Interpolation-Based Efficient Matrix Inversion for MIMO-OFDM Receivers

Moritz Borgmann and Helmut Bölcskei
Communication Technology Laboratory, Swiss Federal Institute of Technology
ETH Zentrum, Sternwartstrasse 7, 8092 Zurich, Switzerland
Email: {moriborg|boelcskei}@nari.ee.ethz.ch

***Abstract*— The use of orthogonal frequency-division multiplexing (OFDM) drastically simplifies receiver design in multiple-input multiple-output (MIMO) wireless systems. Nevertheless, MIMO-OFDM receivers are computationally very demanding since processing is performed on a tone by tone basis with the number of data-carrying tones ranging from 48 (as in the IEEE 802.11a/g standards) to 6817 (as in the DVB-T standard). In this paper, we present a new class of algorithms for computationally efficient channel inversion in MIMO-OFDM zero-forcing receivers. The basic idea of the proposed approach is based on the fact that even though the inverse of a polynomial matrix is generally not polynomial, the adjoint and the determinant will be polynomial, which allows efficient inversion of the individual matrices through interpolation. We perform an in-depth complexity analysis of the new class of interpolation-based inversion algorithms. For the system parameters employed in the IEEE 802.16a standard, we demonstrate computational cost savings of up to 80 % over brute-force per-tone matrix inversion.**

I. Introduction and Outline

The demand for ever increasing data rates in wireless communication systems dictates the use of large bandwidths. *Orthogonal frequency division multiplexing* (OFDM) [1], [2] has become a widely used technique to significantly reduce receiver complexity in broadband wireless systems. Standards employing an OFDM-based physical layer include the IEEE 802.11a/g wireless local area network (WLAN) standard [3], the IEEE 802.16 fixed broadband wireless access (BWA) standard [4], and the European digital audio and video broadcasting standards DAB and DVB-T, respectively.

Multiple-input multiple-output (MIMO) wireless systems [5]–[8] employ multiple antennas at both the transmitting and receiving side of the radio link to improve spectral efficiency. MIMO techniques in combination with OFDM modulation (MIMO-OFDM) [9], [10] have been identified as a promising approach for high spectral efficiency wideband systems: MIMO-OFDM is, among others, under consideration in the IEEE 802.11n high-throughput working group, which aims at extending the data rates of IEEE 802.11a/g based WLANs beyond the current maximum of 54 Mbit/s.

The use of MIMO-OFDM drastically simplifies receiver design by decoupling an intersymbol interference (i.e., frequency-selective) MIMO channel into a set of parallel flat-fading MIMO channels. However, MIMO-OFDM receivers can still be computationally very demanding since preprocessing (e.g., matrix inversion in zero-forcing (ZF) or minimum mean-squared error (MMSE) receivers) has to be performed on a tone by tone basis with the number of tones ranging from 48 (IEEE 802.11 a/g) to 6817 (DVB-T). Successive cancellation methods [11] and sphere decoding [12] require a QR decomposition on a tone by tone basis.

This work was supported in part by the Swiss National Science Foundation under grant 200021-100025.

Contributions: We present a new class of algorithms for computationally efficient matrix inversion in MIMO-OFDM ZF receivers. The extension of our method to MMSE receivers, successive cancellation, and sphere decoding for MIMO-OFDM systems is discussed in [13]. The basic idea of the proposed algorithm is based on the fact that the inverse of a polynomial matrix, although in general not polynomial, can be represented as a matrix with rational entries. As a result, numerator and denominator can be interpolated separately, which, combined with the fact that the number of OFDM tones is typically much larger than the channel order, leads to significant complexity savings when compared to per-tone inversion.

Organization of the Paper: In Section II, we introduce the channel and signal model. Section III presents our novel MIMO channel inversion algorithms. In Section IV, we perform an in-depth analysis of the corresponding computational complexity, and we discuss the implications of the new receiver algorithms for real-world systems. Finally, we summarize our results and discuss open problems in Section V.

Notation: $[\mathbf{A}]_{i,j}$ represents the element in the ith row and jth column of the matrix $\mathbf{A}$. $\mathbf{A}^\dagger$ stands for the pseudoinverse of $\mathbf{A}$. $\det \mathbf{A}$ and $\operatorname{adj} \mathbf{A}$ denote the determinant and adjoint, respectively, of a square matrix $\mathbf{A}$. $\mathbf{F}$ stands for the $N \times N$ FFT matrix with $[\mathbf{F}]_{m,n} = (1/\sqrt{N})\, e^{-j2\pi mn/N}$.

II. MIMO-OFDM

In this section, we shall first introduce the channel and signal model and briefly review the basic principles of channel estimation and interpolation in OFDM systems.

A. Discrete-Time Channel and Signal Model

Channel Model: Throughout the paper, for the sake of simplicity, we limit our discussion to the case of MIMO systems with the same number of transmit and receive antennas, denoted by M. The general case is described in [13].

The transfer function of the matrix-valued channel impulse response is given by

$$\mathbf{H}(e^{j2\pi\theta}) = \sum_{l=0}^{L-1} \mathbf{H}_l \, e^{-j2\pi l\theta}, \quad 0 \le \theta < 1 \tag{1}$$

with the matrix-valued taps $\mathbf{H}_l$ ($l = 0, 1, \ldots, L-1$). The channel model (1) is derived from the assumption of having L resolvable paths, where $L = \lfloor B\tau \rfloor + 1$ with B and τ denoting the signal bandwidth and delay spread, respectively.

MIMO-OFDM: In an OFDM-based MIMO system, the individual signals corresponding to the M transmit antennas are OFDM-modulated before transmission. OFDM demodulation is performed on each of the M received signals, followed by per-tone or joint (across tones) processing. Denoting the number of tones by N, the resulting input-output relation is given by

$$\mathbf{r}_k = \mathbf{H}\big(e^{j2\pi\frac{k}{N}}\big)\mathbf{c}_k + \mathbf{w}_k, \quad k = 0, 1, \ldots, N-1$$

where $\mathbf{c}_k = \begin{bmatrix} c_{k,0} & c_{k,1} & \cdots & c_{k,M-1} \end{bmatrix}^T$ and $\mathbf{r}_k = \begin{bmatrix} r_{k,0} & r_{k,1} & \cdots & r_{k,M-1} \end{bmatrix}^T$, and $\mathbf{w}_k$ is additive complex-valued noise. Here, $c_{k,m}$ stands for the complex-valued data symbol transmitted on the kth tone and the mth antenna, and $r_{k,n}$ denotes the received signal for the kth tone and the nth receive antenna. The channel matrices $\mathbf{H}(e^{j2\pi k/N})$ are obtained by sampling the MIMO channel's transfer function on the unit circle and can hence be interpreted as polynomial matrices of degree $L-1$ in s_k^{-1} with $s_k = e^{j2\pi k/N}$, i.e.,

$$\mathbf{H}\big(e^{j2\pi\frac{k}{N}}\big) = \mathbf{H}(s_k) = \sum_{l=0}^{L-1} \mathbf{H}_l \, s_k^{-l}. \tag{2}$$

In practice, a small subset of the N tones is set aside for pilot symbols and for *virtual tones* at the band edges, which help reduce out-of-band interference and ease the filtering requirements. In the following, we denote the set of indices corresponding to the $D \le N$ tones carrying data symbols by $\mathcal{D} \subseteq \{0, 1, \ldots, N-1\}$.

B. Channel Estimation and Interpolation

Throughout the paper, we will be concerned with *coherent* demodulation. Correspondingly, the receiver requires channel state information (CSI), which is usually obtained by either prepending known symbols to a data frame (*training preamble*) or by interspersing *pilot tones* with data-carrying tones (suitable for a burst transmission mode). Although this paper is not concerned with channel estimation *per se*, our results require understanding of the following basic principles of channel estimation in OFDM systems:

Noise-Free Case: Assume that the receiver has acquired noise-free estimates of $\mathbf{H}(s_k)$ for the P *base points* $k \in \mathcal{P} \subset \{0, 1, \ldots, N-1\}$. Since the M^2 individual scalar subchannels are finite impulse response (FIR) filters of order $L-1$, we can interpolate the $\mathbf{H}(s_k)$ for $k \in \mathcal{P}$ to obtain $\mathbf{H}(s_k)$ for all D data-carrying tones. The resulting interpolation error will be zero provided that the channel is critically sampled, i.e., $P = L$, or oversampled, i.e., $P > L$. In the following, we shall always assume $P \ge L$. The interpolation process can now be summarized briefly as follows. For each of the M^2 scalar subchannels, we collect the noise-free estimates of the corresponding single-input single-output (SISO) channel transfer function at the P base points in the vector $\mathbf{x}$, and the values at the D target tones in the vector $\mathbf{y}$. With the $P \times L$ matrix $\mathbf{G}$ obtained by picking the rows indexed by $\mathcal{P}$ and the first L columns of the FFT matrix $\mathbf{F}$, and the $D \times L$ matrix $\mathbf{E}$ obtained by picking the rows indexed by $\mathcal{D}$ and the first L columns of $\mathbf{F}$, the interpolation procedure amounts to computing

$$\mathbf{y} = \mathbf{E}\mathbf{G}^\dagger\mathbf{x} \tag{3}$$

for every scalar subchannel. Multiplying the vector $\mathbf{x}$ by the matrix $\mathbf{E}\mathbf{G}^\dagger$ is, in general, computationally expensive. In practice, the interpolation is therefore often approximated by applying a (polyphase) interpolation filter to the sequence in $\mathbf{x}$.

Noisy Case: If noisy observations of the channel transfer function at $P \ge L$ tones are available, the solution of the maximum-likelihood or, provided that the channel statistics are known, Bayesian (MMSE) *channel estimation problem* has the same form as (3) with different matrices $\mathbf{E}$ and $\mathbf{G}$ [14], [15]. *Oversampling* the channel transfer function by choosing $P > L$ is instrumental in reducing the estimation error.

III. Efficient Channel Inversion for MIMO-OFDM

Once the channel matrices $\mathbf{H}(s_k)$ for $k \in \mathcal{D}$ have been estimated (interpolated), a ZF receiver computes the unconstrained least squares estimate of the transmitted data vectors according to

$$\hat{\mathbf{c}}_k = \mathbf{H}^{-1}(s_k)\mathbf{r}_k, \quad k \in \mathcal{D}. \tag{4}$$

While (4) has to be evaluated at the symbol rate, the channel inverses $\mathbf{H}^{-1}(s_k)$ can be precomputed and have to be updated only when the channel changes. The focus of this paper is on computationally efficient algorithms for computing the $\mathbf{H}^{-1}(s_k)$ for $k \in \mathcal{D}$ starting from knowledge of $\mathbf{H}(s_k)$ for $k \in \mathcal{P}$.

A. Brute-Force Channel Inversion

The most straightforward approach to channel inversion, i.e., computing the $\mathbf{H}^{-1}(s_k)$ for $k \in \mathcal{D}$, is summarized in the following algorithm:

Algorithm I—Brute-Force Inversion

1) Interpolate $\mathbf{H}(s_k)$, $k \in \mathcal{P}$, to obtain $\mathbf{H}(s_k)$ for $k \in \mathcal{D}$
2) Compute $\mathbf{H}^{-1}(s_k)$ for each $k \in \mathcal{D}$.

It is obvious that the computation of D channel inverses (each of dimension $M \times M$) will in general put a huge computational burden on the receiver, especially if D is large and/or if the channel changes quickly. However, since typically $D \gg L$, the OFDM system essentially highly oversamples the MIMO channel's transfer function, so that the $\mathbf{H}(s_k)$ are

changing slowly across k. Exploiting the correlation between neighboring tones to reduce the computational complexity of algorithm I is the main idea of the new algorithms described in the next subsection.

B. Inversion by Interpolation

Let us start with two simple examples motivating the potential savings that can be obtained by exploiting the correlation of $\mathbf{H}(s_k)$ across k.

The Flat-Fading Case: Consider a flat-fading channel, so that $L = 1$. In this case, we have

$$\mathbf{H}^{-1}(s_k) = \mathbf{H}_0^{-1}, \quad k \in \mathcal{D}$$

which shows that it is sufficient to invert only the single constant matrix $\mathbf{H}_0$ in order to obtain all the D desired channel inverses. A question that arises naturally is whether for $L > 1$ the inverses $\mathbf{H}^{-1}(s_k)$ for $k \in \mathcal{D}$ can be interpolated from a smaller set of inverses. However, the inverse of the polynomial matrix $\mathbf{H}(s_k) = \sum_{l=0}^{L-1} \mathbf{H}_l \, s_k^{-l}$ will not be polynomial unless $\mathbf{H}(s_k)$ is unimodular, i.e., $\det \mathbf{H}(s_k) = \alpha \, s_k^{-n}$ with $n \in \mathbb{N}_0$ and $\alpha \in \mathbb{C}$ an arbitrary constant [16], a condition that is generally not satisfied in practice. We can, therefore, conclude that interpolating the $\mathbf{H}^{-1}(s_k)$ for $k \in \mathcal{D}$ from a smaller set of inverses will, in general, not be possible.

The 2×2 Case: We can, however, exploit the fact that the entries in $\mathbf{H}^{-1}(s_k)$ are rational functions in s_k to formulate an interpolation-based efficient inversion algorithm. Consider, as a simple example, the 2×2 matrices

$$\mathbf{H}(s_k) = \begin{bmatrix} h_{11}(s_k) & h_{12}(s_k) \\ h_{21}(s_k) & h_{22}(s_k) \end{bmatrix}.$$

Assuming that the $\mathbf{H}(s_k)$ are nonsingular, we have

$$\mathbf{H}^{-1}(s_k) = \frac{1}{h_{11}(s_k)h_{22}(s_k) - h_{12}(s_k)h_{21}(s_k)} \times \begin{bmatrix} h_{22}(s_k) & -h_{12}(s_k) \\ -h_{21}(s_k) & h_{11}(s_k) \end{bmatrix}.$$

It is now obvious that both the numerator and denominator of each entry in $\mathbf{H}^{-1}(s_k)$ are polynomial in s_k^{-1}, and can, therefore, be interpolated. This simple observation constitutes the key concept of the new inversion algorithms presented in this paper.

General Case: For an arbitrary number of antennas, we can write

$$\mathbf{H}^{-1}(s_k) = \frac{\operatorname{adj} \mathbf{H}(s_k)}{\det \mathbf{H}(s_k)}. \tag{5}$$

Recall that the determinant of a square $M \times M$ matrix $\mathbf{A}$ is defined recursively by the Laplace expansion

$$\det \mathbf{A} = \sum_{j=1}^{M} (-1)^{i+j} a_{ij} \det \mathbf{A}_{ij} \tag{6}$$

for an arbitrary row $1 \leq i \leq M$ (w.l.o.g., we have expanded the determinant along a row of $\mathbf{A}$). The $(M-1) \times (M-1)$ matrix $\mathbf{A}_{ij}$ is obtained by deleting the ith row and the jth column of $\mathbf{A}$, and $a_{ij} = [\mathbf{A}]_{ij}$. The determinant $\det \mathbf{A}_{ij}$ is a *minor* of $\mathbf{A}$. In the following, we call minors corresponding to $m \times m$ submatrices *m-minors*. We furthermore note that

$$[\operatorname{adj}(\mathbf{A})]_{ji} = (-1)^{i+j} \det \mathbf{A}_{ij}. \tag{7}$$

Realizing that the determinant of a polynomial matrix is polynomial, we can conclude from (5) and (7) that the inverses $\mathbf{H}^{-1}(s_k)$ can be obtained by separately interpolating $\operatorname{adj} \mathbf{H}(s_k)$ and $\det \mathbf{H}(s_k)$. We note that this observation has been made previously in a general context in [17].

The order of an m-minor of $\mathbf{H}(s_k)$ is $L_m - 1$, where

$$L_m = m(L-1) + 1.$$

We can therefore conclude that the number of base points needed for interpolating $\operatorname{adj} \mathbf{H}(s_k)$ and $\det \mathbf{H}(s_k)$ is given by $L_{M-1} \approx (M-1)L$ and $L_M \approx ML$, respectively. Once $\operatorname{adj} \mathbf{H}(s_k)$ and $\det \mathbf{H}(s_k)$ have been interpolated, an additional M^2 divisions need to be performed for each of the D tones. Since the denominator is constant for all matrix elements, alternatively a single division and M^2 multiplications can be performed per tone.

A simple interpolation-based inversion algorithm is described in the following, where $\mathcal{B}_m \subseteq \mathcal{D}$ with $|\mathcal{B}_m| = L_m$ denoting an (arbitrarily chosen) set of base points used for interpolation of m-minors. The base points furthermore have to satisfy the nesting property $\mathcal{B}_{m-1} \subset \mathcal{B}_m$ for $m = 2, 3, \ldots, M$.

Algorithm II-A—Adjoint/Determinant Interpolation

1a) Interpolate $\mathbf{H}(s_k)$, $k \in \mathcal{P}$, to obtain $\mathbf{H}(s_k)$ for $k \in \mathcal{B}_{M-1}$
b) Compute $\operatorname{adj} \mathbf{H}(s_k)$ for $k \in \mathcal{B}_{M-1}$
c) Interpolate $\operatorname{adj} \mathbf{H}(s_k)$, $k \in \mathcal{B}_{M-1}$, to obtain $\operatorname{adj} \mathbf{H}(s_k)$ for $k \in \mathcal{D}$

2a) Pick an arbitrary row of $\mathbf{H}(s_k)$, $k \in \mathcal{P}$, and interpolate it to obtain $\mathbf{H}(s_k)$ for $k \in \mathcal{B}_M$
b) Compute $\det \mathbf{H}(s_k)$, $k \in \mathcal{B}_M$, via Laplace expansion (6) along the row selected in 2a) and using the adjoints obtained in 1c)
c) Interpolate $\det \mathbf{H}(s_k)$, $k \in \mathcal{B}_M$, to obtain $\det \mathbf{H}(s_k)$ for $k \in \mathcal{D}$

3) Evaluate (5) for $k \in \mathcal{D}$.

Note that the choice of the base points $k \in \mathcal{B}_m$ in the intermediate steps is in principle arbitrary; it can, however, have a significant impact on the computational complexity and accuracy of the interpolation filters.

The basic idea underlying algorithm II-A is to perform matrix inversion for approximately ML tones only and to compute the remaining inverses by interpolating $\operatorname{adj} \mathbf{H}(s_k)$ and $\det \mathbf{H}(s_k)$. The number of operations required for the interpolation of an $M \times M$ polynomial matrix scales as $\mathcal{O}(M^2)$, whereas the number of operations required for inversion or adjoint computation scales as $\mathcal{O}(M^3)$. Therefore, roughly speaking, complexity savings over brute force inversion can be obtained if $D \gtrsim ML$, which is the case in many practical

systems. An in-depth complexity analysis, provided in the next section, will corroborate this statement and provide a detailed quantitative complexity comparison.

Based on the observation that computing $\mathrm{adj}\,\mathbf{H}(s_k)$ in step 1b) of algorithm II-A at L_{M-1} base points is potentially very costly, we suggest a modified version of algorithm II-A. The key idea of this modification is to realize that the minors of lower order in the recursive (Laplace) expansion of $\mathrm{adj}\,\mathbf{H}(s_k)$ are polynomials of lower order. It is, therefore, sufficient to compute these minors at a smaller number of base points and to subsequently interpolate the results to a larger number of base points.

Before describing an algorithm that exploits this fact, we make an additional important observation: An $M \times M$ matrix has $\binom{M}{m}^2$ different m-minors. However, when computing the adjoint (or determinant) recursively (through Laplace expansion), not all minors need to be computed at every level m. By proper choice of the rows along which the individual m-minors are expanded, potentially only a subset of the $\binom{M}{m-1}^2$ different $(m-1)$-minors is needed (see [13] for details). Denoting the size of this subset for the mth level as R_m, Table I lists the R_m for the practically relevant cases $M = 2, 3, \ldots, 6$.

We can now summarize the modification of algorithm II-A as:

Algorithm II-B—Space-Frequency Interpolation

1) Interpolate $\mathbf{H}(s_k)$, $k \in \mathcal{P}$, to obtain $\mathbf{H}(s_k)$ for $k \in \mathcal{B}_{M-1}$
2) Compute R_2 2-minors for each $k \in \mathcal{B}_2$. Set $m \leftarrow 2$
3a) If $m = M-1$, go to step 5). Otherwise, Interpolate the m-minors to all tones $k \in \mathcal{B}_{m+1}$
b) Compute R_{m+1} $(m+1)$-minors for each $k \in \mathcal{B}_{m+1}$ using Laplace expansion with the m-minors obtained in 3a) and the $\mathbf{H}(s_k)$ obtained in 1). Set $m \leftarrow m+1$ and go to step 3a)
4) Interpolate $\mathrm{adj}\,\mathbf{H}(s_k)$, $k \in \mathcal{B}_{M-1}$, computed in 3), to obtain $\mathrm{adj}\,\mathbf{H}(s_k)$ for $k \in \mathcal{D}$
5) Continue with steps 2) and 3) of algorithm II-A.

Algorithm II-B employs a nested procedure of minor computation and interpolation and is hence called "space-frequency interpolation".

Inversion by the Leverrier-Fadeev Algorithm: We finally note that there is a large and well-studied class of algorithms for computing the inverse of a polynomial matrix. These algorithms are essentially variations of the Leverrier-Fadeev algorithm for joint determinant and adjoint computation of polynomial matrices (see [18] and references therein). However, Leverrier-Fadeev type algorithms are not suited for the problem at hand due to their high computational complexity and poor numerical properties.

TABLE I

NUMBER OF m-MINORS NEEDED IN COMPUTING THE ADJOINT AND DETERMINANT OF AN $M \times M$ MATRIX BY LAPLACE EXPANSION FOR $m = 2, 3, \ldots, M$ AND TOTAL COUNT OF MULTIPLICATIONS NEEDED FOR ADJOINT COMPUTATION USING LAPLACE EXPANSION

M	R_m (number of m-minors needed)					$c_{\mathrm{adj}}(M)$
	2	3	4	5	6	
2	1					0
3	9	1				18
4	12	16	1			72
5	20	30	25	1		230
6	30	60	45	36	1	600

IV. COMPLEXITY ANALYSIS

So far, we have not been specific about the complexity savings of algorithms II-A and II-B over brute-force inversion (algorithm I). The goal of this section is to introduce a relevant complexity metric, based on which the algorithms described in the previous section will be compared.

A. The Complexity Metric

The definition of a relevant computational complexity metric is, in practice, highly dependent on the platform used for implementation, e.g., a general purpose processor, a digital signal processor (DSP), or a VLSI implementation of a full-custom application specific IC (ASIC). The requirements on high-rate MIMO-OFDM receivers will, in general, mandate the use of dedicated VLSI implementations. We shall, therefore, employ a VLSI-specific computational complexity metric.

Among the basic arithmetic operations in VLSI architectures, the operations governing the overall computational cost are *full complex multiplications*[1] (with two variable operands) and, to a much lesser extent, *multiplications of one variable operand with constant coefficients* (such as, e.g., filter coefficients) [19]. Our complexity metric will therefore be based on the count of these two types of operations only.

The three different algorithms described in the previous section essentially involve the computation of matrix inverses, adjoints, and determinants, as well as interpolation operations.

Interpolation Cost: Interpolation can be accomplished by multiplications with constant coefficients since the interpolation filter coefficients can be precomputed. Moreover, interpolation filters can be implemented efficiently using polyphase structures and a multistage approach [20]. We furthermore assume that the interpolation effort (in terms of number of multiplications) is independent of the number of base points and depends on the number of target points to be interpolated only. This can be achieved by adjusting the filter length to the number of base points, i.e., reducing the filter length for

[1] In fact, divisions are significantly more costly than full multiplications. However, we will show later that in our case the impact of divisions on the total computational cost can be neglected.

increasing number of base points and vice versa. In order to account for the differences between full multiplications and multiplications by a constant, required for interpolation, we quantify the interpolation cost through an equivalent of c_{IP} full multiplications per target tone. In the remainder of the paper, whenever we speak of (computational) cost, we shall mean the equivalent number of full multiplications.

Matrix Inversion Cost: In the following, we denote the cost of computing the adjoint of an $M \times M$ matrix $\mathbf{A}$ by $c_{\mathrm{adj}}(M)$. Once $\mathrm{adj}\,\mathbf{A}$ is known, $\det \mathbf{A}$ can be computed using M multiplications. Computation of $\mathbf{A}^{-1}$ subsequently requires one division and M^2 further multiplications (cf. Section III-B). We emphasize that taking into account the M^2 full multiplications for the cost of inverting the $\mathbf{H}(s_k)$ can be seen as a worst-case analysis, since these multiplications can often be absorbed into the processing stage following the spatial separation step. The cost for the division can be made negligible at the expense of processing delay and will, therefore, be ignored in our complexity analysis. As a result, it follows that the cost of inverting an $M \times M$ matrix using (5) is given by $c_{\mathrm{adj}}(M) + M^2 + M$.

The number of multiplications needed to compute the adjoint of an $M \times M$ matrix through Laplace expansion can be verified to scale in 2^M. More specifically, with R_m in Table I, we obtain

$$c_{\mathrm{adj}}(M) = \sum_{m=2}^{M-1} m R_m.$$

Algorithms I and II-A do not specify the method used for the computation of $\mathrm{adj}\,\mathbf{H}(s_k)$. For large M, the exponential complexity of the Laplace expansion approach becomes prohibitive, so that one will have to resort to a method for adjoint computation with cubic complexity in M. Algorithm II-B, however, implicitly employs the Laplace expansion and can be expected to yield savings for small M only. We shall see later that algorithm II-B yields cost savings over algorithm II-A when M is small and the cost of interpolation is low.

B. Complexity of the Algorithms

We are now in a position to compute the total cost of the three different algorithms described in Section III. Before presenting our detailed results, we note that step 1) in all three algorithms, strictly speaking, performs filtering for noise reduction in the oversampled case ($P > L$) and interpolation in the critically sampled case ($P = L$). In the oversampled case, the values at the base points therefore need to be recomputed; this is not necessary in the critically sampled case. For the sake of simplicity, we shall ignore the corresponding difference in interpolation cost (LM^2). Effectively, this will lead to a slight overestimation of the computational cost in the critically sampled case.

The overall cost of algorithm I is now given by

$$C_{\mathrm{I}} = D\big(c_{\mathrm{adj}}(M) + M^2 + M\big) + DM^2\, c_{\mathrm{IP}}. \tag{8}$$

For the cost of algorithm II-A, we obtain

$$\begin{aligned} C_{\text{II-A}} &= L_{M-1}\, c_{\mathrm{adj}}(M) + L_M M + DM^2 \\ &\quad + \big(DM^2 + D - 1\big)\, c_{\mathrm{IP}} \\ &= \big((M-1)(L-1)+1\big)\, c_{\mathrm{adj}}(M) \\ &\quad + M^2(L-1) + M + DM^2 \\ &\quad + \big(DM^2 + D - 1\big)\, c_{\mathrm{IP}} \\ &\approx ML\,\big(c_{\mathrm{adj}}(M) + M\big) + DM^2 + DM^2\, c_{\mathrm{IP}}. \end{aligned} \tag{9}$$

We can now immediately see the crucial difference between algorithm I and algorithm II-A. Comparing (8) and (9), we observe that the cost for computing the D adjoints (which dominates complexity for large M) in algorithm II-A is proportional to ML and independent of D. The cost for interpolation is approximately the same in both algorithms. In summary, we have essentially reduced the cost for computing D adjoint matrices of size $M \times M$ to the cost of computing ML adjoint matrices of size $M \times M$. We can thus conclude that algorithm II-A will yield savings over algorithm I if $D \gtrsim ML$. For a given number of antennas M, the savings increase with decreasing L.

Finally, the cost of algorithm II-B is given by

$$\begin{aligned} C_{\text{II-B}} &= \sum_{m=2}^{M} m R_m\, L_m + DM^2 \\ &\quad + \left(DM^2 + D - 1 + \sum_{m=2}^{M-2} R_m (L_{m+1} - L_m)\right) c_{\mathrm{IP}}. \end{aligned}$$

Compared to algorithm II-A, algorithm II-B trades inversion cost for interpolation cost. Since the interpolation cost in algorithm II-B is higher than in algorithm II-A, savings can be expected for small c_{IP}.

C. Numerical Results

Based on the results in the previous subsection, we shall next quantify the cost savings of the new algorithms II-A and II-B over brute-force inversion (algorithm I). For the sake of simplicity and concreteness, we assume the parameters of an IEEE 802.16a system [4]: The number of data-carrying tones (out of a total of $N = 256$ tones) is $D = 200$, and the length of the cyclic prefix L_{CP} and hence the maximum value for L is 8, 16, 32, or 64. Note that although individual channel realizations may be shorter than L_{CP}, the system has to be designed for $L = L_{\mathrm{CP}}$ since otherwise dynamic channel length adaptation has to be performed, which is not desirable in practice. For the remainder of this section, we furthermore assume that algorithms I and II-A employ the Laplace expansion for computing the adjoint.

Extent of Complexity Reduction Possible: We start by investigating the fundamental savings of type-II algorithms over brute-force inversion by assuming that the cost of carrying out the filtering operation for interpolation is negligible compared to the cost of computing the adjoints and determinants. For that purpose, we assume for now that $c_{\mathrm{IP}} = 0$. Considering type-II algorithms is sensible only for those parameter settings

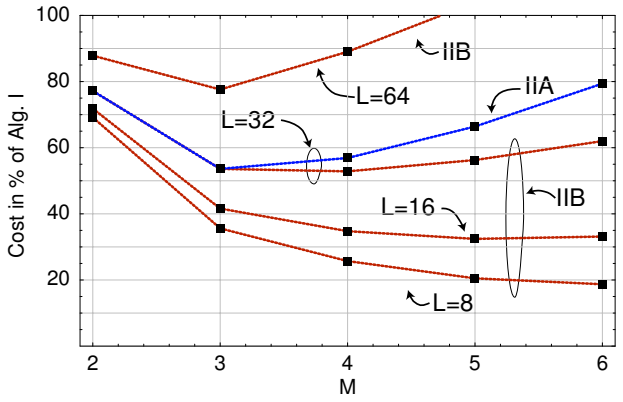

Fig. 1. Cost of algorithm II-B (as percentage of cost of algorithm I) for $D = 200$ tones and varying number of channel taps. Interpolation cost is not taken into account. Cost for algorithm II-A is shown only for $L = 32$.

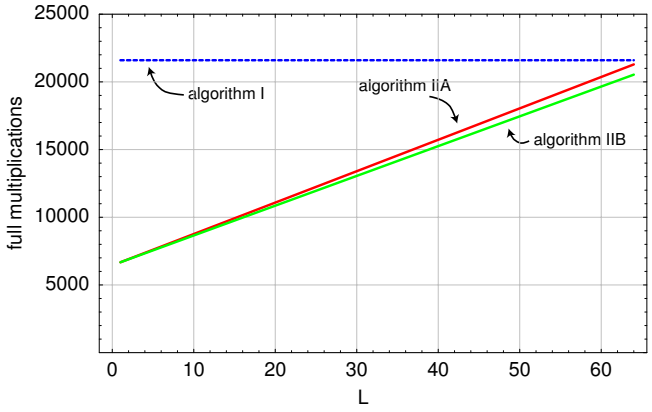

Fig. 2. Total cost for $D = 200$ tones as a function of the number of channel taps L, $c_{\mathrm{IP}} = 1$

that lead to cost savings for $c_{\mathrm{IP}} = 0$. The true savings will always be lower when the cost of interpolation is taken into account.

The results in Fig. 1 show the equivalent number of multiplications for type-II algorithms as percentage of the number of multiplications required by algorithm I (brute-force inversion). We can conclude from Fig. 1 that except for the $L = 64$ case at matrix sizes $M \geq 5$, for all parameter settings in the IEEE 802.16 standard a significant complexity reduction can be obtained by using an interpolation-based inversion algorithm. The complexity savings of type-II algorithms are more pronounced for smaller L, which can be attributed to the fact that the degree of the m-minors decreases with decreasing L. The savings of algorithm II-B over algorithm II-A are more pronounced for larger values of M. This effect is a consequence of the large-M complexity of algorithm II-A being dominated by computing the adjoints in step 1b), which is carried out more efficiently in algorithm II-B.

Case Study for $M = 4$: In order to quantify the savings of type-II algorithms in terms of total cost, including the interpolation cost, it is instructive to consider a fixed matrix

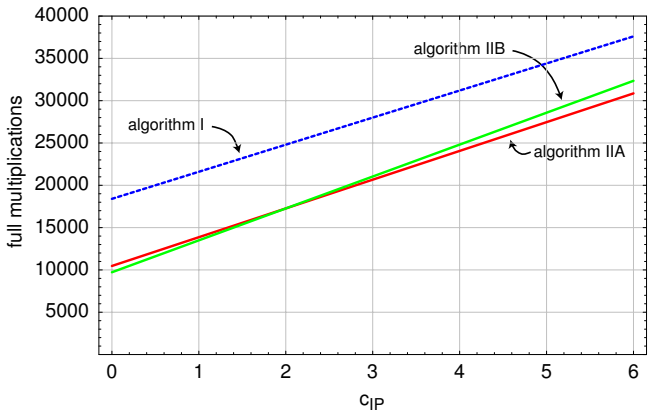

Fig. 3. Impact of interpolation complexity on total cost of channel inversion for a 4×4 system with $D = 200$ tones and $L = 32$ channel taps

size. For $M = 4$, Fig. 2 shows the total cost as a function of L for the three algorithms under consideration. The equivalent cost of interpolation was assumed to be $c_{\mathrm{IP}} = 1$. We can see that the type-II (interpolation-based) algorithms outperform the brute-force inversion algorithm over the entire range of L. The savings of type-II algorithms amount to more than 60 % for $L = 8$ and become negligible for $L = 64$.

Impact of Interpolation Complexity: The relative complexity of algorithms II-A and II-B depends on the interpolation cost, as does the relative amount of cost savings attainable with type-II algorithms over algorithm I. In practice, one can trade interpolation accuracy for computational complexity by employing longer interpolation filters, resulting in improved interpolation accuracy (and increased c_{IP}). In Fig. 3, still assuming $M = 4$, we show the computational cost as a function of c_{IP}. It is evident that the more sophisticated algorithm II-B outperforms the simpler algorithm II-A only at low interpolation costs. In fact, it can be shown that algorithm II-A consistently outperforms algorithm II-B for $c_{\mathrm{IP}} > 2$ (provided that $M = 4$), regardless of D and L. The reason for this behavior lies in the fact that algorithm II-A does not need intermediate interpolation steps, although it performs superfluous computations (the 2-minors are computed L_3 times instead of L_2 times). However, for $c_{\mathrm{IP}} > 2$, the interpolation savings of algorithm II-A with respect to algorithm II-B more than compensate for this overhead. Moreover, in our example with $M = 4$, the cost advantage of algorithm II-B is negligible, so that in practice it is tolerable to consistently use the simpler algorithm II-A.

The advantage of algorithm II-B compared to algorithm II-A at low c_{IP} only begins to be significant for $M \geq 4$. When M increases further, the Laplace expansion-based computation of the adjoint will eventually become inefficient (due to the exponential complexity in M), so that algorithm II-B will cease to be attractive from a complexity point of view, regardless of c_{IP}; instead, algorithm II-A in conjunction with cubic complexity adjoint computation should be used.

V. Conclusions and Outlook

We proposed a new class of computationally efficient algorithms for inverting the channel matrices in zero-forcing MIMO-OFDM receivers. Depending on the number of channel taps and the number of antennas, significant complexity reductions can be realized. For the parameter settings used in the IEEE 802.16 standard, savings of up to 80 % over brute-force tone-by-tone inversion can be obtained. Our analysis also shows that for IEEE 802.11a/g-based systems, the complexity reductions are not significant for $M > 2$.

The general idea reported in this paper is applicable to nonsquare MIMO configurations, minimum mean-squared error (MMSE) receivers, and preprocessing for sphere decoding [13]. An important topic for further research is an investigation of the tradeoff between interpolation filter complexity and matrix inversion accuracy (and consequently bit error rate performance).

Acknowledgment

The authors would like to thank A. Burg for many inspiring discussions.

References

[1] L. J. Cimini, "Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing," *IEEE Trans. Commun.*, vol. 33, no. 7, pp. 665–675, July 1985.

[2] B. LeFloch, M. Alard, and C. Berrou, "Coded orthogonal frequency division multiplex," *Proc. of IEEE*, vol. 83, no. 6, pp. 982–996, June 1995.

[3] *Supplement to IEEE standard for information technology — Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications — high-speed physical layer in the 5 GHz band*, IEEE Std. 802.11a, Rev. 1999.

[4] *IEEE standard for local and metropolitan area networks — Part 16: Air interface for fixed broadband wireless access systems*, IEEE Std. 802.16-2004, Rev. 2004.

[5] I. E. Telatar, "Capacity of multi-antenna Gaussian channels," *European Transactions on Telecommunications*, vol. 10, pp. 585–595, Nov./Dec. 1999.

[6] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Tech. J.*, pp. 41–59, Autumn 1996.

[7] V. Tarokh, N. Seshadri, and A. R. Calderbank, "Space-time codes for high data rate wireless communication: Performance criterion and code construction," *IEEE Trans. Inform. Theory*, vol. 44, no. 2, pp. 744–765, Mar. 1998.

[8] A. J. Paulraj, R. U. Nabar, and D. A. Gore, *Introduction to Space-Time Wireless Communications*. Cambridge, UK: Cambridge Univ. Press, 2003.

[9] G. G. Raleigh and J. M. Cioffi, "Spatio-temporal coding for wireless communication," *IEEE Trans. Commun.*, vol. 46, no. 3, pp. 357–366, 1998.

[10] H. Bölcskei, D. Gesbert, and A. J. Paulraj, "On the capacity of OFDM-based spatial multiplexing systems," *IEEE Trans. Commun.*, vol. 50, no. 2, pp. 225–234, Feb. 2002.

[11] G. J. Foschini, G. D. Golden, R. A. Valenzuela, and P. W. Wolniansky, "Simplified processing for high spectral efficiency wireless communication employing multi-antenna arrays," *IEEE J. Select. Areas Commun.*, vol. 17, no. 11, pp. 1841–1852, Nov. 1999.

[12] E. Viterbo and J. Boutros, "A universal lattice decoder for fading channels," *IEEE Trans. Signal Processing*, vol. 44, no. 7, pp. 1657–1668, July 1996.

[13] M. Borgmann, H. Bölcskei, and D. Cescato, "Interpolation-based efficient channel preprocessing for MIMO-OFDM receivers," *IEEE Trans. Signal Processing*, in preparation.

[14] W. Zhu and M. P. Fitz, "Adaptive Wiener interpolation channel estimation for pilot symbol assisted MIMO-OFDM in low mobility environment," in *Proc. 38th Asilomar Conf. Signals, Syst., Computers*, Pacific Grove, CA, Nov. 2004.

[15] M. Borgmann and H. Bölcskei, "Pragmatic channel estimation for MIMO-OFDM," in preparation.

[16] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Englewood Cliffs (NJ): Prentice Hall, 1993.

[17] A. Schuster and P. Hippe, "Inversion of polynomial matrices by interpolation," *IEEE Trans. Automat. Contr.*, vol. 37, no. 3, pp. 363–365, Mar. 1992.

[18] S. Barnett, "Algebraic manipulation of polynomial matrices: A survey," *IMA J. Math. Contr. Inform.*, vol. 7, no. 3, pp. 249–256, 1990.

[19] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford, UK: Oxford University Press, 2000.

[20] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*. Englewood Cliffs (NJ): Prentice Hall, 1983.