Turbo-HB: A Novel Design and Implementation to Achieve Ultra-Fast Hybrid Beamforming

Yongce Chen, Yan Huang, Chengzhang Li, Y. Thomas Hou and Wenjing Lou
Virginia Tech, Blacksburg, VA, USA

***Abstract*—Hybrid beamforming (HB) architecture has been widely recognized as the most promising solution to mmWave MIMO systems. A major practical challenge for HB is to obtain a solution in $\sim$1 ms, which is an extremely stringent but necessary time requirement for its deployment in the field. In this paper, we present the design and implementation of Turbo-HB, codename for a novel beamforming design under the HB architecture that can obtain the beamforming matrices in about 1 ms. The key ideas in Turbo-HB include (i) reducing the complexity of SVD techniques by exploiting the limited number of channel paths at mmWave frequencies, and (ii) designing and implementing a parallelizable algorithm for a large number of matrix transformations. We validate Turbo-HB by implementing it on an off-the-shelf Nvidia GPU. Through extensive experiments, we show that Turbo-HB can meet $\sim$1 ms timing requirement while delivering competitive throughput performance compared to state-of-the-art algorithms.**

I. Introduction

To further increase the data rate for the next-generation cellular systems, mmWave MIMO systems have been considered as one of the most promising technologies [1, 2]. At such high frequencies, a base station (BS) typically employs hundreds or more antennas to overcome path-loss fading. Due to hardware complexity and energy consumption issues [2, 3], it is difficult to equip a dedicated RF chain for each antenna. To address this problem, the so-called "hybrid architecture," which uses fewer shared RF chains to support a much larger number of antennas (see Fig. 1), has been considered [4–13].

Under an HB architecture, how to perform beamforming remains a critical problem. In practice, a beamforming design must be able to meet a real-time requirement to be useful. By real-time requirement, we mean that one must find a beamforming solution within half of the channel coherence time.[1] For mmWave based cellular systems, this timing requirement can be as short as 1 ms for a mobile user moving at a speed of 10 km/h. Further, such beamforming design must consider a large number of resource blocks (RBs), with each RB supporting multiple active users.

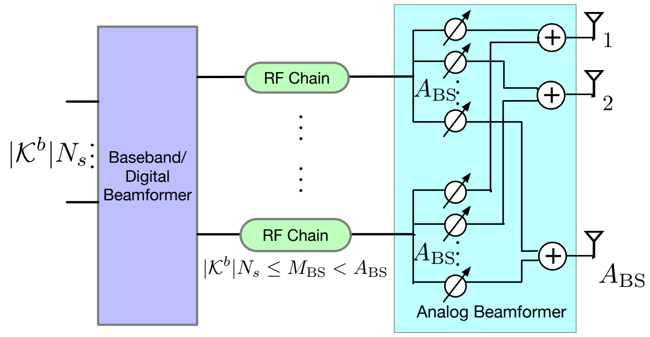


Fig. 1: An HB architecture (BS side).

In the literature, there has been active research on beamforming design under the HB architecture [4–13]. Physical (PHY) layer research in this area attempted to jointly optimize analog and digital beamforming [4–7]. However, the iterative nature of these algorithms makes them difficult to meet the real-time requirement. Further, a joint design requires to estimate and feedback explicit antenna-to-antenna channels, which involves prohibitively large amount of CSI that is difficult to obtain in practice.

To avoid the issues associated with a joint design, a new and practical direction for HB is to follow a sequential design [9–13]. Here, analog beamforming is optimized first and then used as input to optimize the digital beamforming. For analog beamforming, there have been successful designs and system demonstrations in the literature, which are based on beam sweeping/discovering techniques without explicit channel CSI [11–13]. After analog beamforming is applied to both the BS and a user's side, the effective channels seen at the baseband can be obtained through conventional channel estimation.

However, few existing works on sequential design have successfully addressed the digital beamforming problem. Most existing works simply applied traditional beamforming methods such as ZF, MMSE and Block Diagonalization (BD) [14] in the digital domain [9–12]. Although simple, ZF and MMSE typically experience inferior throughput performance for MU-MIMO and mmWave systems, particularly under ill-conditioned channels [11, 15, 16]. Although BD beamforming (and its variants) is shown to improve ZF/MMSE with much better throughput performance [14, 17], it requires many high-dimensional matrix SVD operations, which are of high complexity and require significant computation time.

The goal of developing a beamforming scheme that can meet both real-time requirement and high throughput performance is not trivial. But recent advances in parallel architectures (based on the many-core technology) have shed some new light in this area. In particular, a GPU-based platform

[1] For efficiency, we break up the channel coherence time into two halves. Within each half, we compute beamforming matrices for the next half and transmit data based on beamforming matrices computed in the previous half.

(e.g., those from Nvidia) is particularly promising as it comes with dedicated single-instruction-multiple-data (SIMD) architecture and highly programmable tools such as CUDA. The GPU-based platform offers a new possibility to address many hard problems whose real-time solutions remain open.

In this paper, we present *Turbo-HB,* a GPU-based novel design and implementation to achieve ultra-fast digital beamforming.[2] First, we identify the bottleneck of computation time for BD-type beamforming, which is attributed to the high-dimensional SVD operations. Turbo-HB cuts down the computational complexity by utilizing randomized SVD technique. Second, Turbo-HB accelerates the overall computation time through large-scale parallel computation on a commercial off-the-shelf (COTS) GPU platform. It incorporates a large number of matrix transformations and special engineering efforts such as minimized memory access. The main contributions of this paper are summarized as follows:

- This paper presents Turbo-HB, the first successful HB design that can meet ∼1 ms real-time requirement. This design can be applied to the next-generation cellular systems where a large number of RBs (with MU-MIMO capability) are involved. Our design only relies on a COTS GPU platform and does not require any customized hardware.
- Turbo-HB reduces the computational burden of SVD by exploiting sparsity at mmWave channels. Specifically, it uses only a small number of the most significant singular vectors of a matrix, derived from randomized SVD technique. By limiting operations only to the key information of our interest and judiciously choosing a proper target rank for lower rank approximation, our design drastically reduces the complexity that is involved in the traditional BD-type beamforming.
- Turbo-HB achieves large-scale parallel computation. First, the MU-MIMO beamforming is transformed into a set of parallel single-user MIMO (SU-MIMO) beamforming. Then Turbo-HB is optimized by employing batched matrix operations and minimizing memory access time. Finally, Turbo-HB fully utilizes GPU's processing cores with minimum overhead and takes maximum benefit of large-scale parallel computation.
- We implement Turbo-HB on Nvidia Quadro P6000 GPU using the CUDA programming platform. Extensive experiments are performed to examine both the timing performance and throughput performance. Experimental results show that Turbo-HB is able to obtain the beamforming matrices in ∼1 ms for an MU-MIMO cellular system while achieving similar or better throughput performance by those state-of-the-art algorithms.

II. System Model

Consider a cellular communication scenario in Fig. 2 where a BS serves a set of users $\mathcal{K}$. The BS is equipped with $A_{\rm BS}$ antennas and $M_{\rm BS}$ RF chains. Under HB architecture, $M_{\rm BS} < A_{\rm BS}$. Each user is equipped with $A_{\rm U}$ antennas and $M_{\rm U}$ RF chains, and $M_{\rm U} < A_{\rm U}$. Since the mathematical structure for uplink (UL) and downlink (DL) is symmetric, it is sufficient to study one of them. Therefore, we study DL in this paper.

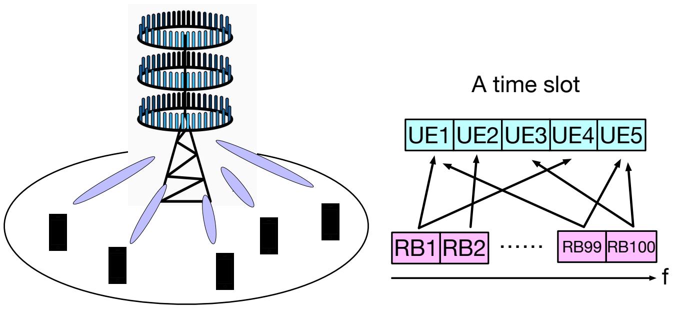


Fig. 2: A cellular system consisting a large number of RBs (with MU-MIMO capability).

Same as a cellular system (e.g., 4G LTE and 5G NR), we consider time-slotted scheduling over a wide bandwidth. Within each time slot, there is a set $\mathcal{B}$ of RBs over the DL bandwidth. For each RB $b \in \mathcal{B}$, a subset of users $\mathcal{K}^b \subseteq \mathcal{K}$ is selected for MU-MIMO transmission, based on some RB allocation strategy (see, e.g., [19, 20]). For the ease of notation, suppose the BS sends N_s data streams to each user.[3] At the user side, since the number of received data streams cannot exceed the number of its RF chains, we have $N_s \leq M_{\rm U}$. Likewise, at the BS we have $|\mathcal{K}^b| N_s \leq M_{\rm BS}$.

Under the HB architecture, beamforming is performed in both digital and analog domains, as shown in Fig. 1. In the digital domain at the BS side, the transmitted signal is processed by an $M_{\rm BS} \times |\mathcal{K}^b| N_s$ baseband precoder $\mathbf{F}_{\rm BB}$. Subsequently, an $A_{\rm BS} \times M_{\rm BS}$ analog precoder $\mathbf{F}_{\rm RF}$ (also known as RF precoder) based on analog circuitry (phase shifters) is applied in the analog domain. Since complex matrix $\mathbf{F}_{\rm RF}$ is implemented with analog phase shifters, each element in the matrix has the same amplitude and differs in its phase, i.e, $|(\mathbf{F}_{\rm RF})_{i,j}| = \frac{1}{\sqrt{A_{\rm BS}}}$, where $(\cdot)_{i,j}$ denotes the (i,j)-th element of matrix $(\cdot)$. In addition, to meet the total power constraint at the BS, $\mathbf{F}_{\rm BB}$ and $\mathbf{F}_{\rm RF}$ must satisfy $||\mathbf{F}_{\rm RF}\mathbf{F}_{\rm BB}||_F^2 \leq P_{\rm T}$, where $P_{\rm T}$ is the total power at the BS and $||\cdot||_F$ denotes the Frobenius norm.

For wireless channels, let $\mathbf{H}_k^b \in \mathbb{C}^{A_{\rm U} \times A_{\rm BS}}$ denote the channel matrix for user $k \in \mathcal{K}$ on RB $b \in \mathcal{B}$, and $\boldsymbol{n}_k^b$ is the $A_{\rm U} \times 1$ vector of i.i.d $\mathcal{CN}(0, \sigma^2)$ additive complex Gaussian noise. Let $\mathbf{F}_{\rm BB}^b$ and $\mathbf{F}_{\rm RF}^b$ denote the baseband precoder and analog precoder for RB b, respectively. Then the received signal of user k on RB b is given by

$$\boldsymbol{y}_k^b = \mathbf{H}_k^b \mathbf{F}_{\rm RF}^b \mathbf{F}_{\rm BB}^b \boldsymbol{s}^b + \boldsymbol{n}_k^b, \quad (k \in \mathcal{K}^b, b \in \mathcal{B}) \qquad (1)$$

where $\boldsymbol{s}^b$ is the signal vector.

At the user side, a symmetric HB structure is employed except with a fewer number of antennas $A_{\rm U}$ and a fewer

[2] By "Turbo," we mean fast and efficient.

[3] With additional notation, our results can be easily extended to the case where the BS sends a different number of data streams to different users.

number of RF chains M_{U}. In analog domain, an $A_{\text{U}} \times M_{\text{U}}$ analog combiner $\mathbf{W}_{\text{RF},k}$ (subject to $|(\mathbf{W}_{\text{RF},k})_{i,j}| = \frac{1}{\sqrt{A_{\text{U}}}}$) is applied. In digital domain, an $M_{\text{U}} \times N_s$ baseband combiner $\mathbf{W}_{\text{BB},k}$ is applied.

After the analog beamformers are applied, the effective channel seen at the baseband is $\widehat{\mathbf{H}}_k^b = \mathbf{W}_{\text{RF},k}^{b\dagger}\mathbf{H}_k^b\mathbf{F}_{\text{RF}}^b$. Denote $\mathbf{F}_{\text{BB},k}^b$ as a sub-matrix of $\mathbf{F}_{\text{BB}}^b = [\mathbf{F}_{\text{BB},1}^b \cdots \mathbf{F}_{\text{BB},k}^b \cdots \mathbf{F}_{\text{BB},|\mathcal{K}^b|}^b]$, where $\mathbf{F}_{\text{BB},k}^b$ consists of N_s columns and corresponds to the baseband signal $\boldsymbol{s}_k^b$ of user k. Then at user k and on RB b, we have the following signal:

$$\begin{aligned}\tilde{\boldsymbol{y}}_k^b = \mathbf{W}_{\text{BB},k}^{b\dagger}\widehat{\mathbf{H}}_k^b\mathbf{F}_{\text{BB},k}^b\boldsymbol{s}_k^b + \sum_{i\in\mathcal{K}^b}^{i\neq k}\mathbf{W}_{\text{BB},k}^{b\dagger}\widehat{\mathbf{H}}_k^b\mathbf{F}_{\text{BB},i}^b\boldsymbol{s}_i^b \\ + \mathbf{W}_{\text{BB},k}^{b\dagger}\mathbf{W}_{\text{RF},k}^{b\dagger}\boldsymbol{n}_k^b, \qquad (k \in \mathcal{K}^b, b \in \mathcal{B})\end{aligned} \tag{2}$$

where $(\cdot)^\dagger$ denotes the conjugate transpose of a matrix.

Therefore, the network throughput in bits/sec/Hz is

$$\begin{aligned}C = \sum_{b\in\mathcal{B}}\sum_{k\in\mathcal{K}^b}\log\Big(\Big|\mathbf{I}_{N_s} + \\ (\mathbf{Q}_k^b)^{-1}\mathbf{W}_{\text{BB},k}^{b\dagger}\widehat{\mathbf{H}}_k^b\mathbf{F}_{\text{BB},k}^b\ \mathbf{F}_{\text{BB},k}^{b\dagger}\widehat{\mathbf{H}}_k^{b\dagger}\mathbf{W}_{\text{BB},k}^b\Big|\Big),\end{aligned} \tag{3}$$

where $(\mathbf{Q}_k^b)^{-1}$ is the covariance matrix of both interference and noise, which is given by

$$\begin{aligned}(\mathbf{Q}_k^b)^{-1} = \sum_{i\in\mathcal{K}^b}^{i\neq k}\mathbf{W}_{\text{BB},k}^{b\dagger}\widehat{\mathbf{H}}_k^b\mathbf{F}_{\text{BB},i}^b\mathbf{F}_{\text{BB},i}^{b\dagger}\widehat{\mathbf{H}}_k^{b\dagger}\mathbf{W}_{\text{BB},k}^b \\ + \sigma^2\mathbf{W}_{\text{BB},k}^{b\dagger}\mathbf{W}_{\text{RF},k}^{b\dagger}\mathbf{W}_{\text{RF},k}^b\mathbf{W}_{\text{BB},k}^b.\end{aligned} \tag{4}$$

Then the throughput optimization problem under the HB architecture can be stated as follows.

OPT-HB

$$\begin{aligned}\max\ & C\left(\mathbf{F}_{\text{RF}}^b, \mathbf{F}_{\text{BB}}^b, \mathbf{W}_{\text{RF},k}^b, \mathbf{W}_{\text{BB},k}^b\right) \\ \text{s.t.}\ & \text{Power constraint: } ||\mathbf{F}_{\text{RF}}^b\mathbf{F}_{\text{BB}}^b||_F^2 \le P_{\text{T}}; \\ & \text{Constant modulus constraints:} \\ & \qquad |(\mathbf{F}_{\text{RF}}^b)_{i,j}| = \frac{1}{\sqrt{A_{\text{BS}}}},\ |(\mathbf{W}_{\text{RF},k}^b)_{m,n}| = \frac{1}{\sqrt{A_{\text{U}}}}; \\ & \text{Index range: } b \in \mathcal{B},\ k \in \mathcal{K}, \\ & \qquad i \in \{1, 2, \cdots, A_{\text{BS}}\},\ j \in \{1, 2, \cdots, M_{\text{BS}}\}, \\ & \qquad m \in \{1, 2, \cdots, A_{\text{U}}\},\ n \in \{1, 2, \cdots, M_{\text{U}}\}.\end{aligned}$$

In problem OPT-HB, the variables are digital and analog beamformers $\mathbf{F}_{\text{RF}}^b, \mathbf{F}_{\text{BB}}^b, \mathbf{W}_{\text{RF},k}^b$ and $\mathbf{W}_{\text{BB},k}^b$, while $P_{\text{T}}, A_{\text{BS}}, A_{\text{U}}, M_{\text{BS}}, M_{\text{U}}$ are constants and $\mathcal{B}$ and $\mathcal{K}^b$ are given sets.

Ideally, a joint optimization of all digital and analog beamformers is required to find a global optimal solution. However, several practical issues make such a joint design impractical. For example, the amount of CSI required is prohibitively large; it is unclear how to estimate the antenna-to-antenna channel $\mathbf{H}_k^b$ through the lens of the RF precoding and combining [8]. A new and practical direction to address HB optimization is to follow a sequential design. Under this approach, analog domain is optimized first and then used as input to optimize the digital domain [9–13]. It has been shown that such a sequential approach can offer competitive performance (compared to those heuristics attempting to solve joint optimization [5, 7, 9, 22]).

Even with a sequential method, for MU-MIMO systems, it would still require enormous computational efforts to find a local optimum [23], due to the high complexity of high-dimensional matrix operations (in addition to non-convex programming). We discuss this problem in detail in the following section.

III. Real-Time Requirement

We now examine the real-time requirement for a practical beamforming design. By real-time we mean a beamforming solution (for all users on all RBs) must be obtained within half of the channel coherence time to be useful.

Consider a typical mmWave MIMO with its center carrier frequency $f_c = 30$ GHz, and a mobile user traveling at a speed of $v = 10$ km/h, which is no slower than most implementations, as demonstrated in [13, 24–26].[4] At this speed, the maximum Doppler shift $f_m = \frac{v}{c} \cdot f_c \approx 277.8$ Hz. The channel coherence time can be empirically given by [28]: $T_c \approx \frac{1}{f_m} = 3.6$ ms. The timing requirement for the execution time of a beamforming algorithm should be even more stringent than this channel coherence time. Within this time interval (i.e., 3.6 ms), only at most the first half of it (i.e., 1.8 ms) can be used for beamforming design so that the second half of 3.6 ms can be used for actual data transmission and computing the beamforming matrices for the subsequent half coherence period. In addition, the 1.8 ms should also consist of channel estimation time and setting the beamforming decision in hardware. These procedures usually take less than 1 ms. Therefore, a solution of digital beamformers to OPT-HB must be obtained within

$$T_{\text{req}} = 1 \ \text{ ms}, \tag{5}$$

to be useful. Note that under HB architecture, analog beamforming is meant to overcome path-loss fading by leveraging the large number of antennas [9, 13]. This part is done on a much larger time scale. In contrast, digital beamforming is to optimize capacity by managing interference among data streams, which heavily depends on fast fading. This part has a much stringent timing requirement T_{req}.

Technical Challenge Digital beamforming for MU-MIMO involves complex operations of matrices with a large number of elements. Traditional techniques such as ZF and MMSE typically experience inferior throughput performance for MU-MIMO and mmWave systems, particularly under ill-conditioned channels [11, 15, 16]. On the other hand, BD-type beamforming is shown to achieve much better throughput performance compared to ZF/MMSE [14]. However, BD involves high-dimensional matrix SVD operations. The computational burden of SVD makes BD not ready for practical use.

[4] Although there are some proposals envisioning to have mmWave support high mobility, unfortunately it is impractical to implement them in practice, due to issues such as huge channel training overhead, multi-user tracking, and so forth [26, 27].

Objective The objective of this paper is to determine digital beamforming ($\mathbf{F}_{\mathrm{BB},k}^{b}$ and $\mathbf{W}_{\mathrm{BB},k}^{b}$) in real-time. Specifically, we want to develop a design that can meet the stringent $\sim$1 ms timing requirement while offering comparable (or better) throughput performance than state-of-the-art approaches.

IV. A Novel Design for Real-time Beamforming

A. Main Ideas

Our main ideas span across two areas.

Low-complexity SVD with high throughput First, we show the bottleneck of computation time for BD-type beamforming is attributed to the high-dimensional SVD operations. We propose to reduce this complexity by using only a small number for the most significant dimensions, leveraging the sparsity of mmWave channels. Specifically, for a $(|\mathcal{K}^b| - 1)M_{\mathrm{U}} \times M_{\mathrm{BS}}$ matrix (for BD beamforming), a standard SVD algorithm takes $O\left(\left[(|\mathcal{K}^b| - 1)M_{\mathrm{U}}\right]^2 M_{\mathrm{BS}}\right)$ floating-point operations (flops) [29]. Thus, applying BD beamforming for $|\mathcal{B}|$ RBs and $|\mathcal{K}^b|$ users at each RB yields at least $O\left(|\mathcal{B}||\mathcal{K}^b| \cdot \left[(|\mathcal{K}^b| - 1)M_{\mathrm{U}}\right]^2 \cdot M_{\mathrm{BS}}\right)$ flops. To reduce this high complexity, we propose to utilize randomized SVD [29], which can cut down the complexity to $O\left(|\mathcal{B}||\mathcal{K}^b| \cdot r^2 \cdot \left[(|\mathcal{K}^b| - 1)M_{\mathrm{U}} + M_{\mathrm{BS}}\right]\right)$, where r is much smaller than $(|\mathcal{K}^b| - 1)M_{\mathrm{U}}$. In essence, randomized SVD is a lower rank SVD approximation method. It works extremely well here, thanks to the limited number of scatterers at mmWave frequencies and thus highly correlated channels.

Interestingly, although Turbo-HB employs a lower rank SVD approximation, it does not mean that the throughput performance would have to deteriorate. Rather, Turbo-HB appears to offer higher throughput performance in most cases. The science behind this behavior is attributed to the following. First, since mmWave channels exhibit a high correlation property, a small set of singular vectors in the lower rank SVD approximation is sufficient to capture the directions of the most significant signal or interference strength. Second, an exact $(|\mathcal{K}^b| - 1)M_{\mathrm{U}} \times M_{\mathrm{BS}}$ matrix SVD (as in BD) aims to cancel *all* inter-user interference exactly (regardless of how small it is). But canceling all inter-user interference requires to project users' signals onto mutually orthogonal subspaces. To achieve such orthogonality, the perceived strength of desired signals at a user is reduced in the process. Since throughput is a function of SINR, it does not help if the perceived strength of desired signals at a user is reduced (for perfect orthogonality). On the other hand, a lower rank SVD approximation allows a certain level of overlapping subspace of different users (as only a small number of major signals preserve mutual orthogonality), which offers us an opportunity to explore the promising beamforming space that has been missed by the BD technique.

Fully functioning parallelism We argue that the $O(\cdot)$-type complexity characterization does not reflect computation time as measured by a wall clock, which is what matters in practice. The latter heavily depends on the underlying problem structure, convergence speed, the number of memory accesses, among others. This motivates us to our second idea, which is to accelerate the computing process in real-time, rather than being confined in $O(\cdot)$ analysis. We propose to design and implement the beamforming algorithm with fully functioning parallelism and minimized memory accesses. We implement our parallel design on a COTS GPU platform, which is a general-purpose computing platform for large-scale parallel computation [18].

Specifically, the MU-MIMO beamforming is first transformed into a set of parallel SU-MIMO beamforming. Then a large number of matrix operations are executed through batch computing. By batched matrix operations (for a large number of RBs and users), Turbo-HB generates a large number of threads that fully occupy a GPU's processing cores and thus reaps the full benefit of GPU's parallel processing capability. Also, along each step of our implementation, we minimize memory accesses to reduce time. For example, batched matrix operations such as matrix multiplications are optimized with the use of fast on-chip shared memory. With proper indexing method for a large number of matrices, we allow consecutive GPU threads to read consecutive (and aligned) memory. As a result, multiple memory accesses can be combined into a single transaction. Further, Turbo-HB limits operations to the key information of our interests (e.g., certain singular vectors) and thus eliminates any unnecessary calculations, parameter passing and memory accesses.

B. Design Details

The task of computing beamforming matrices can be split naturally into three computational stages.

- **Stage A:** Given the partial CSI $\mathbf{V}_k^b$ and $\mathbf{\Sigma}_k^b$ (from $\widehat{\mathbf{H}}_k^b = \mathbf{U}_k^b \mathbf{\Sigma}_k^b \mathbf{V}_k^{b\dagger}$) computed and fed back by each user, we construct matrices $\overline{\mathbf{H}}_k^b$ and $\widetilde{\mathbf{H}}_k^b$ such that $\overline{\mathbf{H}}_k^b$ and $\widetilde{\mathbf{H}}_k^b$ contain all the information that is needed to compute beamforming matrices $\mathbf{F}_{\mathrm{BB},k}$ corresponding to user k. After this stage, the MU-MIMO channel is transformed into a set of parallel SU-MIMO channels.
- **Stage B:** Given matrix $\widetilde{\mathbf{H}}_k^b$, we apply randomized SVD technique for lower rank matrix approximation (with lower computational complexity). Then we obtain $\widetilde{\mathbf{V}}_k^{b(-)}$, which contains the necessary singular vectors to cancel inter-user interference and construct final beamforming matrices.
- **Stage C:** Given matrices $\overline{\mathbf{H}}_k^b$ and $\widetilde{\mathbf{V}}_k^{b(-)}$, we construct the final digital beamforming matrices $\mathbf{F}_{\mathrm{BB},k}$.

In the rest of this section, we offer details of each stage.

Stage A. Each user k estimates the effective channel $\widehat{\mathbf{H}}_k^b$ and computes its SVD as $\widehat{\mathbf{H}}_k^b = \mathbf{U}_k^b \mathbf{\Sigma}_k^b \mathbf{V}_k^{b\dagger}$. User k uses the first N_s columns of $\mathbf{U}_k^b$ as its digital combiner, i.e., $\mathbf{W}_{\mathrm{BB},k}^b$ is set to the first N_s columns of $\mathbf{U}_k^b$. Then to help form digital precoder at BS side, only partial CSI, i.e., $\mathbf{V}_k^b$ and $\mathbf{\Sigma}_k^b$, is required to feed back to the BS (note that Σ_k^b is diagonal and $\mathbf{V}_k^b$ is unitary and thus can be efficiently compressed [30]). Let

$$\overline{\mathbf{H}}_k^b = \Sigma_k^b \mathbf{V}_k^b. \tag{6}$$

Then for our beamforming purpose, $\overline{\mathbf{H}}_k^b$ (an $M_{\text{U}} \times M_{\text{BS}}$ matrix) captures sufficient information of the intended channel from the BS to user k.

Denote $\widetilde{\mathbf{H}}_k^b$ as the concatenation of $\overline{\mathbf{H}}_k^b$'s of all users in $\mathcal{K}^b$ except intended user k, i.e., if $\mathcal{K}^b = \{k\} \bigcup \{1, \cdots, k-1, k+1, \cdots, |\mathcal{K}^b|\}$, then

$$\widetilde{\mathbf{H}}_k^b = \left[\overline{\mathbf{H}}_1^{b\dagger} \cdots \overline{\mathbf{H}}_{k-1}^{b\dagger} \;\; \overline{\mathbf{H}}_{k+1}^{\dagger} \cdots \overline{\mathbf{H}}_{|\mathcal{K}^b|}^{b\dagger}\right]^\dagger \qquad (7)$$

is a $(|\mathcal{K}^b| - 1)M_{\text{U}} \times M_{\text{BS}}$ matrix which captures information of interference channels corresponding to user k.

As $\overline{\mathbf{H}}_k^b$ and $\widetilde{\mathbf{H}}_k^b$ are sufficient to construct the beamforming matrices $\mathbf{F}_{\text{BB},k}$ corresponding to user k, the MU-MIMO channel is transformed into a set of $|\mathcal{K}^b|$ parallel SU-MIMO channels on each RB. Consequently, the remaining Stage B and Stage C can be processed in $|\mathcal{B}| \sum_{b \in \mathcal{B}} |\mathcal{K}^b|$ parallel flows, each of which contributes to one beamforming matrix for one user per RB.

Stage B. To construct beamforming matrix $\mathbf{F}_{\text{BB},k}$ corresponding to user k's signal, we need to make sure that by applying $\mathbf{F}_{\text{BB},k}$ most (if not all) of the interference to user k can be canceled. This can be realized with the help of SVD of interference channel $\widetilde{\mathbf{H}}_k^b$. That is, let

$$\widetilde{\mathbf{H}}_k^b = \widetilde{\mathbf{U}}_k^b \begin{bmatrix} \widetilde{\boldsymbol{\Sigma}}_k^b & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} [\widetilde{\mathbf{V}}_k^{b(+)} \;\; \widetilde{\mathbf{V}}_k^{b(-)}]^\dagger, \qquad (8)$$

where $\widetilde{\mathbf{V}}_k^{b(-)}$ is the last $(M_{\text{BS}} - r)$ columns of the right singular matrix corresponding to the smallest $(M_{\text{BS}} - r)$ singular values of $\widetilde{\mathbf{H}}_k^b$, $\widetilde{\mathbf{V}}_k^{b(+)}$ is the remaining r columns of the right singular matrix, and r is a constant.

Then, if the eigenvalues corresponding to $\widetilde{\mathbf{V}}_k^{b(-)}$ are close to zero, we have

$$\widetilde{\mathbf{H}}_k^b \widetilde{\mathbf{V}}_k^{b(-)} \approx \mathbf{0}, \quad (b \in \mathcal{B}, k \in \mathcal{K}^b). \qquad (9)$$

It follows that

$$\widehat{\mathbf{H}}_j^b \widetilde{\mathbf{V}}_k^{b(-)} \overline{\mathbf{V}}_k^{b(+)} \approx \mathbf{0}, \quad \text{for } j \neq k, \qquad (10)$$

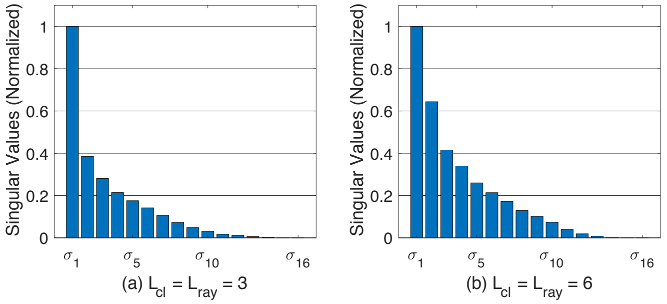
with any choice of $\overline{\mathbf{V}}_k^{b(+)}$ (which is used to differentiate data streams within a user and will be determined later). Therefore, by constructing $\mathbf{F}_{\text{BB},k}$ as

$$\mathbf{F}_{\text{BB},k} = \widetilde{\mathbf{V}}_k^{b(-)} \overline{\mathbf{V}}_k^{b(+)}, \qquad (11)$$

most of the inter-user interference can be suppressed.

Now we have a real-time challenge. Stage B is computation-intensive as a high-dimensional SVD (i.e., Eq. (8)) is required. $\widetilde{\mathbf{H}}_k^b$ is a $(|\mathcal{K}^b| - 1)M_{\text{U}} \times M_{\text{BS}}$ matrix with standard SVD complexity of $O\left(|\mathcal{B}||\mathcal{K}^b| \cdot \left[(|\mathcal{K}^b| - 1)M_{\text{U}}\right]^2 M_{\text{BS}}\right)$ for $|\mathcal{B}|$ RBs. Its computation time can take more than 70% of the total time when not optimized based on our experiment.

In fact, the computation time of matrix SVD (power method) is tightly related to the decaying speed of singular values [31]. For instance, suppose we have a matrix with 4 decreasing singular values $\sigma_1, \sigma_2, \sigma_3$ and σ_4. If $\sigma_1 \gg \sigma_2 \gg \sigma_3 \approx \sigma_4 \approx 0$, then it is computationally fast to obtain the first two singular values (and associated singular vectors), while it takes much longer time to obtain the last two singular values. This observation is especially important for us, since at mmWave frequencies, most signal strength will be concentrated at a few directions due to the limited number of scatterers. As a consequence, it is likely that we encounter several non-zero but close-to-zero singular values. Obtaining these small singular values would consume a lot of time even though such calculations are not necessary in terms of throughput performance (as we shall see in Sec. IV-C).



(a) $L_{\text{cl}} = L_{\text{ray}} = 3$
(b) $L_{\text{cl}} = L_{\text{ray}} = 6$

Fig. 3: Singular values of $\widetilde{\mathbf{H}}_k^b$ (averaged over 100 instances) under different number of scatterers based on mmWave channel modelling.

To verify the singular values of $\widetilde{\mathbf{H}}_k^b$, we conduct the following experiment. We generate 100 instances of $\widetilde{\mathbf{H}}_k^b$ based on mmWave channel model to have $\mathbf{H}_k^b$'s (using the widely adopted mmWave channel model as described in [5]). For analog beamforming, we adopt the well-known DFT-codebook based method [9, 32]. We set $A_{\text{BS}} = 128$, $A_{\text{U}} = 8$, $M_{\text{BS}} = 20$, $M_{\text{U}} = 4$ and $|\mathcal{K}^b| = 5$, thus $\widetilde{\mathbf{H}}_k^b$ is a 16×20 matrix. We investigate two different scattering scenarios: (a) The number of clusters L_{cl} and the number of rays within each cluster L_{ray} are both set to 3; (b) L_{cl} and L_{ray} are both 6 (as typical number of paths for practical mmWave channels [3, 11, 13, 33]). Averaged by 100 instances, the singular values of $\widetilde{\mathbf{H}}_k^b \widetilde{\mathbf{H}}_k^{b\dagger}$ are plotted in Fig. 3. As we expected, the singular values are decaying fast in the beginning but then flatten out. The decaying speed is faster when the number of paths is smaller. More importantly, the last several singular values are pretty small but very close. This means the corresponding directions in the eigenspace have very weak signals but consume much computational effort to differentiate them, which is wasteful.

Following the above analysis, our next objective is to implement a lower rank SVD approximation with lower computational complexity. To do this, we apply randomized SVD technique [29]. The key idea is that by using a random Gaussian matrix $\mathbf{\Omega}$, we can form a rank-r basis $\mathbf{Y}$, where $r < (|\mathcal{K}^b| - 1)M_{\text{U}} < M_{\text{BS}}$ that captures those dominant directions (corresponding to the largest signal strengths). Then the original matrix is projected onto a lower-dimensional subspace (based on basis $\mathbf{Y}$) to compute a standard rank-r SVD.

Based on randomized SVD technique, we develop Algorithm 1 to obtain $\widetilde{\mathbf{V}}_k^{b(-)}$. Algorithm 1 is customized for

```
Algorithm 1: Lightweight SVD
  Given an $m \times n$ matrix $\mathbf{A}$, a target approximation
    rank $r$, and an exponent $q$, this procedure computes
    an approximate last $(n-r)$ right singular vectors of
    $\mathbf{A}$, denoted as $\mathbf{V}$:
1 Generate an $n \times r$ Gaussian matrix $\mathbf{\Omega}$.
2 Form an $m \times r$ matrix $\mathbf{Y} = (\mathbf{A}\mathbf{A}^\dagger)^q \mathbf{A}\mathbf{\Omega}$ by
    multiplying alternately with $\mathbf{A}$ and $\mathbf{A}^\dagger$.
3 Form an $r \times n$ matrix $\mathbf{B} = \mathbf{Y}^\dagger \mathbf{A}$.
4 Form a rank-$r$ $n \times n$ Hermitian matrix $\mathbf{C} = \mathbf{B}^\dagger \mathbf{B}$.
5 Compute ED of the rank-$r$ Hermitian matrix:
    $\mathbf{C} = \check{\mathbf{V}}\mathbf{\Lambda}\check{\mathbf{V}}^\dagger$.
6 Set $\mathbf{V}$ as the last $(n-r)$ columns of $\check{\mathbf{V}}$.
```

our problem to achieve further acceleration. Details of the algorithm's development and their rationales can be found in [34]. $\widetilde{\mathbf{H}}_k^b$ will be used as input for Algorithm 1. As we see in Step 5, due to the lower rank r, only a small-scale eigenvalue decomposition (ED) is required. The complexity of Step 5 is $O\left(r^2 \cdot \left[(|\mathcal{K}^b|-1)M_\text{U} + M_\text{BS}\right]\right)$, reduced from $O\left(\left[(|\mathcal{K}^b|-1)M_\text{U}\right]^2 \cdot M_\text{BS}\right)$. Note that ED is used instead of SVD as only the right singular matrix $\mathbf{V}$ is of our interests. Computing ED can provide $\mathbf{V}$ as well, and it further saves a lot of overhead for memory write/read and parameters passing through the nested build-in functions inside the library, which can lead to significant acceleration. How to choose a proper value of r will be discussed in the next section.

Algorithm 1 in Stage B significantly reduces the computation time of standard SVD operations — the main bottleneck in BD beamforming. The only additional cost is a few more matrix multiplications, which, fortunately, can be parallelized and computed efficiently (more details in Sec. V). Although randomized SVD is an approximation method, we will not analyze its performance here, since it is only an intermediate step for beamforming. Instead, we will discuss and show both the timing and throughput performance by applying randomized SVD in Sec. V.

Stage C. In this stage, we construct the digital beamforming matrices $\mathbf{F}_{\text{BB},k}^b$. For given matrices $\overline{\mathbf{H}}_k^b$ and $\widetilde{\mathbf{V}}_k^{b(-)}$, the product of $\overline{\mathbf{H}}_k^b$ and $\widetilde{\mathbf{V}}_k^{b(-)}$ effectively forms user k's channel with no (or minor) inter-user interference (recall Eq. (10)). Therefore, the optimal beamforming strategy regarding the effective $M_\text{U} \times (M_\text{BS}-r)$ channel $\overline{\mathbf{H}}_k^b \widetilde{\mathbf{V}}_k^{b(-)}$ can be realized based on its SVD, which is given by:

$$\overline{\mathbf{H}}_k^b \widetilde{\mathbf{V}}_k^{b(-)} = \overline{\mathbf{U}}_k^b \overline{\mathbf{\Sigma}}_k^b \left[\overline{\mathbf{V}}_k^{b(+)} \quad \overline{\mathbf{V}}_k^{b(-)}\right]^\dagger, \tag{12}$$

where $\overline{\mathbf{V}}_k^{b(+)}$ is the first N_s columns of right singular matrix and $\overline{\mathbf{V}}_k^{b(-)}$ is the remaining columns. Finally, the digital beamforming matrix $\mathbf{F}_{\text{BB},k}^b$ is given by

$$\mathbf{F}_{\text{BB},k} = \widetilde{\mathbf{V}}_k^{b(-)} \overline{\mathbf{V}}_k^{b(+)}, \tag{13}$$

with an additional normalization to satisfy power constraints.

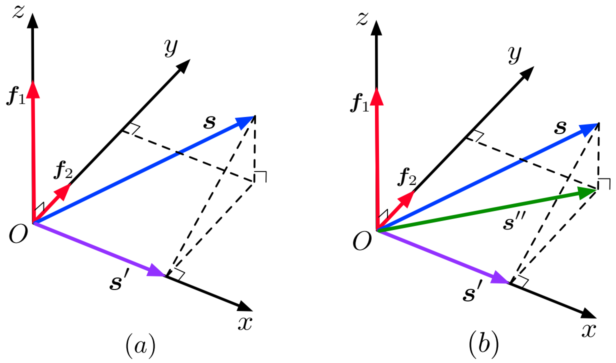


Fig. 4: Signal is projected onto a lower dimensional subspace to avoid interference.

Here we encounter another SVD computation, i.e., Eq. (12). Luckily, the dimension of this to-be-factorized matrix is tied to the number of RF chains at one user, namely M_U, which is typically small (e.g., 1 to 4). $\overline{\mathbf{V}}_k^{b(+)}$ can be derived with the help of Hermitian symmetric matrix ED and matrix multiplication, through the following steps:

- Form an $M_\text{U} \times (M_\text{BS}-r)$ matrix $\mathbf{A}_k^b = \overline{\mathbf{H}}_k^b \widetilde{\mathbf{V}}_k^{b(-)}$;
- Form an $M_\text{U} \times M_\text{U}$ matrix $\mathbf{B}_k^b = \mathbf{A}_k^b \mathbf{A}_k^{b\dagger}$;
- Compute ED of the Hermitian matrix: $\mathbf{B}_k^b = \mathbf{U}_k^b \mathbf{\Lambda}_k^b \mathbf{U}_k^{b\dagger}$;
- Set $\overline{\mathbf{V}}_k^{b(+)}$ to the first N_s columns of $\mathbf{A}_k^{b\dagger} \mathbf{U}_k^b$.

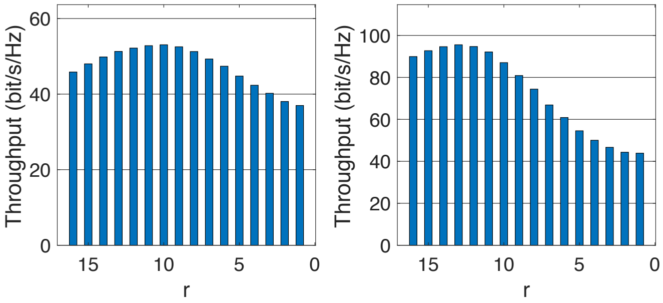
Note that when $M_\text{U} = 1$ or 2, simple and exact closed-form solution for SVD exists [35] and hence this stage can be completed very fast.

C. Approximation with Lower Rank

As we discussed in Sec. IV-B, Turbo-HB applies lower rank approximation to reduce computational complexity. Interestingly, for most cases, our approximation does not sacrifice throughput performance. In this section, we offer some intuition behind it. Then we address the last problem before implementation, which is how to choose a proper value for r.

Let's revisit the SVD of interference channel $\widetilde{\mathbf{H}}_k^b$ as in Eq. (8). In the M_BS-dimensional signal space $[\widetilde{\mathbf{V}}_k^{b(+)} \ \ \widetilde{\mathbf{V}}_k^{b(-)}]$, $\widetilde{\mathbf{V}}_k^{b(-)}$ is an $(M_\text{BS}-r)$-dimensional subspace corresponding to the $(M_\text{BS}-r)$ smallest interference strengths, while $\widetilde{\mathbf{V}}_k^{b(+)}$ is a r-dimensional subspace corresponding to the r largest interference strengths. When standard SVD is performed, we have $r = (|\mathcal{K}^b|-1)M_\text{U}$ (as in conventional BD approach). Then $\widetilde{\mathbf{V}}_k^{b(-)}$ lies *exactly* in the nullspace of $\widetilde{\mathbf{H}}_k^b$, and therefore *all* inter-user interference will be cancelled when $\mathbf{F}_{\text{BB},k}$ is constructed based on $\widetilde{\mathbf{V}}_k^{b(-)}$ (i.e., Eq. (13)). In addition to the high complexity, there is another drawback of such a "perfect" interference cancellation. That is, to achieve mutual orthogonality, one has to project the desired signal onto a subspace with a small number of dimensions. As a result, the perceived desired signal strength at a user is reduced.

In Fig. 4, we use a simple example to illustrate this point. In a 3-dimensional signal space, we have a strong interference $\boldsymbol{f}_1$ along the z axis and a weak interference $\boldsymbol{f}_2$ along the y axis.



(a) SNR = 5dB (b) SNR = 20dB

Fig. 5: Achieved network throughput (averaged over 1,000 instances) as a function of approximation rank r under different SNR.

Now we are going to project a desired signal $\boldsymbol{s}$ (originally in the xyz space) onto some subspace to avoid interference (via beamforming). If perfect interference cancellation is required, then $\boldsymbol{s}$ has to be projected along the x axis to achieve orthogonality to both $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$, leading to a smaller-strength signal $\boldsymbol{s}'$, as shown in Fig. 4(a). However, if only the strong interference $\boldsymbol{f}_1$ is required to be cancelled, then one can project $\boldsymbol{s}$ into a larger dimensional subspace, i.e., xOy plane, resulting in $\boldsymbol{s}''$ as shown in Fig. 4(b). Although $\boldsymbol{s}''$ is interfered with by a weak interference $\boldsymbol{f}_2$, $\boldsymbol{s}''$ can preserve higher signal strength than $\boldsymbol{s}'$, which will lead to a higher SINR (and throughput).

Turbo-HB is purposely designed to explore such a design space by tolerating some level of weak interference. When lower rank SVD approximation is performed, it is meant to only identify r directions corresponding to r strongest interference. Without knowledge of how remaining interference presents, the desired signal will be projected onto a larger dimensional subspace only to avoid the identified interference, preserving greater desired signal strength. This approach is especially effective for scenarios where there is high correlation among the channels or SNR is low. Since in these scenarios, the last few singular values (i.e., corresponding weak interference strengths) are small compared to the power of white noise. Then the dominant term in the denominator of SINR becomes the power of noise, which cannot be suppressed by beamforming. Thus, by tolerating weak interference, desired signal strength is preserved to overcome a bigger issue (the noise), leading to a higher SINR.

Now we address the question of how to choose a proper value for r. Since $0 < r \leq \text{rank}(\widetilde{\mathbf{H}}_k^b) = M_{\text{BS}} - M_{\text{U}}$ and r is an integer, we have $(M_{\text{BS}} - M_{\text{U}})$ possible values for r. If we choose r to be too large (i.e., close to $(M_{\text{BS}} - M_{\text{U}})$), then we will have to get into high-dimensional SVD operations, which are what we try to avoid. On the other hand, if we choose r to be too small, then we may experience serious sacrifice in throughput performance. So the goal is to find an optimal r that offers the best trade-off. Unfortunately, finding the optimal value of r (in terms of maximizing network throughput) is not trivial, as the problem is intractable, due to the large search space and non-convex objective function.

To gain some insight on what value of r should be, we conduct the following experiment. We generate 1,000 channel instances. For each instance, we enumerate all possible r's and calculate its corresponding throughput C. For the time being, we focus only on the objective function (throughput) and defer consideration of computation time till later. In the experiment, we use the same settings as those used in Sec. IV-B. The number of clusters L_{cl} and rays L_{ray} are both set to 3. In Fig. 5, the bars show the achieved network throughput as a function of approximation rank r under different SNR. Note that when $r = 16$, the achieved throughput value (the first blue bar in each figure) is what is achieved by standard SVD (as in traditional BD method). As the value of r decreases, we observe the throughput goes up at first and then goes down. This suggests the lower rank r indeed offers the opportunity for higher throughput, especially at low or moderate SNR scenarios. Under this setting, setting $r = \frac{M_{\text{BS}}}{2} = 10$ would offer better (or comparable) performance than that with $r = 16$ in most instances.

The results in Fig. 5 are averaged over 1,000 channel instances. However, our interest is on a particular channel instance, and the optimal choice of r based on averaging over 1,000 channel instances may not perform well on this particular instance. Therefore, we propose to employ multiple choices of promising r's in parallel and derive multiple beamforming candidates corresponding to these r's. That is, we execute several different lower rank approximations simultaneously, where the set of target rank is given by

$$\mathcal{R} = \{r - \delta, \cdots, \ r - 1 \ , \ r \ , \ r + 1 \ , \cdots, r + \delta\}, \tag{14}$$

where r is around $\frac{M_{\text{BS}}}{2}$ (which may be adjusted according to empirical statistics), and δ is a parameter to control the number of elements in $\mathcal{R}$. As $|\mathcal{R}|$ different lower rank approximations are implemented, we will have $|\mathcal{R}|$ different solutions of $\mathbf{F}_{\text{BB},k}$ for each user on each RB after Stage C. Among these $|\mathcal{R}|$ solutions, we evaluate their throughput performance (i.e., C in Eq. (3)) and choose the one that offers the largest objective value as final beamforming matrix.

V. Implementation and Experimental Evaluation

In this section, we first present our implementation of the design in Sec. IV. Then we present our experimental evaluation of Turbo-HB.

A. Implementation

Our implementation is done on a Dell desktop computer with an Intel Xeon E5-2687W v4 CPU (3.0 GHz) and an Nvidia Quadro P6000 GPU. The programming tool CUDA (version 10.0.130) [36] is used to realize our algorithm and schedule the processing cores. Data communications between CPU and GPU are through a PCIe 3.0 X16 slot with the default configuration.

For a successful implementation of Turbo-HB, we must have a thorough knowledge of the capability and limitation of the GPU and find a way to fit our problem optimally into the platform. In general, the more parallelism and less overhead

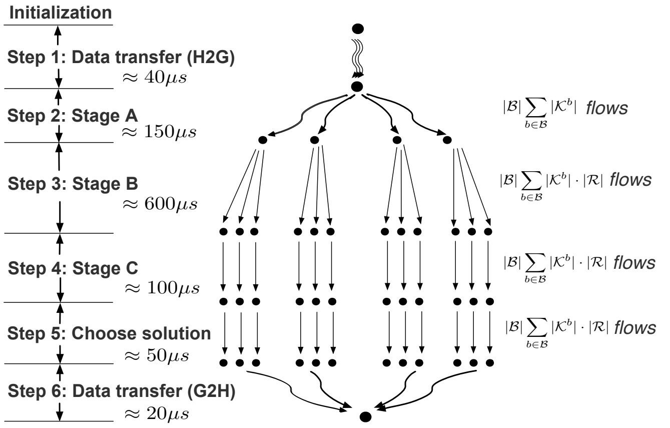

Initialization
Step 1: Data transfer (H2G)
$\approx 40\mu s$
Step 2: Stage A
$\approx 150\mu s$
Step 3: Stage B
$\approx 600\mu s$
Step 4: Stage C
$\approx 100\mu s$
Step 5: Choose solution
$\approx 50\mu s$
Step 6: Data transfer (G2H)
$\approx 20\mu s$
$|\mathcal{B}|\sum_{b\in\mathcal{B}}|\mathcal{K}^b|$ flows
$|\mathcal{B}|\sum_{b\in\mathcal{B}}|\mathcal{K}^b|\cdot|\mathcal{R}|$ flows
$|\mathcal{B}|\sum_{b\in\mathcal{B}}|\mathcal{K}^b|\cdot|\mathcal{R}|$ flows
$|\mathcal{B}|\sum_{b\in\mathcal{B}}|\mathcal{K}^b|\cdot|\mathcal{R}|$ flows

Fig. 6: Workflow of Turbo-HB on GPU.

in the implementation, the better the performance we may achieve. As such, we focus on the following two fronts in our implementation:

1) fully utilize GPU processing cores,
2) minimize memory access time.

The key to fully utilize GPU processing cores is to generate a sufficient amount of parallel workloads to feed the GPU. By our design in Sec. IV, the computations for beamforming matrice are independent among different RBs, different users, and different target ranks. Thus, we can spread out the computation over all available processing cores in parallel. Among all computation steps, the largest number of threads we used is $|\mathcal{B}|\sum_{b\in\mathcal{B}}|\mathcal{K}^b|\cdot|\mathcal{R}|\cdot M_{\text{BS}}^2$ for the ED operation, which is large enough to occupy all available GPU resources (3840 processing cores in our GPU). The workflow of Turbo-HB on GPU is illustrated in Fig. 6. A detailed description of the workflow can be found in [34].

Now we discuss two specific techniques that we have employed in Turbo-HB to enhance parallelism and reduce memory access time.

Batching Batched matrix operations are critical to our problem, as we have to execute a large number of independent matrix operations following the same procedure. As an example, suppose we need to execute hundreds or even thousands of matrix multiplications simultaneously. The programmer needs to generate a kernel with a sufficient number of threads and divide these threads into a number of groups. Then each group computes one or a few matrix multiplications, such that this kernel is able to perform batched matrix multiplications. Similarly, other matrix operations (following the same procedure), such as a large number of independent matrix ED operations, should be programmed in a batched manner to fully occupy the processing cores.

Minimizing global memory access Compared to other types of memory access, accessing global memory is much more time-consuming. We identify two techniques that can help minimize global memory access in our problem.

First, the programmer should carefully coalesce memory access, i.e., consolidating multiple memory accesses into a single transaction. This is particularly important when we handle a large number of matrix operations. The key to memory coalescing is to store the matrices consecutively in the memory with proper indexing. Then the programmer can allow consecutive threads to read consecutive (and aligned) memory and minimize the number of transactions.

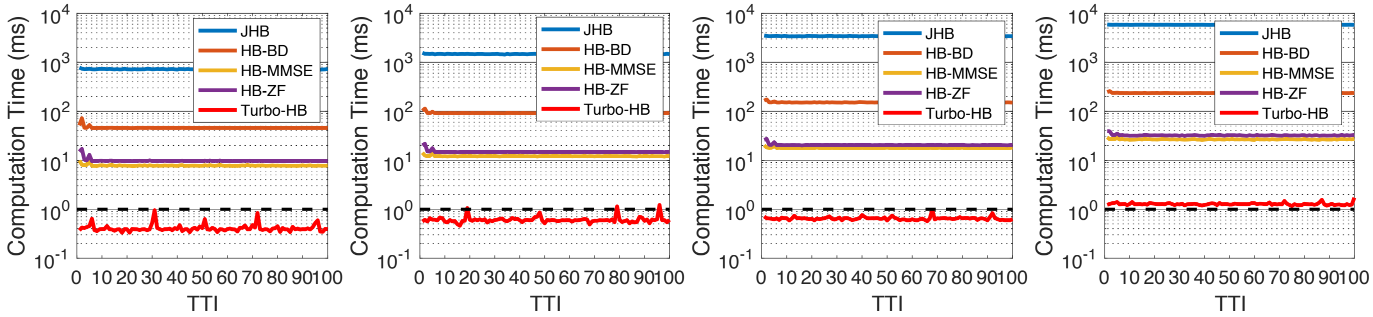
Second, instead of global memory accesses, which is more time-consuming, we can use on-chip shared memory accesses, which is much faster (but with limited storage space). Suppose we want to compute a matrix multiplication $\mathbf{C}_{m\times n} = \mathbf{A}_{m\times l}\mathbf{B}_{l\times n}$. A straightforward way for parallelism is to program each thread to take care of one element of $\mathbf{C}$. Then we need to read $\mathbf{A}$ n times from the global memory and $\mathbf{B}$ m times. In contrast, if matrix multiplication is based on shared memory [36], we only need to read $\mathbf{A}$ for (n / block size) times from the global memory and $\mathbf{B}$ for (m / block size) times. The remaining computations are done by accessing the shared memory.

B. Experimental Results

In this section, we present our experimental results, with a focus on timing and throughput performance. We also compare with other sequential HB schemes. The widely adopted DFT-codebook based method for analog beamforming [9, 32] is used for all schemes. For digital beamforming schemes, we choose HB-BD [9], HB-MMSE and HB-ZF for comparison. We also include one joint analog and digital HB method (JHB) [6] to show its timing performance.

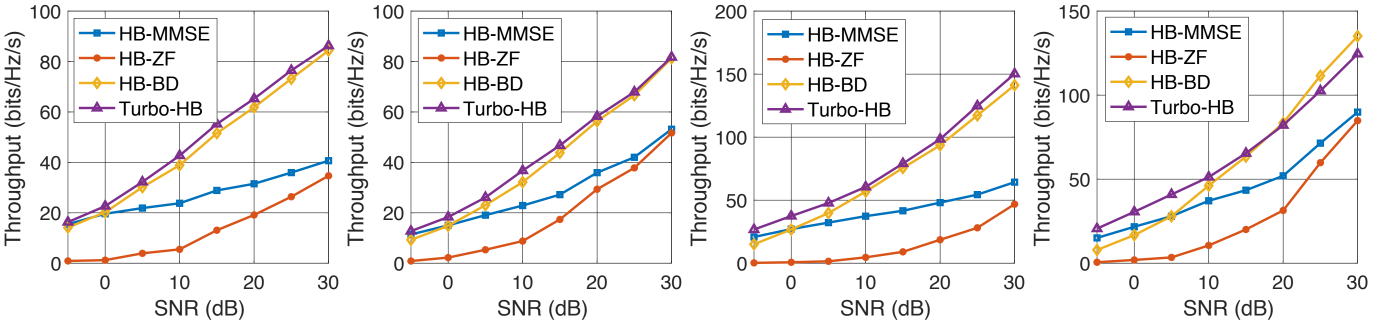
Experimental Setup. We consider a cellular communication scenario with one BS and a number of users. The number of available RBs is $|\mathcal{B}| = 100$. The BS is equipped with 128 antennas and each user is equipped with 16 antennas (a typical number for hybrid architecture at mmWave frequencies [3, 5, 9]). The number of RF chains at the BS varies while the number of RF chains at a user is 2. Each active link is assumed to transmit $N_s = 2$ data streams. To fully exploit concurrent data transmission, the number of users for MU-MIMO on each RB is set to $|\mathcal{K}^b| = \frac{M_{\text{BS}}}{N_s}$. For the channels, we use the widely adopted mmWave channel model as described in [5]. The number of clusters L_{cl}, the number of propagation paths L_{ray} caused by each cluster and SNR (i.e., $\frac{P_{\text{T}}}{\sigma^2}$) will be given under different settings. The angle spread σ_{AS} is set to 5 degrees. We set parameter δ (as defined in Sec. IV-C) to 2.

Timing Performance. We first verify that Turbo-HB can indeed meet ∼1 ms timing requirement. This timing result includes time used for all numerical computations as well as time consumed for data transfer between CPU and GPU. We run the experiment with the following settings: (a) $M_{\text{BS}} = 8$, (b) $M_{\text{BS}} = 12$, (c) $M_{\text{BS}} = 16$, and (d) $M_{\text{BS}} = 20$. For the sequential algorithms (HB-BD, HB-MMSE and HB-ZF), we only count the computation time of digital beamforming part. But for the joint algorithm (JHB), we have to count time consumed both for its digital beamforming and analog beamforming part since they are inseparable. Our GPU-based algorithm is run on CUDA platform while others are run on Matlab platform. Fig. 7 shows the computation time for



(a) M_{BS} = 8, $|K^b|$ = 4 (b) M_{BS} = 12, $|K^b|$ = 6 (c) M_{BS} = 16, $|K^b|$ = 8 (d) M_{BS} = 20, $|K^b|$ = 10

Fig. 7: Comparison of computation time of different schemes under different MU-MIMO scenarios.



(a) L_{cl} = L_{ray} = 3, M_{BS} = 8 (b) L_{cl} = L_{ray} = 6, M_{BS} = 8 (c) L_{cl} = L_{ray} = 3, M_{BS} = 16 (d) L_{cl} = L_{ray} = 6, M_{BS} = 16

Fig. 8: Comparison of throughput achieved by different schemes under different MU-MIMO scenarios.

100 consecutive TTIs. JHB, HB-BD, HB-MMSE and HB-ZF require a computation time on the order of 10^3 ms, 10^2 ms, 10^1 ms and 10^1 ms, respectively. Our experiments show Turbo-HB finds beamforming solution in 0.412 ms, 0.599 ms, 0.647 ms and 1.277 ms under each setting, averaged by 100 TTIs, which is able to meet the timing requirement ($\sim$1 ms) for up to 10 MU-MIMO users for each RB. Note that even though HB-BD may be parallelized for $|\mathcal{B}| = 100$ RBs, it is not possible to cut down the computation time by $100\times$ to meet the timing requirement (from 10^2s of ms to 10^0 ms), as we did with Turbo-HB.

Throughput Performance. Fig. 8 presents the results for network throughput under 4 different settings: (a) $N_{\rm cl} = N_{\rm ray} = 3$, $M_{\rm BS} = 32$, (b) $N_{\rm cl} = N_{\rm ray} = 6$, $M_{\rm BS} = 32$, (c) $N_{\rm cl} = N_{\rm ray} = 3$, $M_{\rm BS} = 16$, and (d) $N_{\rm cl} = N_{\rm ray} = 3$, $M_{\rm BS} = 16$. In all four cases, throughput under conventional HB-MMSE and HB-ZF methods appears to be inferior to others, as MMSE and ZF are not designed for mmWave systems and the poorly conditioned channel greatly degrades MMSE/ZF's performance [11, 15, 16]. In Fig. 8, both Turbo-HB and classical HB-BD offer comparable performance and are better than the others. Specifically, on average over all four settings, Turbo-HB achieved 107.3% throughput performance when normalized with respect to that by HB-BD. Moreover, when the number of channel paths is small and SNR is low, Turbo-HB is able to obtain even better performance. The reason behind this was given in our discussions in Sec. IV-C.

VI. Conclusions

This paper addresses the real-time challenge involved in matrix computation for HB. We presented Turbo-HB, the world-first ultra-fast beamforming design and implementation under the HB architecture. To reduce computation time, we developed low-complexity SVD by exploiting channel sparsity at mmWave frequencies. Further, we developed fully functioning parallelism for Turbo-HB, with optimized matrix operations and minimized memory accesses. We implemented Turbo-HB on a COTS Nvidia GPU and conducted experiments to validate its performance. Our experimental results showed that Turbo-HB is able to find beamforming matrices successfully in $\sim$1 ms. It is also able to offer competitive throughput performance compared with the state-of-the-art algorithms. Turbo-HB offers a practical solution to deploy HB in the field.

Acknowledgements

The first author wishes to thank Prof. Mark Embree from the Department of Mathematics, Virginia Tech, for his insightful discussion on SVD computations. The authors thank the anonymous reviewers for their feedback. This research was supported in part by NSF under grants 1800650 and 1617634. We thank Nvidia AI Lab (NVAIL) in Santa Clara, CA for its unrestricted gift and equipment donation to our research. All opinions expressed in this paper are the authors' and do not necessarily reflect the views and opinions of NSF or Nvidia.

References

[1] W. Hong, K. Baek, Y. Lee, Y. Kim and S. Ko, "Study and prototyping of practically large-scale mmWave antenna systems for 5G cellular devices," *IEEE Comm. Magazine*, vol. 52, no. 9, pp. 63–69, Sept. 2014.

[2] S. Han, C. I, Z. Xu and C. Rowell, "Large-scale antenna systems with hybrid analog and digital beamforming for millimeter wave 5G," *IEEE Comm. Magazine*, vol. 53, no. 1, pp. 186–194, Jan. 2015.

[3] A.F. Molisch, V.V. Ratnam, S. Han, Z. Li, S.L.H. Nguyen, L. Li and K. Haneda, "Hybrid beamforming for massive MIMO: A survey," *IEEE Comm. Magazine*, vol. 55, no. 9, pp. 134–141, Dec. 2017.

[4] W. Ni, X. Dong and W.S. Lu, "Near-optimal hybrid processing for massive MIMO systems via matrix decomposition," *IEEE Trans. on Signal Proc.*, vol. 65, no. 15, pp. 3922–3933, Aug. 2017.

[5] O. El Ayach, S. Rajagopal, S. Abu-Surra, Z. Pi and R.W. Heath Jr., "Spatially sparse precoding in millimeter wave MIMO systems," *IEEE Journal on Selected Areas in Comm.*, vol. 13, no. 3, pp. 1499–1513, Mar. 2014.

[6] T.E. Bogale and L.B. Le, "Beamforming for multiuser massive MIMO systems: Digital versus hybrid analog-digital," in *Proc. of IEEE GLOBECOM*, pp. 4066–4071, Austin, TX, Dec. 2014.

[7] C. Rusu, R. Mendez-Rial, N. Gonzalez-Prelcic and R.W. Heath, "Low complexity hybrid precoding strategies for millimeter wave communication systems," *IEEE Trans. on Wireless Comm.*, vol. 15, no. 12, pp. 8380–8393, Dec. 2016.

[8] A. Alkhateeb, J. Mo, N. Gonzalez-Prelcic and R.W. Heath, "MIMO precoding and combining solutions for millimeter-wave systems," *IEEE Comm. Magazine*, vol. 52, no. 12, pp. 122–131, Dec. 2014.

[9] W. Ni and X. Dong, "Hybrid block diagonalization for massive multiuser MIMO systems," *IEEE Trans. on Comm.*, vol. 64, no. 1, pp. 201–211, Jan. 2016.

[10] L. Liang, W. Xu and X. Dong, "Low-complexity hybrid precoding in massive multiuser MIMO systems," *IEEE Wireless Comm. Letters*, vol. 3, no. 6, pp. 653–656, Dec. 2014.

[11] Y. Ghasempour, M.K. Haider, C. Cordeiro, D. Koutsonikolas and E. Knightly, "Multi-stream beam-training for mmWave MIMO networks," in *Proc. of ACM MobiCom*, pp. 225-–239, New Delhi, India, Jan. 2018.

[12] Y. Ghasempour and E.W. Knightly, "Decoupling beam steering and user selection for scaling multi-user 60 GHz WLANs," in *Proc. of ACM MobiHoc*, pp. 1–10, Chennai, India, July 2017.

[13] S. Sur, I. Pefkianakis, X. Zhang, and K.H. Kim, "Towards scalable and ubiquitous millimeter-wave wireless networks," in *Proc. of ACM MobiCom 2018*, pp. 257–271, New Delhi, India, Oct. 2018.

[14] Q.H. Spencer, A.L. Swindlehurst and M. Haardt, "Zero-forcing methods for downlink spatial multiplexing in multiuser MIMO channels," *IEEE Trans. on Signal Proc.*, vol. 52, no. 2, pp. 461–471, Feb. 2004.

[15] S. K. Mohammed and E. G. Larsson, "Improving the performance of the zero-forcing multiuser MISO downlink precoder through user grouping," *IEEE Trans. on Wireless Comm.*, vol. 15, no. 2, pp. 811–826, Feb. 2016.

[16] V. Stankovic and M. Haardt, "Multi-user MIMO downlink precoding for users with multiple antennas," *Wireless World Research Forum*, pp. 12–14, Toronto, ON, Canada, Nov. 2004.

[17] D. Patil, "Block diagonalization based beamforming," *Master Thesis*, KTH Royal Institute of Technology, Stockholm, Sweden, 2017.

[18] Y. Huang, S. Li, Y.T. Hou and W. Lou, "GPF: a GPU-based design to achieve $\sim$100 μs scheduling for 5G NR," in *Proc. of ACM MobiCom*, pp. 207–222, New Delhi, India, Oct. 2018.

[19] Z. Shen, R. Chen, J.G. Andrews, R.W. Heath and B.L. Evans, "Low complexity user selection algorithms for multiuser MIMO systems with block diagonalization," *IEEE Trans. on Signal Proc.*, vol. 54, no. 9, pp. 3658–3663, Sept. 2006.

[20] X. Zhang and J. Lee, "Low complexity MIMO scheduling with channel decomposition using capacity upperbound," *IEEE Trans. on Comm.*, vol. 56, no. 6, pp. 871–876, June 2008.

[21] W. Yang, G. Durisi and E. Riegler, "On the capacity of large-MIMO block-fading channels," *IEEE Journal on Selected Areas in Comm.*, vol. 31, no. 2, pp. 117–132, Feb. 2013.

[22] A. Alkhateeb, G. Leus and R.W. Heath, "Limited feedback hybrid precoding for multi-user millimeter wave systems," *IEEE Trans. on Wireless Comm.*, vol. 14, no. 11, pp. 6481–6494, July 2015.

[23] S.S. Christensen, R. Agarwal, E. Carvalho and J.M. Cioffi, "Weighted sum-rate maximization using weighted MMSE for MIMO-BC beamforming design," *IEEE Trans. on Wireless Comm.*, vol. 7, no. 12, pp. 4792-–4799, Dec. 2008.

[24] A. Zhou, X. Zhang and H. Ma, "Beam-forecast: facilitating mobile 60 GHz networks via model-driven beam steering," in *Proc. of IEEE INFOCOM*, pp. 1–9, Atlanta, GA, May 2017.

[25] M.K. Haider, Y. Ghasempour, D. Koutsonikolas, and E.W. Knightly, "LiSteer: mmWave beam acquisition and steering by tracking indicator LEDs on wireless APs," in *Proc. of ACM MobiCom*, pp. 273–288, New Delhi, India, Oct. 2018.

[26] E. Bjornson, L. Van der Perre, S. Buzzi and E.G. Larsson, "Massive MIMO in sub-6 GHz and mmWave: physical, practical, and use-case differences," *IEEE Wireless Comm.*, vol. 26, no. 2, pp. 100–108, Apr. 2019.

[27] V. Raghavan, A. Partyka, A. Sampath, S. Subramanian, O.H. Koymen, K. Ravid, J. Cezanne, K. Mukkavilli and J. Li, "Millimeter-wave MIMO prototype: measurements and experimental results," *IEEE Comm. Magazine*, vol. 56, no. 1, pp. 202–209, Jan. 2018.

[28] T.S. Rappaport, *Wireless Communications: Principles and Practice*, Chapter 4, New Jersey: Prentice Hall PTR, 1996. ISBN: 9780133755367.

[29] N. Halko, P.G. Martinsson and J.A. Tropp, "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, May 2011.

[30] IEEE Standards Association, "IEEE standards 802.11ac-2013: enhancements for very high throughput for operation in bands below 6 GHz," Dec. 2013. Available: https://ieeexplore.ieee.org/document/6687187

[31] G.H. Golub and C.F. Van Loan, *Matrix Computations (4th Edition)*, Chapter 8, Johns Hopkins University Press, 2013. ISBN-13: 978-1421407944.

[32] N. Song, H. Sun and T. Yang, "Coordinated hybrid beamforming for millimeter wave multi-user massive MIMO systems," in *Proc. of IEEE GLOBECOM*, pp. 1–6, Washington, DC, Dec. 2016.

[33] T.S. Rappaport, E. Ben-Dor, J.N. Murdock and Y. Qiao, "38 GHz and 60 GHz angle-dependent propagation for cellular & peer-to-peer wireless communications," in *Proc. of IEEE ICC*, pp. 4568–4573, Ottawa, ON, June 2012.

[34] Y. Chen, Y. Huang, C. Li, Y.T. Hou and W. Lou, "Turbo-HB: A Novel Design and Implementation to Achieve Ultra-Fast Hybrid Beamforming," *Technical Report*, Dept. of ECE, Virginia Tech, Jan. 2020. Available: https://sites.google.com/vt.edu/ychen-infocom2020-tr-pdf

[35] J. Blinn, "Consider the lowly 2 x 2 matrix," *IEEE Computer Graphics and Applications*, vol. 16, no. 2, pp. 82–88, Mar. 1996.

[36] Nvidia, "CUDA C programming guide v10.0.130." Available: http://docs. nvidia.com/cuda/cuda-c-programming-guide/index.html