Fast Matrix Inversion Updates for Massive MIMO Detection and Precoding

Francisco Rosário, *Student Member, IEEE,* Francisco A. Monteiro, *Member, IEEE,* and António Rodrigues, *Member, IEEE*

Abstract

In this letter, methods and corresponding complexities for fast matrix inversion updates in the context of massive multiple-input multiple-output (MIMO) are studied. In particular, we propose an on-the-fly method to recompute the zero forcing (ZF) filter when a user is added or removed from the system. Additionally, we evaluate the recalculation of the inverse matrix after a new channel estimation is obtained for a given user. Results are evaluated numerically in terms of bit error rate (BER) using the Neumann series approximation as the initial inverse matrix. It is concluded that, with fewer operations, the performance after an update remains close to the initial one.

Index terms - massive MIMO, ZF, matrix inversion lemma, Neumann series.

I. Introduction

Multiple-input multiple-output (MIMO) and its extension to very large arrays have been a trending topic of research in the past few years. The theoretical advantages of massive MIMO systems are clear: increased spectral capacity while attaining high energy efficiencies [1]. However, with the increase of the number of dimensions, using conventional MIMO algorithms may not be suitable any more in terms of computational efficiency and new methods must emerge.

Exploiting the massive MIMO effect, denoted in the literature as *channel hardening*, it is possible to use low complexity methods, whilst attaining near optimal performance [2]. Namely, linear detection and

F. Rosário and A. Rodrigues are with Instituto de Telecomunicações, and Instituto Superior Técnico, Universidade de Lisboa, Portugal, e-mails: {francisco.rosario@tecnico.ulisboa.pt , ar@lx.it.pt}.

F. A. Monteiro* is with Instituto de Telecomunicações, and ISCTE - Instituto Universitário de Lisboa, Portugal, e-mail: {francisco.monteiro@lx.it.pt}.

This work was funded by FCT (Foundation for Science and Technology) and Instituto de Telecomunicações under project UID/EEA/50008/2013.

precoding techniques are proven to achieve close to maximum diversity. Energy constrained applications, such as base stations (BS) and full-duplex relays, can effectively take advantage of this fact to serve a large number of users simultaneously [3], [4].

Implementation complexity and hardware feasibility have always been carefully evaluated when designing platform targeted algorithms. Without compromising the performance of the system, low complexity methods to retrieve the information from the received signal should be looked for. For these reasons, [5] recently proposed the use of the Neumann series for a fast and efficient approximate inversion method of matrices. In the same work, it has been demonstrated that, as long as the number of BS antennas is much larger than the number of users, block error rates (BLER) similar to the ones with an exact inverse can be achieved, while reducing one order of magnitude in terms of required computations.

In this letter, capitalizing on the results obtained so far in the literature with the Neumann series, we propose and evaluate methods based on the matrix inversion lemma to update the inverse, when a user is added or removed from the system. Similar matrix deflation methods to reduce complexity have been proposed in [6], [7] in the context of V-BLAST systems. Additionally, we provide a method to recompute the inverse when a single user channel estimation has changed. All of the proposed algorithms require a low number of calculations and are memory transfer friendly. This reduction in computation time might be specially useful not only in slow fading environments with a dynamic set of active of users[1], but also in fast fading channels where new channel estimations are obtained very frequently. Using the Neumann series inverse approximation as the input, the impact of the algorithms in the bit error rate (BER) is evaluated and compared via numerical simulations.

II. System Model

Consider a large scale uplink multiuser MIMO system with N antennas at the BS and $M < N$ single antenna users. Each user transmits a symbol from an m-QAM constellation set $\mathcal{A}$ with average power σ_x^2. The resulting transmit vector is denoted by $\mathbf{x} = [x_1, x_2, ..., x_M]^T$, has correlation matrix $\mathbb{E}(\mathbf{x}\mathbf{x}^H) = \sigma_x^2\mathbf{I}$ and is filtered by a channel matrix $\mathbf{H} \in \mathbb{C}^{N \times M}$, whose entries satisfy $\mathcal{CN}(0, 1)$. At the receiver, additive white Gaussian noise (AWGN) is added and the received vector $\mathbf{y} \in \mathbb{C}^{N \times 1}$ can, hence, be expressed as:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \tag{1}$$

where $\mathbf{n} \sim \mathcal{CN}(0, \sigma_n^2)$. Under this model, the signal-to-noise ratio at the BS is defined as $\mathrm{SNR} = M\frac{\sigma_x^2}{\sigma_n^2}$. Perfect knowledge of $\mathbf{H}$ is assumed at the receiver.

[1] Note that performing resource allocation in cellular systems using massive MIMO basically amounts to admission control [8] (i.e., decide which users are active or inactive).

III. Linear Detection and Neumann Series

From the received signal, one is interested in performing hard decisions of $\mathbf{x}$, denoted by $\hat{\mathbf{x}}$. Note that channel coding techniques will not be considered in this paper; though, the results herein presented are straightforwardly extensible to coded transmissions. Further, the proposed methods are also applicable in linear precoding.

A large variety of massive MIMO detection algorithms exist in the literature [9]; however, under the assumption that $M << N$, linear detection methods are proven to perform close to optimal [2]. Then, the matrix inversion-dependent conventional zero forcing (ZF) filter will be considered for analysis purposes (extension to the better performing minimum mean-square error (MMSE) receiver is straightforward). To this end, define the Gram matrix $\mathbf{Z} = \mathbf{H}^H\mathbf{H} \in \mathbb{C}^{M \times M}$ and compute the ZF estimation as:

$$\hat{\mathbf{x}}_{\text{zf}} = \mathcal{Q}((\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H\mathbf{y}) = \mathcal{Q}(\mathbf{Z}^{-1}\mathbf{H}^H\mathbf{y}), \tag{2}$$

where $\mathcal{Q}(\cdot)$ is the quantizer operator to the nearest point in the constellation. Note that most of the computational burden in (2) is in computing the inverses of the Hermitian matrix $\mathbf{Z}$. In particular, the computation of $\mathbf{Z}^{-1}$ using exact inversion methods, such as Cholesky decomposition [10], [11], requires $\mathcal{O}(M^3)$ operations, which might be cumbersome to implement for the case where a large number of users are being served. Using the fact that in massive MIMO systems $\mathbf{Z}$ is an almost diagonal matrix, a hardware-efficient method based on the Neumann series was first proposed in [5] to approximate the required inverse in (2).

It has been proven in [3] that if one applies the decomposition of $\mathbf{Z}$ as $\mathbf{Z} = \mathbf{D} + \mathbf{E}$, where $\mathbf{D}$ is a diagonal matrix with the diagonal entries of $\mathbf{Z}$ and $\mathbf{E}$ the corresponding hollow, then the Neumann series to compute its inverse can be expressed as:

$$\tilde{\mathbf{Z}}_K^{-1} = \sum_{n=0}^{K-1} (-\mathbf{D}^{-1}\mathbf{E})^n \mathbf{D}^{-1}, \tag{3}$$

where K is the number of terms to be computed in the series and $\tilde{\mathbf{Z}}_K^{-1}$ is the K-term approximation of $\mathbf{Z}^{-1}$. Convergence of (3) is only guaranteed if the maximum modulus of eigenvalues of matrix $(\mathbf{I} - \mathbf{D}^{-1}\mathbf{Z})$ is less than 1 and, if this condition is satisfied, then approximation approaches equality as $K \to \infty$ [3]. Moreover, the lower the eigenvalues, the faster the convergence; which holds true when the ratio $\beta = N/M$ is high [12].

Neumann series is a low complexity iterative method; hence it is hardware friendly, unlike conventional inversion methods [13]. As an example, consider the approximation when $K = 3$:

$$\tilde{\mathbf{Z}}_3^{-1} = \underbrace{\mathbf{D}^{-1}}_{\mathbf{A}_0} - \underbrace{(\mathbf{D}^{-1}\mathbf{E})\mathbf{D}^{-1}}_{\mathbf{A}_1} + \underbrace{(\mathbf{D}^{-1}\mathbf{E})(\mathbf{D}^{-1}\mathbf{E}\mathbf{D}^{-1})}_{\mathbf{A}_2}, \tag{4}$$

where $\mathbf{A}_0$, $\mathbf{A}_1$, and $\mathbf{A}_2$ are the $n = 0, 1, 2$ terms in (3). The complexity involved in computing $\mathbf{A}_0$ is M divisions, whilst calculating $\mathbf{A}_1$ and $\mathbf{A}_2$ yields $3M^2 - 3M$ and $16M^3 - 2M$ real-valued multiplications, respectively (these values exploit the existence of zeros in the diagonal and the fact that each of the partial results $\mathbf{A}$ has to be Hermitian). Even though the complexity of (3) scales with $\mathcal{O}(M^3)$ for $K \geq 3$ (same as exact inverse), its recursive nature is of much practical interest.

IV. Inverse Update

Algorithms to efficiently update the inverse of a matrix after a small perturbation are proposed here. The reason behind this study is simple: when a user is added or removed from the system, one is interested in recomputing the new inverse $\mathbf{Z}^{-1}$ in the least time possible in order to maximize data throughput. Having this in mind, it is possible, using the matrix inversion lemma, to decrease the number of computations from $\mathcal{O}(M^3)$ to $\mathcal{O}(M^2)$, hence increasing speed. Additionally, the update of $\mathbf{Z}^{-1}$ after a new channel estimation from a single user is obtained (which corresponds to a rank-1 perturbation in $\mathbf{H}$ and a rank-2 perturbation in $\mathbf{Z}$) is evaluated. Without recomputing the entire inverse, this can be achieved using the Sherman-Morrison formula (a special case of the matrix inversion lemma).

A. Adding and Removing a User

Assume that at a given time instant, the inverse $\mathbf{Z}^{-1} = (\mathbf{H}^H\mathbf{H})^{-1}$ is already computed (via exact inversion or Neumann series) and that a user is added to the system with estimated channel denoted by the column vector $\mathbf{h}_n$. Define the new inflated matrix as $\mathbf{H}_e = [\mathbf{H}\ \mathbf{h}_n]$ (the user is added in the last column but in fact it could have been added in any position). Hence, the new extended Gram matrix denoted by $\mathbf{Z}_e$ is given by:

$$\mathbf{Z}_e = \begin{bmatrix} \mathbf{H}^H \\ \mathbf{h}_n^H \end{bmatrix} \begin{bmatrix} \mathbf{H} & \mathbf{h}_n \end{bmatrix} = \begin{bmatrix} \mathbf{H}^H\mathbf{H} & \mathbf{H}^H\mathbf{h}_n \\ \mathbf{h}_n^H\mathbf{H} & \mathbf{h}_n^H\mathbf{h}_n \end{bmatrix}. \tag{5}$$

In order to find $\mathbf{Z}_e^{-1}$, we use the general result provided by the inverse of a partitioned matrix [14], which is written as follows (see [15, Appendix B]):

$$\begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{F}_{11}^{-1} & -\mathbf{F}_{11}^{-1}\mathbf{P}_{12}\mathbf{P}_{22}^{-1} \\ -\mathbf{P}_{22}^{-1}\mathbf{P}_{21}\mathbf{F}_{11}^{-1} & \mathbf{F}_{22}^{-1} \end{bmatrix}, \tag{6}$$

Algorithm 1 Update $\mathbf{Z}^{-1}$, when a user $\mathbf{h}_p$ is added to $\mathbf{H}$ at position p

```
**Input:** $\mathbf{Z}^{-1}, \mathbf{H}, \mathbf{h}_p$
$\mathbf{t}_1 \leftarrow \mathbf{H}^H \mathbf{h}_p$
$\mathbf{t}_2 \leftarrow \mathbf{Z}^{-1}\mathbf{t}_1$                                  $\triangleright\ 4M^2$
$c \leftarrow 1/(\mathbf{h}_p^H \mathbf{h}_p - \mathbf{t}_1^H \mathbf{t}_2)$                  $\triangleright\ 4M+1$
$\mathbf{t}_3 \leftarrow c\mathbf{t}_2$                                       $\triangleright\ 2M$
$\mathbf{F}_{11}^{-1} \leftarrow \mathbf{Z}^{-1} + c\mathbf{t}_2\mathbf{t}_2^H$                    $\triangleright\ 3M^2+3M$
$\mathbf{Z}_e^{-1} \leftarrow \begin{bmatrix} \mathbf{F}_{11}^{-1} & -\mathbf{t}_3 \\ -\mathbf{t}_3^H & c \end{bmatrix}$
Change last column and last row of $\mathbf{Z}_e^{-1}$ to column $p$ and row $p$
**Output:** $\mathbf{Z}_e^{-1}$
```

where

$$\mathbf{F}_{11} = \mathbf{P}_{11} - \mathbf{P}_{12}\mathbf{P}_{22}^{-1}\mathbf{P}_{21}; \tag{7}$$

$$\mathbf{F}_{22} = \mathbf{P}_{22} - \mathbf{P}_{21}\mathbf{P}_{11}^{-1}\mathbf{P}_{12}. \tag{8}$$

Using the result provided above, $\mathbf{Z}_e^{-1}$ is expressed as [16, Lemma 11.1]:

$$\mathbf{Z}_e^{-1} = \begin{bmatrix} \mathbf{H}^H\mathbf{H} & \mathbf{H}^H\mathbf{h}_n \\ \mathbf{h}_n^H\mathbf{H} & \mathbf{h}_n^H\mathbf{h}_n \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{F}_{11}^{-1} & -c\mathbf{Z}^{-1}\mathbf{H}^H\mathbf{h}_n \\ -c\mathbf{h}_n^H\mathbf{H}\mathbf{Z}^{-H} & c \end{bmatrix}, \tag{9}$$

where $c = 1/(\mathbf{h}_n^H\mathbf{h}_n - \mathbf{h}_n\mathbf{H}\mathbf{Z}^{-1}\mathbf{H}^H\mathbf{h}_n)$ and

$$\mathbf{F}_{11}^{-1} = \mathbf{Z}^{-1} + c\mathbf{Z}^{-1}\mathbf{H}^H\mathbf{h}_n\mathbf{h}_n^H\mathbf{H}\mathbf{Z}^{-H}. \tag{10}$$

Therefore, it is possible to update the inverse of a matrix $\mathbf{Z} = \mathbf{H}^H\mathbf{H}$ when a column is added to $\mathbf{H}$ at position p without explicitly recomputing $\mathbf{Z}_e^{-1}$ or $\tilde{\mathbf{Z}}_{e,K}^{-1}$. The method is summarized in Algorithm 1 and, for comparison purposes, the corresponding number of required real-valued multiplications and divisions is shown on the right hand side (computation of both $\mathbf{H}^H\mathbf{h}_p$ and $\mathbf{h}_p^H\mathbf{h}_p$ were not considered since they are also required in the recomputation of $\mathbf{D}$ and $\mathbf{E}$ in the Neumann series). If the initial $\mathbf{Z}^{-1}$ is the exact inverse, then $\mathbf{Z}_e^{-1}$ is also exact. On the other hand, if an approximation $\tilde{\mathbf{Z}}_K^{-1}$ is used, then a propagation of errors is to be expected. Intuitively, the greater the error, the greater the degradation in performance after the update.

Based on the above results, a similar approach can be conducted when a column from $\mathbf{H}$ is removed. Decompose the original matrix as $\mathbf{H} = [\mathbf{H}_r\ \mathbf{h}_n]$, where $\mathbf{H}_r$ is the deflated matrix from which we want to compute the inverse $\mathbf{Z}_r^{-1} = (\mathbf{H}_r^H\mathbf{H}_r)^{-1}$ and $\mathbf{h}_n$ the column corresponding to user n and that is to be removed. The initial, already computed, inverse matrix $\mathbf{Z}^{-1}$ is given by

$$\mathbf{Z}^{-1} = \begin{bmatrix} \mathbf{H}_r^H \mathbf{H}_r & \mathbf{H}_r^H \mathbf{h}_n \\ \mathbf{h}_n^H \mathbf{H}_r & \mathbf{h}_n^H \mathbf{h}_n \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{F}_{11}^{-1} & -c\mathbf{u} \\ -c\mathbf{u}^H & c \end{bmatrix}, \tag{11}$$

where $\mathbf{u} = \mathbf{Z}_r^{-1}\mathbf{H}_r^H\mathbf{h}_n$. Noting that $\mathbf{F}_{11}^{-1} = \mathbf{Z}_r^{-1} + c\mathbf{u}\mathbf{u}^H$ (from (10)), it is straightforward that

$$\mathbf{Z}_r^{-1} = \mathbf{F}_{11}^{-1} - c\mathbf{u}\mathbf{u}^H. \tag{12}$$

The step-by-step set of operations to compute (12) is outlined in Algorithm 2 (MATLAB indexing notation is used).

B. Updating a User

Formerly, the situations where users are added or removed from the system were considered. Likewise, an update to the inverse when a new channel estimation for user p is obtained can be studied. This operation corresponds to replacing column p of $\mathbf{H}$ with the new estimation $\mathbf{h}_p$. After $\mathbf{H}$ is updated with the new estimation $\mathbf{h}_p$, expressed as $\mathbf{H}_u$, the new product $\mathbf{Z}_u = \mathbf{H}_u^H\mathbf{H}_u$ corresponds to altering column p and row p in the original $\mathbf{Z}$. This rank-2 perturbation can, however, be decomposed into two rank-1 perturbations. Defining a rank-1 perturbation by $\mathbf{a}\mathbf{b}^H$, one can use Sherman-Morrison formula to recompute the inverse after the update, as follows [7]:

$$(\mathbf{Z} + \mathbf{a}\mathbf{b}^H)^{-1} = \mathbf{Z}^{-1} - \frac{(\mathbf{Z}^{-1}\mathbf{a})(\mathbf{b}^H\mathbf{Z}^{-1})}{(1 + \mathbf{b}^H\mathbf{Z}^{-1}\mathbf{a})}. \tag{13}$$

Taking into account (13) and without recomputing the inverse, calculating $\mathbf{Z}_u$ from the previous $\mathbf{Z}$ is feasible. For that purpose, define $\mathbf{z}_{u,p} = \mathbf{H}_u^H\mathbf{h}_p$ and compute $\mathbf{a}_1 = \mathbf{z}_{u,p} - \mathbf{z}_p$, where $\mathbf{z}_p$ is the pth column of $\mathbf{Z}$. Further, make $\mathbf{b}_2 = \mathbf{a}_1^{(0,p)}$, where $\mathbf{a}_1^{(0,p)}$ is the same as $\mathbf{a}_1$ but with the pth entry zeroed. Then, computing $\mathbf{Z}_u^{-1}$ is straightforward:

$$\mathbf{Z}_t^{-1} = (\mathbf{Z} + \mathbf{a}_1\mathbf{e}_p^T)^{-1} = \mathbf{Z}^{-1} - \frac{(\mathbf{Z}^{-1}\mathbf{a}_1)(\mathbf{e}_p^T\mathbf{Z}^{-1})}{(1 + \mathbf{e}_p^T\mathbf{Z}^{-1}\mathbf{a}_1)}; \tag{14}$$

$$\mathbf{Z}_u^{-1} = (\mathbf{Z}_t + \mathbf{e}_p\mathbf{b}_2^H)^{-1} = \mathbf{Z}_t^{-1} - \frac{(\mathbf{Z}_t^{-1}\mathbf{e}_p)(\mathbf{b}_2^H\mathbf{Z}_t^{-1})}{(1 + \mathbf{b}_2^H\mathbf{Z}_t^{-1}\mathbf{e}_p)}, \tag{15}$$

where $\mathbf{e}_p$ is the pth column of the identity matrix $\mathbf{I}$. Equations (14) and (15) require $24M^2 + 8M$ scalar multiplications and 4 divisions; thus, when M is large, updating the inverse via this method might be advantageous over exact inversion in terms of complexity. Note that updating the inverse when only one column in $\mathbf{H}$ is changed can also be regarded as removing and adding a column sequentially (Algorithms 2 and 1). This study will be conducted in the numerical results.

Algorithm 2 Update $\mathbf{Z}^{-1}$, when a user $\mathbf{h}_p$ is removed from $\mathbf{H}$ at position p

```
Input: $\mathbf{Z}^{-1}$, $p$
Change column $p$ and row $p$ of $\mathbf{Z}^{-1}$ to last column and last row
$\mathbf{F}_{11}^{-1} \leftarrow \mathbf{Z}^{-1}(1:(M-1), 1:(M-1))$
$b \leftarrow \mathbf{Z}^{-1}(M, M)$
$\mathbf{t} \leftarrow -\mathbf{Z}^{-1}(1:(M-1), M)$
$\mathbf{Z}_r^{-1} \leftarrow \mathbf{F}_{11}^{-1} - \mathbf{t}\mathbf{t}^H/b$                $\triangleright\ 3M^2 + 3M$
Output: $\mathbf{Z}_r^{-1}$
```

C. *Complexity*

The involved number of real-valued multiplications and divisions to compute $\mathbf{A}_1$ and $\mathbf{A}_2$ and perform the former studied updates is expressed in Table I. It is worth emphasizing that all of the proposed update algorithms are quadratic in complexity. Moreover, if a recomputation of the term $\mathbf{A}_2$ after a small rank perturbation was to be performed, its complexity would still be $\mathcal{O}(M^3)$, since all of its entries would have to be recalculated.

V. Numerical Results

In this section, the impact in BER performance of the algorithms using the ZF estimates in (2) will be evaluated. The approximation $\tilde{\mathbf{Z}}_3^{-1}$ will be used as the initial input inverse matrix, since it provides a good trade-off between performance and complexity [3].

Denote the number of updates by U, corresponding to the number of added, removed or updated users after the last full inverse (in this case $\tilde{\mathbf{Z}}_3^{-1}$) was computed. The output updated inverse matrix after U sequential inflation or deflation operations is designated by $\mathbf{Z}_{n,U}^{-1} \in \mathbb{C}^{(M \pm U)\times(M \pm U)}$. For instance, $U = 2$ means that Algorithm 1 (or 2) was run twice and its input matrices were $\tilde{\mathbf{Z}}_3^{-1}$ and $\mathbf{Z}_{n,1}^{-1}$ for the first and second repetitions, respectively.

The initial number of antennas was set to $M = 8$ and $N = 80$, resulting in a ratio $\beta = 10$, which guarantees the convergence of (3) with very high probability [12]. Further, the constellation size was set to 64-QAM. For comparison purposes, performance using exact inverse and an entire recomputation of the Neumann series approximation before and after the update will also be exhibited. For the given setup ($K = 3$ and $\beta = 10$), a stalling effect in the BER (error floor) for high SNR using the Neumann series is to be expected due to the error associated with the approximation [3].

Fig. 1 depicts the performance after 1 and 2 users are added to the system ($U = 1$ and $U = 2$). The result shows that the degradation in performance when a user is added via matrix inversion lemma is close to negligible, even outperforming the case where the entire recalculation of $\tilde{\mathbf{Z}}_{n,3}^{-1}$ using the Neumann

Table I: Complexity of Neumann series and update operations.

Operation	Real-valued multiplications and divisions
Compute $\mathbf{A}_1$	$3M^2 - 3M$
Compute $\mathbf{A}_2$	$16M^3 - 2M$
Inflate (Alg. 1)	$7M^2 + 9M + 1$
Deflate (Alg. 2)	$3M^2 + 3M$
(14) and (15)	$24M^2 + 8M + 4$

series is done. Additionally, only an insignificant increase in BER is noticed from $U = 1$ to $U = 2$. Therefore, the advantages of the studied method are two-fold: not only inflating via Algorithm 1 achieves better performance than recomputing $\tilde{\mathbf{Z}}_{e,K}^{-1}$, but also a lower number of multiplications are required.

The case where U users are removed from the system is then depicted in Fig. 2. This time, BER performance using the Neumann series approximation is better if a recalculation from scratch is performed. This is related to the smaller error propagation associated with lower-dimensional matrices (higher ratio β). Nevertheless, the gain in complexity provided by Algorithm 2 might be exploited in applications where BER requirements are not too strict (for instance, practical standards recommend BLER=10^{-2} [5]). In addition, this degradation could have been mitigated if a better initial approximation ($K > 3$) of $\mathbf{Z}^{-1}$ had been used.

Finally, Fig. 3 depicts the situation after U different columns in $\mathbf{H}$ are changed (dimensions of the inverse matrices after the update remain the same). As can be inferred, the sequential operations in (14) and (15) using an approximation as initial inverse led to a propagation of errors, resulting in increased BER values. However, it is interesting to note that performing a deflation followed by an inflation (running Algorithms 2 and 1 sequentially) gets better results than computing a 3-term Neumann series from the start. This is explained with the vanished error contribution from “removed” columns in $\mathbf{H}$, since both algorithms return the exact inverse if the initial inverse matrix is also exact. Ultimately, if $U = M$ updates were to be performed, then an exact inverse would be attained regardless of the initial matrix (though, its cost would be similar to one of an exact inverse).

VI. Conclusions

Methods to efficiently compute and update the inverse of a matrix in large MIMO systems were evaluated in this letter. Using ZF linear detection and the Neumann series approximation, results show that, even after low complexity recalculations of the inverse, typical MIMO wireless application performance requirements are fulfilled. This is particularly true when a user is added to the system or a new channel estimation is obtained. The provided algorithms are suitable for hardware implementation and, hence, satisfy the high inversion matrix throughputs requirements of future wireless networks.

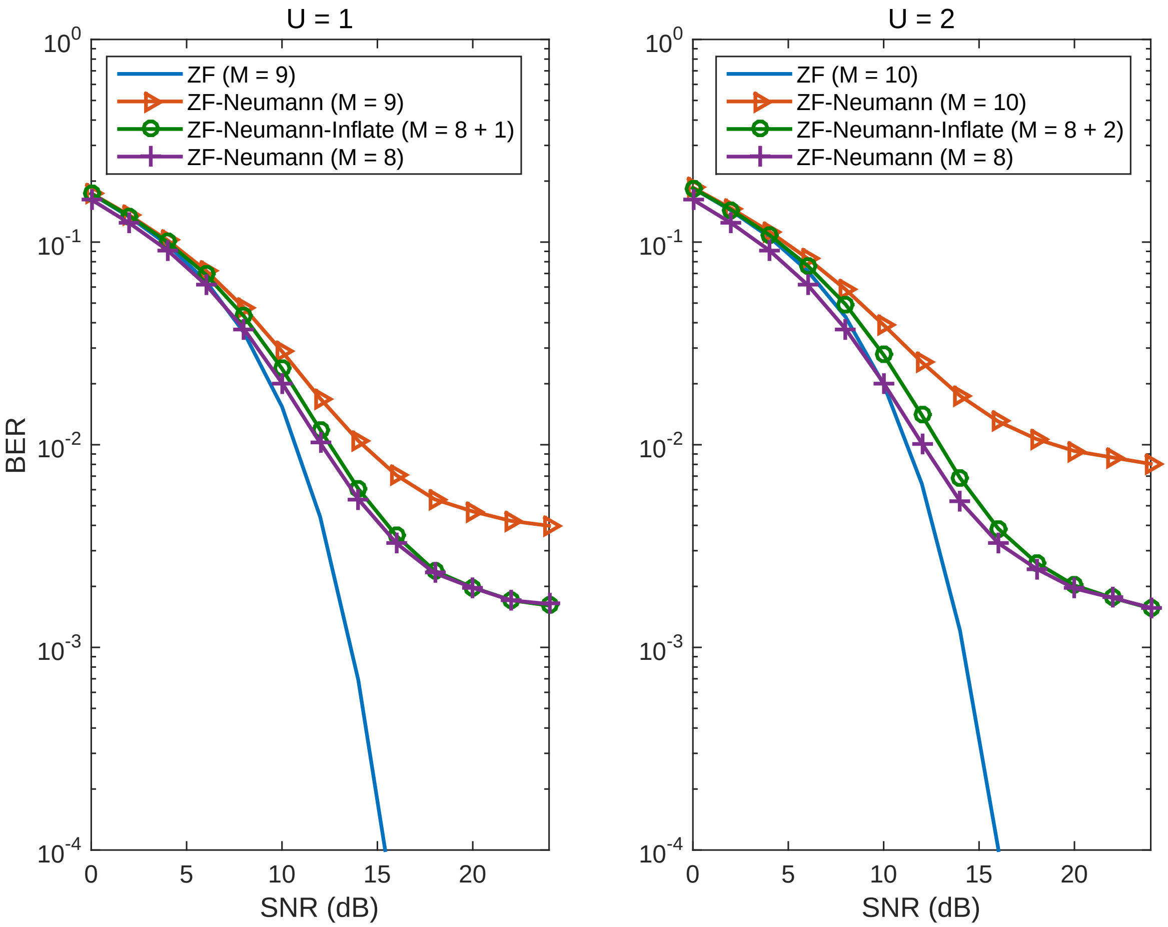

Fig. 1: BER performance comparison after U inflation updates.

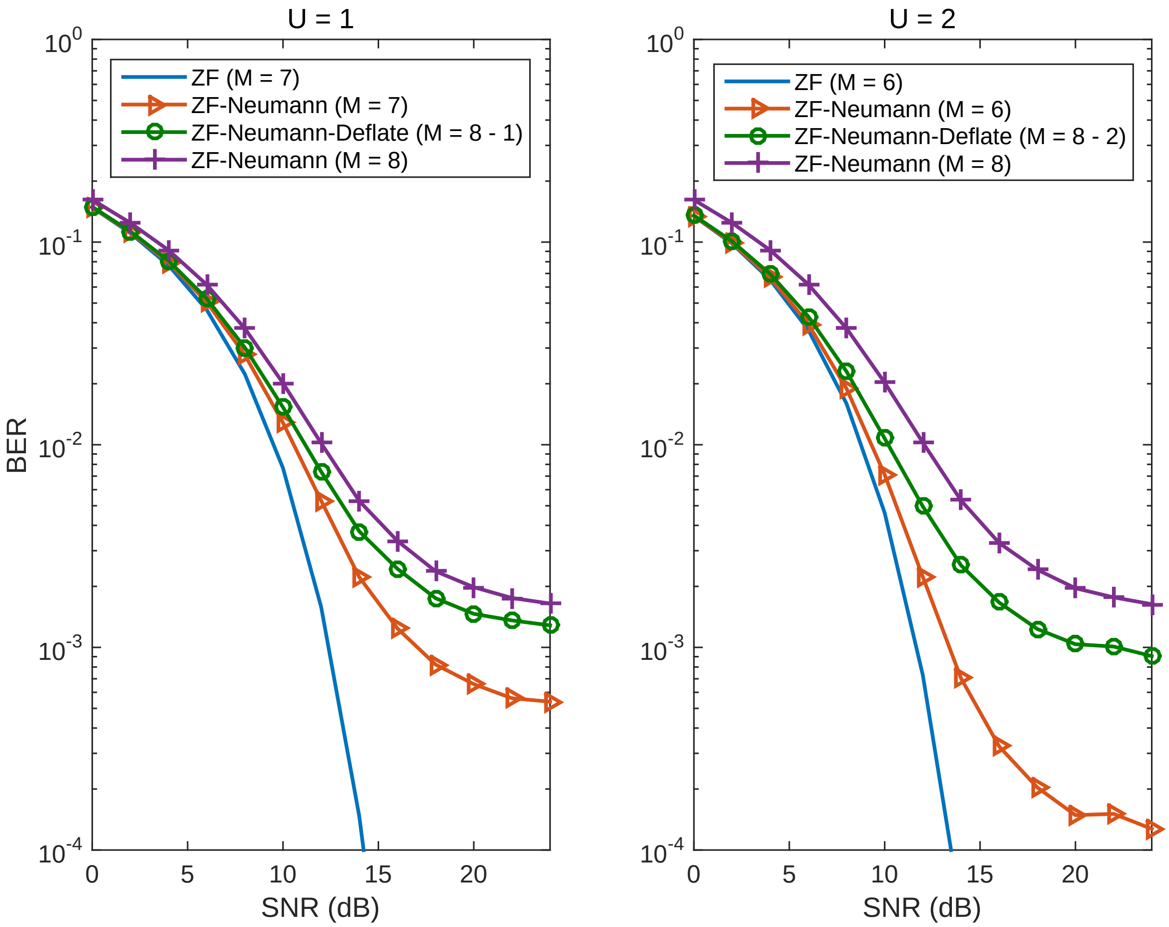

Fig. 2: BER performance comparison after U deflation updates.

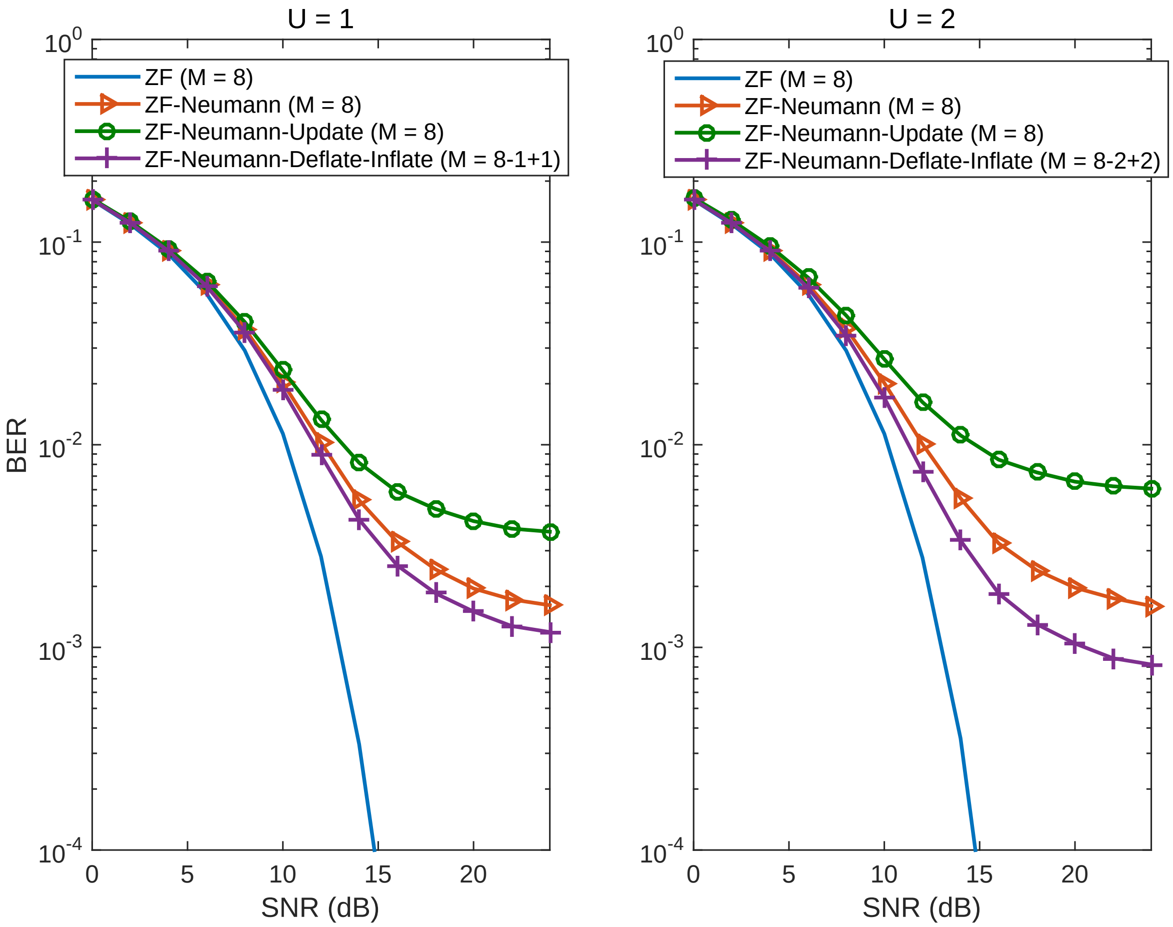

Fig. 3: BER performance comparison after U different column (i.e., channel) updates.

References

[1] E. Björnson, J. Hoydis, M. Kountouris, and M. Debbah, “Massive MIMO systems with non-ideal hardware: Energy efficiency, estimation, and capacity limits,” *IEEE Trans. Inform. Theory*, vol. 60, no. 11, pp. 7112–7139, 2014.

[2] H. Q. Ngo, E. G. Larsson, and T. L. Marzetta, “Energy and spectral efficiency of very large multiuser MIMO systems,” *IEEE Trans. Commun.*, vol. 61, no. 4, pp. 1436–1449, 2013.

[3] M. Wu, B. Yin, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer, “Large-scale MIMO detection for 3GPP LTE: Algorithms and FPGA implementations,” *IEEE J. Sel. Top. Sign. Proces.*, vol. 8, pp. 916–929, 2014.

[4] J. S. Lemos, F. Rosário, F. A. Monteiro, J. Xavier, and A. Rodrigues, “Massive MIMO full-duplex relaying with optimal power allocation for independent multipairs,” in *Proc. IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2015, pp. 306–310.

[5] M. Wu, B. Yin, A. Vosoughi, C. Studer, J. R. Cavallaro, and C. Dick, “Approximate matrix inversion for high-throughput data detection in the large-scale MIMO uplink,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013, pp. 2155–2158.

[6] Y. Shang and X.-G. Xia, “On fast recursive algorithms for V-BLAST with optimal ordered SIC detection,” *IEEE Trans. Wireless Commun.*, vol. 8, no. 6, pp. 2860–2865, 2009.

[7] L. Szczecinski and D. Massicotte, “Low complexity adaptation of MIMO MMSE receivers, implementation aspects,” in *Proc. IEEE Global Commun. Conference (GLOBECOM)*, vol. 4, 2005, pp. 2327–2332.

[8] E. Björnson, E. G. Larsson, and T. L. Marzetta, “Massive MIMO: Ten myths and one critical question,” to appear in *IEEE Commun. Mag.*, 2015. [Online]. Available: http://arxiv.org/abs/1503.06854

[9] X. Ma and Q. Zhou, “Massive MIMO and its detection,” in *MIMO Processing for 4G and Beyond: Fundamentals and Evolution*, M. M. da Silva and F. A. Monteiro, Eds. CRC Press, Taylor & Francis Group, 2014.

[10] A. Burian, J. Takala, and M. Ylinen, “A fixed-point implementation of matrix inversion using Cholesky decomposition,” in *Proc. IEEE 46th Midwest Symposium on Circuits and Systems (MWSCAS)*, vol. 3, 2003, pp. 1431–1434.

[11] A. Rontogiannis, V. Kekatos, and K. Berberidis, “A square-root adaptive V-BLAST algorithm for fast time-varying MIMO channels,” *IEEE Signal Process. Lett.*, vol. 13, no. 5, pp. 265–268, 2006.

[12] D. Zhu, B. Li, and P. Liang, “On the matrix inversion approximation based on Neumann series in massive MIMO systems,” in *Proc. IEEE International Conference on Communications (ICC)*, 2015, pp. 1763–1769.

[13] H. Prabhu, O. Edfors, J. Rodrigues, L. Liu, and F. Rusek, “Hardware efficient approximative matrix inversion for linear pre-coding in massive MIMO,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1700–1703.

[14] W. W. Hager, “Updating the inverse of a matrix,” *SIAM review*, vol. 31, no. 2, pp. 221–239, 1989.

[15] M. J. Beal, “Variational algorithms for approximate Bayesian inference,” Ph.D. dissertation, Gatsby Computational Neuroscience Unit, University College London, UK, 2003.

[16] D. Goldfarb, “Modification methods for inverting matrices and solving systems of linear algebraic equations,” *Mathematics of Computation*, vol. 26, no. 120, pp. 829–852, 1972.