$\|\boldsymbol{b}'_{\mathrm{opt}}\|_1$. Suppose that there exists a vector $\boldsymbol{h}$ that meets Conditions 1) and 2) of Theorem 5. It is clear that this vector $\boldsymbol{h}$ is dual feasible, and furthermore

$$\begin{aligned}\mathrm{Re}\langle \boldsymbol{s}, \boldsymbol{h}\rangle &= \mathrm{Re}\langle \boldsymbol{\Phi}\boldsymbol{b}'_{\mathrm{opt}}, \boldsymbol{h}\rangle \\ &= \mathrm{Re}\langle \boldsymbol{b}'_{\mathrm{opt}}, \boldsymbol{\Phi}^*\boldsymbol{h}\rangle \\ &= \mathrm{Re}\langle \boldsymbol{b}'_{\mathrm{opt}}, \mathrm{sgn}\, \boldsymbol{b}'_{\mathrm{opt}}\rangle \\ &= \|\boldsymbol{b}'_{\mathrm{opt}}\|_1.\end{aligned}$$

To see that $\boldsymbol{b}'_{\mathrm{opt}}$ uniquely solves (2), observe that the third equality can hold only if the support of $\boldsymbol{b}_{\mathrm{opt}}$ equals Λ_{opt}.

Acknowledgment

The author wishes to thank both anonymous referees for their insightful remarks.

References

[1] J. A. Tropp, “Greed is good: Algorithmic results for sparse approximation,” *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[2] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic decomposition by basis pursuit,” *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1999.

[3] D. L. Donoho and X. Huo, “Uncertainty principles and ideal atomic decomposition,” *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2845–2862, Nov. 2001.

[4] M. Elad and A. M. Bruckstein, “A generalized uncertainty principle and sparse representation in pairs of bases,” *IEEE Trans. Inf. Theory*, vol. 48, no. 9, pp. 2558–2567, Sep. 2002.

[5] D. L. Donoho and M. Elad, “Maximal sparsity representation via ℓ_1 minimization,” *Proc. Natl. Acad. Sci.*, vol. 100, pp. 2197–2202, Mar. 2003.

[6] R. Gribonval and M. Nielsen, “Sparse representations in unions of bases,” *IEEE Trans. Inf. Theory*, vol. 49, no. 12, pp. 3320–3325, Dec. 2003.

[7] J.-J. Fuchs, “On sparse representations in arbitrary redundant bases,” *IEEE Trans. Inf. Th.*, vol. 50, no. 6, pp. 1341–1344, Jun. 2004.

[8] R. Gribonval and M. Nielsen, “On the Exponential Convergence of Matching Pursuits in Quasi-Incoherent Dictionaries,” Université de Rennes I, Rennes, France, IRISA Rep. 1619, 2004.

Sum Power Iterative Water-Filling for Multi-Antenna Gaussian Broadcast Channels

Nihar Jindal, *Member, IEEE*, Wonjong Rhee, *Member, IEEE*,
Sriram Vishwanath, *Member, IEEE*, Syed Ali Jafar, *Member, IEEE*,
and Andrea Goldsmith, *Fellow, IEEE*

***Abstract*—In this correspondence, we consider the problem of maximizing sum rate of a multiple-antenna Gaussian broadcast channel (BC). It was recently found that dirty-paper coding is capacity achieving for this channel. In order to achieve capacity, the optimal transmission policy (i.e., the optimal transmit covariance structure) given the channel conditions and power constraint must be found. However, obtaining the optimal transmission policy when employing dirty-paper coding is a computationally complex nonconvex problem. We use duality to transform this problem into a well-structured convex multiple-access channel (MAC) problem. We exploit the structure of this problem and derive simple and fast iterative algorithms that provide the optimum transmission policies for the MAC, which can easily be mapped to the optimal BC policies.**

***Index Terms*—Broadcast channel, dirty-paper coding, duality, multiple-access channel (MAC), multiple-input multiple-output (MIMO), systems.**

I. Introduction

In recent years, there has been great interest in characterizing and computing the capacity region of multiple-antenna broadcast (downlink) channels. An achievable region for the multiple-antenna downlink channel was found in [3], and this achievable region was shown to achieve the sum rate capacity in [3], [10], [12], [16], and was more recently shown to achieve the full capacity region in [14]. Though these results show that the general dirty-paper coding strategy is optimal, one must still optimize over the transmit covariance structure (i.e., how transmissions over different antennas should be correlated) in order to determine the optimal transmission policy and the corresponding sum rate capacity. Unlike the single-antenna broadcast channel (BC), sum capacity is not in general achieved by transmitting to a single user. Thus, the problem cannot be reduced to a point-to-point multiple-input multiple-output (MIMO) problem, for which simple expressions are known. Furthermore, the direct optimization for sum rate capacity is a computationally complex

Manuscript received July 21, 2004; revised December 15, 2004. The work of some of the authors was supported by the Stanford Networking Research Center. The material in this correspondence was presented in part at the International Symposium on Information Theory, Yokohama, Japan, June/July 2003, and at the Asilomar Conference on Signals, Systems, and Computers, Asilomar, CA , Nov. 2002. This work was initiated while all the authors were at Stanford University.

N. Jindal is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: nihar@ece.umn.edu).

W. Rhee is with the ASSIA, Inc., Redwood City, CA 94065 USA (e-mail: wonjong@dsl.stanford.edu).

S. Vishwanath is with the Department of Electrical and Computer Engineering, University of Texas, Austin, Austin, TX 78712 USA (e-mail: sriram@ece.utexas.edu).

S. A. Jafar is with Electronic Engineering and Computer Science, University of California, Irvine, Irvine, CA 92697-2625 USA (e-mail: syed@ece.uci.edu)

A. Goldsmith is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305-9515 USA (e-mail: andrea@systems.stanford.edu).

Communicated by M. Medard, Associate Editor for Communications.

Digital Object Identifier 10.1109/TIT.2005.844082

nonconvex problem. Therefore, obtaining the optimal rates and transmission policy is difficult.[1]

A duality technique presented in [7], [10] transforms the nonconvex downlink problem into a convex sum power *uplink* (multiple-access channel, or MAC) problem, which is much easier to solve, from which the optimal downlink covariance matrices can be found. Thus, in this correspondence we find efficient algorithms to find the sum capacity of the uplink channel, i.e., to solve the following convex optimization problem:

$$\max_{\{\boldsymbol{Q}_i\}_{i=1}^K : \boldsymbol{Q}_i \ge 0,\ \sum_{i=1}^K \mathrm{Tr}(\boldsymbol{Q}_i) \le P} \log \left| \boldsymbol{I} + \sum_{i=1}^K \boldsymbol{H}_i^\dagger \boldsymbol{Q}_i \boldsymbol{H}_i \right|. \tag{1}$$

In this sum power MAC problem, the users in the system have a joint power constraint instead of individual constraints as in the conventional MAC. As in the case of the conventional MAC, there exist standard interior point convex optimization algorithms [2] that solve (1). An interior point algorithm, however, is considerably more complex than our algorithms and does not scale well when there are large numbers of users. Recent work by Lan and Yu based on minimax optimization techniques appears to be promising but suffers from much higher complexity than our algorithms [8]. A steepest descent method was proposed by Viswanathan *et al.*, [13], and an alternative, dual decomposition based algorithm was proposed by Yu in [15]. The complexity of these two algorithms is on the same order as the complexity of the algorithms proposed here. However, we find our algorithms to converge more rapidly, and our algorithms are also considerably more intuitive than either of these approaches. In this correspondence, we exploit the structure of the sum capacity problem to obtain simple iterative algorithms for calculating sum capacity,[2] i.e., for computing (1). This algorithm is inspired by and is very similar to the iterative water-filling algorithm for the conventional individual power constraint MAC problem by Yu, Rhee, Boyd, and Cioffi [17].

This correspondence is structured as follows. In Section II, the system model is presented. In Section III, expressions for the sum capacity of the downlink and dual uplink channels are stated. In Section IV, the basic iterative water-filling algorithm for the MAC is proposed and proven to converge when there are only two receivers. In Sections VI and VII, two modified versions of this algorithm are proposed and shown to converge for any number of users. Complexity analyses of the algorithms are presented in Section VIII, followed by numerical results and conclusions in Sections IX and X, respectively.

II. System Model

We consider a K user MIMO Gaussian broadcast channel (abbreviated as MIMO BC) where the transmitter has M antennas and each receiver has N antennas.[3] The downlink channel is shown in Fig. 1 along with the *dual* uplink channel. The dual uplink channel is a K user multiple-antenna uplink channel (abbreviated as MIMO MAC) where each of the dual uplink channels is the conjugate transpose of the corresponding downlink channel. The downlink and uplink channel are mathematically described as

$$\boldsymbol{y}_i = \boldsymbol{H}_i \boldsymbol{x} + \boldsymbol{n}_i, \quad i = 1, \ldots, K \qquad \text{Downlink channel} \tag{2}$$

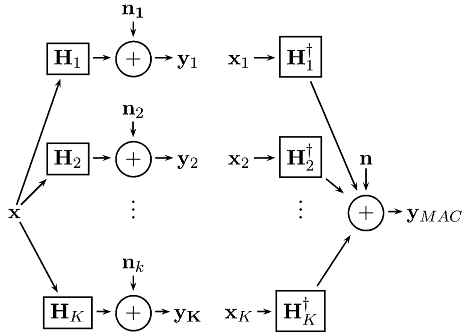


Fig. 1. System models of the MIMO BC (left) and the MIMO MAC (right) channels.

$$\boldsymbol{y}_{\mathrm{MAC}} = \sum_{i=1}^K \boldsymbol{H}_i^\dagger \boldsymbol{x}_i + \boldsymbol{n} \qquad \text{Dual uplink channel} \tag{3}$$

where $\boldsymbol{H}_1, \boldsymbol{H}_2, \ldots, \boldsymbol{H}_K$ are the channel matrices (with $\boldsymbol{H}_i \in \mathbb{C}^{N \times M}$) of Users 1 through K, respectively, on the downlink, the vector $\boldsymbol{x} \in \mathbb{C}^{M \times 1}$ is the downlink transmitted signal, and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K$ (with $\boldsymbol{x}_i \in \mathbb{C}^{N \times 1}$) are the transmitted signals in the uplink channel. This work applies only to the scenario where the channel matrices are fixed and are all known to the transmitter and to each receiver. In fact, this is the only scenario for which capacity results for the MIMO BC are known. The vectors $\boldsymbol{n}_1, \ldots, \boldsymbol{n}_K$ and $\boldsymbol{n}$ refer to independent additive Gaussian noise with unit variance on each vector component. We assume there is a sum power constraint of P in the MIMO BC (i.e., $E[||\boldsymbol{x}||^2] \le P$) and in the MIMO MAC (i.e., $\sum_{i=1}^K E[||\boldsymbol{x}_i||^2] \le P$). Though the computation of the sum capacity of the MIMO BC is of interest, we work with the dual MAC, which is computationally much easier to solve, instead.

Notation: We use boldface to denote vectors and matrices, and $\boldsymbol{H}^\dagger$ refers to the conjugate transpose (i.e., Hermitian) of the matrix $\boldsymbol{H}$. The function $[\cdot]_K$ is defined as

$$[x]_K \triangleq ((x-1) \bmod K) + 1$$

i.e., $[0]_K = K$, $[1]_K = 1$, $[K]_K = K$, and so forth.

III. Sum Rate Capacity

In [3], [10], [12], [16], the sum rate capacity of the MIMO BC (denoted as $\mathcal{C}_{\mathrm{BC}}(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_K, P)$) was shown to be achievable by dirty-paper coding [4]. From these results, the sum rate capacity can be written in terms of the following maximization:

$$\begin{aligned} &\mathcal{C}_{\mathrm{BC}}(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_K, P) \\ &\quad = \max_{\{\boldsymbol{\Sigma}_i\}_{i=1}^K : \boldsymbol{\Sigma}_i \ge 0,\ \sum_{i=1}^K \mathrm{Tr}(\boldsymbol{\Sigma}_i) \le P} \log \left| \boldsymbol{I} + \boldsymbol{H}_1 \boldsymbol{\Sigma}_1 \boldsymbol{H}_1^\dagger \right| \\ &\qquad + \log \frac{\left| \boldsymbol{I} + \boldsymbol{H}_2 (\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2) \boldsymbol{H}_2^\dagger \right|}{\left| \boldsymbol{I} + \boldsymbol{H}_2 \boldsymbol{\Sigma}_1 \boldsymbol{H}_2^\dagger \right|} + \cdots \\ &\qquad + \log \frac{\left| \boldsymbol{I} + \boldsymbol{H}_K (\boldsymbol{\Sigma}_1 + \cdots + \boldsymbol{\Sigma}_K) \boldsymbol{H}_K^\dagger \right|}{\left| \boldsymbol{I} + \boldsymbol{H}_K (\boldsymbol{\Sigma}_1 + \cdots + \boldsymbol{\Sigma}_{K-1}) \boldsymbol{H}_K^\dagger \right|}. \end{aligned} \tag{4}$$

The maximization is performed over downlink covariance matrices $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K$, each of which is an $M \times M$ positive semidefinite matrix. In this correspondence, we are interested in finding the covariance matrices that achieve this maximum. It is easily seen that the objective (4) is not a concave function of $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K$. Thus, numerically finding the maximum is a nontrivial problem. However, in [10], a *duality* is shown to exist between the uplink and downlink which establishes that the dirty paper rate region for the MIMO BC is equal to the capacity region of the dual MIMO MAC (described in (3)). This implies that

[1] In the single transmit antenna BC, there is a similar nonconvex optimization problem. However, it is easily seen that it is optimal to transmit with full power to only the user with the strongest channel. Such a policy is, however, not the optimal policy when the transmitter has multiple antennas.

[2] To compute other points on the boundary of the capacity region (i.e., non-sum-capacity rate vectors), the algorithms in either [13] or [8] can be used

[3] We assume all receivers have the same number of antennas for simplicity. However, all algorithms easily generalize to the scenario where each receiver can have a different number of antennas.

the sum capacity of the MIMO BC is equal to the sum capacity of the dual MIMO MAC (denoted as $\mathcal{C}_{\mathrm{MAC}}(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_K, P)$), i.e.,

$$\mathcal{C}_{\mathrm{BC}}(\boldsymbol{H}_1, \ldots, \boldsymbol{H}_K, P) = \mathcal{C}_{\mathrm{MAC}}(\boldsymbol{H}_1^\dagger, \ldots, \boldsymbol{H}_K^\dagger, P). \tag{5}$$

The sum rate capacity of the MIMO MAC is given by the following expression [10]:

$$\mathcal{C}_{\mathrm{MAC}}(\boldsymbol{H}_1^\dagger, \ldots, \boldsymbol{H}_K^\dagger, P) = \max_{\{\boldsymbol{Q}_i\}_{i=1}^K : \boldsymbol{Q}_i \geq 0, \; \sum_{i=1}^K \mathrm{Tr}(\boldsymbol{Q}_i) \leq P} \log \left| \boldsymbol{I} + \sum_{i=1}^K \boldsymbol{H}_i^\dagger \boldsymbol{Q}_i \boldsymbol{H}_i \right| \tag{6}$$

where the maximization is performed over uplink covariance matrices $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_K$ ($\boldsymbol{Q}_i$ is an $N \times N$ positive semidefinite matrix), subject to power constraint P. The objective in (6) is a concave function of the covariance matrices. Furthermore, in [10, eqs. 8–10], a transformation is provided (this mapping is reproduced in Appendix I for convenience) that maps from uplink covariance matrices to downlink covariance matrices (i.e., from $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_K$ to $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K$) that achieve the same rates and use the same sum power. Therefore, finding the optimal uplink covariance matrices leads directly to the optimal downlink covariance matrices.

In this correspondence, we develop specialized algorithms that efficiently compute (6). These algorithms converge, and utilize the water-filling structure of the optimal solution, first identified for the individual power constraint MAC in [17]. Note that the maximization in (6) is not guaranteed to have a unique solution, though uniqueness holds for nearly all channel realizations. See [17] for a discussion of this same property for the individual power constraint MAC. Therefore, we are interested in finding any maximizing solution to the optimization.

IV. Iterative Water-Filling With Individual Power Constraints

The iterative water-filling algorithm for the conventional MIMO MAC problem was obtained by Yu, Rhee, Boyd, and Cioffi in [17]. This algorithm finds the sum capacity of a MIMO MAC with *individual* power constraints $P_1, \ldots, P_K$ on each user, which is equal to

$$\mathcal{C}_{\mathrm{MAC}}(\boldsymbol{H}_1^\dagger, \ldots, \boldsymbol{H}_K^\dagger, P_1, \ldots, P_K) = \max_{\{\boldsymbol{Q}_i\}_{i=1}^K : \boldsymbol{Q}_i \geq 0, \; \mathrm{Tr}(\boldsymbol{Q}_i) \leq P_i} \log \left| \boldsymbol{I} + \sum_{i=1}^K \boldsymbol{H}_i^\dagger \boldsymbol{Q}_i \boldsymbol{H}_i \right|. \tag{7}$$

This differs from (6) only in the power constraint structure. Notice that the objective is a concave function of the covariance matrices, and that the constraints in (7) are *separable* because there is an individual trace constraint on each covariance matrix. For such problems, it is generally sufficient to optimize with respect to the first variable while holding all other variables constant, then optimize with respect to the second variable, etc., in order to reach a globally optimum point. This is referred to as the block-coordinate ascent algorithm and convergence can be shown under relatively general conditions [1, Sec. 2.7]. If we define the function $f(\cdot)$ as

$$f(\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_K) \triangleq \log \left| \boldsymbol{I} + \sum_{i=1}^K \boldsymbol{H}_i^\dagger \boldsymbol{Q}_i \boldsymbol{H}_i \right| \tag{8}$$

then in the $(n+1)$th iteration of the block-coordinate ascent algorithm

$$\boldsymbol{Q}_i^{(n+1)} \triangleq \arg \max_{\boldsymbol{Q}_i : \boldsymbol{Q}_i \geq 0, \; \mathrm{Tr}(\boldsymbol{Q}_i) \leq P_i} f\left(\boldsymbol{Q}_1^{(n)}, \ldots, \boldsymbol{Q}_{i-1}^{(n)}, \boldsymbol{Q}_i, \boldsymbol{Q}_{i+1}^{(n)}, \ldots, \boldsymbol{Q}_K^{(n)}\right) \tag{9}$$

for $i = [n]_K$ and $\boldsymbol{Q}_i^{(n+1)} = \boldsymbol{Q}_i^{(n)}$ for $i \neq [n]_K$. Notice that only one of the covariances is updated in each iteration.

The key to the iterative water-filling algorithm is noticing that $f(\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_K)$ can be rewritten as

$$\begin{aligned} f(\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_K) &= \log \left| \boldsymbol{I} + \sum_{j \neq i} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j \boldsymbol{H}_j + \boldsymbol{H}_i^\dagger \boldsymbol{Q}_i \boldsymbol{H}_i \right| \\ &= \log \left| \boldsymbol{I} + \sum_{j \neq i} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j \boldsymbol{H}_j \right| \\ &\quad + \log \left| \boldsymbol{I} + \left(\boldsymbol{I} + \sum_{j \neq i} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j \boldsymbol{H}_j \right)^{-1/2} \right. \\ &\quad \left. \times \boldsymbol{H}_i^\dagger \boldsymbol{Q}_i \boldsymbol{H}_i \left(\boldsymbol{I} + \sum_{j \neq i} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j \boldsymbol{H}_j \right)^{-1/2} \right| \end{aligned}$$

for any i, where we have used the property $|\boldsymbol{A}\boldsymbol{B}| = |\boldsymbol{A}||\boldsymbol{B}|$. Therefore, the maximization in (9) is equivalent to the calculation of the capacity of a point-to-point MIMO channel with channel $\boldsymbol{G}_i = \boldsymbol{H}_i \left(\boldsymbol{I} + \sum_{j \neq i} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j^{(n)} \boldsymbol{H}_j \right)^{-1/2}$, thus

$$\boldsymbol{Q}_i^{(n+1)} = \arg \max_{\boldsymbol{Q}_i : \boldsymbol{Q}_i \geq 0, \; \mathrm{Tr}(\boldsymbol{Q}_i) \leq P_i} \log \left| \boldsymbol{I} + \boldsymbol{G}_i^\dagger \boldsymbol{Q}_i \boldsymbol{G}_i \right|. \tag{10}$$

It is well known that the capacity of a point-to-point MIMO channel is achieved by choosing the input covariance along the eigenvectors of the channel matrix and by water-filling on the eigenvalues of the channel matrix [9]. Thus, $\boldsymbol{Q}_i^{(n+1)}$ should be chosen as a *water-fill* of the channel $\boldsymbol{G}_i$, i.e., the eigenvectors of $\boldsymbol{Q}_i^{(n+1)}$ should equal the left eigenvectors of $\boldsymbol{G}_i$, with the eigenvalues chosen by the water-filling procedure.

At each step of the algorithm, exactly one user optimizes his covariance matrix while treating the signals from all other users as noise. In the next step, the next user (in numerical order) optimizes his covariance while treating all other signals, including the updated covariance of the previous user, as noise. This intuitively appealing algorithm can easily be shown to satisfy the conditions of [1, Sec. 2.7] and thus provably converges. Furthermore, the optimization in each step of the algorithm simplifies to water-filling over an effective channel, which is computationally efficient.

If we let $\boldsymbol{Q}_1^*, \ldots, \boldsymbol{Q}_K^*$ denote the optimal covariances, then optimality implies

$$f(\boldsymbol{Q}_1^*, \ldots, \boldsymbol{Q}_K^*) = \max_{\boldsymbol{Q}_i : \boldsymbol{Q}_i \geq 0, \mathrm{Tr}(\boldsymbol{Q}_i) \leq P_i} f(\boldsymbol{Q}_1^*, \ldots, \boldsymbol{Q}_{i-1}^*, \boldsymbol{Q}_i, \boldsymbol{Q}_{i+1}^*, \ldots, \boldsymbol{Q}_K^*) \tag{11}$$

for any i. Thus, $\boldsymbol{Q}_1^*$ is a water-fill of the noise and the signals from all other users (i.e., is a waterfill of the channel $\boldsymbol{H}_1(\boldsymbol{I} + \sum_{j \neq 1} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j^* \boldsymbol{H}_j)^{-1/2}$), while $\boldsymbol{Q}_2^*$ is simultaneously a water-fill of the noise and the signals from all other users, and so forth. Thus, the sum capacity achieving covariance matrices *simultaneously* water-fill each of their respective effective channels [17], with the water-filling levels (i.e., the eigenvalues) of each user determined by the power constraints P_j. In Section V, we will see that similar intuition describes the sum capacity achieving covariance matrices in the MIMO MAC when there is a sum power constraint instead of individual power constraints.

V. Sum Power Iterative Water-Filling

In the previous section, we described the iterative water-filling algorithm that computes the sum capacity of the MIMO MAC subject to individual power constraints [17]. We are instead concerned with computing the sum capacity, along with the corresponding optimal covariance matrices, of a MIMO BC. As stated earlier, this is equivalent to computing the sum capacity of a MIMO MAC subject to a sum power constraint, i.e., computing (12) (see the bottom of the page). If we let $\boldsymbol{Q}_1^*, \ldots, \boldsymbol{Q}_K^*$ denote a set of covariance matrices that achieve the maximum in (12), it is easy to see that similar to the individual power constraint problem, each covariance must be a water-fill of the noise and signals from all other users. More precisely, this means that for every j, the eigenvectors of $\boldsymbol{Q}_i^*$ are aligned with the left eigenvectors of $\boldsymbol{H}_i(\boldsymbol{I} + \sum_{j \neq i} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j^* \boldsymbol{H}_j)^{-1/2}$ and that the eigenvalues of $\boldsymbol{Q}_i^*$ must satisfy the water-filling condition. However, since there is a *sum* power constraint on the covariances, the water level of all users must be equal. This is akin to saying that no advantage will be gained by transferring power from one user with a higher water-filling level to another user with a lower water-filling level. Note that this is different from the individual power constraint problem, where the water level of each user was determined individually and could differ from user to user. In the individual power constraint channel, since each user's water-filling level was determined by his own power constraint, the covariances of each user could be updated one at a time. With a sum power constraint, however, we must update all covariances *simultaneously* to maintain a constant water-level.

Motivated by the individual power algorithm, we propose the following algorithm in which all K covariances are simultaneously updated during each step, based on the covariance matrices from the previous step. This is a natural extension of the per-user sequential update described in Section IV. At each iteration step, we generate an effective channel for *each* user based on the covariances (from the previous step) of all other users. In order to maintain a common water-level, we simultaneously water-fill across all K effective channels, i.e., we maximize the sum of rates on the K effective channels. The nth iteration of the algorithm is described by the following.

1) Generate effective channels

$$\boldsymbol{G}_i^{(n)} = \boldsymbol{H}_i \left(\boldsymbol{I} + \sum_{j \neq i} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j^{(n-1)} \boldsymbol{H}_j \right)^{-1/2} \tag{13}$$

for $i = 1, \ldots, K$.

2) Treating these effective channels as parallel, noninterfering channels, obtain the new covariance matrices $\{\boldsymbol{Q}_i^{(n)}\}_{i=1}^K$ by water-filling with total power P

$$\left\{\boldsymbol{Q}_i^{(n)}\right\}_{i=1}^K = \arg \max_{\{\boldsymbol{Q}_i\}_{i=1}^K : \boldsymbol{Q}_i \geq 0,\ \sum_{i=1}^K \mathrm{Tr}(\boldsymbol{Q}_i) \leq P} \sum_{i=1}^K \log \left| \boldsymbol{I} + \left(\boldsymbol{G}_i^{(n)}\right)^\dagger \boldsymbol{Q}_i \boldsymbol{G}_i^{(n)} \right|.$$

This maximization is equivalent to water-filling the block diagonal channel with diagonals equal to $\boldsymbol{G}_1^{(n)}, \ldots, \boldsymbol{G}_K^{(n)}$. If the singular value decomposition (SVD) of $\boldsymbol{G}_i^{(n)}(\boldsymbol{G}_i^{(n)})^\dagger$ is written as

$$\boldsymbol{G}_i^{(n)} \left(\boldsymbol{G}_i^{(n)}\right)^\dagger = \boldsymbol{U}_i \boldsymbol{D}_i \boldsymbol{U}_i^\dagger$$

with $\boldsymbol{U}_i$ unitary and $\boldsymbol{D}_i$ square and diagonal, then the updated covariance matrices are given by

$$\boldsymbol{Q}_i^{(n)} = \boldsymbol{U}_i \boldsymbol{\Lambda}_i \boldsymbol{U}_i^\dagger \tag{14}$$

where $\boldsymbol{\Lambda}_i = \left[\mu \boldsymbol{I} - (\boldsymbol{D}_i)^{-1}\right]^+$ and the operation $[\boldsymbol{A}]^+$ denotes a component-wise maximum with zero. Here, the water-filling level μ is chosen such that $\sum_{i=1}^K \mathrm{Tr}(\boldsymbol{\Lambda}_i) = P$.

We refer to this as the *original algorithm* [6]. This simple and highly intuitive algorithm does in fact converge to the sum rate capacity when $K = 2$, as we show next.

Theorem 1: The sum power iterative water-filling algorithm converges to the sum rate capacity of the MAC when $K = 2$.

Proof: In order to prove convergence of the algorithm for $K = 2$, consider the following related optimization problem shown in (15) at the bottom of the page.We first show that the solutions to the original sum rate maximization problem in (12) and (15) are the same. If we define $\boldsymbol{A}_1 = \boldsymbol{B}_1 = \boldsymbol{Q}_1$ and $\boldsymbol{A}_2 = \boldsymbol{B}_2 = \boldsymbol{Q}_2$, we see that any sum rate achievable in (12) is also achievable in the modified sum rate in (15). Furthermore, if we define $\boldsymbol{Q}_1 = \frac{1}{2}(\boldsymbol{A}_1 + \boldsymbol{B}_1)$ and $\boldsymbol{Q}_2 = \frac{1}{2}(\boldsymbol{A}_2 + \boldsymbol{B}_2)$, we have

$$\begin{aligned}\log \left| \boldsymbol{I} + \boldsymbol{H}_1^\dagger \boldsymbol{Q}_1 \boldsymbol{H}_1 + \boldsymbol{H}_2^\dagger \boldsymbol{Q}_2 \boldsymbol{H}_2 \right| &\geq \frac{1}{2} \log \left| \boldsymbol{I} + \boldsymbol{H}_1^\dagger \boldsymbol{A}_1 \boldsymbol{H}_1 + \boldsymbol{H}_2^\dagger \boldsymbol{B}_2 \boldsymbol{H}_2 \right| \\ &\quad + \frac{1}{2} \log \left| \boldsymbol{I} + \boldsymbol{H}_1^\dagger \boldsymbol{B}_1 \boldsymbol{H}_1 + \boldsymbol{H}_2^\dagger \boldsymbol{A}_2 \boldsymbol{H}_2 \right|\end{aligned}$$

due to the concavity of $\log(\det(\cdot))$. Since

$$\mathrm{Tr}(\boldsymbol{Q}_1) + \mathrm{Tr}(\boldsymbol{Q}_2) = \frac{1}{2}\mathrm{Tr}(\boldsymbol{A}_1 + \boldsymbol{A}_2 + \boldsymbol{B}_1 + \boldsymbol{B}_2) \leq P$$

any sum rate achievable in (15) is also achievable in the original (12). Thus, every set of maximizing covariances $(\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}_1, \boldsymbol{B}_2)$ maps directly to a set of maximizing $(\boldsymbol{Q}_1, \boldsymbol{Q}_2)$. Therefore, we can equivalently solve (15) to find the uplink covariances that maximize the sum-rate expression in (12).

Now notice that the maximization in (15) has separable constraints on $(\boldsymbol{A}_1, \boldsymbol{A}_2)$ and $(\boldsymbol{B}_1, \boldsymbol{B}_2)$. Thus, we can use the block coordinate ascent method in which we maximize with respect to $(\boldsymbol{A}_1, \boldsymbol{A}_2)$ while holding $(\boldsymbol{B}_1, \boldsymbol{B}_2)$ fixed, then with respect to $(\boldsymbol{B}_1, \boldsymbol{B}_2)$ while holding $(\boldsymbol{A}_1, \boldsymbol{A}_2)$ fixed, and so on. The maximization of (15) with respect to $(\boldsymbol{A}_1, \boldsymbol{A}_2)$ can be written as

$$\max_{\boldsymbol{A}_1, \boldsymbol{A}_2 \geq 0,\ \mathrm{Tr}(\boldsymbol{A}_1 + \boldsymbol{A}_2) \leq P} \log \left| \boldsymbol{I} + \boldsymbol{G}_1^\dagger \boldsymbol{A}_1 \boldsymbol{G}_1 \right| + \log \left| \boldsymbol{I} + \boldsymbol{G}_2^\dagger \boldsymbol{A}_2 \boldsymbol{G}_2 \right| \tag{16}$$

where

$$\boldsymbol{G}_1 = \boldsymbol{H}_1 (\boldsymbol{I} + \boldsymbol{H}_2^\dagger \boldsymbol{B}_2 \boldsymbol{H}_2)^{-1/2}$$

and

$$\boldsymbol{G}_2 = \boldsymbol{H}_2 (\boldsymbol{I} + \boldsymbol{H}_1^\dagger \boldsymbol{B}_1 \boldsymbol{H}_1)^{-1/2}.$$

Clearly, this is equivalent to the iterative water-filling step described in the previous section where $(\boldsymbol{B}_1, \boldsymbol{B}_2)$ play the role of the covariance matrices from the previous step. Similarly, when maximizing with respect to $(\boldsymbol{B}_1, \boldsymbol{B}_2)$, the covariances $(\boldsymbol{A}_1, \boldsymbol{A}_2)$ are the covariance matrices from the previous step. Therefore, performing the cyclic coordinate ascent algorithm on (15) is equivalent to the sum power iterative water-filling algorithm described in Section V.

$$\mathcal{C}_{\mathrm{MAC}}(\boldsymbol{H}_1^\dagger, \ldots, \boldsymbol{H}_K^\dagger, P) = \max_{\{\boldsymbol{Q}_i\}_{i=1}^K : \boldsymbol{Q}_i \geq 0,\ \sum_{i=1}^K \mathrm{Tr}(\boldsymbol{Q}_i) \leq P} \log \left| \boldsymbol{I} + \sum_{i=1}^K \boldsymbol{H}_i^\dagger \boldsymbol{Q}_i \boldsymbol{H}_i \right|. \tag{12}$$

$$\max_{\boldsymbol{A}_1, \boldsymbol{A}_2 \geq 0,\ \boldsymbol{B}_1, \boldsymbol{B}_2 \geq 0,\ \mathrm{Tr}(\boldsymbol{A}_1 + \boldsymbol{A}_2) \leq P,\ \mathrm{Tr}(\boldsymbol{B}_1 + \boldsymbol{B}_2) \leq P} \frac{1}{2} \log \left| \boldsymbol{I} + \boldsymbol{H}_1^\dagger \boldsymbol{A}_1 \boldsymbol{H}_1 + \boldsymbol{H}_2^\dagger \boldsymbol{B}_2 \boldsymbol{H}_2 \right| + \frac{1}{2} \log \left| \boldsymbol{I} + \boldsymbol{H}_1^\dagger \boldsymbol{B}_1 \boldsymbol{H}_1 + \boldsymbol{H}_2^\dagger \boldsymbol{A}_2 \boldsymbol{H}_2 \right|. \tag{15}$$

$$\begin{tikzcd}
\mathbf{A}^{(0)} = \mathbf{Q}^{(-2)} \arrow[d, "\text{Update}"] & \mathbf{B}^{(0)} = \mathbf{Q}^{(-1)} \arrow[d] & \mathbf{C}^{(0)} = \mathbf{Q}^{(0)} \arrow[d] & \mathbf{Q}^{(1)} \triangleq \arg\max_{\mathbf{Q}} f^{exp}(\mathbf{Q}, \mathbf{Q}^{(-1)}, \mathbf{Q}^{(0)}) \\
\mathbf{A}^{(1)} = \mathbf{Q}^{(1)} \arrow[d] & \mathbf{B}^{(1)} = \mathbf{Q}^{(-1)} \arrow[d, "\text{Update}"] & \mathbf{C}^{(1)} = \mathbf{Q}^{(0)} \arrow[d] & \mathbf{Q}^{(2)} \triangleq \arg\max_{\mathbf{Q}} f^{exp}(\mathbf{Q}^{(1)}, \mathbf{Q}, \mathbf{Q}^{(0)}) \\
\mathbf{A}^{(2)} = \mathbf{Q}^{(1)} \arrow[d] & \mathbf{B}^{(2)} = \mathbf{Q}^{(2)} \arrow[d] & \mathbf{C}^{(2)} = \mathbf{Q}^{(0)} \arrow[d, "\text{Update}"] & \mathbf{Q}^{(3)} \triangleq \arg\max_{\mathbf{Q}} f^{exp}(\mathbf{Q}^{(1)}, \mathbf{Q}^{(2)}, \mathbf{Q}) \\
\mathbf{A}^{(3)} = \mathbf{Q}^{(1)} \arrow[d, "\text{Update}"] & \mathbf{B}^{(3)} = \mathbf{Q}^{(2)} \arrow[d] & \mathbf{C}^{(3)} = \mathbf{Q}^{(3)} \arrow[d] & \mathbf{Q}^{(4)} \triangleq \arg\max_{\mathbf{Q}} f^{exp}(\mathbf{Q}, \mathbf{Q}^{(2)}, \mathbf{Q}^{(3)}) \\
\mathbf{A}^{(4)} = \mathbf{Q}^{(4)} & \mathbf{B}^{(4)} = \mathbf{Q}^{(2)} & \mathbf{C}^{(4)} = \mathbf{Q}^{(3)} & \\
 & \bullet\bullet\bullet & &
\end{tikzcd}$$

Fig. 2. Graphical representation of Algorithm 1.

Furthermore, notice that each iteration is equal to the calculation of the capacity of a point-to-point (block-diagonal) MIMO channel. Water-filling is known to be optimal in this setting, and in Appendix II we show that the water-filling solution is the unique solution. Therefore, by [18, p. 228], [1, Ch. 2.7], the block coordinate ascent algorithm converges because at each step of the algorithm there is a unique maximizing solution. Thus, the iterative water-filling algorithm given in Section V converges to the maximum sum rate when $K = 2$. □

However, rather surprisingly, this algorithm does not always converge to the optimum when $K > 2$, and the algorithm can even lead to a strict decrease in the objective function. In Sections VI–IX, we provide modified versions of this algorithm that do converge for all K.

VI. Modified Algorithm

In this section, we present a modified version of the sum power iterative water-filling algorithm and prove that it converges to the sum capacity for any number of users K. This modification is motivated by the proof of convergence of the original algorithm for $K = 2$. In the proof of Theorem 1, a sum of two $\log\det$ functions, with four input covariances is considered instead of the original $\log\det$ function. We then applied the provably convergent cyclic coordinate ascent algorithm, and saw that this algorithm is in fact identical to the sum power iterative algorithm. When there are more than two users (i.e., $K > 2$) we can consider a similar sum of K $\log\det$ functions, and again perform the cyclic coordinate ascent algorithm to provably converge to the sum rate capacity. In this case, however, the cyclic coordinate ascent algorithm is not identical to the original sum power iterative water-filling algorithm. It can, however, be interpreted as the sum power iterative water-filling algorithm with a memory of the covariance matrices generated in the previous $K-1$ iterations, instead of just in the previous iteration.

For simplicity, let us consider the $K = 3$ scenario. Similar to the proof of Theorem 1, consider the following maximization:

$$\begin{aligned}\max \frac{1}{3}\log\left|\boldsymbol{I} + \boldsymbol{H}_1^\dagger \boldsymbol{A}_1 \boldsymbol{H}_1 + \boldsymbol{H}_2^\dagger \boldsymbol{B}_2 \boldsymbol{H}_2 + \boldsymbol{H}_3^\dagger \boldsymbol{C}_3 \boldsymbol{H}_3\right| \\ + \frac{1}{3}\log\left|\boldsymbol{I} + \boldsymbol{H}_1^\dagger \boldsymbol{C}_1 \boldsymbol{H}_1 + \boldsymbol{H}_2^\dagger \boldsymbol{A}_2 \boldsymbol{H}_2 + \boldsymbol{H}_3^\dagger \boldsymbol{B}_3 \boldsymbol{H}_3\right| \\ + \frac{1}{3}\log\left|\boldsymbol{I} + \boldsymbol{H}_1^\dagger \boldsymbol{B}_1 \boldsymbol{H}_1 + \boldsymbol{H}_2^\dagger \boldsymbol{C}_2 \boldsymbol{H}_2 + \boldsymbol{H}_3^\dagger \boldsymbol{A}_3 \boldsymbol{H}_3\right| \quad (17)\end{aligned}$$

subject to the constraints $\boldsymbol{A}_i \geq 0, \boldsymbol{B}_i \geq 0, \boldsymbol{C}_i \geq 0$ for $i = 1, 2, 3$ and

$$\begin{aligned}\mathrm{Tr}(\boldsymbol{A}_1 + \boldsymbol{A}_2 + \boldsymbol{A}_3) &\leq P \\ \mathrm{Tr}(\boldsymbol{B}_1 + \boldsymbol{B}_2 + \boldsymbol{B}_3) &\leq P \\ \mathrm{Tr}(\boldsymbol{C}_1 + \boldsymbol{C}_2 + \boldsymbol{C}_3) &\leq P.\end{aligned}$$

By the same argument used for the two-user case, any solution to the above maximization corresponds to a solution to the original optimization problem in (12). In order to maximize (17), we can again use the cyclic coordinate ascent algorithm. We first maximize with respect to $\boldsymbol{A} \triangleq (\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3)$, then with respect to $\boldsymbol{B} \triangleq (\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{B}_3)$, then with respect to $\boldsymbol{C} \triangleq (\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3)$, and so forth. As before, convergence is guaranteed due to the uniqueness of the maximizing solution in each step [1, Sec. 2.7]. In the two-user case, the cyclic coordinate ascent method applied to the modified optimization problem yields the same iterative water-filling algorithm proposed in Section V where the effective user of each channel is based on the covariance matrices only from the previous step. In general, however, the effective channel of each user depends on covariances which are up to $K-1$ steps old.

A graphical representation of the algorithm for three users is shown in Fig. 2. Here $\boldsymbol{A}^{(n)}$ refers to the triplet of matrices $(\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3)$ after the nth iterate. Furthermore, the function $f^{exp}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ refers to the objective function in (17). We begin by initializing all variables to some $\boldsymbol{A}^{(0)}$, $\boldsymbol{B}^{(0)}$, $\boldsymbol{C}^{(0)}$. In order to develop a more general form that generalizes to arbitrary K, we also refer to these variables as $\boldsymbol{Q}^{(-2)}$, $\boldsymbol{Q}^{(-1)}$, $\boldsymbol{Q}^{(0)}$. Note that each of these variables refers to a triplet of covariance matrices. In step 1, $\boldsymbol{A}$ is updated while holding variables $\boldsymbol{B}$ and $\boldsymbol{C}$ constant, and we define $\boldsymbol{Q}^{(1)}$ to be the updated variable $\boldsymbol{A}^{(1)}$

$$\begin{aligned}\boldsymbol{Q}^{(1)} &\triangleq \boldsymbol{A}^{(1)} \\ &= \arg \max_{\boldsymbol{Q}: \boldsymbol{Q}_i \geq 0,\ \sum_{i=1}^3 \mathrm{Tr}(\boldsymbol{Q}_i) \leq P} f^{exp}(\boldsymbol{Q}, \boldsymbol{B}^{(0)}, \boldsymbol{C}^{(0)}) \quad (18) \\ &= \arg \max_{\boldsymbol{Q}: \boldsymbol{Q}_i \geq 0,\ \sum_{i=1}^3 \mathrm{Tr}(\boldsymbol{Q}_i) \leq P} f^{exp}(\boldsymbol{Q}, \boldsymbol{Q}^{(-1)}, \boldsymbol{Q}^{(0)}). \quad (19)\end{aligned}$$

In step 2, the matrices $\boldsymbol{B}$ are updated with $\boldsymbol{Q}^{(2)} \triangleq \boldsymbol{B}^{(2)}$, and in step 3, the matrices $\boldsymbol{C}$ are updated with $\boldsymbol{Q}^{(3)} \triangleq \boldsymbol{C}^{(3)}$. The algorithm continues cyclically, i.e., in step 4, $\boldsymbol{A}$ is again updated, and so forth. Notice that $\boldsymbol{Q}^{(n)}$ is always defined to be the set of matrices updated in the nth iteration.

In Appendix III, we show that the following is a general formula for $\boldsymbol{Q}^{(n)}$ (see (20) and (21) at the top of the next page), where the effective channel of User i in the nth step is

$$\boldsymbol{G}_i^{(n)} = \boldsymbol{H}_i \left(\boldsymbol{I} + \sum_{j=1}^{K-1} \boldsymbol{H}_{[i+j]_K}^\dagger \boldsymbol{Q}_{[i+j]_K}^{(n-K+j)} \boldsymbol{H}_{[i+j]_K} \right)^{-1/2} \quad (22)$$

where $[x]_K = \mathrm{mod}((x-1), K) + 1$. Clearly, the previous $K-1$ states of the algorithm (i.e., $\boldsymbol{Q}^{(n-K+1)}, \ldots, \boldsymbol{Q}^{(n-1)}$) must be stored in memory in order to generate these effective channels.

$$\boldsymbol{Q}^{(n)} = \arg \max_{\boldsymbol{Q}:\boldsymbol{Q}_i \geq 0,\ \sum_{i=1}^{K} \mathrm{Tr}(\boldsymbol{Q}_i) \leq P} f^{exp}(\boldsymbol{Q}, \boldsymbol{Q}^{(n-K+1)}, \ldots, \boldsymbol{Q}^{(n-1)}) \tag{20}$$

$$= \arg \max_{\boldsymbol{Q}:\boldsymbol{Q}_i \geq 0,\ \sum_{i=1}^{K} \mathrm{Tr}(\boldsymbol{Q}_i) \leq P} \sum_{i=1}^{K} \log \left| \boldsymbol{I} + \left(\boldsymbol{G}_i^{(n)}\right)^\dagger \boldsymbol{Q}_i \boldsymbol{G}_i^{(n)} \right| \tag{21}$$

We now explicitly state the steps of Algorithm 1. The covariances are first initialized to scaled versions of the identity,[4] i.e., $\boldsymbol{Q}_j^{(n)} = \frac{P}{KN}\boldsymbol{I}$ for $j = 1, \ldots, K$ and $n = -(K-2), \ldots, 0$. The algorithm is almost identical to the original sum power iterative algorithm, with the exception that the expression for each effective channel now depends on covariance matrices generated in the previous $K-1$ steps, instead of just on the previous step.

1) Generate effective channels

$$\boldsymbol{G}_i^{(n)} = \boldsymbol{H}_i \left(\boldsymbol{I} + \sum_{j=1}^{K-1} \boldsymbol{H}_{[i+j]_K}^\dagger \boldsymbol{Q}_{[i+j]_K}^{(n-K+j)} \boldsymbol{H}_{[i+j]_K} \right)^{-1/2} \tag{23}$$

for $i = 1, \ldots, K$.

2) Treating these effective channels as parallel, noninterfering channels, obtain the new covariance matrices $\{\boldsymbol{Q}_i^{(n)}\}_{i=1}^K$ by water-filling with total power P

$$\left\{\boldsymbol{Q}_i^{(n)}\right\}_{i=1}^K = \arg \max_{\{\boldsymbol{Q}_i\}_{i=1}^K:\boldsymbol{Q}_i \geq 0,\ \sum_{i=1}^{K} \mathrm{Tr}(\boldsymbol{Q}_i) \leq P} \sum_{i=1}^{K} \log \times \left| \boldsymbol{I} + (\boldsymbol{G}_i^{(n)})^\dagger \boldsymbol{Q}_i \boldsymbol{G}_i^{(n)} \right|.$$

We refer to this as *Algorithm 1*. Next we prove convergence to the sum rate capacity:

Theorem 2: Algorithm 1 converges to the sum rate capacity for any K.

Proof: Convergence is shown by noting that the algorithm is the cyclic coordinate ascent algorithm applied to the function $f^{exp}(\cdot)$. Since there is a unique (water-filling) solution to the maximization in step 2, the algorithm converges to the sum capacity of the channel for any number of users K.[5] More precisely, convergence occurs in the objective of the expanded function

$$\lim_{n \to \infty} f^{exp}(\boldsymbol{Q}^{(n-K+1)}, \ldots, \boldsymbol{Q}^{(n)}) = \mathcal{C}_{\mathrm{MAC}}(\boldsymbol{H}_1^\dagger, \ldots, \boldsymbol{H}_K^\dagger, P). \tag{24}$$

Convergence is also easily shown in the original objective function $f(\cdot)$ because the concavity of the $\log(\det())$ function implies

$$f\left(\frac{1}{K} \sum_{l=n-K+1}^{n} \boldsymbol{Q}_1^{(l)}, \ldots, \frac{1}{K} \sum_{l=n-K+1}^{n} \boldsymbol{Q}_K^{(l)} \right) \geq f^{exp}\left(\boldsymbol{Q}^{(n-K+1)}, \ldots, \boldsymbol{Q}^{(n)}\right).$$

Thus, if we average over the covariances from the previous K iterations, we get

$$\lim_{n \to \infty} f\left(\frac{1}{K} \sum_{l=n-K+1}^{n} \boldsymbol{Q}_1^{(l)}, \ldots, \frac{1}{K} \sum_{l=n-K+1}^{n} \boldsymbol{Q}_K^{(l)} \right) = \mathcal{C}_{\mathrm{MAC}}(\boldsymbol{H}_1^\dagger, \ldots, \boldsymbol{H}_K^\dagger, P). \tag{25}$$

□

Though the algorithm does converge quite rapidly, the required memory is a drawback for large K. In Section VII, we propose an additional modification to reduce the required memory.

VII. Alternative Algorithm

In the preceding section, we described a convergent algorithm that requires memory of the covariance matrices generated in the previous $K-1$ iterations, i.e., of $K(K-1)$ matrices. In this section, we propose a simplified version of this algorithm that relies solely on the covariances from the previous iteration, but is still provably convergent. The algorithm is based on the same basic iterative water-filling step, but in each iteration, the updated covariances are a weighted sum of the old covariances and the covariances generated by the iterative water-filling step. This algorithm can be viewed as Algorithm 1 with the insertion of an averaging step after each iteration.

A graphical representation of the new algorithm (referred to as Algorithm 2 herein) for $K = 3$ is provided in Fig. 3. Notice that the initialization matrices are chosen to be all equal. As in Algorithm 1, in the first step $\boldsymbol{A}$ is updated to give the temporary variable $\boldsymbol{S}^{(1)}$. In Algorithm 1, we would assign $(\boldsymbol{A}^{(1)}, \boldsymbol{B}^{(1)}, \boldsymbol{C}^{(1)}) = (\boldsymbol{S}^{(1)}, \boldsymbol{B}^{(0)}, \boldsymbol{C}^{(0)})$, and then continue by updating $\boldsymbol{B}$, and so forth. In Algorithm 2, however, before performing the next update (i.e., before updating $\boldsymbol{B}$), the three variables are *averaged* to give

$$\boldsymbol{Q}^{(1)} \triangleq \frac{1}{3}(\boldsymbol{S}^{(1)} + \boldsymbol{Q}^{(0)} + \boldsymbol{Q}^{(0)}) = \frac{1}{3}\boldsymbol{S}^{(1)} + \frac{2}{3}\boldsymbol{Q}^{(0)}$$

and we set

$$(\boldsymbol{A}^{(1)}, \boldsymbol{B}^{(1)}, \boldsymbol{C}^{(1)}) = (\boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(1)}).$$

Notice that this averaging step does not decrease the objective, i.e., $f^{exp}(\boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(1)}) \geq f^{exp}(\boldsymbol{S}^{(1)}, \boldsymbol{Q}^{(0)}, \boldsymbol{Q}^{(0)})$, as we show later. This is, in fact, crucial in establishing convergence of the algorithm.

After the averaging step, the update is again performed, but this time on $\boldsymbol{B}$. The algorithm continues in this manner. It is easy to see that the averaging step essentially eliminates the need to retain the previous $K-1$ states in memory, and instead only the previous state (i.e., $\boldsymbol{Q}^{(n-1)}$) needs to be stored. The general equations describing the algorithm are

$$\boldsymbol{S}^{(n)} = \arg \max_{\boldsymbol{Q}} f^{exp}(\boldsymbol{Q}, \boldsymbol{Q}^{(n-1)}, \ldots, \boldsymbol{Q}^{(n-1)}) \tag{26}$$

$$\boldsymbol{Q}^{(n)} = \frac{1}{K}\boldsymbol{S}^{(n)} + \frac{K-1}{K}\boldsymbol{Q}^{(n-1)}. \tag{27}$$

The maximization in (26) that defines $\boldsymbol{S}^{(n)}$ is again solved by the water-filling solution, but where the effective channel depends only on the covariance matrices from the previous state, i.e., $\boldsymbol{Q}^{(n-1)}$.

[4] The algorithm converges from *any* starting point, but for simplicity we have chosen to initialize using the identity covariance. In Section IX we discuss the large advantage gained by using the original algorithm for a few iterations to generate a considerably better starting point.

[5] Notice that the modified algorithm and the original algorithm in Section V are equivalent only for $K = 2$.

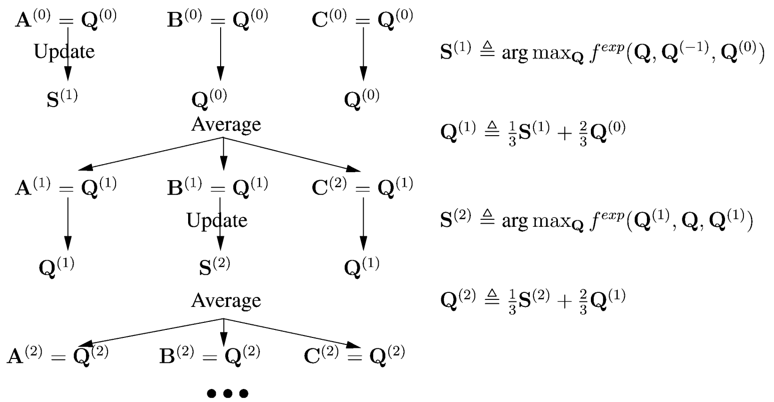

$\mathbf{A}^{(0)} = \mathbf{Q}^{(0)}$
$\mathbf{B}^{(0)} = \mathbf{Q}^{(0)}$
$\mathbf{C}^{(0)} = \mathbf{Q}^{(0)}$
Update
$\mathbf{S}^{(1)} \triangleq \arg\max_{\mathbf{Q}} f^{exp}(\mathbf{Q}, \mathbf{Q}^{(-1)}, \mathbf{Q}^{(0)})$
$\mathbf{S}^{(1)}$
$\mathbf{Q}^{(0)}$
$\mathbf{Q}^{(0)}$
Average
$\mathbf{Q}^{(1)} \triangleq \frac{1}{3}\mathbf{S}^{(1)} + \frac{2}{3}\mathbf{Q}^{(0)}$
$\mathbf{A}^{(1)} = \mathbf{Q}^{(1)}$
$\mathbf{B}^{(1)} = \mathbf{Q}^{(1)}$
$\mathbf{C}^{(2)} = \mathbf{Q}^{(1)}$
Update
$\mathbf{S}^{(2)} \triangleq \arg\max_{\mathbf{Q}} f^{exp}(\mathbf{Q}^{(1)}, \mathbf{Q}, \mathbf{Q}^{(1)})$
$\mathbf{Q}^{(1)}$
$\mathbf{S}^{(2)}$
$\mathbf{Q}^{(1)}$
Average
$\mathbf{Q}^{(2)} \triangleq \frac{1}{3}\mathbf{S}^{(2)} + \frac{2}{3}\mathbf{Q}^{(1)}$
$\mathbf{A}^{(2)} = \mathbf{Q}^{(2)}$
$\mathbf{B}^{(2)} = \mathbf{Q}^{(2)}$
$\mathbf{C}^{(2)} = \mathbf{Q}^{(2)}$
•••

Fig. 3. Graphical representation of Algorithm 2 for $K = 3$.

After initializing $\boldsymbol{Q}^{(0)}$, the algorithm proceeds as follows.[6]

1) Generate effective channels for each use

$$\boldsymbol{G}_i^{(n)} = \boldsymbol{H}_i \left(\boldsymbol{I} + \sum_{j\neq i} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j^{(n-1)} \boldsymbol{H}_j\right)^{-1/2}, \qquad i = 1, \dots, K. \tag{28}$$

2) Treating these effective channels as parallel, noninterfering channels, obtain covariance matrices $\{\boldsymbol{S}_i^{(n)}\}_{i=1}^K$ by water-filling with total power P

$$\left\{\boldsymbol{S}_i^{(n)}\right\}_{i=1}^K = \arg \max_{\{\boldsymbol{S}_i\}_{i=1}^K : \boldsymbol{S}_i \geq 0, \sum_{i=1}^K \mathrm{Tr}(\boldsymbol{S}_i) \leq P} \sum_{i=1}^K \log \times \left| \boldsymbol{I} + \left(\boldsymbol{G}_i^{(n)}\right)^\dagger \boldsymbol{S}_i \boldsymbol{G}_i^{(n)} \right|.$$

3) Compute the updated covariance matrices $\boldsymbol{Q}_i^{(n)}$ as

$$\boldsymbol{Q}_i^{(n)} = \frac{1}{K}\boldsymbol{S}_i^{(n)} + \frac{K-1}{K}\boldsymbol{Q}_i^{(n-1)}, \qquad i = 1, \dots, K. \tag{29}$$

Algorithm 2 (which first appeared in [11]) differs from the original algorithm only in the addition of the third step.

Theorem 3: Algorithm 2 converges to the sum rate capacity for any K.

Proof: Convergence of the algorithm is proven by showing that Algorithm 1 is equivalent to Algorithm 2 with the insertion of a nondecreasing (in the objective) operation in between every iteration. The spacer step theorem of [18, Ch. 7.11] asserts that if an algorithm satisfying the conditions of the global convergence theorem [18, Ch. 6.6] is combined with *any* series of steps that do not decrease the objective, then the combination of these two will still converge to the optimal. The cyclic coordinate ascent algorithm does indeed satisfy the conditions of the global convergence theorem, and later we prove that the averaging step does not decrease the objective. Thus, Algorithm 2 converges.[7]

Consider the n-iteration of the algorithm, i.e.,

$$(\boldsymbol{Q}^{(n-1)}, \dots, \boldsymbol{Q}^{(n-1)}) \to (\boldsymbol{S}^{(n)}, \boldsymbol{Q}^{(n-1)}, \dots, \boldsymbol{Q}^{(n-1)}) \tag{30}$$

$$\to \left(\frac{1}{K}\boldsymbol{S}^{(n)} + \frac{K-1}{K}\boldsymbol{Q}^{(n-1)}, \dots, \frac{1}{K}\boldsymbol{S}^{(n)} + \frac{K-1}{K}\boldsymbol{Q}^{(n-1)}\right) \tag{31}$$

where the mapping in (30) is the cyclic coordinate ascent algorithm performed on the first set of matrices, and the mapping in (31) is the averaging step. The first step is clearly identical to Algorithm 1, while the second step (i.e., the averaging step) has been added. We need only show that the averaging step is nondecreasing, i.e.,

$$f^{exp}(\boldsymbol{S}^{(n)}, \boldsymbol{Q}^{(n-1)}, \dots, \boldsymbol{Q}^{(n-1)}) \leq f^{exp}\left(\frac{1}{K}\boldsymbol{S}^{(n)} + \frac{K-1}{K}\boldsymbol{Q}^{(n-1)}, \dots, \frac{1}{K}\boldsymbol{S}^{(n)} + \frac{K-1}{K}\boldsymbol{Q}^{(n-1)}\right). \tag{32}$$

Notice that we can rewrite the left-hand side as

$$\begin{aligned} & f^{exp}(\boldsymbol{S}^{(n)}, \boldsymbol{Q}^{(n-1)}, \dots, \boldsymbol{Q}^{(n-1)}) \\ &= \frac{1}{K}\sum_{i=1}^K \log \left| \boldsymbol{I} + \boldsymbol{H}_i^\dagger \boldsymbol{S}_i^{(n)} \boldsymbol{H}_i + \sum_{j\neq i} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j^{(n-1)} \boldsymbol{H}_j \right| \\ &\leq \log \left| \frac{1}{K}\sum_{i=1}^K \left(\boldsymbol{I} + \boldsymbol{H}_i^\dagger \boldsymbol{S}_i^{(n)} \boldsymbol{H}_i + \sum_{j\neq i} \boldsymbol{H}_j^\dagger \boldsymbol{Q}_j^{(n-1)} \boldsymbol{H}_j\right) \right| \\ &= \log \left| \boldsymbol{I} + \sum_{j=1}^K \boldsymbol{H}_j^\dagger \left(\frac{1}{K}\boldsymbol{S}_j^{(n)} + \frac{K-1}{K}\boldsymbol{Q}_j^{(n-1)}\right) \boldsymbol{H}_j \right| \\ &= f^{exp}\left(\frac{1}{K}\boldsymbol{S}^{(n)} + \frac{K-1}{K}\boldsymbol{Q}^{(n-1)}, \dots, \frac{1}{K}\boldsymbol{S}^{(n)} + \frac{K-1}{K}\boldsymbol{Q}^{(n-1)}\right) \end{aligned}$$

where the inequality follows from the concavity of the $\log|\cdot|$ function. Since the averaging step is nondecreasing, the algorithm converges. More precisely, this means $f^{exp}(\boldsymbol{Q}^{(n)}, \dots, \boldsymbol{Q}^{(n)})$ converges to the sum capacity. Since this quantity is equal to $f(\boldsymbol{Q}^{(n)})$, we have

$$\lim_{n\to\infty} f(\boldsymbol{Q}^{(n)}) = \mathcal{C}_{\mathrm{MAC}}(\boldsymbol{H}_1^\dagger, \dots, \boldsymbol{H}_K^\dagger, P). \tag{33}$$

□

VIII. Complexity Analysis

In this section, we provide complexity analyses of the three proposed algorithms and other algorithms in the literature. Each of the three proposed algorithms here have complexity that increases *linearly* with K, the number of users. This is an extremely desirable property when considering systems with large numbers of users (i.e., 50 or 100 users). The linear complexity of our algorithm is quite easy to see if one goes through the basic steps of the algorithm. For simplicity, we consider Algorithm 1, which is the most complex of the algorithms. Calculating the effective channels in step 1 requires calculating the total interference seen by each user (i.e., a term of the form of $|\boldsymbol{I} + \sum_{j\neq i} \boldsymbol{H}_i^\dagger \boldsymbol{Q}_i \boldsymbol{H}_i|$). A running sum of such a term can be maintained, such that calculating the effective channel of each user requires only a finite number of subtractions and additions. The water-filling operation in step 2 can also be performed in linear time by taking the SVD of each of the effective

[6] As discussed in Section IX, the original algorithm can be used to generate an excellent starting point for Algorithm 2.

[7] There is also a technical condition regarding compactness of the set with larger objective than the objective evaluated for the initialization matrices that is trivially satisfied due to the properties of Euclidean space.

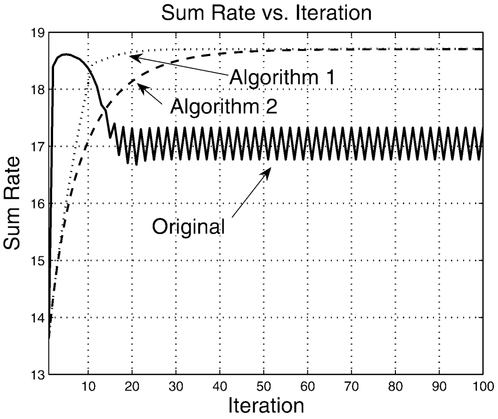


Fig. 4. Algorithm comparison for a divergent scenario.

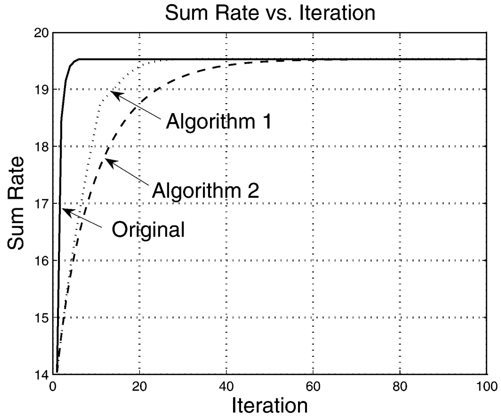


Fig. 5. Algorithm comparison for a convergent scenario.

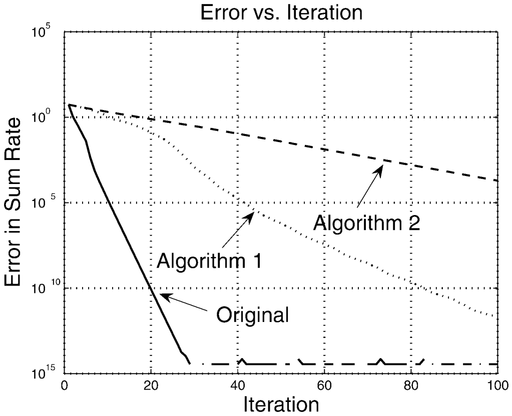


Fig. 6. Error comparison for a convergent scenario.

channels and then water-filling. It is important not to perform a standard water-filling operation on the block diagonal channel, because the size of the involved matrices grow with K. In general, the key idea behind the linear complexity of our algorithm is that the entire input space is never considered (i.e., only $N \times N$ and $M \times M$ matrices, and never matrices whose size is a function of K, are considered). This, however, is not true of general optimization methods which do not take advantage of the structure of the sum capacity problem.

Standard interior point methods have complexity that is cubic with respect to the dimensionality of the input space (i.e., with respect to K, the number of users), due to the complexity of the inner Newton iterations [2]. The minimax-based approach in [8] also has complexity that is cubic in K because matrices whose size is a function of K are inverted in each step. For very small problems, this is not significant, but for even reasonable values of K (i.e., $K = 10$ or $K = 20$) this increase in complexity makes such methods computationally prohibitive.

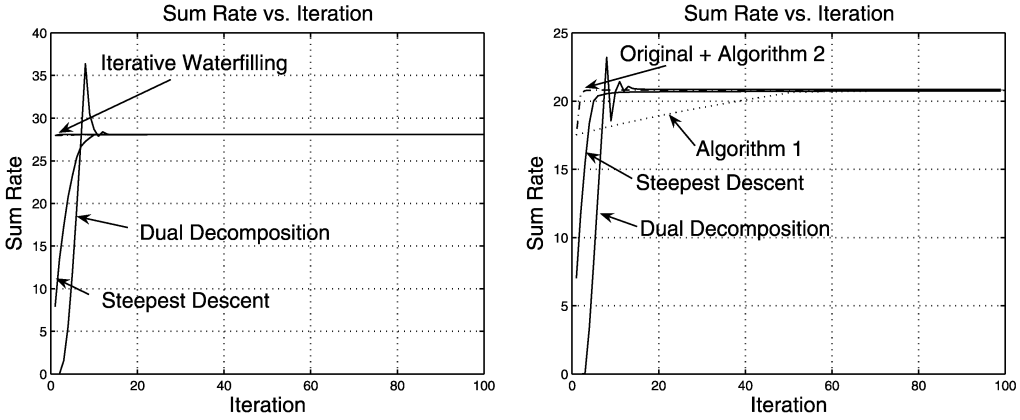
The other proposed specialized algorithms [13], [15] are also linear in complexity (in K). However, the steepest descent algorithm proposed in [13] requires a line search in each step, which does not increase the complexity order but does significantly increase run time. The dual decomposition algorithm proposed in [15] requires an inner optimization to be performed within each iteration (i.e., user-by-user iterative water-filling [17] with a fixed water level, instead of individual power constraints, must be performed repeatedly), which significantly increases run time. Our sum power iterative water-filling algorithms, on the other hand, do not require a line search or an inner optimization within each iteration, thus leading to a faster run time. In addition, we find the iterative water-filling algorithms to converge faster than the other linear complexity algorithms for almost all channel realizations. Some numerical results and discussion of this are presented in Section IX.

IX. Numerical Results

In this section, we provide some numerical results to show the behavior of the three algorithms. In Fig. 4, a plot of sum rate versus iteration number is provided for a 10–user channel with four transmit and four receive antennas. In this example, the original algorithm does not converge and can be seen to oscillate between two suboptimal points. Algorithms 1 and 2 do converge, however, as guaranteed by Theorems 2 and 3. In general, it is not difficult to randomly generate channels for which the original algorithm does not converge and instead oscillates between suboptimal points. This divergence occurs because not only can the original algorithm lead to a decrease in the sum rate, but additionally there appear to exist suboptimal points between which the original algorithm can oscillate, i.e., point 1 is generated by iteratively waterfilling from point 2, and *vice versa*.

In Fig. 5, the same plot is shown for a different channel (with the same system parameters as in Fig. 4: $K = 10, M = N = 4$) in which the original algorithm does in fact converge. Notice that the original algorithm performs best, followed by Algorithm 1, and then Algorithm 2. The same trend is seen in Fig. 6, which plots the error in capacity. Additionally, notice that all three algorithms converge linearly, as expected for this class of algorithms. Though these plots are only for a single instantiation of channels, the same ordering has always occurred, i.e., the original algorithm performs best (in situations where it converges) followed by Algorithm 1 and then Algorithm 2.

The fact that the original algorithm converges faster than the modified algorithms is intuitively not surprising, because the original algorithm updates matrices at a much faster rate than either of the modified versions of the algorithm. In Algorithm 1, there are K covariances for each user (corresponding to the K previous states) that are averaged to yield the set of covariances that converge to the optimal. The most recently updated covariances therefore make up only a fraction $1/K$ of the average, and thus the algorithm moves relatively slowly. In Algorithm 2, the updated covariances are very similar to the covariances from the previous state, as the updated covariances are equal to $(K-1)/K$ times the previous state's covariances plus only a factor of $1/K$ times the covariances generated by the iterative water-filling step. Thus, it should be intuitively clear that in situations where the original algorithm actually converges, convergence is much faster for the original algorithm than for either of the modified algorithms. From the plot it is clear that the performance difference between the original algorithm and Algorithms 1 and 2 is quite significant. At the end of this section, however, we discuss how the original algorithm can be combined with either Algorithm 1 or 2 to improve performance considerably while still maintaining guaranteed convergence. Of the two modified algorithms, Algorithm 1 is almost always seen to outperform Algorithm 2. However, there does not appear to be an intuitive explanation for this behavior.



(a) (b)

Fig. 7. Comparison of linear complexity algorithms. (a) Ten-user system with $M = 10, N = 1$ (b) Fifty-user system with $M = 5, N = 1$.

In Fig. 7(a) sum rate is plotted for the three iterative water-filling algorithms (original, Algorithm 1, and Algorithm 2), the steepest descent method [13], and the dual decomposition method [15], for a channel with $K = 10$, $M = 10$, and $N = 1$. The three iterative water-filling algorithms perform nearly identically for this channel, and three curves are in fact superimposed on one and other in the figure. Furthermore, the iterative water-filling algorithms converge more rapidly than either of the alternative methods. The iterative water-filling algorithms outperform the other algorithms in many scenarios, and the gap is particularly large when the number of transmit antennas (M) and users (K) are large. It should be noted that there are certain situations where the steepest descent and dual decomposition algorithms outperform the iterative water-filling algorithm, in particular when the number of users is much larger than the number of antennas. Fig. 7(b) contains a convergence plot of a 50-user system with $M = 5$ and $N = 1$. Algorithm 1 converges rather slowly precisely because of the large number of users (i.e., because the covariances can only change at approximately a rate of $1/K$ in each iteration, as discussed earlier). Notice that both the steepest descent and dual decomposition algorithms converge faster. However, the results for a *hybrid* algorithm are also plotted here (referred to as "Original + Algorithm 2"). In this hybrid algorithm, the original iterative water-filling algorithm is performed for the first five iterations, and then Algorithm 2 is used for all subsequent iterations. The original algorithm is essentially used to generate a good starting point for Algorithm 2. This hybrid algorithm converges, because the original algorithm is only used a finite number of times, and is seen to outperform any of the other alternatives. In fact, we find that the combination of the original algorithm with either Algorithm 1 or 2 converges extremely rapidly to the optimum and outperforms the alternative linear complexity approaches in the very large majority of scenarios, i.e., for any number of users and antennas. This is true even for channels for which the original algorithm itself does not converge, because running the original algorithm for a few iterations still provides an excellent starting point.

X. Conclusion

In this correspondence we proposed two algorithms that find the sum capacity achieving transmission strategies for the multiple-antenna BC. We use the fact that the Gaussian broadcast and MAC's are duals in the sense that their capacity regions, and therefore their sum capacities, are equal. These algorithms compute the sum capacity achieving strategy for the dual MAC, which can easily be converted to the equivalent optimal strategies for the BC. The algorithms exploit the inherent structure of the MAC and employ a simple iterative water-filling procedure that provably converges to the optimum. The two algorithms are extremely similar, as both are based on the cyclic coordinate ascent and use the single-user water-filling procedure in each iteration, but they offer a simple tradeoff between performance and required memory. The convergence speed, low complexity, and simplicity make the iterative water-filling algorithms extremely attractive methods to find the sum capacity of the multiple-antenna BC.

Appendix I
MAC BC Transformation

In this appendix, we restate the mapping from uplink covariance matrices to downlink matrices. Given uplink covariances $\boldsymbol{Q}_1, \dots, \boldsymbol{Q}_K$, the transformation in [10, eqs. 8–10] outputs downlink covariance matrices $\Sigma_1, \dots, \Sigma_K$ that achieve the same rates (on a user-by-user basis, and thus also in terms of sum rate) using the same sum power, i.e., with

$$\sum_{i=1}^{K} \mathrm{Tr}(\boldsymbol{Q}_i) = \sum_{i=1}^{K} \mathrm{Tr}(\boldsymbol{\Sigma}_i).$$

For convenience, we first define the following two quantities:

$$\boldsymbol{A}_i \triangleq \boldsymbol{I} + \boldsymbol{H}_i \left(\sum_{l=1}^{i-1} \boldsymbol{\Sigma}_l \right) \boldsymbol{H}_i^\dagger, \quad \boldsymbol{B}_i \triangleq \boldsymbol{I} + \sum_{l=i+1}^{K} \boldsymbol{H}_l^\dagger \boldsymbol{Q}_l \boldsymbol{H}_l \tag{34}$$

for $i = 1, \dots, K$. Furthermore, we write the SVD decomposition of $\boldsymbol{B}_i^{-1/2}\boldsymbol{H}_i^\dagger\boldsymbol{A}_i^{-1/2}$ as $\boldsymbol{B}_i^{-1/2}\boldsymbol{H}_i^\dagger\boldsymbol{A}_i^{-1/2} = \boldsymbol{F}_i\boldsymbol{D}_i\boldsymbol{G}_i^\dagger$, where $\boldsymbol{D}_i$ is a square and diagonal matrix.[8] Then, the equivalent downlink covariance matrices can be computed via the following transformation:

$$\boldsymbol{\Sigma}_i = \boldsymbol{B}_i^{-1/2}\boldsymbol{F}_i\boldsymbol{G}_i^\dagger\boldsymbol{A}_i^{1/2}\boldsymbol{Q}_i\boldsymbol{A}_i^{1/2}\boldsymbol{G}_i\boldsymbol{F}_i^\dagger\boldsymbol{B}_i^{-1/2} \tag{35}$$

beginning with $i = 1$. See [10] for a derivation and more detail.

Appendix II
Uniqueness of Water-Filling Solution

In this appendix, we show there is a unique solution to the following maximization:

$$\max_{\boldsymbol{Q} \geq 0,\ \mathrm{Tr}(\boldsymbol{Q}) \leq P} \log \left| \boldsymbol{I} + \boldsymbol{H}\boldsymbol{Q}\boldsymbol{H}^\dagger \right| \tag{36}$$

for any nonzero $\boldsymbol{H} \in \mathbb{C}^{N \times M}$ for arbitrary M, N. This proof is identical to the proof of optimality of water-filling in [9, Sec. 3.2], with the addition of a simple proof of uniqueness.

Since $\boldsymbol{H}^\dagger\boldsymbol{H} \in \mathbb{C}^{M \times M}$ is Hermitian and positive semi-definite, we can diagonalize it and write $\boldsymbol{H}^\dagger\boldsymbol{H} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^\dagger$ where $\boldsymbol{U} \in \mathbb{C}^{M \times M}$ is unitary and $\boldsymbol{D} \in \mathbb{R}^{M \times M}$ is diagonal with nonnegative entries. Since the ordering of the columns of $\boldsymbol{U}$ and the entries of $\boldsymbol{D}$ are arbitrary and because $\boldsymbol{D}$ must have at least one strictly positive entry (because

[8] Note that the standard SVD command in MATLAB does not return a square and diagonal $\boldsymbol{D}_i$. This is accomplished by using the "0" option in the SVD command in MATLAB, and is referred to as the "economy size" decomposition.

$\boldsymbol{H}$ is not the zero matrix), for simplicity, we assume $\boldsymbol{D}_{ii} > 0$ for $i = 1, \ldots, L$ and $\boldsymbol{D}_{ii} = 0$ for $i = L+1, \ldots, M$ for some $1 \le L \le M$. Using the identity $|\boldsymbol{I} + \boldsymbol{AB}| = |\boldsymbol{I} + \boldsymbol{BA}|$, we can rewrite the objective function in (36) as

$$\begin{aligned}\log|\boldsymbol{I} + \boldsymbol{HQH}^\dagger| &= \log|\boldsymbol{I} + \boldsymbol{QH}^\dagger\boldsymbol{H}| = \log|\boldsymbol{I} + \boldsymbol{QUDU}^\dagger| \\ &= \log|\boldsymbol{I} + \boldsymbol{U}^\dagger\boldsymbol{QUD}|. \qquad (37)\end{aligned}$$

If we define $\boldsymbol{S} \triangleq \boldsymbol{U}^\dagger\boldsymbol{QU}$, then $\boldsymbol{Q} = \boldsymbol{USU}^\dagger$. Since $\text{Tr}(\boldsymbol{AB}) = \text{Tr}(\boldsymbol{BA})$ and $\boldsymbol{U}$ is unitary, we have

$$\text{Tr}(\boldsymbol{S}) = \text{Tr}(\boldsymbol{U}^\dagger\boldsymbol{QU}) = \text{Tr}(\boldsymbol{QUU}^\dagger) = \text{Tr}(\boldsymbol{Q}).$$

Furthermore, $\boldsymbol{S} \ge 0$ if and only if $\boldsymbol{Q} \ge 0$. Therefore, the maximization can equivalently be carried out over $\boldsymbol{S}$, i.e.,

$$\max_{\boldsymbol{S} \ge 0,\ \text{Tr}(\boldsymbol{S}) \le P} \log|\boldsymbol{I} + \boldsymbol{SD}|. \qquad (38)$$

In addition, each solution to (36) corresponds to a different solution of (38) via the invertible mapping $\boldsymbol{S} = \boldsymbol{U}^\dagger\boldsymbol{QU}$. Thus, if the maximization in (36) has multiple solutions, the maximization in (38) must also have multiple solutions. Therefore, it is sufficient to show that (38) has a unique solution, which we prove next.

First we show by contradiction that any optimal $\boldsymbol{S}$ must satisfy $\boldsymbol{S}_{ij} = 0$ for all $i, j > L$. Consider an $\boldsymbol{S} \ge 0$ with $\boldsymbol{S}_{ij} \ne 0$ for some $i > L$ and $j > L$. Since

$$|\boldsymbol{S}_{ij}| \le \sqrt{\boldsymbol{S}_{ii}\boldsymbol{S}_{jj}}, \qquad \text{for any } \boldsymbol{S} \ge 0$$

this implies $\boldsymbol{S}_{ii} > 0$ and $\boldsymbol{S}_{jj} > 0$, i.e., at least one diagonal entry of $\boldsymbol{S}$ is strictly positive below the Lth row/column. Using Hadamard's inequality [5] and the fact that $\boldsymbol{D}_{ii} = 0$ for $i > L$, we have

$$|\boldsymbol{I} + \boldsymbol{SD}| \le \prod_{i=1}^{M}(1 + \boldsymbol{S}_{ii}\boldsymbol{D}_{ii}) = \prod_{i=1}^{L}(1 + \boldsymbol{S}_{ii}\boldsymbol{D}_{ii}).$$

We now construct another matrix $\boldsymbol{S}'$ that achieves a strictly larger objective than $\boldsymbol{S}$. We define $\boldsymbol{S}'$ to be diagonal with

$$\boldsymbol{S}'_{ii} = \begin{cases} \boldsymbol{S}_{11} + \sum_{i=L+1}^{M}\boldsymbol{S}_{ii}, & i = 1 \\ \boldsymbol{S}_{ii}, & i = 2, \ldots, L \\ 0, & i = L+1, \ldots, M. \end{cases} \qquad (39)$$

Clearly $\boldsymbol{S}' \ge 0$ and

$$\text{Tr}(\boldsymbol{S}') = \sum_{i=1}^{L}\boldsymbol{S}'_{ii} = \boldsymbol{S}_{11} + \sum_{i=L+1}^{M}\boldsymbol{S}_{ii} + \sum_{i=2}^{L}\boldsymbol{S}_{ii} = \text{Tr}(\boldsymbol{S}).$$

Since $\boldsymbol{S}'$ is diagonal, the matrix $\boldsymbol{S}'\boldsymbol{D}$ is diagonal and we have

$$\begin{aligned}\log|\boldsymbol{I} + \boldsymbol{S}'\boldsymbol{D}| = \log\prod_{i=1}^{L}(1 + \boldsymbol{S}'_{ii}\boldsymbol{D}_{ii}) &> \log\prod_{i=1}^{L}(1 + \boldsymbol{S}_{ii}\boldsymbol{D}_{ii}) \\ &\ge \log|\boldsymbol{I} + \boldsymbol{SD}|\end{aligned}$$

where the strict inequality is due to the fact that $\boldsymbol{S}'_{11} > \boldsymbol{S}_{11}$ and $\boldsymbol{D}_{11} > 0$. Therefore, the optimal $\boldsymbol{S}$ must satisfy $\boldsymbol{S}_{ij} = 0$ for all $i, j > L$.

Next we show by contradiction that any optimal $\boldsymbol{S}$ must also be diagonal. Consider any $\boldsymbol{S} \ge 0$ that satisfies the above condition ($\boldsymbol{S}_{ij} = 0$ for all $i, j > L$) but is not diagonal, i.e., $\boldsymbol{S}_{kj} \ne 0$ for some $k \ne j$ and $k, j \le L$. Since $\boldsymbol{D}$ is diagonal and $\boldsymbol{D}_{ii} > 0$ for $i = 1, \ldots, L$, the matrix $\boldsymbol{SD}$ is not diagonal because $(\boldsymbol{SD})_{kj} = \boldsymbol{S}_{kj}\boldsymbol{D}_{jj} \ne 0$. Since Hadamard's inequality holds with equality only for diagonal matrices, we have

$$\log|\boldsymbol{I} + \boldsymbol{SD}| < \log\prod_{i=1}^{L}(1 + \boldsymbol{S}_{ii}\boldsymbol{D}_{ii}).$$

Let us define a diagonal matrix $\boldsymbol{S}'$ with $\boldsymbol{S}'_{ii} = \boldsymbol{S}_{ii}$ for $i = 1, \ldots, M$. Clearly, $\text{Tr}(\boldsymbol{S}') = \text{Tr}(\boldsymbol{S})$ and $\boldsymbol{S}' \ge 0$. Since $\boldsymbol{S}'$ is diagonal, the matrix $\boldsymbol{S}'\boldsymbol{D}$ is diagonal and thus

$$\log|\boldsymbol{I} + \boldsymbol{S}'\boldsymbol{D}| = \log\prod_{i=1}^{L}(1 + \boldsymbol{S}_{ii}\boldsymbol{D}_{ii}) > \log|\boldsymbol{I} + \boldsymbol{SD}|.$$

Therefore, the optimal $\boldsymbol{S}$ must be diagonal, as well as satisfy $\boldsymbol{S}_{ij} = 0$ for $i, j > L$.

Therefore, in order to find *all solutions* to (38), it is sufficient to only consider the class of diagonal, positive semidefinite matrices $\boldsymbol{S}$ that satisfy $\boldsymbol{S}_{ij} = 0$ for all $i, j > L$ and $\text{Tr}(\boldsymbol{S}) \le P$. The positive semidefinite constraint is equivalent to $\boldsymbol{S}_{ii} \ge 0$ for $i = 1, \ldots, L$, and the trace constraint gives $\sum_{i=1}^{L}\boldsymbol{S}_{ii} \le P$. Since

$$\log|\boldsymbol{I} + \boldsymbol{S}'\boldsymbol{D}| = \log\prod_{i=1}^{L}(1 + \boldsymbol{S}'_{ii}\boldsymbol{D}_{ii})$$

for this class of matrices, we need only consider the following maximization:

$$\max_{\{\boldsymbol{S}_{ii}\}_{i=1}^{L}:\boldsymbol{S}_{ii} \ge 0,\ \sum_{i=1}^{L}\boldsymbol{S}_{ii} \le P} \sum_{i=1}^{L}\log(1 + \boldsymbol{S}_{ii}\boldsymbol{D}_{ii}). \qquad (40)$$

Since $\boldsymbol{D}_{ii} > 0$ for $i = 1, \ldots, L$, the objective in (40) is a strictly concave function, and thus has a unique maximum. Thus, (38) has a unique maximum, which implies that (36) also has a unique maximum.

Appendix III
Derivation of Algorithm 1

In this appendix, we derive the general form of Algorithm 1 for an arbitrary number of users. In order to solve the original sum rate capacity maximization in (12), we consider an alternative maximization

$$\max_{\boldsymbol{S}(1), \ldots, \boldsymbol{S}(K)} f^{exp}(\boldsymbol{S}(1), \ldots, \boldsymbol{S}(K)) \qquad (41)$$

where we define $\boldsymbol{S}(i) \triangleq (\boldsymbol{S}(i)_1, \ldots, \boldsymbol{S}(i)_K)$ for $i = 1, \ldots, K$ with $\boldsymbol{S}(i)_j \in \mathbb{C}^{N \times N}$, and the maximization is performed subject to the constraints $\boldsymbol{S}(i)_j \ge 0$ for all i, j and

$$\sum_{j=1}^{K}\text{Tr}(\boldsymbol{S}(i)_j) \le P, \qquad \text{for } i = 1, \ldots, K.$$

The function $f^{exp}(\cdot)$ is defined as

$$\begin{aligned}&f^{exp}(\boldsymbol{S}(1), \ldots, \boldsymbol{S}(K)) \\ &\quad = \frac{1}{K}\sum_{i=1}^{K}\log\left|\boldsymbol{I} + \sum_{j=1}^{K}\boldsymbol{H}_j^\dagger\boldsymbol{S}([j-i+1]_K)_j\boldsymbol{H}_j\right|. \qquad (42)\end{aligned}$$

In the notation used in Section VI, we would have $\boldsymbol{A} = \boldsymbol{S}(1)$, $\boldsymbol{B} = \boldsymbol{S}(2)$, $\boldsymbol{C} = \boldsymbol{S}(3)$. As discussed earlier, every solution to the original sum rate maximization problem in (12) corresponds to a solution to (41), and *vice versa*. Furthermore, the cyclic coordinate ascent algorithm can be used to maximize (41) due to the separability of the constraints on $\boldsymbol{S}(1), \ldots, \boldsymbol{S}(K)$. If we let $\{\boldsymbol{S}(i)^{(n)}\}_{i=1}^{K}$ denote the nth iteration of the cyclic coordinate ascent algorithm, then (43) (at the bottom of the page) holds for

$$\boldsymbol{S}(l)^{(n)} = \begin{cases} \arg\max_{\boldsymbol{S}} f^{exp}(\boldsymbol{S}(1)^{(n-1)}, \ldots, \boldsymbol{S}(m-1)^{(n-1)}, \boldsymbol{S}, \boldsymbol{S}(m+1)^{(n-1)}, \ldots, \boldsymbol{S}(K)^{(n-1)}) & l = m \\ \boldsymbol{S}(l)^{(n-1)} & l \ne m \end{cases} \qquad (43)$$

$$\boldsymbol{Q}^{(n)} = \arg\max_{\boldsymbol{Q}} f^{exp}\left(\boldsymbol{Q}, \boldsymbol{Q}^{(n-K+1)}, \ldots, \boldsymbol{Q}^{(n-1)}\right) \tag{48}$$

$$= \arg \max_{\boldsymbol{Q}: \boldsymbol{Q}_i \geq 0,\ \sum_{i=1}^{K} \mathrm{Tr}(\boldsymbol{Q}_i) \leq P} \sum_{i=1}^{K} \log \left| \boldsymbol{I} + \boldsymbol{H}_i^{\dagger} \boldsymbol{Q}_i \boldsymbol{H}_i + \sum_{j=1}^{K-1} \boldsymbol{H}_{[i+j]_K}^{\dagger} \boldsymbol{Q}_{[i+j]_K}^{(n-K+j)} \boldsymbol{H}_{[i+j]_K} \right| \tag{49}$$

$$= \arg \max_{\boldsymbol{Q}: \boldsymbol{Q}_i \geq 0,\ \sum_{i=1}^{K} \mathrm{Tr}(\boldsymbol{Q}_i) \leq P} \sum_{i=1}^{K} \log \left| \boldsymbol{I} + \left(\boldsymbol{G}_i^{(n)}\right)^{\dagger} \boldsymbol{Q}_i \boldsymbol{G}_i^{(n)} \right|. \tag{50}$$

$l = 1, \ldots, K$, where $m = [n]_K$. For each n, we define $\boldsymbol{Q}^{(n)}$ to be the updated matrices in that iteration

$$\begin{aligned} \boldsymbol{Q}^{(n)} &\triangleq \boldsymbol{S}(m)^{(n)} && (44)\\ &= \arg\max_{\boldsymbol{S}} f^{exp}(\boldsymbol{S}(1)^{(n-1)}, \ldots, \boldsymbol{S}(m-1)^{(n-1)}, \boldsymbol{S}, \\ &\qquad \boldsymbol{S}(m+1)^{(n-1)}, \ldots, \boldsymbol{S}(K)^{(n-1)}) && (45)\\ &= \arg\max_{\boldsymbol{S}} f^{exp}(\boldsymbol{S}, \boldsymbol{S}(m+1)^{(n-1)}, \ldots, \boldsymbol{S}(K)^{(n-1)}, \\ &\qquad \boldsymbol{S}(1)^{(n-1)}, \ldots, \boldsymbol{S}(m-1)^{(n-1)}) && (46) \end{aligned}$$

where in the final step we used the fact that

$$\begin{aligned} &f^{exp}(\boldsymbol{S}(1), \ldots, \boldsymbol{S}(K)) \\ &\quad = f^{exp}(\boldsymbol{S}(l), \ldots, \boldsymbol{S}(K), \boldsymbol{S}(1), \ldots, \boldsymbol{S}(l-1)) \quad (47) \end{aligned}$$

for any l due to the circular structure of f^{exp} and the uniqueness of the water-filling solution to (46). Plugging in recursively for $\boldsymbol{Q}^{(n)}$ for all n, we get (48)–(50) at the top of the page. The final maximization is equivalent to water-filling over effective channels $\boldsymbol{G}_j$, given by

$$\boldsymbol{G}_i^{(n)} = \boldsymbol{H}_i \left(\boldsymbol{I} + \sum_{j=1}^{K-1} \boldsymbol{H}_{[i+j]_K}^{\dagger} \boldsymbol{Q}_{[i+j]_K}^{(n-K+j)} \boldsymbol{H}_{[i+j]_K} \right)^{-1/2} \tag{51}$$

for $i = 1, \ldots, K$.

Acknowledgment

The authors wish to thank Daniel Palomar and Tom Luo for helpful discussions regarding convergence issues.

References

[1] D. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.

[2] S. Boyd and L. Vandenberghe, *Introduction to Convex Optimization With Engineering Applications*. Stanford, CA: Course Reader, Stanford Univ., 2001.

[3] G. Caire and S. Shamai (Shitz), "On the achievable throughput of a multiantenna Gaussian broadcast channel," *IEEE Trans. Inf. Theory*, vol. 49, no. 7, pp. 1691–1706, Jul. 2003.

[4] M. Costa, "Writing on dirty paper," *IEEE Trans. Inf. Theory*, vol. IT-29, no. 3, pp. 439–441, May 1983.

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[6] N. Jindal, S. Jafar, S. Vishwanath, and A. Goldsmith, "Sum power iterative water-filling for multi-antenna Gaussian broadcast channels," in *Proc. Asilomar Conf. Signals, Systems, and Computers*, Asilomar, CA, 2002.

[7] N. Jindal, S. Vishwanath, and A. Goldsmith, "On the duality of Gaussian multiple-access and broadcast channels," *IEEE Trans. Inf. Theory*, vol. 50, no. 5, pp. 768–783, May 2004.

[8] T. Lan and W. Yu, "Input optimization for multi-antenna broadcast channels and per-antenna power constraints," in *Proc. IEEE GLOBECOM*, vol. 1, Nov. 2004, pp. 420–424.

[9] E. Telatar, "Capacity of multi-antenna Gaussian channels," *Europ. Trans. on Telecomm.*, vol. 10, no. 6, pp. 585–596, Nov. 1999.

[10] S. Vishwanath, N. Jindal, and A. Goldsmith, "Duality, achievable rates, and sum-rate capacity of MIMO broadcast channels," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2658–2668, Oct. 2003.

[11] S. Vishwanath, W. Rhee, N. Jindal, S. A. Jafar, and A. Goldsmith, "Sum power iterative water-filling for Gaussian vector broadcast channels," in *Proc. IEEE Int. Symp. Information Theory*, Yokohama, Japan, Jun./Jul. 2003, p. 467.

[12] P. Viswanath and D. N. C. Tse, "Sum capacity of the vector Gaussian broadcast channel and uplink-downlink duality," *IEEE Trans. Inf. Theory*, vol. 49, no. 8, pp. 1912–1921, Aug. 2003.

[13] H. Viswanathan, S. Venkatesan, and H. C. Huang, "Downlink capacity evaluation of cellular networks with known interference cancellation," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 802–811, Jun. 2003.

[14] H. Weingarten, Y. Steinberg, and S. Shamai, "The capacity region of the Gaussian MIMO broadcast channel," in *Proc. Conf. Information Sciences and Systems*, Princeton, NJ, Mar. 2004.

[15] W. Yu, "A dual decomposition approach to the sum power Gaussian vector multiple-access channel sum capacity problem," in *Proc. Conf. Information Sciences and Systems (CISS)*, Baltimore, MD, 2003.

[16] W. Yu and J. M. Cioffi, "Sum capacity of a Gaussian vector broadcast channels," *IEEE Trans. Inf. Theory*, vol. 50, no. 9, pp. 1875–1892, Sep. 2002.

[17] W. Yu, W. Rhee, S. Boyd, and J. Cioffi, "Iterative water-filling for Gaussian vector multiple-access channels," *IEEE Trans. Inf. Theory*, vol. 50, no. 1, pp. 145–152, Jan. 2004.

[18] W. Zangwill, *Nonlinear Programming: A Unified Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1969.

Design of Efficient Second-Order Spectral-Null Codes

Ching-Nung Yang

***Abstract*—An efficient recursive method has been proposed for the encoding/decoding of second-order spectral-null codes, via concatenation by Tallini and Bose. However, this method requires the appending of one, two, or three extra bits to the information word, in order to make a balanced code, with the length being a multiple of $\mathbf{4}$; this introduces redundancy. Here, we introduce a new *quasi*-second-order spectral-null code with the length $\equiv \mathbf{2} \pmod{\mathbf{4}}$ and extend the recursive method of Tallini and Bose, to achieve a higher code rate.**

***Index Terms*—Balanced code, dc-free codes, high-order spectral-null codes.**

I. Introduction

In some applications, such as digital transmission and recording systems, we want to achieve a larger level of rejection of the low-frequency components for *dc-free* (referred to as balanced or zero-disparity) codes. These codes are so called "high-order spectral-null codes"

Manuscript received December 10, 1003; revised November 27, 2004.

The author is with the Department of Computer Science and Information Engineering, National Dong Hwa University, Shou-Feng, Taiwan, R.O.C. (e-mail: cnyang@mail.ndhu.edu.tw).

Communicated by Ø. Ytrehus, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2005.844085