Solving the ZF Receiver Equation for MIMO Systems Under Variable Channel Conditions Using the Block Fourier Algorithm

João Carlos Silva, Rui Dinis, António Rodrigues
Instituto Superior Técnico/Instituto de Telecomunicações/ CAPS-IST
Torre Norte 11-11, Av. Rovisco Pais 1, 1049-001
Lisboa, Portugal,
joao.carlos.silva@lx.it.pt, rdinis@ist.utl.pt,
antonio.rodrigues@lx.it.pt

Francisco Cercas, Nuno Souto, Sérgio Jesus
ISCTE / Instituto de Telecomunicações/ADETTI
Torre Norte 11-08, Av. Rovisco Pais 1, 1049-001
Lisboa, Portugal
francisco.cercas@lx.it.pt, nuno.souto@lx.it.pt, sj23@clix.pt

***Abstract*—**

The ZF (Zero Forcing) algorithm is one of the best linear receivers for DS-CDMA (Direct Sequence-Code Division Multiple Access). However, for the case of MIMO/BLAST (Multiple Input, Multiple Output / Bell Laboratories Layered Space Time) with high loading, the perceived complexity of the ZF receiver is taken as too big, and thus other types of receivers are employed, yielding worse results. In this paper, we investigate the complexity of the solution to the MIMO ZF receiver's equations using Block-Fourier algorithms, for both steady and unsteady channel situations.

Keywords- Zero Forcing, MIMO,Block Fourier, unsteady channels.

I. INTRODUCTION

The ZF algorithm (explained in detail in [1]), or algorithms based on them, are equalizers that are essential for compensating the various error sources that are present in the wireless communication link, such as ISI (Inter-Symbolic Interference) and MAI (Multiple Access Interference), which become more significant as the loading of the system is increased, with special incidence on MIMO systems.

The Block-Fourier algorithms, presented in [2],[3] for the ZF algorithm, are only suitable for constant channel conditions. In this work new versions of these algorithms are derived, capable of dealing with detection in unsteady channels with speeds up to 100km/h. These new algorithms are based in the partitioned block-Fourier algorithms of earlier works [2],[3], but extra steps were added to take in consideration the channel change from partition to partition. Inside each partition the channel is considered constant (thus providing approximate, yet reasonable results). The new algorithms, although more computationally expensive than the ones presented in earlier works, are not as expensive as the Gauss or Cholesky algorithms (that provide exact results).

The paper is organized as follows. Section II describes the Block Fourier (BF) algorithm. Section III introduces the Zero Forcing (ZF) algorithm, alongside the notion of matrix partitioning for solving the system equation. Simulation results are discussed in section IV, and the conclusions are given on Section V.

II. BLOCK FOURIER ALGORITHM

The BF algorithm can be easily applied for circulant matrices. Since in our case, the matrices we deal with are block-circulant, we will exploit this structure.

A. Diagonalizing Circulant Matrices

A circulant matrix is a square Toeplitz matrix with each column being a rotated version of the column to the left of it:

$$\mathbf{C} = \begin{bmatrix} c_1 & c_n & c_{n-1} & \cdots & c_2 \\ c_2 & c_1 & c_n & \cdots & c_3 \\ c_3 & c_2 & c_1 & \cdots & c_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n & c_{n-1} & c_3 & \cdots & c_1 \end{bmatrix} \quad (1)$$

The interesting property of circulant matrices is that its eigenvectors matrix is equal to the orthonormal DFT matrix F_n' of corresponding dimension n. F_n' can be written as:

$$\mathbf{F}_n' = \sqrt{n} * \mathbf{F}_n \quad (2)$$

where F_n is the non-orthonormal DFT matrix:

$$\mathbf{F}_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix} \quad (3)$$

with $\omega = e^{-j2\pi/n}$; $j = \sqrt{-1}$.

Using this property a circulant matrix $\mathbf{C}$, can be decomposed in:

$$\mathbf{C} = \mathbf{F}_n'^{-1}\mathbf{\Lambda}\mathbf{F}_n' = \mathbf{F}^{-1}\mathbf{\Lambda}\mathbf{F} \quad (4)$$

where Λ is a diagonal matrix that contains the eigenvalues of C. The Λ matrix can be easily computed from:

$$\mathbf{\Lambda} = diag(\mathbf{FC}(:,1)) \quad (5)$$

where $diag(\mathbf{x})$ represents the diagonal matrix whose diagonal elements are taken from the x vector. Substituting $\mathbf{C}$ by (4) in the linear system:

$$\mathbf{Cx} = \mathbf{b} \quad (6)$$

and solving for x results in:

$$\mathbf{x} = \mathbf{F}^{-1}\mathbf{\Lambda}^{-1}\mathbf{F}\mathbf{b} \tag{7}$$

Equation (7) can be computed efficiently with three discrete Fourier transforms and the inversion of the diagonal matrix $\mathbf{\Lambda}$. The complete solution would require $3n^2$ complex multiplication/addition pairs for the three DFT matrix/vector multiplications; n complex divisions to invert $\mathbf{\Lambda}$; and n complex multiplications to multiply $\mathbf{\Lambda}^{-1}$ by $\mathbf{Fb}$.

Using the Cooley & Tukey Fast Fourier Transform algorithm the complex multiplication/addition operation pairs needed to compute a size n DFT [2] is:

$$\mathbf{N}_{FFT}^{C\oplus\otimes} = \frac{n\log_2 n}{2} \tag{8}$$

This means that equation (7) can be computed spending only $n\left(\frac{3}{2}\log_2 n + 2\right)$ complex floating point operations (considering each multiplication/addition pair as one operation). The memory requirements to solve the system using such algorithm are also very modest. It's only necessary to keep in memory two size n vectors: the $\mathbf{b}$ vector and the first column of $\mathbf{C}$. All operations can be made in-place as the solution vector $\mathbf{x}$ replaces $\mathbf{b}$. Further economy can be achieved if $\mathbf{C}$ is sparse (band or block diagonal for example).

B. Application to Block-Circulant Matrices

A block-circulant matrix can be visualized as a circulant matrix where each element is a matrix instead of a scalar value. Consider a block-circulant matrix $\mathbf{C}_{(PQ)}$ composed by $N\times N$ blocks of size $P\times Q$. If $\mathbf{C}_{(PQ)} \in \mathbb{C}^{NP\times NQ}$ is block-circulant it must satisfy:

$$\mathbf{C}_{(PQ)i,j} = \mathbf{C}_{(PQ)\tilde{i},\tilde{j}} \tag{9}$$

with $\tilde{i} = ((i+P-1) \bmod NP)+1$, $\tilde{j} = ((j+Q-1) \bmod NQ)+1$. This means that each element of $\mathbf{C}_{(PQ)}$ is repeated P rows below and Q columns to the right of it. Indices that exceed the NP lines or NQ columns wrap around to the first lines and first columns, respectively. From now on we will consider only square block matrices with $N\times N$ blocks, so the N index will be omitted for simplicity. The block dimensions of the matrix will be represented in subscript between curve brackets, and in the case of square blocks matrices, only a dimension represented.

To deal with block circulant matrices we need to introduce the block-Fourier transformation. The block-Fourier matrix is defined as:

$$\mathbf{F}_{(K)} = \mathbf{F}_N \otimes \mathbf{I}_K \tag{10}$$

where $\mathbf{F}_N$ is a non-orthonormal DFT matrix of dimension N as defined in (2); $\mathbf{I}_K$ is the K size identity matrix; and $\otimes$ denotes the Kronecker product.

Similar to the last sub-section, a block-circulant matrix can also be decomposed using block-Fourier transforms:

$$\mathbf{C}_{(PQ)} = \mathbf{F}_{(P)}^{-1}\mathbf{\Lambda}_{(PQ)}\mathbf{F}_{(Q)} \tag{11}$$

where $\Lambda_{(PQ)}$ is a block-diagonal matrix computed from:

$$\mathbf{\Lambda}_{(PQ)} = diag_{(PQ)}(\mathbf{F}_{(Q)}\mathbf{C}_{(PQ)}(:,1:Q)) \tag{12}$$

where $diag_{(PQ)}(\mathbf{x})$ represents the block-diagonal matrix whose block-elements are the $P\times Q$ sized blocks of x. Similarly to the circulant systems, a block-circulant system can also be efficiently solved using the block-Fourier decomposition. The block-circulant system:

$$\mathbf{C}_{(P)}\mathbf{x} = \mathbf{b} \tag{13}$$

with $\mathbf{C}_{(P)} \in \mathbb{C}^{NP\times NP}$; $\mathbf{x} \in \mathbb{C}^{NP}$; $\mathbf{b} \in \mathbb{C}^{NP}$. It can be solved with:

$$\mathbf{x} = \mathbf{F}_{(P)}^{-1}\mathbf{\Lambda}_{(P)}^{-1}\mathbf{F}_{(P)}\mathbf{b} \tag{14}$$

If the blocks are not square the Moore-Penrose pseudoinverse concept can be used. Consider a block-circulant matrix $\mathbf{C}_{(PQ)} \in \mathbb{C}^{NP\times NQ}$, with $P\times Q$ sized blocks.

$$\mathbf{C}_{(PQ)}\mathbf{x} = \mathbf{b}, \tag{15}$$

$\mathbf{C}_{(PQ)} \in \mathbb{C}^{NP\times NQ}$; $\mathbf{x} \in \mathbb{C}^{NQ}$; $\mathbf{b} \in \mathbb{C}^{NP}$; $Q \le P$.

The system (15) can be solved using the Moore-Penrose pseudo-inverse of the complex matrix $\mathbf{C}_{(PQ)}$:

$$\mathbf{x} = \left(\mathbf{C}_{(PQ)}^H\mathbf{C}_{(PQ)}\right)^{-1}\mathbf{C}_{(PQ)}^H\mathbf{b} \tag{16}$$

if $\mathbf{C}_{(PQ)}^H\mathbf{C}_{(PQ)}$ is invertible.

This solution is the least squares solution (ZF) to the system (15), as previously shown. Applying the block-Fourier decomposition of (11) to (16) results:

$$\mathbf{x} = \left[\left(\mathbf{F}_{(P)}^{-1}\mathbf{\Lambda}_{(PQ)}\mathbf{F}_{(Q)}\right)^H\mathbf{F}_{(P)}^{-1}\mathbf{\Lambda}_{(PQ)}\mathbf{F}_{(Q)}\right]^{-1}\cdot \cdot\left(\mathbf{F}_{(P)}^{-1}\mathbf{\Lambda}_{(PQ)}\mathbf{F}_{(Q)}\right)^H\mathbf{b} \tag{17}$$

with $\mathbf{\Lambda}_{(PQ)}$ defined as in (12).

Equation (17) can be simplified, considering that $\mathbf{F}_{(K)}^H = \mathbf{F}_{(K)}^{-1}$:

$$\mathbf{x} = \mathbf{F}_{(Q)}^{-1}\left[\mathbf{\Lambda}_{(PQ)}^{H}\mathbf{\Lambda}_{(PQ)}\right]^{-1}\mathbf{\Lambda}_{(PQ)}^{H}\mathbf{F}_{(P)}\mathbf{b} \tag{18}$$

This solution can be computed with only three block-Fourier transforms, the inversion of $\mathbf{\Lambda}_{(PQ)}^{H}\mathbf{\Lambda}_{(PQ)}$ and the multiplication of two block diagonal matrices by a column vector. The multiplication

$$\left[\mathbf{\Lambda}_{(PQ)}^{H}\mathbf{\Lambda}_{(PQ)}\right]^{-1} * \mathbf{\Lambda}_{(PQ)}^{H} * \left(\mathbf{F}_{(P)}\mathbf{b}\right) \tag{19}$$

must be performed from right to left to minimize the number of operations required, since a matrix-vector multiplication is faster than a matrix-matrix one. Regarding that:

$$\left[\mathbf{\Lambda}_{(PQ)}^{H}\mathbf{\Lambda}_{(PQ)}\right]^{-1} \in \mathbb{C}^{NQ\times NQ} \tag{20}$$

$\mathbf{\Lambda}_{(PQ)}^{H} \in \mathbb{C}^{NQ\times NP}$, $\mathbf{F}_{(P)}\mathbf{b} \in \mathbb{C}^{NP}$

the multiplication (19) requires $(NP+NQ)NQ$ pairs of complex multiplications/additions. From the definition of block-Fourier transform in (10) it can be shown that given

$$\mathbf{y} = \mathbf{F}_{(K)}\mathbf{x} \tag{21}$$

with $\mathbf{x},\mathbf{y} \in \mathbb{C}^{NK}$; $\mathbf{F}_{(K)} \in \mathbb{C}^{NK\times NK}$, we have:

$$\mathbf{y}\left(i:K:(N-1)K+i\right) = \mathbf{F}_N\mathbf{x}\left(i:K:(N-1)K+i\right) \tag{22}$$

with $1 \le i \le K$, where F_N represents a non-orthonormal DFT matrix of dimension N as defined in (3).

This simply means that each block-Fourier transform of block-size K can be decomposed in K Fourier transforms of size N. Furthermore, each Fourier transform can be executed independently of each other and thus advantage of parallel hardware implementations can be taken. Taking this into consideration, recalling equation (8) and considering that are needed two block-Fourier transforms of block-size Q and one of block-size P to compute (18), the number of operations required for that three Fourier transforms is:

$$N_{3\,block-FFT}^{C\oplus\otimes} = (2Q+P)\frac{N\log_2 N}{2} \tag{23}$$

Due to finite precision round errors, the R_k matrices may not be Hermitian positive definite even if $C_{(PQ)}^H C_{(PQ)}$ is Hermitian positive definite. This can be corrected simply by removing the imaginary part of the diagonal elements and zeroing all other elements that have complex modulus below some threshold value, before applying the Cholesky factorization. This new simplified versions require the same number of floating point operations as derived before because null elements operations were not considered from the beginning.

III. ZF DETECTOR

The solution of the Zero-Forcing detector is

$$\hat{d} = \left(T^H T\right)^{-1} T^H e \tag{24}$$

This is the shortest length least squares solution of:

$$e = T\,d \tag{25}$$

where T is not square, in general. Figure 1 (left) represents the structure of T.

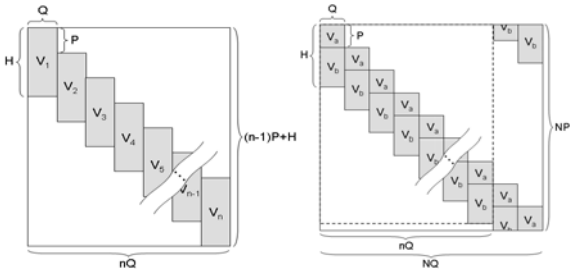

Figure 1 – Block Structure of the T matrix (left), Extended T matrix, constant channel (right)

T is a block matrix with V_n blocks disposed along its diagonal.

All N blocks are equal in a constant channel condition. Even if the channel varies slowly it can be a reasonably approximation to consider all V_n block equal as we will investigate later. In a constant channel condition, it's easy to extend matrix T to become block-circulant, simply by adding extra block-columns to it, as shown in Figure 1 (right). The elements below the last V_a block wrap around to the top of the columns. The number of extra columns needed to make T block-circulant is:

$$E = N - n = \lceil H / P \rceil - 1 \tag{26}$$

The resulting matrix is block-circulant with $N \times N$ blocks of $P \times Q$ size. This new matrix will be represented as $T_{(PQ)}$. Now equation (24) can be transformed in:

$$\hat{d}^{\approx} = \left(T_{(PQ)}^H T_{(PQ)}\right)^{-1} T_{(PQ)}^H e', \tag{27}$$

$T_{(PQ)} \in \mathbb{C}^{NP \times NQ}$; $\hat{d}^{\approx} \in \mathbb{C}^{NQ}$; $e' \in \mathbb{C}^{NP}$

and solved with the Fourier method, as done for equation (15) in last section. The e' vector can be obtained from e by padding at its end with $(N-n+1)P - H$ zeros, and $\hat{d}$ can be extracted from the first nQ elements of $\hat{d}^{\approx}$.

There are two approximations in the transformation of T in $T_{(PQ)}$: all V_n blocks were made equal and extra columns/lines were added to the matrix. If all blocks were made equal to the first block, the approximation would be increasingly worse (directly correlated to the speed) towards the last block.

A better approximation would be expected if a middle block was used. Let us use the middle block of T if n is even or the left-middle block if n is odd:

$$V = V_{(\lfloor n/2 \rfloor)} \tag{28}$$

Using this method, better approximations in a wider central range can be attained, as shown in Figure 2.

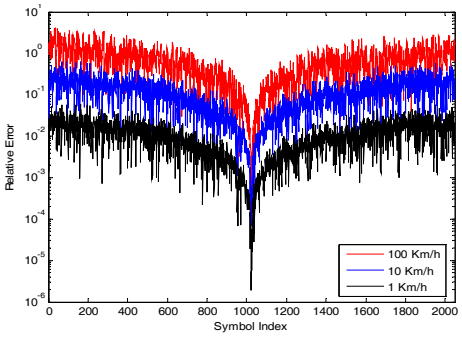

Figure 2 – Constant channel approximation relative error (PedestrianA, minimum load, 4x4 antennas) – Middle block.

Nevertheless, the approximation is very poor for the firsts and lasts elements of the $\hat{d}$ vector. The last pictures make obvious that the methods presented in [2],[3], although valid for a constant channel condition, are not very useful if the channel changes, even if low speeds are considered.

We will first concentrate on constant channel conditions. First we will determine the error level introduced by the addition of extra columns/lines to the T matrix that transform it in the block-circulant matrix $T_{(PQ)}$, as explained before and the error of the block-Fourier algorithm.

A. *Constant Channel Conditions*

Table 1 show the error level introduced in the determination of vector $\hat{d}$ by each phase of the detection process in 10 constant channel situations. Each value is the maximum complex modulus of the relative difference between the estimated $\hat{d}$ and real d obtained in 100 runs with distinct random d vectors. The "Estimation" column indicates the maximum error of the estimation if equation (24) is solved directly. The "Circulant" column shows the maximum error introduced by the extension of the T matrix to become $T_{(PQ)}$ relative to the estimated solution. The "Fourier1" and "Fourier2" columns show the maximum error introduced by the use of the Fourier algorithms 1 and 2 compared with the circulant system solution.

Since a SF=16 is considered, the minimum load situation corresponds to 0 interferers while the maximum load situation correspond to 15 interferes (with each user having only 1 physical channel). As can be seen the errors are very low for all the tested matrices and are only slightly above the floating point precision used. Furthermore the T matrix extension is not the main error cause.

It can be seen that the error level is constant along the entire symbol vector and no beginning neither end high error levels appear, since no multipath interference from adjacent blocks is being considered. Excluding the midamble from the detection and splitting the process in two independent detections, each one involving only data symbols, some errors can be introduced in the beginning of the second data chunk. This can

be corrected by including some symbols of the mid-amble in the second detection process.

Table 1 – Maximum absolute error introduced by each phase of the Zero-Forcing detector

	Antennas	Load	Estimation	Circulant	Fourier1	Fourier2
Pedestrian A	SISO	Min	4,E-16	5,E-16	2,E-15	3,E-15
		Max	9,E-14	1,E-13	1,E-13	1,E-13
	MIMO 2x2	Min	8,E-15	6,E-15	1,E-14	1,E-14
		Max	4,E-14	5,E-14	5,E-14	4,E-14
	MIMO 4x4	Min	4,E-14	2,E-14	7,E-14	5,E-14
Vehicular A	SISO	Min	7,E-16	8,E-16	3,E-15	3,E-15
		Max	1,E-14	3,E-14	3,E-14	3,E-14
	MIMO 2x2	Min	2,E-14	7,E-15	3,E-14	2,E-14
		Max	1,E-12	2,E-12	2,E-12	2,E-12
	MIMO 4x4	Min	5,E-14	3,E-14	8,E-14	7,E-14

Table 2 – Number of operations required by the Block-Fourier algorithm

Step	Rep.	N. of Operations
Tf	P	$\frac{QN\log_2 N}{2}$
ef	P	$\frac{N\log_2 N}{2}$
Σ_k	N	PQ^2
R_k	N	$\frac{Q^3+3Q^2+2Q}{6}$
R_k^{-1}	N	Q^2
df	N	Q^2+Q+QP^2
d	Q	$\frac{N\log_2 N}{2}$
Total		$=N\left[\frac{Q^3+15Q^2+8Q}{6}+PQ^2+QP^2+(Q+QP+P)\frac{\log_2 N}{2}\right]$

B. Partitioning

The algorithms proposed in last sub-section reveal already many parallel paths that could be exploited for parallel processing in adequate hardware. Nevertheless the algorithm remains globally sequential since it only determines the estimated $\hat{d}^{\approx}$ vector at the end.

Figure 3 illustrates an approach to split the extended Zero-Forcing equation (27) in smaller systems. Figure 3 represents $S_Q = T_{(PQ)}^H T_{(PQ)}$ and the estimated vector $\hat{d}^{\approx}$. The idea is to split the S_Q matrix in smaller ones. This can be an reasonable approximation because the S_Q has the greater values concentrated around the diagonal, and decreasing modulus as we get far away from the diagonal. This means that each element of vector $\hat{d}^{\approx}$ depends mainly of the same index value of vector $T_{(PQ)}^H e'$ and it depends less and less of the values of it as we get farther from that same index value.

Since each partition just approximates well the middle values, the $\hat{d}^{\approx}$ values of the beginning and end of each partition must be discarded. In the simulations it will be discarded the first l^- and last l^+ elements of each $\hat{d}^{\approx}$ partition.

Note that since the block-Fourier algorithm will be applied at each partition, and because each partition has to be approximated as a block-circulant matrix, high error will appear also in the first elements of the first partition and in the last elements of the last partition. This would not happen if each partition would be solved with an exact method like Gauss or Cholesky. This is the reason why those elements are also discarded in Figure 3.

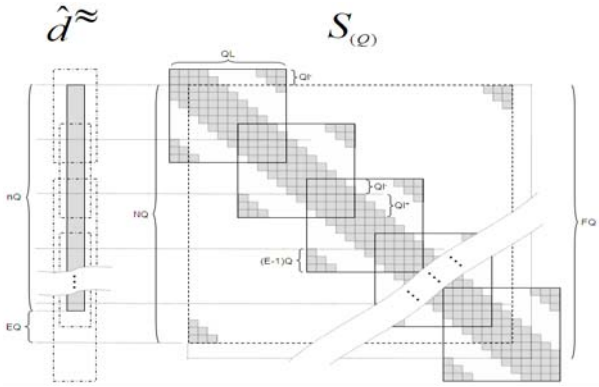

Figure 3 – Partitioning block-Fourier algorithm

The overlapping length must be carefully selected accordingly the precision required. The bigger the overlapping the better the approximation but more expensive will be the computation. In the derivation of the algorithms a prelap l^- and postlap l^+ were defined in a similar way as done in [3] but we shall always use $l^- = l^+$, since there is no advantage of defining dissimilar overlapping lengths. The partition length can be selected according the number of intended partitions and the overlapping selected, and this is usually determined by the hardware structure.

C. Unsteady Channel Conditions

The block-Fourier algorithm presented so far works well just under constant channel conditions or with very slow changing speeds. Even at 1 Km/h significant errors arise. The standard block-Fourier algorithm cannot be adapted for unsteady-channel conditions because it just works for block-circulant matrices, but the partitioned version can be easily adapted just by using in each partition a different block of the original *T* matrix. If we use the middle block in each partition of the original T matrix to construct each extended approximate $T_{(PQ)}$, the block-Fourier algorithm can be used for each partition as done in the last sub-section, and $\hat{d}^{\approx}$ obtained from the middle elements of each partition computation.

The overlapping length must be selected accordingly the precision required as for the constant channel. The bigger the overlapping the better the approximation but more expensive will be the computation.

The partition length now must be also selected according the precision required: bigger partitions will approximate quickly changing channels worst, but smaller partitions will require too much computational power. Partition and overlapping length can also be determined by the hardware structure available, if some kind of parallel processing is available in an already developed hardware platform. Also, very small partitions or very long overlapping can become incompatible making the required error level just not attainable for high speed channels.

IV. Simulation Results

The Monte Carlo was employed; 100 random data vectors were created and used with each of the T matrices to create the correspondent $\boldsymbol{e}$ vectors from $\mathbf{e = T d}$. Then all the

algorithms were applied in turn to estimate d from each e vector. Finally the estimated and original d vectors were compared and the wrong bits count up.

This procedure was repeated for different velocities (0,1,10 and 100 km/h) and antennas configuration (SISO and MIMO 2x2).

Table 3 present the BER results for 1 and 2 antennas. The "Est" algorithm represents the "exact" solution of the ZF equation, i.e. its solution with an algorithm that does not include extra approximations further from the numeric floating point precision of the simulator. As expected the "Est" algorithm always estimated the correct data vector, since no noise was considered in the simulation.

The "Fourier" algorithm corresponds to the unpartitioned Fourier algorithm described earlier. The "Fourier_m" uses the middle block of the ***T*** matrix while "Fourier" uses the first block. As expected, the use of the middle block gives the best results, except for low speeds, were the both are equivalent.

"FourierP_c" refers to the partitioned Fourier algorithm for constant channels while "FourierP_v" is the correspondent unsteady channel version. As expected "FourierP_v" gives always better results, except for very low speeds, were "FourierP_c" can give equivalent results with less floating point operations. The numbers after "FourierP_c" or "FourierP_v" indicate the number of blocks used in each partition and the pre-lap and post-lap blocks number. For example "FourierP_v_008_002_002" means an 8 blocks partition with 2 blocks pre-lap and 2 blocks post-lap algorithm.

From Table 3, it can be concluded that is advantageous to reduce the size of partitions, especially for high speeds (as expected, since in each partition the channel is approximated as constant). It's also easy to see that, for a particular partition size, better results are attained as larger laps are used.

For low speeds the size of the partitions does not have a so high influence in the correctness of the estimation, but greater lap sizes are also advantageous as noticed for high speeds. The conjunction of these two factors makes hard to find the best algorithm for high speeds, since small partitions can not have large overlaps. In a similar way, for low speeds, large overlaps imply very large partition and a compromise as to be made in each situation. Nevertheless, it's clear that for high speeds the most important factor is the size of the partitions, while for low speeds the overlap size is the key factor.

V. CONCLUSIONS

The Block-Fourier algorithms presented in [2],[3] for the zero-forcing algorithm under constant channel conditions where also tested under unsteady channel situations, having revealed useless in conditions of medium or high speeds. New versions of those algorithms, capable of dealing with detection in unsteady channels with speeds until 100km/h were derived and tested.

These new algorithms where based in the partitioned block-Fourier algorithms of [2],[3], but extra steps were added to take in consideration the channel change from partition to partition. Inside each partition the channel is considered constant. The new algorithms, although more computationally expensive than the original block-Fourier ones, are not so expensive as the Gauss or Cholesky ones (even if optimized versions were considered).

The best algorithm must be selected according the channel conditions: for almost constant channels constant block-Fourier algorithms could be used with good results, while for high speeds the new block-Fourier algorithms proposed must be used, preferably with small sized partitions.

Table 3 – ZF simulations results for 1 and 2 antennas (BER)

Interferers			0					0			
Channel model environment			Pedestrian A					Pedestrian A			
Antennas			1					2			
Number of Bits (Mbits)			10,24					20,48			
Velocity (Km/h)			0	1	10	100	0	1	10	100
Est			0	0	0	0	0	0	0	0
Fourier			0	5,6E-4	9,8E-3	2,7E-1	0	0,0E+0	4,5E-3	2,6E-1
Fourier_m			0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_128_000_000	128	0	0	1,8E-4	7,6E-4	9,5E-2	0	0,0E+0	6,4E-4	1,2E-1
FourierP_c_128_002_002	128	2	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_128_004_004	128	4	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_128_008_008	128	8	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_128_016_016	128	16	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_128_032_032	128	32	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_064_000_000	64	0	0	1,9E-4	7,7E-4	9,5E-2	0	0,0E+0	6,7E-4	1,2E-1
FourierP_c_064_002_002	64	2	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_064_004_004	64	4	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_064_008_008	64	8	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_064_016_016	64	16	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_032_000_000	32	0	0	2,1E-4	8,0E-4	9,5E-2	0	0,0E+0	7,1E-4	1,2E-1
FourierP_c_032_002_002	32	2	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_032_004_004	32	4	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_032_008_008	32	8	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_016_000_000	16	0	0	2,4E-4	8,5E-4	9,5E-2	0	0,0E+0	8,2E-4	1,2E-1
FourierP_c_016_002_002	16	2	0	1,8E-4	7,5E-4	9,5E-2	0	0,0E+0	6,4E-4	1,2E-1
FourierP_c_016_004_004	16	4	0	1,7E-4	7,5E-4	9,5E-2	0	0,0E+0	6,2E-4	1,2E-1
FourierP_c_008_000_000	8	0	0	3,1E-4	9,5E-4	9,5E-2	0	0,0E+0	1,1E-3	1,2E-1
FourierP_c_008_002_002	8	2	0	2,0E-4	7,5E-4	9,5E-2	0	0,0E+0	7,3E-4	1,2E-1
FourierP_c_004_000_000	4	0	0	5,4E-4	1,2E-3	9,5E-2	0	0,0E+0	1,7E-3	1,2E-1
FourierP_c_004_001_001	4	1	0	3,6E-4	7,9E-4	9,5E-2	0	0,0E+0	1,1E-3	1,2E-1
FourierP_v_128_000_000	128	0	0	7,6E-5	1,8E-4	6,2E-3	0	0,0E+0	1,5E-5	1,0E-2
FourierP_v_128_002_002	128	2	0	6,5E-5	1,7E-4	6,0E-3	0	0,0E+0	2,2E-6	8,8E-3
FourierP_v_128_004_004	128	4	0	5,9E-5	1,8E-4	6,1E-3	0	0,0E+0	3,2E-6	8,3E-3
FourierP_v_128_008_008	128	8	0	5,0E-5	2,2E-4	4,8E-3	0	0,0E+0	6,8E-6	6,3E-3
FourierP_v_128_016_016	128	16	0	3,1E-5	1,5E-4	2,9E-3	0	0,0E+0	1,5E-5	4,3E-3
FourierP_v_128_032_032	128	32	0	1,9E-5	7,2E-5	9,3E-4	0	0,0E+0	9,8E-7	1,7E-3
FourierP_v_064_000_000	64	0	0	2,2E-5	8,4E-5	9,6E-4	0	0,0E+0	2,3E-5	1,7E-3
FourierP_v_064_002_002	64	2	0	3,0E-5	4,0E-5	8,3E-4	0	0,0E+0	3,8E-6	1,5E-3
FourierP_v_064_004_004	64	4	0	3,8E-5	3,2E-5	7,7E-4	0	0,0E+0	8,7E-6	1,3E-3
FourierP_v_064_008_008	64	8	0	5,5E-5	6,4E-5	5,4E-4	0	0,0E+0	1,2E-5	9,9E-4
FourierP_v_064_016_016	64	16	0	1,9E-5	5,9E-5	1,2E-4	0	0,0E+0	9,8E-7	5,9E-4
FourierP_v_032_000_000	32	0	0	2,2E-5	6,6E-5	1,7E-4	0	0,0E+0	4,5E-5	6,6E-4
FourierP_v_032_002_002	32	2	0	1,5E-5	3,2E-5	1,2E-4	0	0,0E+0	2,5E-6	5,0E-4
FourierP_v_032_004_004	32	4	0	0,0E+0	2,9E-5	8,8E-5	0	0,0E+0	4,0E-6	4,0E-4
FourierP_v_032_008_008	32	8	0	1,9E-5	2,4E-5	8,9E-5	0	0,0E+0	3,5E-6	1,2E-4
FourierP_v_016_000_000	16	0	0	2,3E-5	3,8E-5	1,4E-4	0	0,0E+0	8,9E-5	2,7E-4
FourierP_v_016_002_002	16	2	0	0,0E+0	1,4E-5	6,1E-5	0	0,0E+0	1,0E-6	2,1E-4
FourierP_v_016_004_004	16	4	0	0,0E+0	1,1E-5	2,2E-5	0	0,0E+0	4,9E-8	2,0E-4
FourierP_v_008_000_000	8	0	0	2,2E-6	1,6E-5	7,8E-5	0	0,0E+0	1,6E-4	3,7E-4
FourierP_v_008_002_002	8	2	0	0,0E+0	2,3E-6	1,3E-5	0	0,0E+0	3,4E-7	1,5E-4
FourierP_v_004_000_000	4	0	0	2,3E-6	6,4E-6	9,1E-5	0	0,0E+0	2,7E-4	4,1E-4
FourierP_v_004_001_001	4	1	0	0,0E+0	0,0E+0	2,1E-5	0	0,0E+0	3,9E-7	1,8E-4

VI. REFERENCES

[1] S. Kay, "Fundamentals of Statistical Signal Processing: Estimation Theory", Englewood Cliffs, NJ: Prentice-Hall, 1993, pg391

[2] M. Vollmer, M. Haardt, J. Gotze, "Comparative Study of Joint-Detection Techniques for TD-CDMA Based Mobile Radio Systems", IEEE Sel. Areas Comm., vol.19, no.8, August 2001.

[3] R. Machauer, M. Iurascu, F. Jondral, "FFT Speed Multiuser Detection for High Rate Data Mode in UTRA-FDD", IEEE VTS 54th, vol.1, 2001.