Householder Dice: A Matrix-Free Algorithm for Simulating Dynamics on Gaussian and Random Orthogonal Ensembles

Yue M. Lu

***Abstract*—This paper proposes a new algorithm, named Householder Dice (HD), for simulating dynamics on dense random matrix ensembles with rotational invariance. Examples include the Gaussian ensemble, the Haar-distributed random orthogonal ensemble, and their complex-valued counterparts. A "direct" approach to the simulation, where one first generates a dense $n \times n$ matrix from the ensemble, requires at least $\mathcal{O}(n^2)$ resource in space and time. The HD algorithm overcomes this $\mathcal{O}(n^2)$ bottleneck by using the principle of deferred decisions: rather than fixing the entire random matrix in advance, it lets the randomness unfold with the dynamics. At the heart of this matrix-free algorithm is an adaptive and recursive construction of (random) Householder reflectors. These orthogonal transformations exploit the group symmetry of the matrix ensembles, while simultaneously maintaining the statistical correlations induced by the dynamics. The memory and computation costs of the HD algorithm are $\mathcal{O}(nT)$ and $\mathcal{O}(nT^2)$, respectively, with T being the number of iterations. When $T \ll n$, which is nearly always the case in practice, the new algorithm leads to significant reductions in runtime and memory footprint. Numerical results demonstrate the promise of the HD algorithm as a new computational tool in the study of high-dimensional random systems.**

***Index Terms*—Dynamics, message passing, Haar measure, Householder reflection, random matrices**

I. Introduction

To do research involving large random systems, one must make a habit of experimenting on the computer. Indeed, computer simulations help verify theoretical results and provide new insights, not to mention that they can also be incredibly fun. For many problems in statistical learning, random matrix theory, and statistical physics, the simulations that one encounters are often given as an iterative process in the form of

$$\boldsymbol{x}_{t+1} = f_t(\boldsymbol{M}_t\boldsymbol{x}_t, \boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_{t-d}), \qquad \text{for } 1 \le t \le T. \tag{1}$$

Here, $\boldsymbol{M}_t$ is either $\boldsymbol{Q}$ or $\boldsymbol{Q}^{\mathsf{T}}$, where $\boldsymbol{Q}$ is a random matrix; $f_t(\cdot)$ denotes some general vector-valued function that maps $\boldsymbol{M}_t\boldsymbol{x}_t$ and a few previous iteration vectors $\{\boldsymbol{x}_{t-i}\}_{0\le i\le d}$ to the next one $\boldsymbol{x}_{t+1}$; and T is the total number of iterations.

Y. M. Lu is with the John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA (e-mail: yuelu@seas.harvard.edu). The initial part of this work was done during his sabbatical at the École normale supérieure (ENS) in Paris, France in Fall 2019. He thanks colleagues at the ENS for their hospitality and stimulating discussions. This work was supported by the Harvard FAS Dean's Fund for Promising Scholarship, by the chaire CFM-ENS "Science des donnees", and by the US National Science Foundation under grants CCF-1718698 and CCF-1910410.

With suitable definitions of the mappings $f_t(\cdot)$, the formulation in (1) includes many well-known algorithms as its special cases. A classical example is to use iterative methods [1] to compute the extremal eigenvalues/eigenvectors of a (spiked) random matrix [2], [3]. Other examples include approximate message passing on dense random graphs [4]–[8], and gradient descent algorithms for solving learning and estimation problems with random design [9], [10]. In this paper, we show that all of these algorithms can be simulated by an efficient *matrix-free* scheme, if the random matrix $\boldsymbol{Q}$ is drawn from an ensemble with rotational invariance. Examples of such ensembles include the i.i.d. Gaussian (i.e. the rectangular Ginibre) ensemble, the Haar-distributed random orthogonal ensemble, the Gaussian orthogonal ensemble, and their complex-valued counterparts.

What is wrong with the standard way of simulating (1), where we first draw a sample $\boldsymbol{Q}$ from the matrix ensemble and then carry through the iterations? This direct approach is straightforward to implement, but it cannot handle large dimensions. To see this, suppose that $Q \in \mathbb{R}^{m\times n}$ with $m \asymp n$. We shall also assume that the computational cost of the nonlinear mapping $f_t(\cdot)$ is $\mathcal{O}(n)$. It follows that, at each iteration of (1), most of the computation is spent on the matrix-vector multiplication $\boldsymbol{M}_t\boldsymbol{x}_t$, at a cost of $\mathcal{O}(n^2)$ work. It is not at all obvious that one can do much better: Merely generating an $n \times n$ Gaussian matrix already requires $O(n^2)$ resource in computation and storage. When n is large, n^2 is huge. In practice, this $\mathcal{O}(n^2)$ bottleneck means that one cannot simulate (1) at a dimension much larger than $n = 10^4$ on a standard computer (in a reasonable amount of time). However, there are many occasions, especially in the study of high-dimensional random systems, where one does wish to simulate large random matrices. A common workaround is to choose a moderate dimension (*e.g.*, $n = 1000$), repeat the simulation over many independent trials, and then average the results to reduce statistical fluctuations. In addition to having to spend extra time on the repeated trials, this strategy can still suffer from strong finite size effects, making it a poor approximation of the true high-dimensional behavior of the underlying random systems. (An example is given in Section II-B to illustrate this issue.)

In this paper, we propose a new algorithm, named *Householder Dice* (HD), for simulating the dynamics in (1) on the Gaussian, Haar, and other related random matrix ensembles. Our new approach is *statistically-equivalent* to the direct approach discussed above, but the memory and computation

costs of the HD algorithm are $\mathcal{O}(nT)$ and $\mathcal{O}(nT^2)$, respectively, where T is the number of iterations. In many problems, T is much smaller than n. Typically, $T = \mathcal{O}(\text{polylog}(n))$. In such cases, the new algorithm leads to significant reductions in runtime and memory footprint. In the numerical examples presented in Section II, we show that the crossover value of n at which the HD algorithm outperforms the direct approach can be as low as 500. The speedup becomes orders of magnitude greater for $n \geq 10^4$. Moreover, the HD algorithm expands the limits of what could be done on standard computers by making it tractable to perform dense random matrix experiments in dimensions as large as $n = 10^7$.

The basic idea of the HD algorithm follows the so-called principle of deferred decisions [11]. Intuitively, each iteration of (1) only probes $\boldsymbol{Q}$ in a one-dimensional space spanned by $\boldsymbol{x}_t$. Thus, if the total number of iterations $T \ll n$, we only need to expose the randomness of $\boldsymbol{Q}$ over a few low-dimensional subspaces. It is then clearly wasteful to fix and store in memory the full matrix in advance. The situation is analogous to that of simulating a simple random walk for T steps. We can let the random choices gradually unfold with the progress of the walk, fixing only the randomness that must be revealed at any given step. The challenge in our problem though is that the dynamics in (1) can create a complicated dependence structure between the random matrix $\boldsymbol{Q}$ and the iteration vectors $\boldsymbol{x}_t, \boldsymbol{x}_{t-1} \ldots, \boldsymbol{x}_0$. Nevertheless, we show that this dependence structure can be exactly accounted for by an adaptive and recursive construction of (random) Householder reflectors [12], [13] which exploit the inherent group symmetry of the matrix ensembles.

Using Householder reflectors to speed up random matrix experiments is not a new idea. It is well-known [14], [15] that a Haar-distributed random orthogonal matrix can be factorized as a product of Householder reflectors. This leads to an efficient way of generating a random orthogonal matrix with $\mathcal{O}(n^2)$ operations (rather than the $\mathcal{O}(n^3)$ cost associated with a full QR decomposition on a Gaussian matrix). Householder reflectors have also been applied to reduce a Gaussian matrix to a particularly simple random bidiagonal form [16], [17]. This clever factorization leads to an $\mathcal{O}(n^2)$ algorithm for simulating the spectrum densities of Gaussian and Wishart matrices. (Recall that a standard eigenvalue decomposition on a dense matrix requires $\mathcal{O}(n^3)$ work in practice.) The proposed HD algorithm differs from the previous work in that it is a truly *matrix-free* construction. With the progress of the dynamics, it gradually builds a recursive set of (random) Householder reflectors based on the current iteration vector $\boldsymbol{x}_t$ and the history of the iterations up to this point. This adaptive, "on-the-fly" construction is essential for us to capture the correlation structures generated by the dynamics without fixing the matrix in advance.

The rest of the paper is organized as follows. We first present in Section II a few motivating examples to showcase the applications of the HD algorithm. Section III contains the main technical results of this paper. After a brief review of the basic properties of the Haar measure (on classical matrix groups) and Householder reflectors, we present the construction of the proposed algorithm for the Gaussian and random orthogonal ensembles. Theorems 1 and 2 establish the statistical equivalence of the HD algorithm and the direct approach to simulating (1). Generalizations to complex-valued and other related ensembles are discussed in Section III-D. We conclude the paper in Section IV.

II. Numerical Examples

Before delving into technical details, it is helpful to go through a few motivating applications that show how the HD algorithm can significantly speed up the simulation tasks.[1]

A. Lasso with Random Designs

In the first example, we consider the simulation of the lasso estimator widely used in statistics and machine learning. The goal is to estimate a sparse vector $\boldsymbol{\beta}^* \in \mathbb{R}^n$ from its noisy linear observation given by

$$\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{\beta}^* + \boldsymbol{w},$$

where $\boldsymbol{Q} \in \mathbb{R}^{m \times n}$ is a design (or covariate) matrix, and $\boldsymbol{w} \sim \mathcal{N}(0, \sigma_w^2 \boldsymbol{I})$ denotes the noise in $\boldsymbol{y}$. The lasso estimator is formulated as an optimization problem

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{2}$$

where $\widehat{\boldsymbol{\beta}}$ is an estimate of $\boldsymbol{\beta}^*$ and $\lambda > 0$ is a regularization parameter.

A popular method for solving (2) is the iterative soft-thresholding algorithm (ISTA) [19]:

$$\boldsymbol{x}_{t+1} = \eta_{\lambda\tau}[\boldsymbol{x}_t + \tau\boldsymbol{Q}^{\mathsf{T}}(\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{x}_t)], \qquad 0 \leq t < T, \tag{3}$$

where $\tau > 0$ denotes the step size and $\eta_{\lambda\tau}(x) = \text{sign}(x)\max\{|x| - \lambda\tau, 0\}$ is an element-wise soft-thresholding operator. In many theoretical studies of lasso, one assumes that the design matrix is random with i.i.d. normal entries, *i.e.* $Q_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \frac{1}{m})$. In this case, ISTA is an iterative process on a Gaussian matrix $\boldsymbol{Q}$ and its transpose. With some change of variables, we can rewrite (3) as a special case of the general dynamics given in (1), with one iteration of (3) mapped to two iterations of (1).

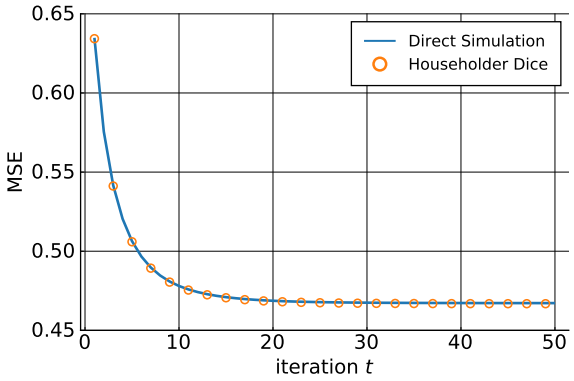
We simulate the ISTA dynamics using both the proposed HD algorithm and the direct simulation approach that fixes the Gaussian matrix $\boldsymbol{Q}$ in advance. In our experiments, the target sparse vector $\boldsymbol{\beta}^*$ has i.i.d. entries drawn from the Bernoulli-Gaussian prior

$$\beta_i^* \sim \rho\delta(\beta) + (1-\rho)\frac{1}{\sqrt{2\pi\sigma_s^2}}\exp\Big\{-\frac{\beta^2}{2\sigma_s^2}\Big\},$$

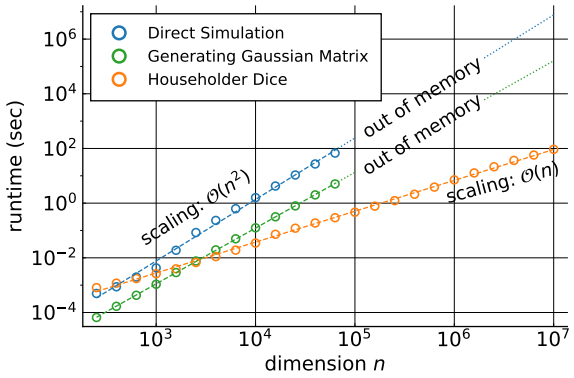
where $0 < \rho < 1$ and $\sigma_s > 0$ are two constants. The design matrix $\boldsymbol{Q}$ is of size $m \times n$ with $m = \lfloor n/2 \rfloor$.

Figure 1(a) shows the mean-squared error (MSE) $e^{(t)} \overset{\text{def}}{=} \frac{1}{n}\|\boldsymbol{x}_t - \boldsymbol{\beta}^*\|^2$ at each iteration of the dynamics, obtained by averaging over 10^5 independent trials. The dimension here is $n = 1000$. The results from the HD algorithm (the red circles

[1] All of the numerical experiments presented in this section have been done in Julia [18]. The source code implementing the HD algorithm is available online at https://github.com/yuelusip/HouseholderDice.



(a)



(b)

Fig. 1. Simulating the ISTA dynamics (3) using two approaches: the standard approach where the random matrix $\boldsymbol{Q}$ is generated in advance, and the proposed HD algorithm. (a) The time-varying MSE averaged over 10^5 independent trials, with the results from the two approaches matching. (b) Runtime versus the matrix dimension n, shown in log-log scale. In all the experiments, the parameters are set to $T = 50$, $\lambda = 2$, $\tau = 0.3$, $\rho = 0.2$, $\sigma_s = 2$ and $\sigma_w = 0.1$.

in the figure) match those from the standard approach (the blue line). This is expected, since Householder Dice is designed to be statistically equivalent to the direct approach. However, the two simulation approaches behave very differently in runtime and memory footprint, as shown in Figure 1(b). When we increase the dimension n, the runtime of the standard approach exhibits a quadratic growth rate $\mathcal{O}(n^2)$, whereas the runtime of the HD algorithm scales linearly with n. For comparison, we also plot in the figure the runtime for merely generating an i.i.d. Gaussian matrix $\boldsymbol{Q}$ of size $m \times n$.

For small dimensions ($250 \leq n < 500$), the HD algorithm takes slightly more time than the direct approach, likely due to the additional overhead in implementing the former. Starting from $n \geq 500$, it becomes the more efficient choice. In fact, for $n \geq 2500$, the HD algorithm can simulate the ISTA dynamics (for 50 iterations) in less time than it takes to generate the Gaussian matrix. For dimensions beyond $n = 10^5$, Householder Dice becomes the only feasible method, as implementing the direct approach would require more memory than available on the test computer (equipped with 32 GB of RAM). Finally, for $n = 10^7$, the runtime for the HD algorithm is 92 seconds, whereas by extrapolation the direct approach would have taken 7.7×10^6 seconds (approximately 89 days).

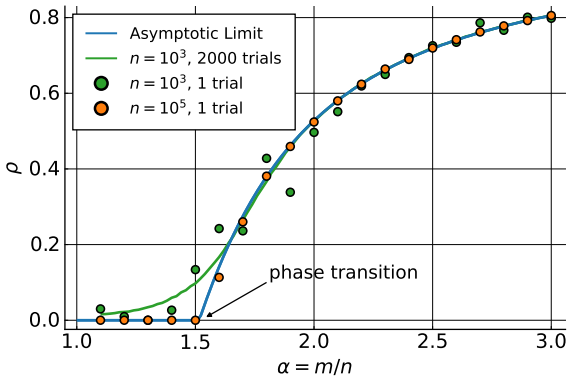


Fig. 2. Simulating the spectral method given in (4) and comparing the empirical results against the asymptotic predictions given in [24]. The result for $n = 10^3$ shows strong statistical fluctuations. This can be reduced by averaging over multiple independent trials, but the average curve still suffers from strong finite size effects, especially near the phase transition point. At $n = 10^5$, the match between the empirical results and the theoretical curve is nearly perfect in any (typical) trial.

B. Spectral Method for Generalized Linear Models

In the second example, we consider a spectral method [20]–[23] with applications in signal estimation and exploratory data analysis. Let $\boldsymbol{\xi}$ be an unknown vector in $\mathbb{R}^n$ and $\{\boldsymbol{a}_i\}_{1\leq i\leq m}$ a set of sensing vectors. We seek to estimate $\boldsymbol{\xi}$ from a number of generalized linear measurements $\left\{y_i = f(\boldsymbol{a}_i^\mathsf{T}\boldsymbol{\xi})\right\}_{1\leq i\leq m}$, where $f(\cdot)$ is some function modeling the acquisition process. The spectral method works as follows. Let

$$\boldsymbol{D} \overset{\text{def}}{=} \frac{1}{m}\boldsymbol{A}\,\mathrm{diag}\left\{y_1,\ldots,y_m\right\}\boldsymbol{A}^\mathsf{T}, \tag{4}$$

where $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m]$ is a matrix whose columns are the sensing vectors. Denote by $\boldsymbol{x}_1$ a normalized eigenvector associated with the largest eigenvalue of $\boldsymbol{D}$. This vector $\boldsymbol{x}_1$ is then our estimate of $\boldsymbol{\xi}$, up to a scaling factor. The performance of the spectral method is usually given in terms of the squared cosine similarity $\rho(\boldsymbol{\xi}, \boldsymbol{x}_1) = \frac{(\boldsymbol{\xi}^\mathsf{T}\boldsymbol{x}_1)^2}{\|\boldsymbol{\xi}\|^2\|\boldsymbol{x}_1\|^2}$.

Asymptotic limits of $\rho(\boldsymbol{\xi}, \boldsymbol{x}_1)$ have been derived for the cases where $\boldsymbol{A}$ is an i.i.d. Gaussian matrix [22], [23] or a subsampled random orthogonal matrix [24]. In our experiment, we consider the latter setting. Assume $m = \lfloor \alpha n \rfloor$ for some $\alpha > 1$. We can write

$$\boldsymbol{A} = \begin{bmatrix}\boldsymbol{I}_n & \boldsymbol{0}_{n\times(m-n)}\end{bmatrix}\boldsymbol{Q},$$

where $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is a random orthogonal matrix drawn from the Haar distribution.

We simulate the spectral method and compare its empirical performance with the asymptotic limit given in [24]. In our experiment, the measurement model is set to be $y_i = \tanh\left(\left|\boldsymbol{a}_i^\mathsf{T}\boldsymbol{\xi}\right|\right)$. We compute the leading eigenvector $\boldsymbol{x}_1$ by using the Krylov-Schur algorithm [1]. Just like the standard power method, this algorithm involves the repeated multiplication of $\boldsymbol{D}$ with some vectors, but it typically needs fewer iterations to achieve the same accuracy. With the forms of $\boldsymbol{D}$ and $\boldsymbol{A}$ given above, the algorithm can again be regarded as a special case of the general dynamics in (1). We use the HD algorithm for the simulation and show the results in Figure 2

for two different matrix dimensions: $n = 10^3$ and $n = 10^5$. Observe that, at $n = 10^3$, there is still noticeable fluctuations between the actual performance of the spectral method (shown as green dots in the figure) and the theoretical prediction (the blue line). To get a better match, the standard practice is to do many independent trials (2000 in our experiment) and average the results. This gives us the green curve in the figure. Averaging can indeed reduce statistical fluctuations, but there are still strong finite size effects, especially near the phase transition point. This is a case where the capability of the proposed HD algorithm to handle large matrices becomes particularly attractive: when we increase the dimension to $n = 10^5$, the empirical results match the theoretical curve very closely in any (typical) trial, with no need for averaging over repeated simulations. In terms of runtime, it takes the HD algorithm less than 4 seconds on average to obtain an extremal eigenvalue/eigenvector of $\boldsymbol{D}$ for $n = 10^5$.

III. Main Results

Notation: In what follows, $\boldsymbol{e}_i$ denotes the ith natural basis vector, and $\boldsymbol{Z}_i \stackrel{\text{def}}{=} \boldsymbol{I} - \boldsymbol{e}_i\boldsymbol{e}_i^{\mathsf{T}}$. For $i \le j$, we use $\boldsymbol{Z}_{i:j}$ as a shorthand notation for $\prod_{i\le k\le j} \boldsymbol{Z}_k$. The dimension of $\boldsymbol{Z}_i$ and $\boldsymbol{Z}_{i:j}$ is either $m \times m$ or $n \times n$, which will be made clear from the context. For any $\boldsymbol{v} \in \mathbb{R}^n$, the "slicing" operation that takes a subset of $\boldsymbol{v}$ is denoted by

$$\boldsymbol{v}[i:j] \stackrel{\text{def}}{=} [v_i, v_{i+1}, \ldots, v_j]^{\mathsf{T}},$$

where $1 \le i \le j \le n$. Block diagonal matrices will be written as

$$\begin{bmatrix} \boldsymbol{A}_1 & \\ & \boldsymbol{A}_2 \end{bmatrix},$$

where the zero entries in the off-diagonal blocks are omitted. We use

$$\mathbb{O}(n) \stackrel{\text{def}}{=} \{\boldsymbol{M} \in \mathbb{R}^{n\times n} : \boldsymbol{M}\boldsymbol{M}^{\mathsf{T}} = \boldsymbol{I}_n\}$$

to denote the set of $n \times n$ orthogonal matrices, and $\mathbb{U}(n) \stackrel{\text{def}}{=} \{\boldsymbol{M} \in \mathbb{C}^{n\times n} : \boldsymbol{M}\boldsymbol{M}^{*} = \boldsymbol{I}_n\}$ its complex-valued counterpart. We will be mainly focusing on two real-valued random matrix ensembles: Ginibre(m, n) represents the ensemble of $m \times n$ matrices with i.i.d. standard normal entries, and Haar(n) represents the ensemble of random orthogonal matrices drawn from the Haar measure on $\mathbb{O}(n)$. The generalizations to the complex-valued cases and other closely related ensembles will be discussed in Section III-D.

A. Preliminaries

The ensembles Ginibre(m, n) and $\mathbb{O}(n)$ share an important property: they are both *invariant* with respect to multiplications by orthogonal matrices. For example, for any $\boldsymbol{G}$ drawn from Ginibre(m, n), it is easy to verify that

$$\boldsymbol{G} \sim \text{Ginibre}(m, n) \implies \boldsymbol{U}\boldsymbol{G}\boldsymbol{V} \sim \text{Ginibre}(m, n), \tag{5}$$

where $\boldsymbol{U} \in \mathbb{O}(m), \boldsymbol{V} \in \mathbb{O}(n)$ are any two deterministic or *random* orthogonal matrices independent of $\boldsymbol{G}$.

Rotational-invariant (or more precisely, group translation-invariant) properties similar to (5) are actually what defines the Haar measure. We call a probability measure μ on $\mathbb{O}(n)$ a Haar measure if

$$\mu(\mathcal{A}) = \mu(\boldsymbol{U} \circ \mathcal{A}) = \mu(\mathcal{A} \circ \boldsymbol{U}) \tag{6}$$

for any measurable subset $\mathcal{A} \subset \mathbb{O}(n)$ and any fixed $\boldsymbol{U} \in \mathbb{O}(n)$. Here, $\boldsymbol{U}\circ\mathcal{A} \stackrel{\text{def}}{=} \{\boldsymbol{U}\boldsymbol{V} : \boldsymbol{V} \in \mathcal{A}\}$ and $\mathcal{A}\circ\boldsymbol{U}$ is defined similarly. It is a classical result (see, e.g., [25, Theorem 5.14]) that there is one, and only one, translation-invariant probability measure in the sense of (6) on $\mathbb{O}(n)$. In fact, the theorem holds in much greater generality. For example, it remains true for any compact Lie group, which includes $\mathbb{O}(n)$ [and $\mathbb{U}(n)$] as its special case.

An additional property of $\mathbb{O}(n)$, $\mathbb{U}(n)$ (and compact Lie groups in general) is that left-invariance [the first equality in (6)] implies right-invariance (the second equality), and vice versa. This then allows us to have a simplified characterization of the Haar measure on $\mathbb{O}(n)$. Specifically, to show that a random orthogonal matrix $\boldsymbol{Q} \sim$ Haar(n), it is sufficient to verify that

$$\boldsymbol{Q} \stackrel{d}{=} \boldsymbol{U}\boldsymbol{Q}$$

for any fixed $\boldsymbol{U} \in \mathbb{O}(n)$, where $\stackrel{d}{=}$ means that two random variables have the same distribution. We will use this convenient characterization in Section III-C, when we establish the statistical equivalence between the proposed HD algorithm and the direct simulation of (1).

Finally, we recall the construction of Householder reflectors [12], [13] from numerical linear algebra, as they will play important roles in our subsequent discussions. Given a vector $\boldsymbol{v} \in \mathbb{R}^n$, how can we build an orthogonal matrix $\boldsymbol{H}$ such that $\boldsymbol{H}\boldsymbol{v} = \|\boldsymbol{v}\|\,\boldsymbol{e}_1$? This is exactly the problem addressed by Householder reflectors, defined here as

$$\boldsymbol{H}(\boldsymbol{v}) \stackrel{\text{def}}{=} -\operatorname{sign}(v_1)\Big(\boldsymbol{I} - 2\,\frac{\boldsymbol{u}\boldsymbol{u}^{\mathsf{T}}}{\boldsymbol{u}^{\mathsf{T}}\boldsymbol{u}}\Big), \tag{7}$$

where $\boldsymbol{u} = \boldsymbol{v} + \operatorname{sign}(v_1)\|\boldsymbol{v}\|\,\boldsymbol{e}_1$, and $\operatorname{sign}(v_1) = 1$ if $v_1 \ge 0$ and -1 otherwise. The choice of the sign in (7) helps improve numerical stability by making sure that the denominator $\boldsymbol{u}^{\mathsf{T}}\boldsymbol{u}$ stays bounded away from zero. (See [13, Lecture 10] for details.)

By construction, $\boldsymbol{H}(\boldsymbol{v})$ is a symmetric matrix whose eigenvalues are equal to ± 1. It follows that $\boldsymbol{H}(\boldsymbol{v}) \in \mathbb{O}(n)$. Moreover, we can verify from direct calculations that

$$\boldsymbol{H}(\boldsymbol{v})\boldsymbol{e}_1 = \boldsymbol{v}/\|\boldsymbol{v}\| \quad \text{and} \quad \boldsymbol{H}(\boldsymbol{v})\boldsymbol{v} = \|\boldsymbol{v}\|\,\boldsymbol{e}_1. \tag{8}$$

Geometrically, $\boldsymbol{H}(\boldsymbol{v})$ represents a reflection across the exterior (or interior) angle bisector of $\boldsymbol{v}/\|\boldsymbol{v}\|$ and $\boldsymbol{e}_1$. It is widely used in numerical linear algebra thanks to its low memory/computational costs. The matrix $\boldsymbol{H}(\boldsymbol{v})$ itself can be efficiently represented with $\mathcal{O}(n)$ space, and matrix-vector multiplications involving $\boldsymbol{H}(\boldsymbol{v})$ only require $\mathcal{O}(n)$ work.

For any $\boldsymbol{p} \in \mathbb{R}^n$ and $1 \le k \le n$, we define a generalized Householder reflector as

$$\boldsymbol{H}_k(\boldsymbol{p}) \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{I}_{k-1} & \\ & \boldsymbol{H}(\boldsymbol{p}[k:n]) \end{bmatrix}, \tag{9}$$

where $\boldsymbol{H}(\cdot)$ is the reflector defined in (7), and $\boldsymbol{p}[k:n]$ denotes a subvector obtained by removing the first $k-1$ elements of

$\boldsymbol{p}$. The construction in (7) requires that the reflecting vector $\boldsymbol{p}[k:n]$ be nonzero. In order for (9) to be always well-defined, we set $\boldsymbol{H}_k(\boldsymbol{p}) = \boldsymbol{I}_n$ if $\boldsymbol{p}[k:n] = \boldsymbol{0}$. Recall the notation $\boldsymbol{Z}_{1:k}$ introduced at the beginning of the section. It is easy to verify that

$$\boldsymbol{Z}_{1:k}\boldsymbol{H}_k(\boldsymbol{p})\boldsymbol{p} = \boldsymbol{0}, \tag{10}$$

which means that the orthogonal transformation $\boldsymbol{H}_k(\boldsymbol{p})$ can turn the last $n-k$ entries of $\boldsymbol{p}$ to zero. We will use this property in the construction of the HD algorithm.

B. Gaussian Random Matrices

We start by considering the case where the random matrix $\boldsymbol{Q}$ in the dynamics (1) has i.i.d. Gaussian entries, *i.e.*, $\boldsymbol{Q} \sim \text{Ginibre}(m,n)$. In addition, we shall always assume that $\boldsymbol{Q}$ is independent of the initial condition $\{\boldsymbol{x}_1, \boldsymbol{x}_0, \ldots, \boldsymbol{x}_{1-d}\}$.

Suppose that the first step of (1) is in the form of $\boldsymbol{x}_2 = f_1(\boldsymbol{Q}\boldsymbol{x}_1, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{1-d})$, i.e., $\boldsymbol{M}_1 = \boldsymbol{Q}$. How do we simulate this step without generating the entire Gaussian matrix $\boldsymbol{Q}$? This can be achieved by a simple observation:

$$\boldsymbol{Q} \overset{d}{=} \boldsymbol{g}_1\boldsymbol{e}_1^\mathsf{T} + \boldsymbol{G}_1\boldsymbol{Z}_1 \overset{d}{=} (\boldsymbol{g}_1\boldsymbol{e}_1^\mathsf{T} + \boldsymbol{G}_1\boldsymbol{Z}_1)\boldsymbol{R}_1 \sim \text{Ginibre}(m,n), \tag{11}$$

where $\boldsymbol{Z}_1 = \boldsymbol{I} - \boldsymbol{e}_1\boldsymbol{e}_1^\mathsf{T}$, $\boldsymbol{R}_1 \overset{\text{def}}{=} \boldsymbol{H}_1(\boldsymbol{x}_1)$ is a (generalized) Householder reflector defined in (9), $\boldsymbol{g}_1 \sim \text{Ginibre}(m,1)$ is a Gaussian vector, and $\boldsymbol{G}_1 \sim \text{Ginibre}(m,n)$ is an independent Gaussian matrix. Here and subsequently, whenever we generate new random vectors and matrices, they are always independent of each other and of the σ-algebra generated by all the other random variables constructed up to that point. For example, $\boldsymbol{g}_1$ and $\boldsymbol{G}_1$ in (11) are understood to be independent of the initial condition $\{\boldsymbol{x}_1, \boldsymbol{x}_0, \ldots, \boldsymbol{x}_{1-d}\}$. In (11), the first equality (in distribution) is obvious, and the second equality is due to the rotational invariance of the Ginibre ensemble. (Recall (5) and the fact that $\boldsymbol{R}_1$ is an orthogonal matrix.)

The new representation

$$\boldsymbol{Q}^{(1)} = (\boldsymbol{g}_1\boldsymbol{e}_1^\mathsf{T} + \boldsymbol{G}_1\boldsymbol{Z}_1)\boldsymbol{R}_1 \tag{12}$$

looks like a rather convoluted way of writing an i.i.d. Gaussian matrix, but it turns out to be the right choice for efficient simulations. To see this, we use the property of the Householder reflector [see (8)] which gives us $\boldsymbol{R}_1\boldsymbol{x}_1 = \boldsymbol{H}_1(\boldsymbol{x}_1)\boldsymbol{x}_1 = \|\boldsymbol{x}_1\|\,\boldsymbol{e}_1$ and thus $\boldsymbol{Z}_1\boldsymbol{R}_1\boldsymbol{x}_1 = \boldsymbol{0}$. It follows that

$$\boldsymbol{Q}^{(1)}\boldsymbol{x}_1 = \|\boldsymbol{x}_1\|\,\boldsymbol{g}_1.$$

Thus, to simulate the first step of the dynamics, we only need to generate a Gaussian vector $\boldsymbol{g}_1$. The more expensive Gaussian matrix $\boldsymbol{G}_1$ does not need to be revealed (yet), as it is invisible to $\boldsymbol{x}_1$.

It is helpful to consider two more iterations to see how this idea can be applied recursively. Suppose that the second iteration takes the form of $\boldsymbol{x}_3 = f_2(\boldsymbol{Q}\boldsymbol{x}_2, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{2-d})$. In general, $\boldsymbol{x}_2$ will have a nonzero component in the space orthogonal to $\boldsymbol{x}_1$, and thus the Gaussian matrix $\boldsymbol{G}_1$ in (12) is no longer invisible to $\boldsymbol{x}_2$, meaning that $\boldsymbol{G}_1\boldsymbol{Z}_1\boldsymbol{R}_1\boldsymbol{x}_2 \neq \boldsymbol{0}$. However, we can use the trick in (11) again by writing

$$\boldsymbol{G}_1 \overset{d}{=} (\boldsymbol{g}_2\boldsymbol{e}_2^\mathsf{T} + \boldsymbol{G}_2\boldsymbol{Z}_2)\boldsymbol{R}_2 \sim \text{Ginibre}(m,n), \tag{13}$$

where $\boldsymbol{g}_2 \sim \text{Ginibre}(m,1)$, $\boldsymbol{G}_2 \sim \text{Ginibre}(m,n)$, and $\boldsymbol{R}_2 \overset{\text{def}}{=} \boldsymbol{H}_2(\boldsymbol{R}_1\boldsymbol{x}_2)$ is again a generalized Householder reflector in (9). The subscript in $\boldsymbol{H}_2$ should not be overlooked, as it signifies the precise way the matrix is constructed. (Recall (9) for the notation convention we use.)

Observe that $\boldsymbol{R}_2$ commutes with $\boldsymbol{Z}_1$. Substituting (13) into (12) then allows us to write

$$\boldsymbol{Q}^{(2)} = \boldsymbol{u}_1\boldsymbol{v}_1^\mathsf{T} + \boldsymbol{u}_2\boldsymbol{v}_2^\mathsf{T} + \boldsymbol{G}_2\boldsymbol{Z}_{1:2}\boldsymbol{R}_2\boldsymbol{R}_1 \sim \text{Ginibre}(m,n), \tag{14}$$

where $\boldsymbol{u}_1 \overset{\text{def}}{=} \boldsymbol{g}_1$, $\boldsymbol{u}_2 \overset{\text{def}}{=} \boldsymbol{g}_2$, $\boldsymbol{v}_1 \overset{\text{def}}{=} \boldsymbol{R}_1\boldsymbol{e}_1$, and $\boldsymbol{v}_2 \overset{\text{def}}{=} \boldsymbol{R}_1\boldsymbol{R}_2\boldsymbol{e}_2$. Just like what happens in (12), there is again no need to explicitly generate the dense Gaussian matrix $\boldsymbol{G}_2$ in (14). To see this, we note that $\boldsymbol{Z}_{1:2}\boldsymbol{R}_2\boldsymbol{R}_1\boldsymbol{x}_2 = \boldsymbol{Z}_{1:2}\boldsymbol{H}_2(\boldsymbol{R}_1\boldsymbol{x}_2)\boldsymbol{R}_1\boldsymbol{x}_2 = \boldsymbol{0}$, where the second equality is due to (10). It follows that

$$\boldsymbol{Q}^{(2)}\boldsymbol{x}_2 = (\boldsymbol{v}_1^\mathsf{T}\boldsymbol{x}_2)\boldsymbol{u}_1 + (\boldsymbol{v}_2^\mathsf{T}\boldsymbol{x}_2)\boldsymbol{u}_2.$$

So far we have only been considering the case where we access $\boldsymbol{Q}$ from the right. For the third iteration, let us suppose that we access $\boldsymbol{Q}$ from the left, i.e., $\boldsymbol{x}_4 = f_3(\boldsymbol{Q}^\mathsf{T}\boldsymbol{x}_3, \boldsymbol{x}_3, \ldots, \boldsymbol{x}_{3-d})$. The idea is similar. Let

$$\boldsymbol{G}_2 = \boldsymbol{L}_1(\boldsymbol{e}_1\boldsymbol{g}_3^\mathsf{T} + \boldsymbol{Z}_1\boldsymbol{G}_3) \sim \text{Ginibre}(m,n), \tag{15}$$

where $\boldsymbol{L}_1 \overset{\text{def}}{=} \boldsymbol{H}_1(\boldsymbol{x}_3)$, $\boldsymbol{g}_3 \sim \text{Ginibre}(n,1)$, and $\boldsymbol{G}_3 \sim \text{Ginibre}(m,n)$. Substituting (15) into (14) gives us

$$\boldsymbol{Q}^{(3)} = \textstyle\sum_{1\le i\le 3} \boldsymbol{u}_i\boldsymbol{v}_i^\mathsf{T} + \boldsymbol{L}_1\boldsymbol{Z}_1\boldsymbol{G}_3\boldsymbol{Z}_{1:2}\boldsymbol{R}_2\boldsymbol{R}_1 \sim \text{Ginibre}(m,n),$$

where $\boldsymbol{u}_3 \overset{\text{def}}{=} \boldsymbol{L}_1\boldsymbol{e}_1$ and $\boldsymbol{v}_3 \overset{\text{def}}{=} \boldsymbol{R}_1\boldsymbol{R}_2\boldsymbol{Z}_{1:2}\boldsymbol{g}_3$. Moreover, $[\boldsymbol{Q}^{(3)}]^\mathsf{T}\boldsymbol{x}_3 = \sum_{i\le 3}(\boldsymbol{u}_i^\mathsf{T}\boldsymbol{x}_3)\boldsymbol{v}_i$.

The general idea should now be clear. Rather than fixing the entire Gaussian matrix in advance, we let the random choices gradually unfold as the iteration goes on, generating only the randomness that must be revealed at each step. Continuing this process for T steps, we reach the HD algorithm for the Ginibre ensemble, summarized in Algorithm 1. Its memory and computational costs can be determined as follows.

During its operation, Algorithm 1 keeps track of $2T$ vectors $\{\boldsymbol{u}_t \in \mathbb{R}^m, \boldsymbol{v}_t \in \mathbb{R}^n\}_{t\le T}$ and T Householder reflectors

$$\left\{\boldsymbol{L}_i \in \mathbb{R}^{m\times m}\right\}_{i\le \ell_T} \text{ and } \left\{\boldsymbol{R}_i \in \mathbb{R}^{n\times n}\right\}_{i\le r_T},$$

where ℓ_T (resp. r_T) records the number of times we have used $\boldsymbol{Q}^\mathsf{T}$ (resp. $\boldsymbol{Q}$) in the T iterations of the dynamics. Clearly, $r_T + \ell_T = T$. Thanks to the structures of the Householder reflectors in (7), the total memory footprint of Algorithm 1 is $\mathcal{O}((m+n)T)$. At each iteration, computations mainly take place in lines 6–9 (or lines 13–16 if $\boldsymbol{M}_t = \boldsymbol{Q}^\mathsf{T}$). Since the matrices used there are always products of Householder reflectors, these steps require $\mathcal{O}((m+n)t)$ operations. As t ranges from 1 to T, the computational complexity of Algorithm 1 is thus $\mathcal{O}((m+n)T^2)$.

Remark 1. *In line 6 and line 13, Algorithm 1 recursively constructs two products of (generalized) Householder reflectors. Readers familiar with numerical linear algebra will recognize that this process is essentially the Householder algorithm for QR factorization [13, Lecture 10]. Special data structures have been developed (see, e.g., [26]) to efficiently represent and operate on such products of reflectors.*

Algorithm 1 Simulating (1) on Ginibre(m,n) using Householder Dice

Require: The initial condition $\{\boldsymbol{x}_1, \boldsymbol{x}_0, \ldots, \boldsymbol{x}_{1-d}\}$, and the number of iterations $T \le \min\{m,n\}$

Set $r=0$, $\ell=0$, $\boldsymbol{L}_0 = \boldsymbol{I}_m$, and $\boldsymbol{R}_0 = \boldsymbol{I}_n$.
for $t = 1, \ldots, T$ **do**
  if $\boldsymbol{M}_t = \boldsymbol{Q}$ **then**
    $r \leftarrow r+1$
    Generate $\boldsymbol{g}_t \sim$ Ginibre$(m,1)$
    $\boldsymbol{R}_r = \boldsymbol{H}_r(\boldsymbol{R}_{r-1} \ldots \boldsymbol{R}_1 \boldsymbol{R}_0 \boldsymbol{x}_t)$
    $\boldsymbol{u}_t = \boldsymbol{L}_0 \boldsymbol{L}_1 \ldots \boldsymbol{L}_\ell \boldsymbol{Z}_{1:\ell} \boldsymbol{g}_t$
    $\boldsymbol{v}_t = \boldsymbol{R}_0 \boldsymbol{R}_1 \ldots \boldsymbol{R}_r \boldsymbol{e}_r$
    $\boldsymbol{y}_t = \sum_{i \le t} (\boldsymbol{v}_i^\mathsf{T} \boldsymbol{x}_t) \boldsymbol{u}_i$
  else
    $\ell \leftarrow \ell + 1$
    Generate $\boldsymbol{g}_t \sim$ Ginibre$(n,1)$
    $\boldsymbol{L}_\ell = \boldsymbol{H}_\ell(\boldsymbol{L}_{\ell-1} \ldots \boldsymbol{L}_1 \boldsymbol{L}_0 \boldsymbol{x}_t)$
    $\boldsymbol{u}_t = \boldsymbol{L}_0 \boldsymbol{L}_1 \ldots \boldsymbol{L}_\ell \boldsymbol{e}_\ell$
    $\boldsymbol{v}_t = \boldsymbol{R}_0 \boldsymbol{R}_1 \ldots \boldsymbol{R}_r \boldsymbol{Z}_{1:r} \boldsymbol{g}_t$
    $\boldsymbol{y}_t = \sum_{i \le t} (\boldsymbol{u}_i^\mathsf{T} \boldsymbol{x}_t) \boldsymbol{v}_i$
  end if
  $\boldsymbol{x}_{t+1} = f_t(\boldsymbol{y}_t, \boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_{t-d})$
end for

We can now exhibit the statistical equivalence of the HD algorithm and the direct simulation approach.

Theorem 1. *Fix $T \le \min\{m,n\}$, and let $\{\boldsymbol{x}_t : 1-d \le t \le T+1\}$ be a sequence of vectors generated by Algorithm 1. Let $\{\widetilde{\boldsymbol{x}}_t : 1-d \le t \le T+1\}$ be another sequence obtained by the direct approach to simulating (1), where we use the same initial condition (i.e. $\widetilde{\boldsymbol{x}}_t = \boldsymbol{x}_t$ for $1-d \le t \le 1$) but generate a full matrix $\boldsymbol{Q} \sim$ Ginibre(m,n) in advance. The joint probability distribution of $\{\boldsymbol{x}_t\}$ is equivalent to that of $\{\widetilde{\boldsymbol{x}}_t\}$.*

Proof. We start by describing the general structure of the algorithm. At the t-th iteration, the algorithm keeps the following representation of the matrix $\boldsymbol{Q}$:

$$\boldsymbol{Q}^{(t)} = \sum_{i \le t} \boldsymbol{u}_i \boldsymbol{v}_i^\mathsf{T} + \underbrace{\boldsymbol{L}_0 \boldsymbol{L}_1 \ldots \boldsymbol{L}_{\ell_t}}_{\text{Householder}} \boldsymbol{Z}_{1:\ell_t} \boldsymbol{G}_t \boldsymbol{Z}_{1:r_t} \underbrace{\boldsymbol{R}_{r_t} \ldots \boldsymbol{R}_1 \boldsymbol{R}_0}_{\text{Householder}}, \quad (16)$$

where $\boldsymbol{G}_t \sim$ Ginibre(m,n) is a Gaussian matrix independent of the σ-algebra generated by all the other random variables constructed up to this point, and ℓ_t (resp. r_t) denotes the number of times we have used $\boldsymbol{Q}^\mathsf{T}$ (resp. $\boldsymbol{Q}$) in the first t iterations of the dynamics. To lighten the notation, we will omit the subscript in the remainder of the proof and simply write them as ℓ and r.

The vectors $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}$ and the Householder reflectors $\{\boldsymbol{L}_i\}$, $\{\boldsymbol{R}_i\}$ in (16) are constructed recursively, as follows. We start with $\boldsymbol{L}_0 = \boldsymbol{I}_m$ and $\boldsymbol{R}_0 = \boldsymbol{I}_n$. At the t-th iteration (for $1 \le t \le T$), if $\boldsymbol{M}_t = \boldsymbol{Q}$ (i.e. if we need to compute $\boldsymbol{Q}\boldsymbol{x}_t$), we add a new Householder reflector

$$\boldsymbol{R}_r = \boldsymbol{H}_r(\boldsymbol{R}_{r-1} \ldots \boldsymbol{R}_1 \boldsymbol{R}_0 \boldsymbol{x}_t)$$

and two new "basis" vectors

$$\boldsymbol{u}_t = \boldsymbol{L}_0 \boldsymbol{L}_1 \ldots \boldsymbol{L}_\ell \boldsymbol{Z}_{1:\ell} \boldsymbol{g}_t \quad \text{and} \quad \boldsymbol{v}_t = \boldsymbol{R}_0 \boldsymbol{R}_1 \ldots \boldsymbol{R}_r \boldsymbol{e}_r,$$

where $\boldsymbol{g}_t \sim$ Ginibre$(m,1)$. The procedure for the case of $\boldsymbol{M}_t = \boldsymbol{Q}^\mathsf{T}$ is completely analogous: we add a new Householder reflector $\boldsymbol{L}_\ell$ (on the left) and construct the basis vectors $\boldsymbol{u}_t, \boldsymbol{v}_t$ accordingly.

It is important to note that the Gaussian matrix $\boldsymbol{G}_t$ in (16) is never explicitly constructed in the algorithm. Assume without loss of generality that $\boldsymbol{M}_t = \boldsymbol{Q}$. Let $\boldsymbol{p} = \boldsymbol{R}_{r-1} \ldots \boldsymbol{R}_1 \boldsymbol{R}_0 \boldsymbol{x}_t$. We then have

$$\boldsymbol{Z}_{1:r} \boldsymbol{R}_r \ldots \boldsymbol{R}_1 \boldsymbol{R}_0 \boldsymbol{x}_t = \boldsymbol{Z}_{1:r} \boldsymbol{H}_r(\boldsymbol{p}) \boldsymbol{p} = \boldsymbol{0},$$

where the second equality is due to (10). Consequently, $\boldsymbol{G}_t$ remains invisible to $\boldsymbol{x}_t$, and

$$\boldsymbol{Q}^{(t)} \boldsymbol{x}_t = \sum_{i \le t} (\boldsymbol{v}_i^\mathsf{T} \boldsymbol{x}_t) \boldsymbol{u}_i.$$

To prove the assertion of the theorem, it suffices to show that, for all $1 \le t \le T$, $\boldsymbol{Q}^{(t)}$ has the correct distribution, namely $\boldsymbol{Q}^{(t)} \sim$ Ginibre(m,n) and $\boldsymbol{Q}^{(t)}$ is independent of the initial condition $\{\boldsymbol{x}_1, \boldsymbol{x}_0, \ldots, \boldsymbol{x}_{1-d}\}$. This is clearly true for $t=1$, based on our discussions around (12). Now suppose that the condition on the distribution has been verified for $\boldsymbol{Q}^{(t)}$ for some $t \ge 1$, . To go to $t+1$, we rewrite the Gaussian matrix $\boldsymbol{G}_t$ in (16) by using a decomposition similar to (13). Specifically, if $\boldsymbol{M}_t = \boldsymbol{Q}$, we write

$$\boldsymbol{G}_t \overset{d}{=} (\boldsymbol{g}_{t+1} \boldsymbol{e}_{r+1}^\mathsf{T} + \boldsymbol{G}_{t+1} \boldsymbol{Z}_{r+1}) \boldsymbol{R}_{r+1} \sim \text{Ginibre}(m,n), \quad (17)$$

where $\boldsymbol{g}_{t+1} \sim$ Ginibre$(m,1)$, $\boldsymbol{G}_{t+1} \sim$ Ginibre(m,n), and $\boldsymbol{R}_{r+1} \overset{\text{def}}{=} \boldsymbol{H}_{r+1}(\boldsymbol{R}_r \ldots, \boldsymbol{R}_1 \boldsymbol{R}_0 \boldsymbol{x}_{t+1})$. (The decomposition for the case where $\boldsymbol{M}_t = \boldsymbol{Q}^\mathsf{T}$ is completely analogous.)

That the new representation on the right-hand side of (17) has the same distribution as $\boldsymbol{G}_t$ is due to the rotational invariance of the Ginibre ensemble [see (5)]. Substituting (17) into (16) allows us to conclude that the matrix

$$\sum_{i \le t} \boldsymbol{u}_i \boldsymbol{v}_i^\mathsf{T} + \boldsymbol{L}_0 \ldots \boldsymbol{L}_\ell \boldsymbol{Z}_{1:\ell} (\boldsymbol{g}_{t+1} \boldsymbol{e}_{r+1}^\mathsf{T} + \boldsymbol{G}_{t+1} \boldsymbol{Z}_{r+1}) \boldsymbol{R}_{r+1} \boldsymbol{Z}_{1:r} \boldsymbol{R}_r \ldots \boldsymbol{R}_0 \quad (18)$$

satisfies the required condition on its distribution. By construction, $\boldsymbol{R}_{r+1}$ commutes with $\boldsymbol{Z}_{1:r}$. [Recall (9).] This simple observation allows us to check that the matrix in (18) is exactly $\boldsymbol{Q}^{(t+1)}$. By induction on t from 1 to T, we then complete the proof. □

C. Haar-Distributed Random Orthogonal Matrices

We now turn to the case where $\boldsymbol{Q}$ is a Haar-distributed random orthogonal matrix. The construction of the HD algorithm relies on the following factorization of the Haar measure on $\mathbb{O}(n)$.

Lemma 1. *Let $\boldsymbol{g} \sim$ Ginibre$(n,1)$, $\boldsymbol{Q}_{n-1} \sim$ Haar$(n-1)$, and $\boldsymbol{v} \in \mathbb{R}^n$, all of which are independent. Then*

$$\boldsymbol{H}_1(\boldsymbol{g}) \begin{bmatrix} 1 & \\ & \boldsymbol{Q}_{n-1} \end{bmatrix} \boldsymbol{H}_1(\boldsymbol{v}) \sim \text{Haar}(n). \quad (19)$$

Proof. Let $\boldsymbol{M}$ denote the left-hand side of (19). It is sufficient to show that $\boldsymbol{M} \overset{d}{=} \boldsymbol{U}\boldsymbol{M}$ for any fixed $\boldsymbol{U} \in \mathbb{O}(n)$. The

statement of the lemma then follows from the fact that the Haar measure is the unique (left) translation-invariant measure on $\mathbb{O}(n)$.

For any nonzero vector $\boldsymbol{x} \in \mathbb{R}^n$, we denote by $\boldsymbol{B}(\boldsymbol{x}) \in \mathbb{R}^{n\times(n-1)}$ the submatrix consisting of the last $n-1$ columns of $\boldsymbol{H}_1(\boldsymbol{x})$. It is also useful to notice that the first column of $\boldsymbol{H}_1(\boldsymbol{x})$ is $\boldsymbol{x}/\|\boldsymbol{x}\|$. Thus, $\boldsymbol{H}_1(\boldsymbol{x}) = \left[\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} \mid \boldsymbol{B}(\boldsymbol{x})\right]$. The following observation is easy to verify. For any fixed $\boldsymbol{U} \in \mathbb{O}(n)$, there exists some $\boldsymbol{R} \in \mathbb{O}(n-1)$ such that

$$\boldsymbol{U}\boldsymbol{B}(\boldsymbol{x}) = \boldsymbol{B}(\boldsymbol{U}\boldsymbol{x})\boldsymbol{R}.$$

Applying this to $\boldsymbol{B}(\boldsymbol{g})$ [in $\boldsymbol{H}_1(\boldsymbol{g})$] then allows us to write

$$\boldsymbol{U}\boldsymbol{M} = \boldsymbol{H}_1(\boldsymbol{U}\boldsymbol{g}) \begin{bmatrix} 1 & \\ & \boldsymbol{R}\boldsymbol{Q}_{n-1} \end{bmatrix} \boldsymbol{H}_1(\boldsymbol{v}),$$

where $\boldsymbol{R}$ is an orthogonal matrix independent of $\boldsymbol{Q}_{n-1}$ and $\boldsymbol{v}$. Since the joint distribution of $(\boldsymbol{U}\boldsymbol{g}, \boldsymbol{R}\boldsymbol{Q}_{n-1}, \boldsymbol{v})$ is equal to that of $(\boldsymbol{g}, \boldsymbol{Q}_{n-1}, \boldsymbol{v})$ in (19), we must have $\boldsymbol{M} \overset{d}{=} \boldsymbol{U}\boldsymbol{M}$. □

The HD algorithm exploits the factorization in (19) to speed up the simulation of Haar random matrices. Before presenting the algorithm in its full generality, we first illustrate how it unfolds in the first two iterations of (1). For simplicity, we assume that $\boldsymbol{M}_1 = \boldsymbol{M}_2 = \boldsymbol{Q}$. For the first iteration, we use (19) to write $\boldsymbol{Q}$ as

$$\boldsymbol{Q}^{(1)} = \boldsymbol{L}_1 \begin{bmatrix} 1 & \\ & \boldsymbol{Q}_{n-1} \end{bmatrix} \boldsymbol{R}_1 \sim \text{Haar}(n), \tag{20}$$

where $\boldsymbol{R}_1 = \boldsymbol{H}_1(\boldsymbol{x}_1)$, $\boldsymbol{L}_1 = \boldsymbol{H}_1(\boldsymbol{g}_1)$, $\boldsymbol{g}_1 \sim \text{Ginibre}(n,1)$ and $\boldsymbol{Q}_{n-1} \sim \text{Haar}(n-1)$. Using the property of Householder reflectors given in (8), we have

$$\boldsymbol{Q}^{(1)}\boldsymbol{x}_1 = \|\boldsymbol{x}_1\|\, \boldsymbol{H}_1(\boldsymbol{g}_1)\boldsymbol{e}_1 = \frac{\|\boldsymbol{x}_1\|}{\|\boldsymbol{g}_1\|}\, \boldsymbol{g}_1.$$

Notice that only a Gaussian vector $\boldsymbol{g}_1$ is needed here, and that the matrix $\boldsymbol{Q}_{n-1}$ is invisible.

To simulate the second iteration, we apply the factorization (19) recursively to write $\boldsymbol{Q}_{n-1}$ as

$$\begin{aligned} \boldsymbol{Q}_{n-1} &= \boldsymbol{H}_1(\boldsymbol{g}_2[2:n]) \begin{bmatrix} 1 & \\ & \boldsymbol{Q}_{n-2} \end{bmatrix} \boldsymbol{H}_1(\boldsymbol{p}[2:n]) \\ &\sim \text{Haar}(n-1), \end{aligned} \tag{21}$$

where $\boldsymbol{g}_2 \sim \text{Ginibre}(n,1)$, $\boldsymbol{Q}_{n-2} \sim \text{Haar}(n-2)$, and $\boldsymbol{p} = \boldsymbol{R}_1\boldsymbol{x}_2$. Substituting (21) into (20) then gives us

$$\boldsymbol{Q}^{(2)} = \boldsymbol{L}_1\boldsymbol{L}_2 \begin{bmatrix} \boldsymbol{I}_2 & \\ & \boldsymbol{Q}_{n-2} \end{bmatrix} \boldsymbol{R}_2\boldsymbol{R}_1, \tag{22}$$

where $\boldsymbol{L}_2 = \boldsymbol{H}_2(\boldsymbol{g}_2)$ and $\boldsymbol{R}_2 = \boldsymbol{H}_2(\boldsymbol{p})$. By construction, the vector $\boldsymbol{R}_2\boldsymbol{R}_1\boldsymbol{x}_2$ has nonzero entries only in the first two coordinates. It follows that

$$\boldsymbol{Q}^{(2)}\boldsymbol{x}_2 = \boldsymbol{L}_1\boldsymbol{L}_2\boldsymbol{R}_2\boldsymbol{R}_1\boldsymbol{x}_2,$$

with $\boldsymbol{Q}_{n-2}$ in (22) remaining invisible.

Continuing this process, we see a simple pattern emerging. We summarize it in Algorithm 2. In general, the algorithm recursively constructs two sequences of Householder reflectors

Algorithm 2 Simulating (1) on Haar(n) using Householder Dice

Require: The initial condition $\{\boldsymbol{x}_1, \boldsymbol{x}_0, \dots, \boldsymbol{x}_{1-d}\}$, and the number of iterations $T \le n$

1: Set $\boldsymbol{L}_0 = \boldsymbol{I}_n$, and $\boldsymbol{R}_0 = \boldsymbol{I}_n$.
2: **for** $t = 1, \dots, T$ **do**
3: Generate $\boldsymbol{g}_t \sim \text{Ginibre}(n,1)$
4: **if** $\boldsymbol{M}_t = \boldsymbol{Q}$ **then**
5: $\boldsymbol{p}_t = \boldsymbol{R}_{t-1} \dots \boldsymbol{R}_1\boldsymbol{R}_0\boldsymbol{x}_t$
6: $\boldsymbol{R}_t = \boldsymbol{H}_t(\boldsymbol{p}_t)$
7: $\boldsymbol{L}_t = \boldsymbol{H}_t(\boldsymbol{g}_t)$
8: $\boldsymbol{y}_t = \boldsymbol{L}_1 \dots \boldsymbol{L}_t\boldsymbol{R}_t\boldsymbol{p}_t$
9: **else**
10: $\boldsymbol{p}_t = \boldsymbol{L}_{t-1} \dots \boldsymbol{L}_1\boldsymbol{L}_0\boldsymbol{x}_t$
11: $\boldsymbol{L}_t = \boldsymbol{H}_t(\boldsymbol{p}_t)$
12: $\boldsymbol{R}_t = \boldsymbol{H}_t(\boldsymbol{g}_t)$
13: $\boldsymbol{y}_t = \boldsymbol{R}_1 \dots \boldsymbol{R}_t\boldsymbol{L}_t\boldsymbol{p}_t$
14: **end if**
15: $\boldsymbol{x}_{t+1} = f_t(\boldsymbol{y}_t, \boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \dots, \boldsymbol{x}_{t-d})$
16: **end for**

$\{\boldsymbol{L}_t, \boldsymbol{R}_t\}_{t\le T}$, starting from $\boldsymbol{L}_0 = \boldsymbol{R}_0 = \boldsymbol{I}_n$. At the t-th iteration, we first generate a new Gaussian vector $\boldsymbol{g} \sim \text{Ginibre}(n,1)$. Suppose $\boldsymbol{M}_t = \boldsymbol{Q}$, we compute

$$\boldsymbol{p}_t = \boldsymbol{R}_{t-1} \dots \boldsymbol{R}_1\boldsymbol{R}_0\boldsymbol{x}_t \tag{23}$$

and add two reflectors $\boldsymbol{R}_t = \boldsymbol{H}_t(\boldsymbol{p}_t)$ and $\boldsymbol{L}_t = \boldsymbol{H}_t(\boldsymbol{g}_t)$. The algorithm then proceeds to the next iteration by letting $\boldsymbol{x}_{t+1} = f_t(\boldsymbol{y}_t, \boldsymbol{x}_t, \dots, \boldsymbol{x}_{t-d})$, where $\boldsymbol{y}_t = \boldsymbol{L}_1 \dots \boldsymbol{L}_t\boldsymbol{R}_t\boldsymbol{p}_t$. The steps the algorithm takes if $\boldsymbol{M} = \boldsymbol{Q}^\mathsf{T}$ are exactly symmetric, with the roles of $\{\boldsymbol{R}_i\}$ and $\{\boldsymbol{L}_i\}$ switched. The computational and memory complexity of Algorithm 2 is similar to that of Algorithm 1. Specifically, the Householder reflectors can be efficiently represented by the corresponding reflection vectors, at a cost of $\mathcal{O}(nT)$ space. At each iteration, the matrix-vector multiplications in lines 5, 8, 10 and 13 can all be implemented in $\mathcal{O}(nt)$ operations (thanks to the Householder structure). Therefore, the total computational complexity is $\mathcal{O}(nT^2)$.

Finally, we establish the statistical "correctness" of Algorithm 2 in the following theorem.

Theorem 2. *Fix $T \le n$, and let $\{\boldsymbol{x}_t : 1-d \le t \le T+1\}$ be a sequence of vectors generated by Algorithm 2. Let $\{\widetilde{\boldsymbol{x}}_t : 1-d \le t \le T+1\}$ be another sequence obtained by the direct approach to simulating (1), where we use the same initial condition (i.e. $\widetilde{\boldsymbol{x}}_t = \boldsymbol{x}_t$ for $1-d \le t \le 1$) but generate a random orthogonal matrix $\boldsymbol{Q} \sim$ Haar(n) in advance. The joint probability distribution of $\{\boldsymbol{x}_t\}$ is equivalent to that of $\{\widetilde{\boldsymbol{x}}_t\}$.*

Proof. The proof is very similar to that of Theorem 1. At the t-th iteration, the algorithm has constructed a representation of the random orthogonal matrix $\boldsymbol{Q}$ as

$$\boldsymbol{Q}^{(t)} = \boldsymbol{L}_1\boldsymbol{L}_2 \dots \boldsymbol{L}_t \begin{bmatrix} \boldsymbol{I}_t & \\ & \boldsymbol{Q}_{n-t} \end{bmatrix} \boldsymbol{R}_t \dots \boldsymbol{R}_2\boldsymbol{R}_1, \tag{24}$$

where $\{\boldsymbol{L}_i, \boldsymbol{R}_i\}_{i\le t}$ is a collection of Householder reflectors, and $\boldsymbol{Q}_{n-t} \sim \text{Haar}(n-t)$ is an $(n-t)\times(n-t)$ random

orthogonal matrix independent of the σ-algebra generated by all the other random variables constructed up to this point. We shall have established the theorem if we prove the following two claims for $1 \le t \le T$: (a) $\boldsymbol{Q}^{(t)} \sim \text{Haar}(n)$ and $\boldsymbol{Q}^{(t)}$ is independent of the initial condition $\{\boldsymbol{x}_t\}_{1-d\le t\le 1}$; (b) If $\boldsymbol{M}_t = \boldsymbol{Q}$ in (1), then

$$\boldsymbol{Q}^{(t)}\boldsymbol{x}_t = \boldsymbol{L}_1 \dots \boldsymbol{L}_t \boldsymbol{R}_t \boldsymbol{p}_t, \tag{25}$$

where $\boldsymbol{p}_t$ is as defined in (23). If $\boldsymbol{M}_t = \boldsymbol{Q}^{\mathsf{T}}$, then $[\boldsymbol{Q}^{(t)}]^{\mathsf{T}}\boldsymbol{x}_t = \boldsymbol{R}_1\boldsymbol{R}_2\dots\boldsymbol{R}_t\boldsymbol{L}_t\dots\boldsymbol{L}_2\boldsymbol{L}_1\boldsymbol{x}_t$.

Claim (a) can be proved by induction. We have already established its correctness for $t = 1$. [See (20).] To propagate the claim from iteration t to $t+1$, we simply apply Lemma 1 to rewrite $\boldsymbol{Q}_{n-t}$ in (24) as

$$\boldsymbol{Q}_{n-t} \overset{d}{=} \boldsymbol{H}_1(\boldsymbol{g}_{t+1}[t+1:n]) \begin{bmatrix} 1 & \\ & \boldsymbol{Q}_{n-t-1} \end{bmatrix} \boldsymbol{H}_1(\boldsymbol{p}_{t+1}[t+1:n]) \sim \text{Haar}(n-t),$$

where $\boldsymbol{g}_{t+1} \sim \text{Ginibre}(n,1)$, $\boldsymbol{Q}_{n-t-1} \sim \text{Haar}(n-t-1)$, and $\boldsymbol{p}_{t+1} = \boldsymbol{R}_t \dots \boldsymbol{R}_2\boldsymbol{R}_1\boldsymbol{x}_{t+1}$. (This is for the case of $\boldsymbol{M}_{t+1} = \boldsymbol{Q}$, but the treatment for the case of $\boldsymbol{M}_{t+1} = \boldsymbol{Q}^{\mathsf{T}}$ is completely analogous.) Substituting this equivalent representation into (24) gives us $\boldsymbol{Q}^{(t+1)}$.

To establish Claim (b), we again assume without loss of generality that $\boldsymbol{M}_t = \boldsymbol{Q}$. By the definition of $\boldsymbol{p}_t$ in (23) and that of $\boldsymbol{R}_t$, we have

$$\boldsymbol{Q}^{(t)}\boldsymbol{x}_t = \boldsymbol{L}_1\boldsymbol{L}_2\dots\boldsymbol{L}_t \begin{bmatrix} \boldsymbol{I}_t & \\ & \boldsymbol{Q}_{n-t} \end{bmatrix} \boldsymbol{H}_t(\boldsymbol{p}_t)\boldsymbol{p}_t.$$

Using (10), we can then verify the expression given in (25). □

D. Other Random Matrix Ensembles

The Gaussian and Haar ensembles studied above can serve as building blocks for simulating other related random matrix ensembles. For example, consider the classical Gaussian orthogonal ensemble (GOE). A symmetric $n \times n$ matrix G is drawn from GOE(n) if $\{G_{ij}\}_{1\le i\le j\le n}$ are independent random variables, with $G_{ii} \sim \mathcal{N}(0,2)$ and $G_{ij} \sim \mathcal{N}(0,1)$ for $i < j$. Clearly,

$$\boldsymbol{Q} \sim \text{Ginibre}(n,n) \implies \frac{1}{\sqrt{2}}(\boldsymbol{Q} + \boldsymbol{Q}^{\mathsf{T}}) \sim \text{GOE}(n).$$

It follows that a single matrix-vector multiplication involving $\boldsymbol{G} \sim \text{GOE}(n)$ can be simulated via two matrix-vector multiplications involving a nonsymmetric Gaussian matrix, i.e.,

$$\boldsymbol{y} = \boldsymbol{G}\boldsymbol{x} \implies \widehat{\boldsymbol{y}} = \boldsymbol{Q}\boldsymbol{x} \text{ and } \boldsymbol{y} = (\boldsymbol{Q}^{\mathsf{T}}\boldsymbol{x} + \widehat{\boldsymbol{y}})/\sqrt{2}.$$

As a second example, we consider random matrices in the form of

$$\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}, \tag{26}$$

where $\boldsymbol{U} \sim \text{Haar}(m)$ and $\boldsymbol{V} \sim \text{Haar}(n)$ are two independent random orthogonal matrices, and $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ is a rectangular diagonal matrix independent of $\boldsymbol{U}, \boldsymbol{V}$. Matrices like these often appear in the study of free probability theory [27]. They are also used as a convenient model for matrices whose singular vectors are *generic* [6]–[8]. Strictly speaking, Theorem 2 only applies to the case where the dynamics operates on a single random orthogonal matrix. However, it is obvious from the proof that the idea applies to more general dynamics involving a finite number of independent random orthogonal matrices. Thus, Algorithm 2 can be easily adapted to handle the matrix ensemble given in (26).

Finally, we note that the constructions of the HD algorithm can be generalized to the complex-valued cases, with the random matrices drawn from the complex Ginibre ensemble, the Haar ensemble on the unitary group $\mathbb{U}(n)$, and the Gaussian unitary ensemble, respectively. We avoid repetitions, as most changes in such generalizations are straightforward (such as replacing $\boldsymbol{M}^{\mathsf{T}}$ by $\boldsymbol{M}^*$). In what follows, we only present the formula for a complex version of the Householder reflector, as it might be less well-known.

Let $\boldsymbol{v} \in \mathbb{C}^n$ be a nonzero vector. Write $v_1/\|\boldsymbol{v}\| = re^{i\theta}$, where r is a nonnegative real number. (When $v_1 = 0$, we have $r = 0$ and set $\theta = 0$.) We define a unitary reflector [28, pp. 48–49] as

$$\boldsymbol{H}(\boldsymbol{v}) = (-e^{-i\theta})\Big[\boldsymbol{I}_n - \frac{(\boldsymbol{v}/\|\boldsymbol{v}\| + e^{i\theta}\boldsymbol{e}_1)(\boldsymbol{v}/\|\boldsymbol{v}\| + e^{i\theta}\boldsymbol{e}_1)^*}{1+r}\Big]. \tag{27}$$

It is easy to check that $\boldsymbol{H}(\boldsymbol{v})$ is a unitary matrix such that $\boldsymbol{H}(\boldsymbol{v})\boldsymbol{v} = \|\boldsymbol{v}\|\,\boldsymbol{e}_1$ and $\boldsymbol{H}^*(\boldsymbol{v})\boldsymbol{e}_1 = \boldsymbol{v}/\|\boldsymbol{v}\|$. Moreover, if $\boldsymbol{v}$ is real, then (27) reduces to the Householder reflector given in (7).

IV. Conclusion

We proposed a new algorithm called Householder Dice for simulating dynamics on several dense random matrix ensembles with rotational invariance. Rather than fixing the entire random matrix in advance, the new algorithm is matrix-free, generating only the randomness that must be revealed at any given step of the dynamics. The name of the algorithm highlights the central role played by an adaptive and recursive construction of (random) Householder reflectors. These orthogonal transformations exploit the group symmetry of the matrix ensembles, while simultaneously maintaining the statistical correlations induced by the dynamics. Numerical results demonstrate the promise of the HD algorithm as a new computational tool in the study of high-dimensional random systems.

References

[1] G. W. Stewart, "A Krylov–Schur algorithm for large eigenproblems," *SIAM J. Matrix Anal. Appl.*, vol. 23, no. 3, pp. 601–614, 2002.

[2] J. Baik, G. B. Arous, and S. Péché, "Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices," *Ann. Probab.*, vol. 33, pp. 1643–1697, Sept. 2005.

[3] F. Benaych-Georges and R. R. Nadakuditi, "The eigenvalues and eigenvectors of finite, low rank perturbations of large random matrices," *Adv. Math.*, vol. 227, pp. 494–521, May 2011.

[4] E. Bolthausen, "An iterative construction of solutions of the TAP equations for the Sherrington–Kirkpatrick model," *Commun. Math. Phys.*, no. 325, pp. 333–366, 2014.

[5] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inf. Theory*, vol. 57, pp. 764 –785, Feb. 2011.

[6] M. Opper, B. Cakmak, and O. Winther, "A theory of solving TAP equations for Ising models with general invariant random matrices," *J. Phys. A*, vol. 49, no. 11, p. 114002, 2016.

[7] S. Rangan, P. Schniter, and A. K. Fletcher, "Vector approximate message passing," *IEEE Trans. Inf. Theory*, vol. 65, pp. 6664–6684, Oct 2019.
[8] Z. Fan, "Approximate message passing algorithms for rotaltionally invariant matrices," *arXiv:2008.11892*, 2020.
[9] E. J. Candes, X. Li, and M. Soltanolkotabi, "Phase retrieval via Wirtinger flow: Theory and algorithms," *IEEE Trans. Inf. Theory*, vol. 61, no. 4, pp. 1985–2007, 2015.
[10] S. Goldt, M. S. Advani, A. M. Saze, F. Krzakala, and L. Zdeborová, "Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup," in *Advances in Neural Information Processing Systems 32*, 2019.
[11] M. Mitzenmacher and E. Upfal, *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.
[12] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," *J. ACM*, vol. 5, no. 4, pp. 339–342, 1958.
[13] L. N. Trefethen and D. Bau III., *Numerical Linear Algebra*. Philadelphia, PA: SIAM, 1997.
[14] G. W. Stewart, "The efficient generation of random orthogonal matrices with an application to condition numbers," *SIAM J. Numer. Anal.*, vol. 17, no. 3, pp. 403–425, 1980.
[15] F. Mezzadri, "How to generate random matrices from the classical compact groups," *Notice of the AMS*, vol. 54, pp. 592–604, May 2007.
[16] J. W. Silverstein, "The smallest eigenvalue of a large dimensional Wishart matrix," *Ann. Probab.*, vol. 13, no. 4, pp. 1364–1368, 1985.
[17] A. Edelman, *Eigenvalues and Condition Numbers of Random Matrices*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1989.
[18] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Rev.*, vol. 59, no. 65–98, 2017.
[19] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
[20] K.-C. Li, "On principal hessian directions for data visualization and dimension reduction: Another application of Stein's lemma," *J. Am. Stat. Assoc*, vol. 87, no. 420, pp. 1025–1039, 1992.
[21] P. Netrapalli, P. Jain, and S. Sanghavi, "Phase retrieval using alternating minimization," in *Advances in Neural Information Processing Systems*, pp. 2796–2804, 2013.
[22] Y. M. Lu and G. Li, "Phase transitions of spectral initialization for high-dimensional nonconvex estimation," *Information and Inference*, vol. 9, pp. 507–541, September 2020.
[23] M. Mondelli and A. Montanari, "Fundamental limits of weak recovery with applications to phase retrieval," in *Proceedings of Machine Learning Research*, vol. 75, 2018.
[24] R. Dudeja, M. Bakhshizadeh, J. Ma, and A. Maleki, "Analysis of spectral methods for phase retrieval with random orthogonal matrices," *IEEE Trans. Inf. Theory*, vol. 66, pp. 5182–5203, Aug 2020.
[25] W. Rudin, *Functional Analysis*. New York: McGraw-Hill, 2nd ed., 1991.
[26] R. Schreiber and C. V. Loan, "A storage-efficient WY representation for products of Householder transformations," *SIAM J. Sci. Stat. Comput.*, vol. 10, January 1989.
[27] J. A. Mingo and R. Speicher, *Free Probability and Random Matrices*. New York, NY: Springer Science & Business Media, 2017.
[28] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. Oxford, UK: Clarendon Press, Apr. 1988.

Yue M. Lu is Gordon McKay Professor of Electrical Engineering and of Applied Mathematics at Harvard University. He received the M.Sc. degree in mathematics and the Ph.D. degree in electrical engineering, both in 2007, from the University of Illinois at Urbana-Champaign. He received the Most Innovative Paper Award of IEEE International Conference on Image Processing (ICIP) in 2006, the Best Student Paper Award of IEEE ICIP in 2007, and the Best Student Presentation Award at the 31st SIAM SEAS Conference in 2007. Student papers supervised and coauthored by him won the Best Student Paper Award of IEEE International Conference on Acoustics, Speech and Signal Processing in 2011, the Best Student Paper Award of IEEE Global Conference on Signal and Information Processing (GlobalSIP) in 2014, and the Best Student Paper (First Prize) of the IEEE CAMSAP Workshop in 2017. He is a recipient of the ECE Illinois Young Alumni Achievement Award.

His research interests include theoretical and algorithmic aspects of high-dimensional statistical signal and information processing. He is currently serving as a member of the IEEE Signal Processing Theory and Methods Technical Committee, a member of the Machine Learning for Signal Processing Technical Committee, and an Associate Editor of the IEEE Transactions on Signal Processing.