Implementation of a Real-Time Wireless Interference Alignment Network

Jackson W. Massey, Jonathan Starr, Seogoo Lee, Dongwook Lee, Andreas Gerstlauer, and Robert W. Heath Jr.
Wireless Networking and Communications Group, Dept. of Elec. and Comp. Engineering
The University of Texas at Austin
Email: {jackson.massey, jonstarr, sglee, dongwook.lee}@utexas.edu, {gerstl, rheath}@ece.utexas.edu

***Abstract*—Interference alignment (IA) is a cooperative transmission technique for the interference channel.This paper describes two testbeds that implement real-time Multiple-input multiple-output (MIMO) IA for a network with three 2-antenna user pairs using software defined radio techniques: a PC-based testbed for rapid prototyping of potential IA protocols and an embedded testbed for evaluating IA under real-world computational constraints. The IA implementations rely on a wired backbone to share global channel state information (CSI) and a shared clock for frequency and timing synchronization. The testbeds are used to demonstrate the viability of IA, and to compare its robustness with several alternative transmission strategies, such as 2×2 MIMO TDMA, in terms of sum-rates. Results show that we are able to successfully achieve over-the-air IA in our three-user 2×2 MIMO testbed. The paper highlights key challenges with the practical realization of IA that are encountered while developing the testbed and identifies areas for future research.**

I. Introduction

Multiple-input multiple-output (MIMO) interference alignment (IA) is a transmission strategy for interference channels that scales the network's sum-rate linearly, at high signal-to-noise ratio (SNR), with the number of users, under certain dimensionality requirements [1]. This is achieved by cooperatively designing linear precoders that are applied to the transmitted signals such that the interfering signals at the receivers can be reduced to a subspace orthogonal to the desired signal. As a result, the desired signal can be projected into the interference-free subspace. Most of the work in IA has been theoretical and validated only with simulations that are based on several assumptions, which may not hold in reality. Therefore, for MIMO IA to become practical, real-world, over-the-air MIMO IA systems must be implemented to validate theoretical assumptions and demonstrate feasibility.

Previous work on practical IA applications includes the approach in [2] where channel measurements were performed in indoor and outdoor environments and subsequently applied to IA simulations. In their work, however, IA was not performed over-the-air, ignoring the complications that can arise from frequency and time synchronization. In [3], a testbed was created that uses interference alignment and cancellation. The paper discusses MAC and PHY layer issues, including how to extend the system to more antennas, but only implements a 2×2 system. More recently, in [4], a 3-user MIMO IA testbed (6×6 total number of antennas) was created that implements IA over-the-air and operates at 5 GHz. This testbed compares IA and TDMA, but does not examine the computational costs. IA is known to be computationally demanding, making it crucial to study fundamental limits and practically achievable performance on the type of mobile platforms that will eventually be the target of any real-world IA deployment.

In this paper, we describe the implementation two 3-user, 2×2 MIMO IA testbeds using a software defined radio (SDR) platform [5]. The testbeds use the same USRP hardware but one does the processing using PCs while the other uses an embedded platform[1]. We describe the setup of the testbed, the insights gained and the challenges we encountered in going from theory to reality. We present results demonstrating that large-scale over-the-air IA is achievable in a a real testbed. Furthermore, we optimize the implementation and reduce computational complexity as a first step toward an embedded IA implementation.

II. Interference Alignment Background

Consider a MIMO system with K-users that has N_{tx} transmit antennas at transmitter k and N_{rx} receive antennas at receiver m. All of the users send N_s streams of data using orthogonal frequency-division multiplexing (OFDM) with N subcarriers. Applying a precoder matrix $\mathbf{F}_k$ at each of the transmitters yields the following signal at the kth receiver, where $\mathbf{y}_k$ is the $N_{\text{rx}} \times 1$ received signal vector, $\mathbf{H}_{k,m}$ is the $N_{\text{rx}} \times N_{\text{tx}}$ channel matrix from transmitter m to receiver k, $\mathbf{F}_k$ is the precoder applied at transmitter k, $\mathbf{s}_k$ is the $N_s \times 1$ symbol vector sent by transmitter k, and $\mathbf{v}_k$ is a complex vector of i.i.d. circularly symmetric white Gaussian noise with a covariance matrix $\mathbb{E}[\mathbf{v}_k \mathbf{v}_k^*] = \sigma_v^2 \mathbf{I}_{N_k}$:

$$\mathbf{y}_k[n] = \mathbf{H}_{k,k}[n]\mathbf{F}_k[n]\mathbf{s}_k[n] + \sum_{m \neq k} \mathbf{H}_{k,m}[n]\mathbf{F}_m[n]\mathbf{s}_m[n] + \mathbf{v}_k[n]. \quad (1)$$

Applying a receiving filter at the kth receiving node, where $\mathbf{W}_k$ is the receiving filter at receiver k, yields

$$\mathbf{W}_k[n]\mathbf{y}_k[n] = \mathbf{W}_k[n]\mathbf{H}_{k,k}[n]\mathbf{F}_k[n]\mathbf{s}_k[n] + \sum_{m \neq k} \mathbf{W}_k[n]\mathbf{H}_{k,m}[n]\mathbf{F}_m[n]\mathbf{s}_m[n] + \mathbf{W}_k[n]\mathbf{v}_k[n]. \quad (2)$$

[1]The code for both testbeds is available at: http://www.profheath.org/research/interference-alignment/mimo-ofdm-interference-alignment-testbed/

TABLE I: OFDM Parameters

FFT Length	128
Cyclic Prefix Length	6
Number of Null Subcarriers	23
Number of Pilot Subcarriers	9
Number of Data Subcarriers	96

The precoders and receiving filters are chosen such that

$$\mathbf{W}_k[n]\mathbf{H}_{k,m}[n]\mathbf{F}_m[n]\mathbf{s}_m[n] \approx 0 \text{ for } m \neq n \tag{3}$$

leaving

$$\mathbf{W}_k[n]\mathbf{y}_k[n] = \mathbf{W}_k[n]\mathbf{H}_{k,k}[n]\mathbf{F}_k[n]\mathbf{s}_k[n] + \mathbf{W}_k[n]\mathbf{v}_k[n] \tag{4}$$

at the kth receiving node.

In this paper, we utilize an approach that calculates precoders and receiving filters using the alternating minimization of interference leakage (AMIL) algorithm developed in [6]. AMIL forms two minimization problems and alternates between them, minimizing one problem while holding the other constant until a threshold is reached.

It should be noted that the 3-user case with $M_k = 2$ and $N_m = 2$ does have an analytical solution [1]. The AMIL algorithm, however, was implemented for this paper as a more general solution allowing the testbed to be reconfigured to a different user setup later if desired.

In its ideal form as presented so far, IA makes the following theoretical assumptions that are not always feasible in practice: (i) the devices are synchronized in frequency or have at most one carrier frequency offset between all of the transmitters and all of the receivers; (ii) the devices are synchronized in time such that all of the packets are sent at the same time; and (iii) perfect channel state information (CSI) is obtained for the IA algorithm to calculate the precoders and receiving filters. Additionally, several practical challenges must be overcome for IA to become feasible as described: (i) the CSI feedback must be received within the channel coherence time or the CSI is no longer valid and (ii) the IA algorithm must be able to run in real-time on a given hardware.

III. System Setup

Interference alignment is applied to a testbed with 3 user pairs that each use OFDM and 2×2 MIMO to communicate. This setup requires 6 transmitters (working together in pairs) and 6 receivers (also working in pairs). The system is designed to operate in the 2.4 GHz frequency range or the 5 GHz frequency range. The channel bandwidth would ideally be 20 MHz, but is actually lower due to hardware constraints discussed in Section IV. The OFDM parameters used for the testbed are listed in Table I.

Each transmission contains a set of training data for signal detection, carrier frequency offset correction, and channel estimation. The first set of training data is comprised of repeated Zadoff-Chu sequences. For the implementation in Section IV, a length 17 Zadoff-Chu sequence is repeated 5 times followed by a length 29 Zadoff-Chu sequence that is repeated 3 times. This part of the training is the same for all of the antennas at the transmitter. It is used at the receiver to detect when a signal is received and to estimate the carrier frequency offset. The next part of the training is a set of OFDM symbols that are used to estimate the channels between the antennas. Because it is important to have sufficient training data to estimate the CSI correctly, eight training OFDM symbols are used.

TABLE II: Testbed Hardware

Part Description	Part Number
USRP Base Package	Ettus USRP N210 / NI USRP-2921
RF Daughterboard	Ettus XCVR2450
Antenna	Ettus VERT2450

The testbed uses two computers that are each equipped with dual-core Intel Xeon 2.67 GHz processors and 12 GB of memory. One computer is used to control the transmitters while the other is used to control the receivers, as shown in Figure 1. The two computers are connected to the same network and use TCP/IP for the feedback channel.

Twelve Ettus Research USRP N210s transceivers are used [5]. Transmitter or receiver USRP pairs are controlled using a single gigabit Ethernet connection, which is shared between the two USRPs using Ettus Research MIMO cables. A function generator provides a 10 MHz clock and PPS signal (0-5 V, 1 Hz square-wave) to all of the transceivers in order to synchronize their frequency and time, respectively. It should be noted that the MIMO cable can also share clock and PPS signals, but that this approach is not used because of potential phase delays due to the signal traveling over different electrical lengths. Our setup, by contrast, uses a fully balanced tree to distribute the reference signal. Table II lists the hardware used for the USRP setup. Figure 1 shows a diagram of the setup and pictures of the transmitters and receivers.

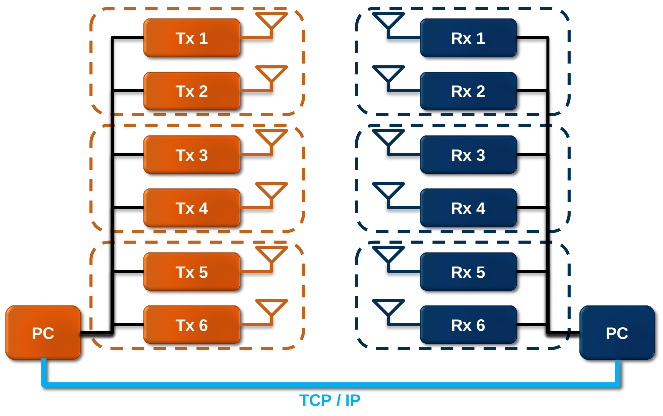
IV. PC Implementation

Our first testbed implements the system setup in LabVIEW on the PCs. LabVIEW is a graphical programming language that is often used to quickly develop systems by controlling hardware with blocks of LabVIEW code called virtual instruments [7]. LabVIEW includes hardware drivers for the USRPs and has several communication toolkits to aid in the development of SDR systems. In our testbed, the hardware control and system was implemented natively in LabVIEW. Most of the communication algorithms, such as the channel estimation and the AMIL algorithm, by contrast, were first written and tested in MATLAB and then ported over to LabVIEW using embedded MathScript blocks.

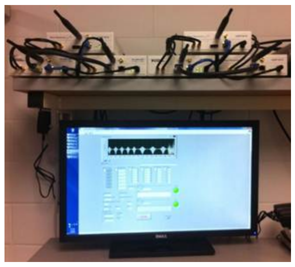
A MIMO-only version of the testbed was first implemented to become familiar with the setup and learn about respective constraints before adding additional complications, such as feedback, which are required for an IA implementation. In the remainder of this section, we describe the challenges associated with each implementation, and how they were overcome. Finally, we present some results of measurements taken on the testbed.

A. MIMO Spatial Multiplexing

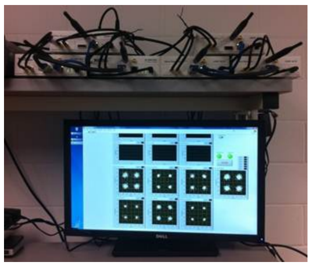
A MIMO implementation requires synchronization (both in time and frequency) between all of the transmitters and receivers in the system. The MIMO code developed for the



(a) Block diagram



(b) Transmitter



(c) Receiver

Fig. 1: IA testbed setup using PCs and USRPs. The dashed line boxes signify the different users. The TCP/IP cable on the bottom connects the two PCs for the feedback channel. A function generator is connected to each transmitter and receiver for synchronization.

testbed initially started with a 2×2 setup to reduce complexity for debugging. In this setup, the MIMO cable was used to share the clock and PPS signals across two devices. Extending to higher order MIMO schemes, such as 3×3 or 6×6, requires a different approach. We utilized a function generator as the external source for synchronization.

Another constraint that we encountered in the MIMO testbed was that the USRPs required a continuous stream of data for transmission or else an underflow occurs at the transmitter. Thus, to turn off the transmitter between packets, zeros need to be continuously fed into the transmitters. This forces the sampling rate of the USRPs to be much lower than the sampling rate of a single packet transmission (1 $\mathrm{MS/s}$ versus > 20 $\mathrm{MS/s}$). Ideally, a bursty mode of operation would be used in which each transmission is scheduled as needed and the transmitter and receiver are off when not in use (instead of transmitting/receiving zeros). Work is currently being done to update the testbed to use this approach.

B. *Interference Alignment*

IA was implemented by modifying the MIMO testbed to include linear precoding, precoder computation, and CSI feedback. In our testbed, a training packet is first sent to determine the CSI. Then, the receivers' PC performs the AMIL algorithm to calculate the IA precoders. The receiver PC then sends the precoders to the transmitter PC via the TCP/IP feedback channel. Finally, the transmitters apply the precoders and then send the data packet. Alternatively, the raw CSI could be sent and the AMIL algorithm could be performed at both the transmitters and receivers.

The feedback channel requires scheduling within the system. Given the continuous transmission constraint mentioned in the previous subsection, the transmitter PC must continuously send zeros to the USRP buffers while receiving the TCP/IP packets and applying the precoders on the data. To accomplish this, a producer-consumer setup was created, where the producer loop collects the feedback, applies the precoders, and then adds the samples to be transmitted to a queue. The consumer loop then checks the queue for elements - transmitting the queued elements if they are present or transmitting zeros if the queue is empty. The producer and consumer loops are run in parallel in LabVIEW.

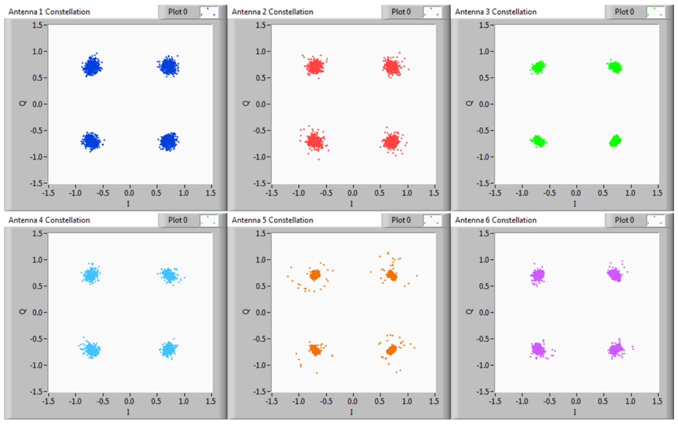


Fig. 2: 6×6 MIMO constellations for each antenna.

C. *Results*

The received constellations for each antenna of a 6×6 MIMO transmission are shown in Figure 2. The constellations indicate that the time and frequency synchronization is working for the MIMO testbed. Similar constellations were also received with the IA testbed. The signal-to-interference-plus-noise ratio (SINR) and SNR for user k, SINR_k and SNR_k, are estimated using the distance between the equalized symbols and the transmitted symbols during an IA and 2×2 MIMO TDMA transmission, respectively. The system SNR is calculated using the measured signal energy during the MIMO TDMA and the noise energy between transmitted packets. To demonstrate IA's effectiveness, the network's sum-rate, $\mathrm{R}_{\text{sum, IA}}$ and $\mathrm{R}_{\text{sum, TDMA}}$, are calculated in a manner similar to [4]. For example, $\mathrm{R}_{\text{sum, IA}} = \sum_{k=1}^{3}\log_2(1+\mathrm{SINR}_k)$. The network's sum-rates are plotted in Figure 3 versus SNR for measured results. The plotted lines show the linear fit for the data. The measured results for the indoor setup lay in between the indoor results in [2].

V. Embedded Implementation

The ultimate objective of our project is to implement the MIMO IA system on embedded platforms. The prototype implementation using LabVIEW is targeting a PC-type setup

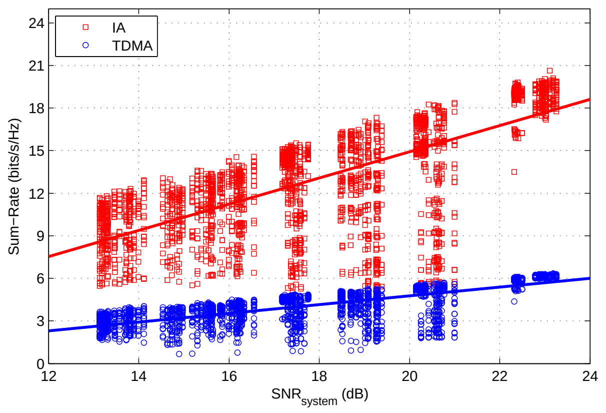


Fig. 3: The network sum-rate vs. SNR for the IA testbed.

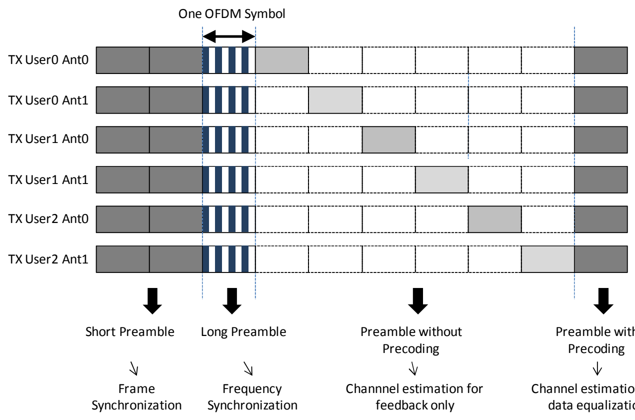


Fig. 4: Preamble structure for embedded IA.

and is not readily portable to generic embedded targets. Toward deploying the MIMO IA system in realistic embedded contexts, we converted all LabVIEW blocks into an embedded C/C++ implementation that can be ported to a wider variety of embedded or PC target platforms.

To design a portable, C/C++-based SDR implementation, we use a GNU Radio environment running on top of a Linux operating system [8]. GNU Radio is a free and open-source software development toolkit that provides signal processing blocks to implement software radios. In the GNU Radio platform, a signal processing system is described as a synchronous data flow (SDF) graph, where each processing block is a node of the overall SDF chain. GNU Radio directly supports the USRP devices as RF front ends, allowing us to use the same hardware setup as is described in the previous section.

All developed code is setup to support cross-compilation to other target platforms. We specifically support a setup with USRPs connected to an ARM-based host system running Linux. Basic functionality of the GNU Radio setup on ARM systems is verified by cross-compiling code on a Texas Instruments (TI) Panda board [9], which is built around TI's OMAP4460 mobile applications processor containing a dual-core ARM Cortex-A9 CPU running Ubuntu Linux 11.10.

Widely used, low-cost mobile development platforms, such as Panda boards, emulate a typical embedded environment as used, for example, in many modern smart phones. With built-in floating-point hardware and two cores running at 1.2GHz, implementation of software-defined radio solutions on such platforms is becoming feasible. Raw computational power, however, is still not comparable to PCs or workstations. General embedded platforms, moreover, do not readily support high-level, library-based development environments, such as LabVIEW, requiring us to reimplement the complete system in a more efficient and reduced form directly in C/C++.

A. System Specification

To realize a MIMO IA system in tightly constrained embedded contexts, careful attention has to be paid to the computational complexity of the implementation. To reduce the complexity, several changes were made to the LabVIEW implementation. In this paper, we thereby restrict the scope to a pure software solution, where FPGA-based hardware acceleration is part of future work.

1) Frame Structure: Some of the most computationally intensive blocks in any OFDM system are the internal fast Fourier transforms (FFTs). The computational complexity of these FFTs is directly proportional to the number of subcarriers in each OFDM symbol, and hence the number of FFT points. In the LabVIEW implementation, the FFT point was 128. In the embedded setup, however, it was reduced to 64 in order to save both computation and bandwidth.

Perfect synchronization in time and frequency is one of the most important aspects and requirements of an IA system. In addition to the modification in FFT size, we design our preamble structure to satisfy these tight requirements while reducing synchronization complexity. Figure 4 shows our modified preamble structure. The synchronization methods are modified and implemented in C++ to support this new preamble. There are four types of preambles in our system:

Short Preamble We achieve time synchronization and coarse frequency synchronization using correlation-based algorithms with this preamble.

Long Preamble We use the frequency domain pilot subcarriers of a long preamble to find the integer frequency offset.

Uncoded CSI Preamble This preamble is only for channel feedback. Each transmitter and each Tx antenna send their preambles at different time-orthogonal moments. The preamble does not experience precoding or decoding. Hence, the receiver can estimate the uncoded wireless channel.

Coded CSI Preamble This preamble experiences precoding at transmitters. With the preamble, we estimate the coded wireless channel, which is needed to decode data at receivers.

B. Block Implementation

Figure 5 shows the overall block diagram of our embedded IA system. For the embedded implementation, we fully converted all signal processing blocks of the MIMO/IA OFDM LabVIEW system into C++ code for use within the overall GNU Radio environment. The functionality of most of the blocks is the same as described in the previous sections. The IA precoding/decoding matrix calculation, which is the block

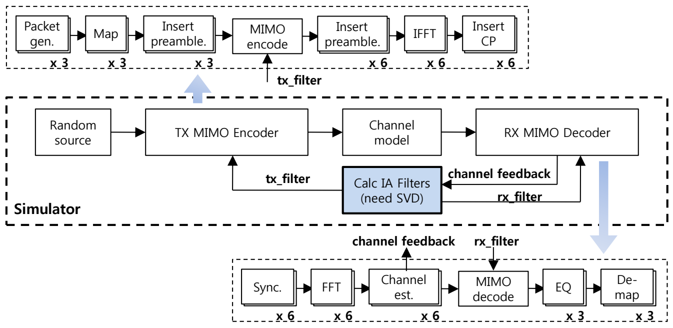


Fig. 5: Block diagram of embedded IA implementation.

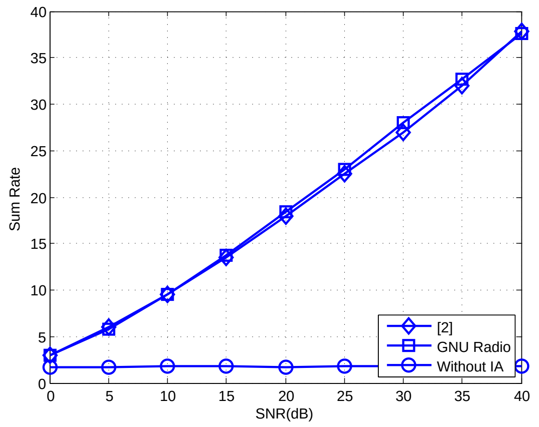


Fig. 6: Simulated sum-rate vs. SNR.

with the highest computational complexity, was redesigned in C++. This includes optimized C++ implementations of complex mathematical functions, such as the singular value decomposition (SVD). We use a well-known method for computing the SVD in two stages [10]. The first stage decomposes the target matrix into a real upper triangular matrix, and the second stage then finds the SVD of this converted matrix.

We implemented both a 6x6 MIMO and an IA system in the GNU Radio setup. The over-the-air 6×6 MIMO system is implemented with USRPs. The IA system is currently only for simulation, and will be extended to over-the-air operation in the future. With the exception of the FFT blocks taken from the GNU Radio library, our final embedded implementation consists of a total of 11,500 lines of custom C++ code.

C. Results

Figure 6 shows the network sum-rate from IA simulation in the GNU Radio environment. Results show that our embedded IA implementation is verified to have the same sum-rate as the simulated IA from [2], where both systems are simulated under Raleigh flat fading channel conditions. Furthermore, in Figure 7, the relationship between the number of iterations of alternating minimization and the achievable sum-rate is presented. The figure shows that the number of iterations needed to achieve a peak sum-rate depends on the SNR, which is also consistent with the results from [2].

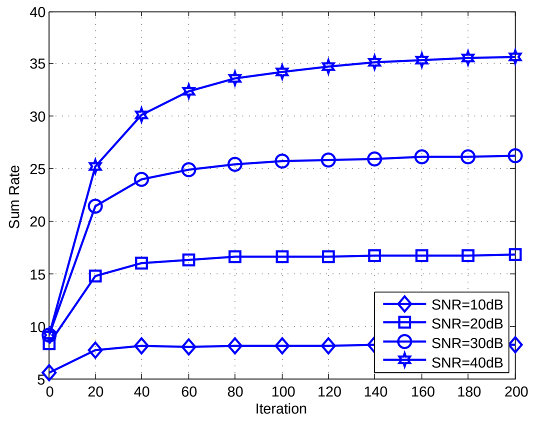


Fig. 7: Simulated sum-rate vs. iteration.

VI. Conclusion and Future Work

We were able to successfully achieve over-the-air 2×2 MIMO IA with 3 users on the popular USRP SDR prototyping hardware. Two new testbeds are created to explore the potential of IA in real-world settings, an important factor that is often overlooked in theoretical papers. The first testbed, implemented on two PCs, shows that IA achieves higher network sum-rates in practice as well as theory. Additional insights were gained during implementation, particularly with respect to synchronization requirements. The second testbed addresses the feasibility of IA for embedded systems. Various modifications to the system are proposed to reduce the computational complexity. Results from both testbeds demonstrate the achievable performance gains and the effectiveness of IA.

Future work involves implementing over-the-air feedback, using distributed time and frequency synchronization, and further optimizing the code for both platforms to achieve higher performance.

Acknowledgments

This work was funded in part by the Army Research Laboratory W911NF-10-1-0420 and the DARPA IT-MANET Program W911NF-07-1-0028. The authors would like to thank Erik Luther from National Instruments for his assistance.

References

[1] V. R.Cadambe *et al.*, "Interference alignment and degrees of freedom of the k-user interference channel," *IEEE Trans. Inf. Theory*, vol. 54, pp. 3425–3441, Aug. 2008.
[2] O. E.Ayach *et al.*, "The feasibility of interference alignment over measured MIMO-OFDM channels," *IEEE Trans. Veh. Technol.*, vol. 59, pp. 4309–4321, Nov. 2010.
[3] S.Gollakota *et al.*, "Interference alignment and cancellation," in *SIGCOMM*, 2009.
[4] O.Gonzalez *et al.*, "Experimental validation of interference alignment techniques using a multiuser MIMO testbed," in *Int. ITG Workshop on Smart Antennas*, 2011.
[5] Ettus Research. https://www.ettus.com/product/details/UN210-KIT.
[6] S. W.Peters *et al.*, "Interference alignment via alternating minimization," in *Acoust., Speech, and Signal Process. IEEE Int. Conf. on*, 2009.
[7] National Instruments. http://www.ni.com/labview/.
[8] E.Blossom *et al.* GNU radio. http://gnuradio.org/.
[9] Texas Instruments. Pandaboard References. http://pandaboard.org/.
[10] S.Qiao *et al.*, "Computing the singular values of 2-by-2 complex matrices," 2002. [Online]. Available: http://www.cas.mcmaster.ca/~qiao/publications/zsvd2.pdf