the double banded matrix at stage k:

$$\left[\begin{array}{cc} 0 & I \\ E_k & D_k \end{array}\right] \tag{4.29}$$

we find a realization for the left inverse as:

$$\left[\begin{array}{cc} -D_k^{\dagger}E_k & D_k^{\dagger} \\ -D_k^{\dagger}E_k & D_k^{\dagger} \end{array}\right] \tag{4.30}$$

which is best computed by diagram inversion as

$$x_{k+1} = y_k = -D_k^{\dagger}(E_k x_k + u_k) \tag{4.31}$$

with no starting state ($y_1 = x_2 = D_1^{\dagger}u_1$) and last state x_{n+1} disappearing. The generalized left inverse $A^{\dagger}$ computed in this way is lower triangular, its diagonal (block) entries are given by

$$[A^{\dagger}]_{kk} = D_k^{\dagger} \tag{4.32}$$

while its off diagonal block entries are:

$$[A^{\dagger}]_{ij} = -D_i^{\dagger}E_i D_{i+1}^{\dagger}E_{i+1}\cdots D_{j-1}^{\dagger}E_{j-1}D_j^{\dagger} \tag{4.33}$$

These expressions (entries of the inverse matrix) can also be found directly, by recursion (in the fashion of Crout-Doolittle).

An important remark is in order here: the pseudo inverse so determined is not unique. If the vector x whose oblique projection x_1 we want to determine is a precise linear combination

of the columns of A, then the result will be same for all pseudo-inverses. However, if x is not a precise linear combination and contains 'noise', then it definitely matters which pseudo-inverse is actually used, and it may be advantageous to use the pseudo-inverse producing a minimal norm result - see further theory on the matter.

In our present case, the matrix D_k has more structure, it is double-Toeplitz. This can be used to advantage to compute its inverse using the theory of matrices with low displacement rank. We leave such further enhancements to the actual desing engineers!

5 Discussion

In the two algorithms presented here, a pseudo-inverse of a matrix T plays a central role. When T has structure - and in the case of WCDMA code matrices it has lots of it! - then it becomes essential to exploit that structure in the computation of the desired pseudo-inverse. In [4] our low complexity methods have been evaluated in an actual concrete application with very promising results. In this presentation we have concentrated on two cases which may have a more general interest than just the WCDMA context. Engineering a WCDMA receiver would require a number of additional concerns, which go beyond the basic theory. Different users may have codes that are in some ways not commensurate with each other, so that the matrix formulation becomes quite complex (although each user individually has the same structure). Also, the properties of the noise must be exploited to maximum benefit. And finally, the identification of the channel may be necessary. This would involve different methods than those presented here, namely a factor analysis on the remainder matrix. These problems are left to the more specialized literature.

References

[1] P. Dewilde and A.-J. van der Veen. *Time-varying Systems and Computations.* Kluwer, 1998.

[2] P. Dewilde and A.-J. van der Veen. Inner-outer factorization and the inversion of locally finite systems of equations. *Linear Algebra and its Applications*, page *to appear*, 2000.

[3] M. Morf, M. Dobbins, J. Friedlander, and T. Kailath. Square-root algorithms for parallel processing in optimal estimation. *Automatica*, vol. 15:299–306, 1979.

[4] L. Tong, A.-J. van der Veen, and P. Dewilde. A new decorrelating rake receiver for long-code wcdma. In *Proceedings 2002 Conference on Information Sciences Systems.* Princeton University, March 2002.