Interference Alignment via Adaptively Controlled Perturbations

Hadi Ghauch [1], Taejoon Kim [2], Mats Bengtsson [1], Mikael Skoglund [1]
[1] *School of Electrical Engineering and the ACCESS Linnaeus Center, KTH Royal Institute of Technology, Stockholm, Sweden*
[2] *Department of Electronic Engineering, City University of Hong Kong, Kowloon Tong, Hong Kong*
ghauch@kth.se, taejokim@cityu.edu.hk, mats.bengtsson@ee.th.se, skoglund@ee.kth.se

***Abstract*—In this work, we study the so-called leakage minimization problem, within the context of interference alignment (IA). For that purpose, we propose a novel approach based on adaptively controlled perturbations of the leakage function, and show how the latter can be used as a mechanism to control the algorithm's convergence (and thus tradeoff convergence speed for reliability). Although the proposed scheme falls under the broad category of stochastic optimization, we show through simulations that it has a quasi-deterministic convergence that we exploit to improve on the worst case performance of its predecessor, resulting in significantly better sum-rate capacity and average cost function value.**

I. Introduction

Interference Alignment (IA) is a recent technique that has been shown to achieve the optimal degrees-of-freedom of the K-user MIMO interference channel [1], [2]. Although extremely promising at first, later work has revealed that IA comes with many challenges that first need to be addressed, before any of its promised gains can be harnessed, namely the need for channel extensions, time / frequency varying channels, etc. One such challenge that our work addresses, is the need to find transmit and receive filters that satisfy the IA conditions (since the original closed-form beamforming design proposed in [1] is only applicable to cases with no more than 3 users). As a result, people turned their attention to trying to solve this problem using numerical methods, and since the inception of IA, a plethora of iterative algorithms have been proposed for that purpose. Furthermore, those algorithms generally fall into one of following broad categories (based on the criterion they are trying to optimize): leakage minimization ([3]), (weighted) mean-squared error minimization ([4],[5]), and sum-rate maximization ([6], [7]). However, they all seem to suffer from common limitations.

Firstly, due to their construction, it is rather *challenging to mathematically characterize the behaviour* of such algorithms. Furthermore, they are *not amenable to performance analysis* mainly due to the absence of any mechanism that can control convergence, thus resulting in lack of guarantees on their average performance (in most cases, only convergence to local optima can be shown). In addition, since many of them fall under the broad umbrella of greedy algorithms, their *performance is largely hindered by the presence of many local optima* in their respective cost functions. Moreover, algorithms that fall under the distributed category (namely distributed IA [3]), employ the so-called ping-pong transmit and receive filter updates exploiting channel reciprocity, and thus may suffer from practical implementation issues (e.g., synchronization, applicability to TDD systems only, etc.).

The approach that we present in this work addresses the above limitations by adopting a novel paradigm, i.e. by *introducing a mechanism to control the convergence*. As a result, metrics such as convergence speed and reliability (defined shortly after), can be traded-off to achieve the desired performance. Although not addressed in this paper, the addition of this so-called convergence control mechanism is a stepping stone for performance analysis. Moreover, we show that the proposed scheme improves the worst case performance of distributed IA (in view of fairness of comparison, we use distributed IA [3] as our benchmark), and thus achieves *better average performance*, in terms of convergence speed, reliability, and sum-rate performance.

In the following, we use lowercase letters to denote scalars, while bold lowercase and uppercase letters denote vectors and matrices, respectively. $\lambda_i[\boldsymbol{A}]$ denotes the i^{th} eigenvalue of $\boldsymbol{A}$ (assuming the eigenvalues are sorted in increasing order of magnitude). The † symbol denotes the conjugate transpose of a matrix (Hermitian). Moreover, κ denotes the set of integers $\kappa = \{1, ..., K\}$ and $\boldsymbol{I}_n$ denotes the $n \times n$ identity matrix. Finally, $(\boldsymbol{V})^\perp$ is the orthogonal complement of a given subspace $\boldsymbol{V}$, $\mathcal{U}(n, m)$ denotes the space of unitary $n \times m$ matrices, and $d_c(\boldsymbol{M}_1, \boldsymbol{M}_2) = (\sqrt{m - \|\boldsymbol{M}_1^\dagger \boldsymbol{M}_2\|_F^2}\ )/\sqrt{m}$ is the normalized chordal distance (on the Grassmann manifold), between two subspaces $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ in $\mathcal{U}(n, m)$.

Part of this work has been performed in the framework of the FP7 project ICT-317669 METIS, which is partly funded by the European Union. The authors would like to acknowledge the contributions of their colleagues in METIS, although the views expressed are those of the authors and do not necessarily represent the project.

II. Signal Model and Problem Statement

We consider a K-user MIMO interference channel, where the received signal (after linear filtering), is given by

$$\begin{aligned}\hat{\boldsymbol{x}}^{[k]} &= \boldsymbol{U}^{[k]^\dagger}\boldsymbol{y}^{[k]} \\ &= \boldsymbol{U}^{[k]^\dagger}\boldsymbol{H}^{[kk]}\boldsymbol{V}^{[k]}\boldsymbol{x}^{[k]} + \boldsymbol{U}^{[k]^\dagger}\sum_{\substack{j=1,\\ j\neq k}}^{K}\boldsymbol{H}^{[kj]}\boldsymbol{V}^{[j]}\boldsymbol{x}^{[j]} + \boldsymbol{U}^{[k]^\dagger}\boldsymbol{z}^{[k]}\end{aligned} \tag{1}$$

where the first term represents the desired signal, and the second one denotes undesired inter-user interference. In the above, $\boldsymbol{y}^{[k]}$ is the N-dimensional signal at receiver k, $\boldsymbol{H}^{[kj]}$ is the $N \times M$ channel matrix from transmitter j to receiver k (assumed to have i.i.d Gaussian entries), $\boldsymbol{V}^{[j]}$ and $\boldsymbol{U}^{[k]}$ are the $M \times d$ precoding and $N \times d$ receive filters of transmitter j and receiver k ($(k,j) \in \kappa^2$), respectively. Furthermore, $\boldsymbol{z}^{[k]}$ is the N-dimensional zero-mean AWGN vector with covariance matrix $\sigma^2\boldsymbol{I}_N$, and $\boldsymbol{x}^{[k]}$ the d-dimensional vector of transmit symbols intended to receiver k, with covariance matrix $E[\boldsymbol{x}^{[k]}\boldsymbol{x}^{[k]^\dagger}] = (P/d)\boldsymbol{I}_d$. We also denote the interference covariance matrix at receiver k by,

$$\boldsymbol{Q}^{[k]} = \sum_{\substack{j=1\\ j\neq k}}^{K}(P/d)\boldsymbol{H}^{[kj]}\boldsymbol{V}^{[j]}\boldsymbol{V}^{[j]^\dagger}\boldsymbol{H}^{[kj]^\dagger}. \tag{2}$$

The main premise of IA is to find transmit and receive filters such that every receiver would be able to suppress all inter-user interference (i.e., second term in (1)). A common approach, first proposed by [3], is to iteratively minimize the so-called interference leakage, by alternately optimizing the transmit and receive filters, and exploiting channel reciprocity. If zero-forcing receivers are assumed (where the columns of $\boldsymbol{U}^{[k]}$ are chosen such as to span the subspace with lowest interference at receiver k - as described in [3]), we can write the associated optimization problem as follows,

$$\begin{aligned}(P1) \quad &\min_{\boldsymbol{V}^{[1]},\dots,\boldsymbol{V}^{[K]}} \; f = \sum_{k=1}^{K} f^{[k]} = \sum_{k=1}^{K}\sum_{i=1}^{d}\lambda_i[\boldsymbol{Q}^{[k]}] \\ &s.t. \;\; \boldsymbol{V}^{[k]^\dagger}\boldsymbol{V}^{[k]} = \boldsymbol{I}_d, \quad k \in \kappa\end{aligned}$$

where $f^{[k]}$ and f are the interference leakage at receiver k and the total interference leakage, respectively. It is well known that the leakage minimization problem is non-convex, and thus very little is known about global solutions to the above problem. Thus we restrict ourselves in this work to feasible IA problems, (as defined in [8]), because the latter have known global minima . In addition, because not much is known about the form of IA solutions for generic systems, the only measure of the "closeness of a solution" (that a particular algorithm yields) to the global minimum, is whether the leakage value is lower than an arbitrarily small tolerance ϵ. With this in mind we define the convergence reliability measure, as the probability that the leakage is reduced below a predetermined fixed threshold ϵ, after l iterations of the algorithm, i.e.,

$$R_\epsilon(l) = Pr[f_l \leq \epsilon], \;\; l = 0, 1, \dots \tag{3}$$

In the next section, we will motivate our reasons for adopting such an approach, in the sense that it addresses the limitations of a wide class of previously proposed algorithms (as discussed in Section I).

If the IA problem is feasible, then there exists at least a set of transmit and receive filters that satisfies the IA conditions, and for which $f = 0$. Since f is non-negative, then $f = 0$ is the global minimum of $(P1)$.

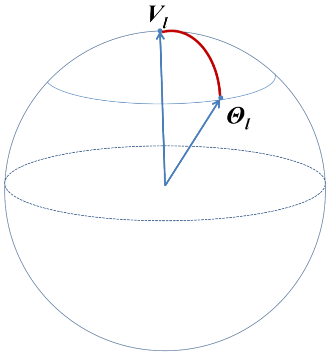

Fig. 1. Transmit fitter update (in prevalent approaches)

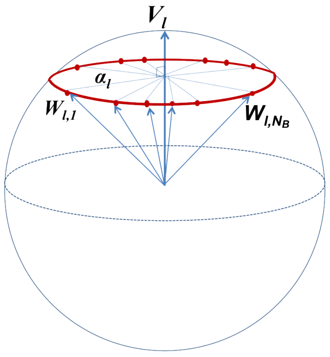

Fig. 2. Transmit filter update (proposed)

III. Proposed Algorithm

A. Algorithm Description and Motivation

We first present the proposed update rule (later detailed and motivated), as

$$\begin{aligned}&\boldsymbol{V}_{l+1}^{[j]} = \sqrt{1-\alpha_l^2}\,\boldsymbol{V}_l^{[j]} + \alpha_l\,\boldsymbol{\Theta}_l^{[j]} \\ &0 \leq \alpha_l \leq 1, \; \boldsymbol{\Theta}_l^{[j]} = \tilde{\boldsymbol{V}}_l^{[j]}\,\boldsymbol{F}_l^{[j]}, \quad j \in \kappa, \quad l = 0, 1, 2, \dots\end{aligned} \tag{4}$$

where $\tilde{\boldsymbol{V}}_l^{[j]} \in (\boldsymbol{V}_l^{[j]})^\perp, \tilde{\boldsymbol{V}}_l^{[j]} \in \mathcal{U}(M,d)$, and $\boldsymbol{F}_l^{[j]} \in \mathcal{U}(d,d)$ is a random isotropic unitary matrix. Note that although it is clear from (4) that there is an implicit constraint on d ($d \leq M/2$), it is not restrictive (since IA anyway requires each transmitter to use no more than half of its total signaling dimensions, for the alignment to be possible).

A possible interpretation of (4) is done by drawing an analogy to statistics. Thus, (4) implies that we iteratively refine our estimate of the current solution $\boldsymbol{V}_l^{[j]}$ (a random variable), by adding another random variable $\boldsymbol{\Theta}_l^{[j]}$. By choosing $\boldsymbol{\Theta}_l^{[j]}$ to be uncorrelated with $\boldsymbol{V}_l^{[j]}$ we would be adding "information" that is not already included in $\boldsymbol{V}_l^{[j]}$. However, since in our case the variables are deterministic, requiring $\boldsymbol{V}_l^{[j]}$ and $\boldsymbol{\Theta}_l^{[j]}$ to be uncorrelated is equivalent to requiring them to be orthogonal, making the analogy complete. Note as well that another motivation for adopting an update rule such as (4), is that it is the optimal subspace representation if one wishes to express $\boldsymbol{V}_{l+1}^{[j]}$, in terms of elements that are missing from $\boldsymbol{V}_l^{[j]}$.

First, we illustrate the fundamental differences between our proposed algorithm, and the prevalent approaches to solving the leakage minimization problem. Note that the latter can be modeled by the following generic update rule: $\boldsymbol{V}_{l+1}^{[j]} =$

$\boldsymbol{V}_l^{[j]} + \gamma_l \boldsymbol{\Theta}_l^{[j]}, \boldsymbol{V}_l^{[j]} \in \mathcal{U}(M,d), \boldsymbol{\Theta}_l^{[j]} \in \mathcal{U}(M,d),\ 0 \leq \gamma_l \leq 1, \forall\ j \in \kappa,\ l = 0,1,2,...$, i.e., by moving according to some fixed predetermined direction $\boldsymbol{\Theta}_l^{[j]}$ (such as [9] and [7] that we use for comparison, due to the presence of unitary constraints as well). Thus, given fixed $\boldsymbol{V}_l^{[j]}$ and $\boldsymbol{\Theta}_l^{[j]}$, a search is performed over γ_l. As a result, the search space for $\boldsymbol{V}_{l+1}^{[j]}$ (the set of "candidate solutions") is the geodesic between $\boldsymbol{V}_l^{[j]}$ and $\boldsymbol{\Theta}_l^{[j]}$ (Fig. 1). It can be seen that since $\boldsymbol{V}_l^{[j]}$ and $\boldsymbol{\Theta}_l^{[j]}$ are not orthogonal in such approaches, the said search space may be limited, and we may be "missing out" on potentially good solutions. On the other hand, with our approach (as will be detailed later), given α_l, the search space at every iteration is given by the circumference of the circle of radius α_l, and centred around $\boldsymbol{V}_l^{[j]}$, as shown in Fig. 2 (note that if in addition, a line search over α_l is performed at every iteration, this would correspond to an exhaustive search over the entire optimization space).

For such reasons, it can be seen that such an approach is expected to yield better performance. Furthermore, since $\boldsymbol{\Theta}_l^{[j]}$ represents a "greedy direction" in many cases (gradient of the cost function such as [9] and [7], or the direction that minimizes the MSE such as [4]), such algorithms are susceptible to get trapped in local optima. This is effectively remedied in our approach since $\boldsymbol{\Theta}_l^{[j]}$ has a stochastic element.

B. Algorithm Details

We will first present the main steps of the algorithm, and detail them later on.

```
Start with random unitary $\boldsymbol{V}_1^{[1]},...,\boldsymbol{V}_1^{[K]}$
for $l = 0,1,2,....$
    Compute $\alpha_l$
    if $(f(\boldsymbol{V}_l^{[1]},...,\boldsymbol{V}_l^{[K]}) \leq \epsilon)$ then stop
    Generate $\mathcal{B}_l^{[j]} = \{\boldsymbol{F}_{l,n}^{[j]}\}_{n=1}^{N_B},\ \forall j \in \kappa$
    Generate $\tilde{\boldsymbol{V}}_l^{[j]} \in (\boldsymbol{V}_l^{[j]})^{\perp},\ \forall j \in \kappa$
    Compute $\boldsymbol{W}_{l,n}^{[j]} = \sqrt{1-\alpha_l^2}\ \boldsymbol{V}_l^{[j]} + \alpha_l(\tilde{\boldsymbol{V}}_l^{[j]})\boldsymbol{F}_{l,n}^{[j]}, \forall n,\ \forall j$
    $\tilde{\boldsymbol{F}}_l^{[j]} = \underset{n}{argmin}\ f(\boldsymbol{W}_{l,n}^{[1]},...,\boldsymbol{W}_{l,n}^{[K]}),\ \forall j \in \kappa$
    $\boldsymbol{V}_{l+1}^{[j]} = \sqrt{1-\alpha_l^2}\ \boldsymbol{V}_l^{[j]} + \alpha_l\ \tilde{\boldsymbol{V}}_l^{[j]}\tilde{\boldsymbol{F}}_l^{[j]},\ \forall j \in \kappa$
end
```

Firstly a codebook $\mathcal{B}_l^{[j]} = \{\boldsymbol{F}_{l,n}^{[j]} \in \mathcal{U}(d,d)\}_{n=1}^{N_B}$ of isotropic random unitary matrices is generated. Then "candidate solutions" for the next iteration are computed according to the update rule (4), $\boldsymbol{W}_{l,n}^{[j]} = \sqrt{1-\alpha_l^2}\ \boldsymbol{V}_l^{[j]} + \alpha_l \tilde{\boldsymbol{V}}_l^{[j]}\boldsymbol{F}_{l,n}^{[j]}, \forall n$. Stated differently, the update is done by *perturbing the current solution along its orthogonal direction, on the circumference of a circle of radius* α_l, on the unitary hypersphere (Fig. 2). In fact, it can be easily verified that all the generated candidate solutions all lie at a distance α_l from the current solution $\boldsymbol{V}_l^{[j]}$, since $d_c(\boldsymbol{V}_l^{[j]}, \boldsymbol{W}_{l,n}^{[j]}) = (\sqrt{d - \|\boldsymbol{V}_l^{[j]\dagger}\boldsymbol{W}_{l,n}^{[j]}\|_F^2})/\sqrt{d} = (\sqrt{d-(1-\alpha_l^2)\|I_d\|_F^2})/\sqrt{d} = \alpha_l,\ \forall\ l,\ \forall\ n$. Then, out of the N_B candidate solutions, we select the one that results in the lowest value of the cost function. We note as well that this update preserves the unitary constraint in $(P1)$ (it is easy to verify that $\boldsymbol{V}_{l+1}^{[j]} \in \mathcal{U}(M,d)$ if $\boldsymbol{V}_l^{[j]} \in \mathcal{U}(M,d)$), and ensures that the optimization is performed on the Grassmann manifold.

As we just saw, the essence of the algorithm is to perturb around the current solution (by generating random codewords along the circumference of a circle). However, the "amount" of perturbation that is introduced at every iteration (that we refer to as perturbation radius), i.e. α_l, is crucial to the algorithm's operation. A heuristic for choosing the perturbation radius, is shortly discussed next.

C. Choice of perturbation radius

It is easy to see that fixing α_l is a problematic choice - even if chosen to be relatively large or small, since in the former the algorithm will never converge, while in the latter, convergence will be extremely slow. This said, we quickly realize that a smart heuristic would be to choose α_l as decreasing with l . This issue is addressed more thoroughly by deriving analytical bounds that govern the choice of α_l, in Section IV

IV. Convergence Analysis

A. Convergence to a stationary point

In this section we derive analytical expressions relating the change in the leakage at receiver k, to α_l. In the derivations below, we assume single stream transmission for simplicity (although they can be extended to the case where $d > 1$) . For that purpose, we first use the update rule in (4), to write

$$\boldsymbol{Q}_{l+1}^{[k]} = \sum_{\substack{j=1 \\ j\neq k}}^{K} (P/d)\boldsymbol{H}^{[kj]}\boldsymbol{V}_{l+1}^{[j]}\boldsymbol{V}_{l+1}^{[j]\dagger}\boldsymbol{H}^{[kj]\dagger}$$
$$= (1-\alpha_l^2)\boldsymbol{Q}_l^{[k]} + \alpha_l\sqrt{1-\alpha_l^2}\boldsymbol{D}_l^{[k]} + \alpha_l^2\boldsymbol{E}_l^{[k]} \qquad (5)$$

where $\boldsymbol{Q}_l^{[k]}$ and $\boldsymbol{E}_l^{[k]}$ are positive semi-definite, while $\boldsymbol{D}_l^{[k]}$ is Hermitian (we omitted the expressions for $\boldsymbol{E}_l^{[k]}$ and $\boldsymbol{D}_l^{[k]}$). Now using (5), we express the change in the leakage at receiver k, as

$$\Delta f_l^{[k]} = \left|f_{l+1}^{[k]} - f_l^{[k]}\right| = \left|\ \lambda_1\left[\boldsymbol{Q}_{l+1}^{[k]}\right] - \lambda_1[\boldsymbol{Q}_l^{[k]}]\ \right|$$
$$= \left|\ \lambda_1[\underbrace{\boldsymbol{Q}_l^{[k]}}_{A}\underbrace{-\alpha_l^2\boldsymbol{Q}_l^{[k]} + \alpha_l\sqrt{1-\alpha_l^2}\boldsymbol{D}_l^{[k]} + \alpha_l^2\boldsymbol{E}_l^{[k]}}_{B}] - \lambda_1[\underbrace{\boldsymbol{Q}_l^{[k]}}_{A}]\ \right|$$
$$\leq \| -\alpha_l^2\boldsymbol{Q}_l^{[k]} + \alpha_l\sqrt{1-\alpha_l^2}\boldsymbol{D}_l^{[k]} + \alpha_l^2\boldsymbol{E}_l^{[k]}\|_2$$
$$\leq \alpha_l^2\|\boldsymbol{Q}_l^{[k]}\|_2 + \alpha_l\sqrt{1-\alpha_l^2}\|\boldsymbol{D}_l^{[k]}\|_2 + \alpha_l^2\|\boldsymbol{E}_l^{[k]}\|_2$$
$$\leq \alpha_l\left(\|\boldsymbol{Q}_l^{[k]}\|_2 + \|\boldsymbol{D}_l^{[k]}\|_2 + \|\boldsymbol{E}_l^{[k]}\|_2\right) \overset{\triangle}{=} c_o\alpha_l. \qquad (6)$$

Note that the first inequality follows from the fact that $|\lambda_1[A+B] - \lambda_1[A]\ | \leq \|B\|_2$ for symmetric matrices A, B where $\|B\|_2 = \max(|\lambda_1[B]|, |\lambda_N[B]|)$ is the l_2 norm applied to symmetric matrices [10] . Furthermore, the second inequality follows from the triangle inequality, while the last one follows from the following trivial bounds: $\alpha_l\sqrt{1-\alpha_l^2} = \sqrt{\alpha_l^2 - \alpha_l^4} \leq \alpha_l$ and $\alpha_l^2 \leq \alpha_l$. Furthermore, note that each of the norms in (6) can be bounded by a constant term (it is easy to verify that they can be trivially bounded by $\sum_{j=1,\ j\neq k}^{k} \lambda_N[\boldsymbol{H}^{[kj]^\dagger}\boldsymbol{H}^{[kj]}]$) thus ensuring that none of them scales up, as the iteration number l increases. If α is decreasing in l, then the bound on the change in the function, Δf_l, is decreasing as well, thus ensuring convergence of the proposed update to a stationary point of the proposed algorithm. Note that this entirely supports the heuristic that was discussed in Section III-C, for picking α_l as decreasing.

A stationary point is defined as any iteration index l, corresponding to a change in the function that is below a predefined tolerance δ. Thus we give the following definition.

Definition 1. *A point f_l is stationary w.r.t. the algorithm, if $f_l \in \mathcal{S}$, where $\mathcal{S} = \{f_{p+1}\ :\ |f_{p+1} - f_p| < \delta\},\ \delta > 0,\ p = 0, 1, 2, ...$*

Theorem 1. *If $\{\alpha_l\}_{l=0}^{\infty}$ is a monotonically decreasing sequence in l, such that $0 \leq \alpha_l \leq 1,\ \forall\ l$, and $\lim\limits_{l\to\infty}\ \alpha_l = 0$, the algorithm always converges to a stationary point $f_{st} \in \mathcal{S}$.*

Proof: Using (6) we can write $\Delta f_l^{[k]} = |f_{l+1}^{[k]} - f_l^{[k]}| \leq c_o\alpha_l$. Then, for every $\delta > 0$, there exists some integer m such that $|f_{m+1}^{[k]} - f_m^{[k]}| \leq c_o\alpha_m \leq \delta$ (since $\lim\limits_{l\to\infty}\ \alpha_l = 0$). Thus, the sequence $\{f_l^{[k]}\}_l$ is a Cauchy sequence, and thus it converges to some point-wise limit, i.e., $f_l^{[k]} \to f_{st}^{[k]},\ \forall\ k \in \kappa$. Thus, we conclude that $f_l = \sum_k f_l^{[k]}\ \to \sum_k f_{st}^{[k]} \triangleq f_{st}$, implying that the algorithm will always converge to a stationary point f_{st} which will be the solution that the algorithm yields ■

Remark 1. *It can be easily inferred from (6), that the faster / slower α_l decays, the faster / slower the algorithm converges to a stationary point. However, the latter theorem does not give any insights about the average quality of this stationary point (this will be the object of the discussion in the next subsection, and simulations will later show that fast decaying sequences, yield stationary points with worse average values).*

B. Slow versus fast decaying perturbations

Although we motivated the need for α_l to be decreasing, many times over, we still have to distinguish between two important cases: slow and fast decaying sequences (with respect to some metric that we need not specify here). As discussed in Section III-B, the perturbation is done by generating a set of N_B "candidate solutions" (that will act as the search space). It was shown in the latter section that the said candidate solutions all lie at a chordal distance of α_l from the current one, i.e., $d_c(\boldsymbol{V}_l^{[j]}, \boldsymbol{W}_{l,n}^{[j]}) = \alpha_l,\ j \in \kappa,\ \forall\ n = 1, ..., N_B$ (Fig 1). It is not too difficult to see that if N_B and α_l are relatively large, then the search space is large enough such that function is not constant over it. Then there exists an index $n^* \leq N_B$ in the codebook that results in a lower function value, i.e. $\exists(n_1^*, ..., n_K^*)$ such that $f(\boldsymbol{W}_{l,n_1^*}^{[1]}, ..., \boldsymbol{W}_{l,n_K^*}^{[K]}) \leq f(\boldsymbol{V}_l^{[1]}, ..., \boldsymbol{V}_l^{[K]})$. Following this line of thought, as α_l is reduced, so is the size of the search domain. Then one can see that if α_l *is drastically reduced, then it is possible (and even likely) that we have discarded potentially global solutions, and would be settling for local minima.* With this in mind, if the sequence α_l is fast decaying, although convergence to a stationary point is fast (refer to $Definition$ 1), the resulting cost function value however, is not good on average (mostly local optima). Conversely, if α_l is slow decaying, the search space defined by $d_c(\boldsymbol{V}_l^{[j]}, \boldsymbol{W}_{l,n}^{[j]})$ does not decrease significantly at each iteration. Thus, *although convergence to a stationary point is slower in this case, it is steady and is likely to escape local minima, thus resulting in stationary points with better values.*

Remark 2. *Furthermore, for slow decaying sequences, rough preliminary analytical results seem to suggest that the average cost function value can be upperbounded by a deterministic sequence, i.e., $\mathbb{E}[f_l] \leq K_s\alpha_l^2$ (K_s is a constant).*

C. Exponentially decaying perturbations

Motivated by the need to modify the decay of α_l , we chose exponentially decaying sequences, i.e. $\{\alpha_l\}_l = 10^{-l/(2T)}$, because they provide a convenient way to make the decay faster / slower by decreasing / increasing the parameter $T > 0$. If the result of $Remark$ 2 holds, it implies that for such sequences, the ergodic performance is solely determined by the choice of the sequence - a deterministic function, i.e. $\mathbb{E}[f_l] \leq K_s\alpha_l^2$. Thus, this yields an upperbound on the average number of iterations, η_T , required to reach a desired tolerance ϵ, i.e. η_T is defined as $\eta_T = \{l\ :\ K_s\alpha_l^2 \approx \epsilon\} = \{l\ :\ K_s 10^{-l/T} \approx \epsilon\} \approx T\ log_{10}(K_s/\epsilon)$.

We simulate in the next section various average performance metrics (average convergence, convergence reliability, and ergodic sum-rate), and compared them to the well known distributed IA algorithm [3].

V. Numerical Results

A. 2×2, 3-user MIMO IC, $d = 1$

1) Average convergence: We first simulated the average behaviour of our scheme, for different values of the decay parameter T (by averaging over 500 realizations of both our scheme and distributed IA). We use a codebook size of $N_B = 32$, resulting in slightly higher - but comparable - running time of our algorithm, w.r.t. distributed IA. Several conclusions can be drawn from Fig. 3.

First, we observe that the average cost function value (solid lines) has a fast decaying phase, whose initial slope (for small values of l) can be approximated by α_l^2 (dashed lines), after which it reaches a stationary point. However, it is clear that the said stationary point is worse for fast decaying sequences.

Thus, we now see that there is a tradeoff on T: for small values (e.g. $T = 8$) we have fast convergence to a stationary point. Conversely, for larger values of T we have a slow but steady decay in the function, resulting in stationary points with better average values. This observation *supports both Remark* 1 *and the discussion that followed in Section IV-B*

Furthermore, interestingly, we see that the results in Fig. 3 *seems to support the claim that we made in Remark* 2 stating that the average cost function value, $\mathbb{E}[f_l]$, is upperbounded by $K_s\alpha_l^2$, for slow decaying sequences. This being said, it can be seen that as T increases (α_l decays slower), both $\mathbb{E}[f_l]$ (solid lines), and α_l^2 (dashed lines) have the same slope (implying that the bound in *Remark* 2 is correct, up to a scaling constant K_s that only induces a vertical shift in the semi-log scale of Fig. 3, and that can thus be neglected for our purposes).

Finally, Fig. 3 shows the significant gains in average cost function value, of our approach over distributed IA (regardless of the choice of T). However, since the average will be dominated by just a small number of realizations, Fig. 3 does not convey the full picture.

2) An in-depth look at the convergence of distributed IA: Motivated by this last observation, we plotted overlaid individual convergence curves for both algorithms (Fig. 4). It is clear that there are cases when distributed IA (dotted lines) offers an extremely poor performance - although it has a much better performance than our scheme in many cases. Unfortunately, since the cost function average will be dominated by those "worst case scenarios", it is now clear why our scheme (dashed lines) will perform better on average: it has a robust, quasi-deterministic performance, i.e. it follows the sequence decay α_l^2, except for the few cases where it goes to local optima. This is due to the fact that the decay in α_l is "too fast" for this particular realization, and the algorithms end up at a local minima (exactly the effect that we earlier described in Section IV-B). Based on this same discussion, this can be remedied by using sequences with slower decay, i.e., the number of such "worst case scenarios" can be reduced by increasing T. Summing up, the steady performance, significantly improves the "worst case scenarios" - a limiting factor on the average performance, and thus resulting in significant gains that were seen in Fig. 3.

3) Convergence Reliability: We simulate what we earlier referred to as convergence reliability (3), using a tolerance $\epsilon = 10^{-6}$. Two conclusions can be drawn by observing Fig 5).

Firstly, the tradeoff on T that we earlier discussed, is reiterated: for small values of T, fast convergence can be achieved at the expense of a lower convergence reliability, e.g. for $T = 8$, the algorithm reaches the specified tolerance in less than 60 iterations with probability 90%. By increasing T to 12, we can improve this figure to 98%, at the expense of a slower convergence. Thus, by *changing T, we alter the the convergence of the algorithm to achieve the desired behaviour*.

Interestingly, we observe as well, that as T increases,

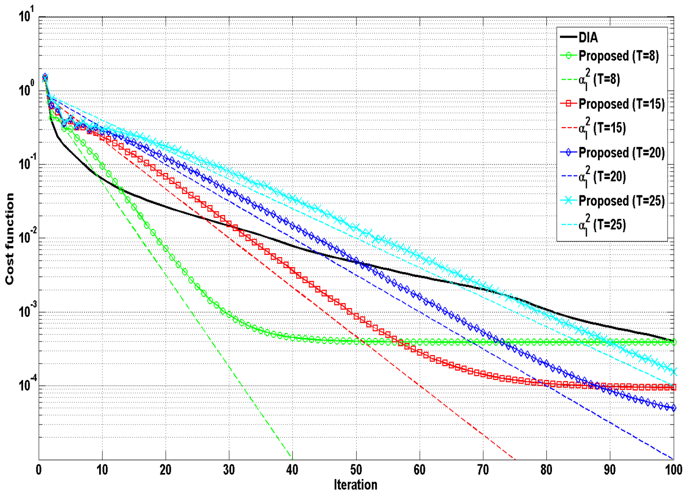

Fig. 3. Average convergence of proposed scheme for different values of T, and of Distributed IA

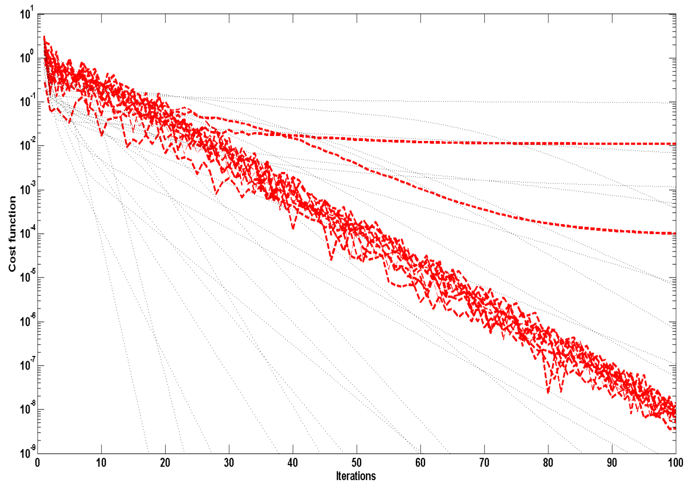

Fig. 4. Snapshots of individual convergence curves for both distributed IA (dotted lines), and our proposed scheme (dashed lines)

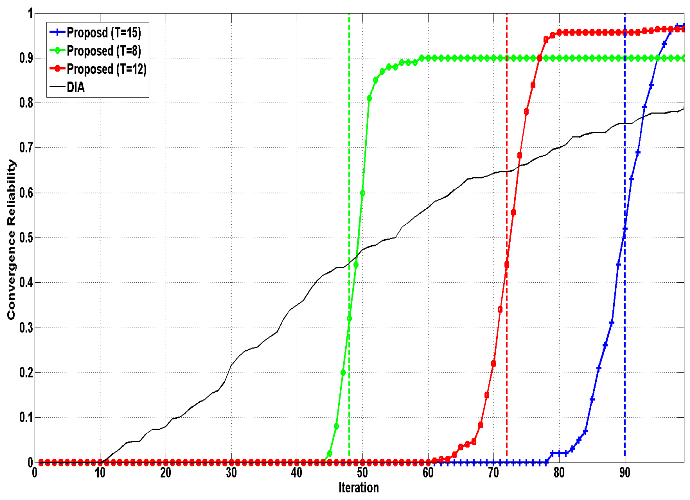

Fig. 5. Convergence reliability, for 2×2 MIMO IC, $K = 3, d = 1, \epsilon = 10^{-6}$

the simulated convergence reliability (solid lines) starts to approximate a unit step function $u(l-\eta_T)$ (dashed lines) located at η_T. As a result, one might be tempted to conjecture that as $T \to \infty$, $R_\epsilon(l) \approx u(l-\eta_T)$. If the latter result can be proved analytically (for arbitrarily small ϵ), it implies that the convergence of this algorithm to the specified tolerance ϵ happens almost surely (when T is large).

Lastly, we observe in Fig 5 that there are *significant improvements over Distributed IA, both in terms of convergence speed and reliability*.

4) Sum-rate performance: Although our algorithm seems to outperform Distributed IA in all aspects, it is however unrealistic to expect such improvements in terms of sum-rate (since distributed IA gets arbitrarily close the sum-rate degrees-of-freedom upperbound). As a matter of fact, any IA algorithm will have this property, if left to run for a large enough number of iterations (an impractical assumption). Motivated by this observation, and exploiting the tradeoff that our algorithm offers (i.e. by picking a small T to achieve fast convergence) we simulate the sum-rate performance of the single stream 3-user 2×2 MIMO IC, while fixing the number of iterations for both algorithms. We can see from Fig 6 that for a small number of iterations, e.g. 40, both algorithms (shown in solid lines) fail to achieve the high-SNR slope of the sum-rate function, as predicted by IA. However, we see that this effect is significantly more pronounced in Distributed IA, as the gap between both algorithms grows with increasing SNR.

VI. Conclusion and Future Work

We exploited the geometry of the unitary constraints to propose a novel leakage minimization algorithm, based on adaptively controlled perturbations. We showed that by decreasing the magnitude of the latter, with each iteration, we can control the convergence speed and reliability. As a result, with such an approach convergence speed can be traded-off for convergence reliability to achieve the desired behaviour. We also shed some light on some undesirable characteristics of distributed IA related to the fact that its average cost function value is limited by the some worse case performances. As a result, exploiting both the flexibility and quasi-deterministic convergence of our proposed algorithm allowed us to improve on the latter cases, and thus result in significant average performance gains (both in terms of average cost function value, sum-rate capacity, and convergence reliability).

We note that the work that was done here is a first step towards designing algorithms whose ergodic performance can be mathematically analysed. Related problems left for future work include performance analysis of the proposed scheme, analysis of the effect of codebook size, and the possibility of a distributed implementation of the above scheme.

adaptively

Acknowledgment

The authors wish to acknowledge the many stimulating discussions and the valuable help of Rasmus Brandt in reviewing this paper.

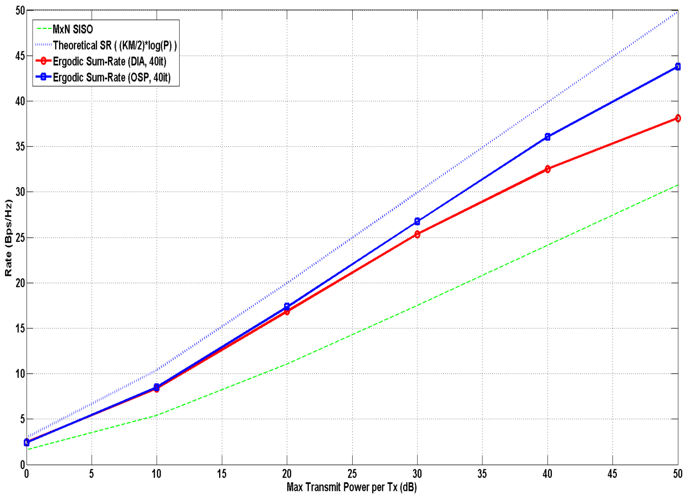

Fig. 6. Ergodic sum-rate of 2×2 MIMO IC, $K = 3,\ d = 1$, 40 iterations

References

[1] V. Cadambe and S. Jafar, "Interference alignment and degrees of freedom of the k -user interference channel," *Information Theory, IEEE Transactions on*, vol. 54, no. 8, pp. 3425–3441, 2008.

[2] T. Gou and S. Jafar, "Degrees of freedom of the k user m times n mimo interference channel," *Information Theory, IEEE Transactions on*, vol. 56, no. 12, pp. 6040–6057, 2010.

[3] K. Gomadam, V. R. Cadambe, and S. A. Jafar, "A distributed numerical approach to interference alignment and applications to wireless interference networks," *IEEE Transactions on Information Theory*, vol. 57, pp. 3309–3322, June 2011.

[4] D. Schmidt, C. Shi, R. Berry, M. Honig, and W. Utschick, "Minimum mean squared error interference alignment," in *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, pp. 1106 –1110, Nov. 2009.

[5] S. W. Peters and R. W. Heath, "Cooperative algorithms for MIMO interference channels," *IEEE Transactions on Vehicular Technology*, vol. 60, pp. 206–218, Jan. 2011.

[6] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4331–4340, 2011.

[7] I. Santamaria, O. Gonzalez, R. W. Heath, and S. W. Peters, "Maximum sum-rate interference alignment algorithms for MIMO channels," in *2010 IEEE Global Communications Conference (GLOBECOM 2010)*, pp. 1–6, IEEE, Dec. 2010.

[8] C. M. Yetis, T. Gou, S. A. Jafar, and A. H. Kayran, "On feasibility of interference alignment in MIMO interference networks," *IEEE Transactions on Signal Processing*, vol. 58, pp. 4771–4782, Sept. 2010.

[9] H. Ghauch and C. Papadias, "Interference alignment: A one-sided approach," in *2011 IEEE Global Communications Conference (GLOBECOM 2011)*, pp. 1 –5, Dec. 2011.

[10] G. H. Golub and C. F. Van Loan, *Matrix computations (3rd ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.