Linear Algebra and its Applications 620 (2021) 297–321

Contents lists available at ScienceDirect

Linear Algebra and its Applications

www.elsevier.com/locate/laa

Sampling the eigenvalues of random orthogonal and unitary matrices ☆

Massimiliano Fasi [a,*], Leonardo Robol [b]

[a] *School of Science and Technology, Örebro University, Sweden*
[b] *Dipartimento di Matematica, Università di Pisa, Italy*

ARTICLE INFO

Article history:
Received 23 September 2020
Accepted 24 February 2021
Available online 8 March 2021
Submitted by Z. Bao

MSC:
15B10
15B52
65F15

Keywords:
Random matrix
Unitary matrix
Orthogonal matrix
Eigenvalue sampling
Haar distribution

ABSTRACT

We develop an efficient algorithm for sampling the eigenvalues of random matrices distributed according to the Haar measure over the orthogonal or unitary group. Our technique samples directly a factorization of the Hessenberg form of such matrices, and then computes their eigenvalues with a tailored core-chasing algorithm. This approach requires a number of floating-point operations that is quadratic in the order of the matrix being sampled, and can be adapted to other matrix groups. In particular, we explain how it can be used to sample the Haar measure over the special orthogonal and unitary groups and the conditional probability distribution obtained by requiring the determinant of the sampled matrix be a given complex number on the complex unit circle.

☆ The work of the first author was supported by the Royal Society, the Wenner-Gren Foundations grant UPD2019-0067, and the Istituto Nazionale di Alta Matematica INdAM-GNCS Project 2019. The second author is a member of the research group GNCS, and his work has been partially supported by a GNCS/INdAM project "Giovani Ricercatori" 2018.

* Corresponding author.
E-mail addresses: massimiliano.fasi@oru.se (M. Fasi), leonardo.robol@unipi.it (L. Robol).

https://doi.org/10.1016/j.laa.2021.02.031

1. Introduction

Random matrix theory, introduced by Wishart [1] about 90 years ago, investigates the properties of matrices whose entries are random variables. The quantities of interest range from the joint probability distribution of the matrix elements to the asymptotic behavior of its eigenvalues and singular values [2], and applications stretch from nuclear physics [3,4], wireless networks [5], and neuroscience [6] to numerical analysis [7,8] and number theory [9]. Random matrix theory is still a very active area of research [10]. We refer the interested reader to the survey by Edelman and Rao [7] for a general introduction to the topic, and to the monographs by Forrester [11] and Mehta [2] for a more complete discussion. The general mechanisms by which random matrix theory can be employed to solve practical problems are discussed by Edelman and Wang [12].

In applications, one is often interested in random matrices with a given structure. In quantum mechanics, for example, the energy levels of a system are described by the eigenvalue of its Hamiltonian, a Hermitian operator on an infinite-dimensional Hilbert space. By approximating this space by a Hilbert space of finite dimension, one can reduce the problem of finding the energy levels to that of solving a Hermitian eigenvalue problem. The true Hamiltonian, however, is typically not known, thus it is customary to make statistical assumptions on the distribution of its entries, enforcing only the symmetry of the operator. The distribution of the eigenvalues of random symmetric and Hermitian matrices has been extensively studied [2,13,14] and an algorithm for sampling the eigenvalues of uniformly distributed Hermitian matrices has been developed by Edelman, Sutton, and Wang [15].

Here we are interested in the orthogonal group $O(n) = \{Q \in \mathbb{R}^{n\times n} \mid Q^TQ = I_n\}$ and in the unitary group $U(n) = \{U \in \mathbb{C}^{n\times n} \mid U^*U = I_n\}$, where I_n denotes the identity matrix of order n and A^T and A^* denote the transpose and the conjugate transpose of A, respectively. It is easy to prove that the determinant of an orthogonal or unitary matrix lies on the unit circle, and that the special orthogonal group $SO(n) = \{Q \in O(n) \mid \det Q = 1\}$ and the special unitary group $SU(n) = \{U \in U(n) \mid \det U = 1\}$ are subgroups of $O(n)$ and $U(n)$, respectively. $O(n)$ is made of two connected components, the already mentioned $SO(n)$ and one in which all matrices have determinant -1, which we denote by $O^-(n)$. Clearly, the latter is not a group.

Random unitary matrices find application in quantum physics where they are employed, for example, to model scattering matrices and Floquet operators [11, Section 2.1]. Random orthogonal matrices, on the other hand, are used in statistical mechanics to characterize the behavior of certain log-gas systems [11, Section 2.9].

For a group G, the measure μ such that $\mu(G) = 1$ is a normalized left or right Haar measure if for any $Q \in G$ and any measurable $\mathcal{G} \subset G$ it satisfies $\mu(Q\mathcal{G}) = \mu(\mathcal{G})$ or $\mu(\mathcal{G}Q) = \mu(\mathcal{G})$, respectively. For compact Lie groups, it can be shown that the left and right measures are unique and coincide. Hence, since $O(n)$, $SO(n)$, $U(n)$, and $SU(n)$ are all compact Lie groups [16, Chapter 1], they have a unique normalized (left and right) Haar measure [17, § 58, § 60].

We consider the problem of sampling efficiently the joint eigenvalue distribution of unitary (or orthogonal) matrices distributed according to the Haar measure. Numerically, this may be obtained by sampling matrices from the desired group uniformly, and then computing their eigenvalues by relying, for instance, on the QR iteration. The latter step requires $\mathcal{O}(n^3)$ floating-point operations (flops) to sample the n eigenvalues of a matrix of order n. The key observation is that in order to accomplish this task it is not necessary to explicitly sample matrices from the corresponding group, but it suffices to understand the distribution of their Hessenberg forms, which we analyze in detail in Section 4. The main advantage of this approach is that unitary or orthogonal matrices in Hessenberg form can be diagonalized in $\mathcal{O}(n^2)$ flops. We will exploit this to derive the algorithm discussed in Section 5, which has quadratic complexity and linear storage requirements.

The algorithm we propose can efficiently sample the joint distribution of the eigenvalues of Haar-distributed matrices from any of the Lie groups $O(n)$, $SO(n)$, $U(n)$, and $SU(n)$. In Section 6 we show that the empirical phase and spacing of eigenvalues sampled by our algorithm follow the corresponding theoretical distributions for $U(n)$, and then we explore empirically the distribution of the eigenvalues of matrices from the Haar distribution of $SU(n)$, $O(n)$, $SO(n)$, and $O^-(n)$, for which fewer theoretical results are available.

Our starting point is an algorithm proposed by Stewart [18] for sampling random matrices from the Haar distribution of $O(n)$. We recall this approach and the subsequent generalization to $U(n)$, due to Diaconis and Shahshahani [19], in Section 3. This technique exploits an algorithm for the QR factorization based on Householder transformations, which we revise in Section 2.

These techniques require the sampling of $\mathcal{O}(n^2)$ random variables, and need $\mathcal{O}(n^2)$ memory for storing the result. We provide an alternative and more compact formulation for the Hessenberg form obtained by the algorithms above, which requires the sampling of $\mathcal{O}(n)$ random variables and $\mathcal{O}(n)$ storage. We show that this formulation can be used to compute the eigenvalues in $\mathcal{O}(n^2)$ floating-point operations by leveraging the unitary QR algorithm in [20].

The use of a condensed factorization for storing random matrices has been already explored, for instance, by Edelman and Ure [21], who sample unitary matrices by taking random Schur parameters [22]. Methods similar to the technique presented in this work might be obtained by representing the Hessenberg forms of unitary Haar-distributed matrices using Schur parameters, or similarly condensed forms such as CMV matrices [23], and then using a quadratic method to compute their eigenvalues [22,24–28].

The approach discussed here is based on the unitary QR algorithm in [20, Section 5], The latter can be seen as a special case of the rootfinding algorithm of Aurentz et al. [25], which has been proven to be backward stable and compares favorably with the methods above in terms of performance.

Finally, we introduce some notation. Throughout the manuscript, we use capital letters (A, B, ...) to denote matrices, lower case letters (u, v, ...) to denote vectors, and lower case Greek letters (α, β, ...) to denote scalars. We indicate the entries of matrices and

vectors using a subscript notation, so that a_{ij} denotes the entry in position (i, j) of the matrix A and v_k refers to the kth element of the vector v. We use the same notation for random variables.

We denote by $\mathcal{N}_{\mathbb{R}}(\mu, \sigma^2)$ the Gaussian distribution centered at $\mu \in \mathbb{R}$ with variance σ^2, and by $\mathcal{N}_{\mathbb{R}}^{(m,n)}$ the distribution of $m \times n$ random matrices with independently distributed Gaussian entries, that is,

$$X \sim \mathcal{N}_{\mathbb{R}}^{(m,n)} \iff x_{ij} \sim \mathcal{N}_{\mathbb{R}}(0, 1), \qquad i = 1, \dots, m, \quad j = 1, \dots n.$$

The chi-squared distribution with k degrees of freedom, denoted by $\chi^2_{\mathbb{R}}(k)$, is the distribution of the sum of the squares of k independent Gaussian random variables, and is formally defined by

$$\gamma \sim \chi^2_{\mathbb{R}}(k) \iff \gamma = \sum_{i=1}^{k} \delta_i^2, \qquad \delta_i \sim \mathcal{N}_{\mathbb{R}}^{(k,1)}.$$

These are real-valued distributions. The complex counterpart of $\mathcal{N}_{\mathbb{R}}(\mu, \sigma^2)$ is denoted by $\mathcal{N}_{\mathbb{C}}(\mu, \sigma^2)$ and defined by

$$\gamma + \mathrm{i}\delta \sim \mathcal{N}_{\mathbb{C}}(\mu, \sigma^2) \iff \gamma \sim \mathcal{N}_{\mathbb{R}}\left(\mathrm{Re}(\mu), \frac{\sigma^2}{2}\right) \text{ and } \delta \sim \mathcal{N}_{\mathbb{R}}\left(\mathrm{Im}(\mu), \frac{\sigma^2}{2}\right), \quad \text{with } \gamma \perp\!\!\!\perp \delta,$$

where the notation $\gamma \perp\!\!\!\perp \delta$ indicates that the random variables γ and δ are independent. The distribution $\mathcal{N}_{\mathbb{C}}^{(m,n)}$ is defined as in the real case, and we can define the complex analogue of the $\chi^2_{\mathbb{R}}(k)$ distribution as

$$\gamma \sim \chi^2_{\mathbb{C}}(n-1)(k) \iff \gamma = \sum_{i=1}^{k} |\delta_i|^2, \qquad \delta_i \sim \mathcal{N}_{\mathbb{C}}^{(k,1)}.$$

It is easy to prove that $\chi^2_{\mathbb{C}}(n-1)(k) \sim \chi^2_{\mathbb{R}}(2k)/2$.

2. Householder transformations and QR factorization

In this section we briefly recall some basic facts about Householder transformations, and discuss how they can be employed to compute the QR factorization of a square matrix.

Let $v \in \mathbb{C}^n$ be a nonzero vector. The matrix

$$P(v) = I_n - \frac{2}{\|v\|_2^2} v v^* \tag{2.1}$$

is a Householder transformation. It is easy to verify that $P(v)$ is unitary and Hermitian, and in particular is orthogonal and symmetric if the entries of v are real. This implies

that $P(v)^2 = I_n$. Moreover, computing the action of $P(v)$ on a vector requires only $\mathcal{O}(n)$ flops, instead of the $\mathcal{O}(n^2)$ flops that would be needed for a generic $n \times n$ matrix-vector product.

Householder transformations are a convenient tool to zero out the trailing entries of a nonzero vector $u \in \mathbb{C}^n$. For instance, let $\theta_1 = \operatorname{Arg} u_1$, where $\operatorname{Arg}\colon \mathbb{C} \to (-\pi, \pi]$ denotes the principal value of the argument function. Then the matrix $P(v)$ for $v = u + \mathrm{e}^{\mathrm{i}\theta_1}\|u\|_2 e_1$ is such that $P(v)u = -\mathrm{e}^{\mathrm{i}\theta_1}\|u\|_2 e_1$, where e_i denotes the ith column of the identity matrix. In order to zero out only the last $n-k$ components of $u \in \mathbb{C}^{n\times n}$, it suffices to consider the block matrix

$$\widehat{P}_k(u) := \begin{bmatrix} I_{k-1} & \\ & P(v) \end{bmatrix}, \qquad v := u_{k:n} + \mathrm{e}^{\mathrm{i}\theta_k}\|u_{k:n}\|_2 e_1, \qquad \theta_k := \operatorname{Arg} u_k, \tag{2.2}$$

where $u_{i:j} \in \mathbb{C}^{j-i+1}$ denotes the vectors that contains the entries of u from the ith to the jth inclusive.

Any matrix $A \in \mathbb{R}^{n\times n}$ has the QR factorization $A =: QR$, where $Q \in O(n)$ and $R \in \mathbb{R}^{n\times n}$ is upper triangular [29, Theorem 5.2.1]. If A is nonsingular, this factorization is unique up to the sign of the diagonal entries of R. This result can be extended to complex matrices: any nonsingular matrix $A \in \mathbb{C}^{n\times n}$ has a unique QR factorization $A =: QR$, where $Q \in U(n)$ and $R \in \mathbb{C}^{n\times n}$ is upper triangular with real positive entries along the diagonal [30, Theorem 7.2]. More generally, the QR factorization of a full-rank matrix is unique as long as the phases of the diagonal entries of R are fixed.

In many of the following proofs, it will be useful to assume that the matrix A under consideration has full rank. This is typically not restrictive, since rank-deficient matrices are a zero-measure set in $\mathcal{N}_{\mathbb{R}}^{(n,n)}$ and $\mathcal{N}_{\mathbb{C}}^{(n,n)}$; we will comment on this fact in further detail when needed.

We now explain how to compute efficiently the QR factorization of an $n \times n$ complex matrix A by means of Householder reflections [29, Section 5.2.2]. The corresponding procedure for real matrices can be obtained by employing real Householder reflectors. Let the matrix $A^{(0)} := A$ be partitioned by columns as

$$A^{(0)} = \begin{bmatrix} A_1^{(0)} & \cdots & A_n^{(0)} \end{bmatrix},$$

and let $\widetilde{P}_1 := \widehat{P}_1\big(A_1^{(0)}\big)$. We obtain that

$$\widetilde{P}_1 A^{(0)} = \begin{bmatrix} r_{11} & c \\ 0 & A^{(1)} \end{bmatrix}, \qquad r_{11} = -\mathrm{e}^{\mathrm{i}\theta}\left\|A_1^{(0)}\right\|, \quad A^{(1)} \in \mathbb{C}^{(n-1)\times(n-1)}, \quad c \in \mathbb{C}^{1\times(n-1)},$$

where θ denotes the complex sign of the top left element of the matrix $A^{(0)}$. If we apply this procedure recursively to the trailing submatrix $A^{(1)}$, after $n-1$ steps we obtain

$$\widetilde{P}_{n-1} \cdots \widetilde{P}_1 A =: R, \qquad \widetilde{P}_k := \widehat{P}_k\big(A_1^{(k-1)}\big), \qquad k = 1, \ldots, n-1, \tag{2.3}$$

where the $Q := \widetilde{P}_1 \cdots \widetilde{P}_{n-1}$ is unitary and R is upper triangular. This algorithm produces the matrix R and the factors $\widetilde{P}_i$ for $i = 1, \ldots, n-1$ in $4n^3/3$ flops [29, Section 5.2.2]. Computing the matrix Q explicitly requires an additional $4n^3/3$ flops [29, Section 5.1.6], but by exploiting the structure one can compute the action of Q on a vector or matrix with only n^2 and n^3 flops, respectively.

In order to normalize the factorization, note that if D is the diagonal matrix such that $d_{ii} = -\mathrm{e}^{-\mathrm{i}\theta_i}$ where θ_i is defined as in (2.2), then the matrix DR has positive real entries along the diagonal. Therefore, the normalized factorization with positive diagonal entries in the upper triangular factor is:

$$A = \widetilde{Q}\widetilde{R}, \qquad \widetilde{R} := D^* R, \quad \widetilde{Q} := \widetilde{P}_1 \ldots \widetilde{P}_{n-1} D, \quad D := -\operatorname{diag}(\mathrm{e}^{\mathrm{i}\theta_1}, \ldots, \mathrm{e}^{\mathrm{i}\theta_n}), \tag{2.4}$$

where $\widetilde{P}_1, \ldots, \widetilde{P}_{n-1}$ are as in (2.3).

A matrix $H \in \mathbb{C}^{n\times n}$ is in upper Hessenberg form if $h_{ij} = 0$ when $i > j+1$. Any square matrix is unitarily similar to an upper Hessenberg matrix, that is, for any $A \in \mathbb{C}^{n\times n}$ there exists a matrix $U_A \in U(n)$ such that $U_A A U_A^*$ is an upper Hessenberg matrix.

3. The Haar measure and Stewart's algorithm

Birkhoff and Gulati [31, Theorem 4] note that if the QR factorization $A =: QR$ of an $n \times n$ matrix $A \sim \mathcal{N}_{\mathbb{R}}^{(n,n)}$ is normalized so that the entries along the diagonal of R are all positive, then Q is distributed according to the Haar measure over $O(n)$.

This observation suggests a straightforward method for sampling Haar distributed matrices from $O(n)$. One can simply generate an $n \times n$ real matrix $A \sim \mathcal{N}_{\mathbb{R}}^{(n,n)}$, compute its QR decomposition, and normalize it as discussed in Section 2. This procedure is easy to implement, since efficient and numerically stable routines for computing the QR factorization are available in most programming languages.

The computational cost of this technique can be approximately halved by computing the QR factorization implicitly. Stewart [18] proposes an algorithm that does not explicitly generate the random matrix A, but produces the transpose of the matrix Q in factored form as $D\widetilde{P}_1 \ldots \widetilde{P}_{n-1}$, where $\widetilde{P}_k := \widehat{P}_k\big(x^{(k)}\big)$ for some random vector $x^{(k)} \sim \mathcal{N}_{\mathbb{R}}^{(n,1)}$, and D is an $n \times n$ diagonal sign matrix whose entries are computed as on line 4 of Algorithm 3.1.

This algorithm readily generalizes to the complex case, as suggested by Diaconis and Shahshahani [19] and discussed in detail by Mezzadri [32]. In order to sample Haar-distributed random matrices from $U(n)$, it suffices to generate vectors with entries drawn from the standard complex normal distribution $\mathcal{N}_{\mathbb{C}}(0, 1)$, and replace the real sign function by its complex generalization $e^{\mathrm{i}\operatorname{Arg}(z)}$ for $z \in \mathbb{C}$.

We outline this approach in Algorithm 3.1. The function UMULT(X, $\mathbb{F}$) computes the action of a Haar-distributed matrix from $O(n)$ (if $\mathbb{F} = \mathbb{R}$) or $U(n)$ (if $\mathbb{F} = \mathbb{C}$) on the rectangular matrix $X \in \mathbb{C}^{n\times m}$. In order to determine the computational cost of the algorithm, note that asymptotically only the two matrix-vector products and the matrix

Algorithm 3.1: Action of a matrix from the Haar distribution.

1 **function** UMULT($X \in \mathbb{C}^{n\times m}, \mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$)

Compute the matrix QX, where Q is an orthogonal (if $\mathbb{F} = \mathbb{R}$) or unitary (if $\mathbb{F} = \mathbb{C}$) matrix from the Haar distribution.

2 **for** $k \leftarrow 2$ **to** n **do**

3 $v \sim \mathcal{N}_{\mathbb{F}}^{(k,1)}$

4 $d_{n-k+1} \leftarrow -\mathrm{e}^{\mathrm{i}\operatorname{Arg} v_1}$

5 $u \leftarrow v - d_{n-k+1}\|v\|_2 e_1$

6 $u \leftarrow u/\|u\|_2$

7 $X =: \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \begin{matrix} \}n-k \\ \}k \end{matrix}$

8 $X \leftarrow \begin{bmatrix} X_1 \\ X_2 - (2u)(u^* X_2) \end{bmatrix}$

9 $z \sim \mathcal{N}_{\mathbb{F}}(0,1)$

10 **return** $\operatorname{diag}(d_1, \ldots, d_{n-1}, -\mathrm{e}^{\mathrm{i}\operatorname{Arg} z}) \cdot X$

sum on line 8 are significant. Therefore, each iteration of the for loop starting on line 2 requires $4km$ flops, and the computational cost of Algorithm 3.1 is approximately $2n^2m$ flops.

In order to sample Haar-distributed matrices, it suffices to set X to I_n. In this case, the computational cost of the algorithm can be reduced by taking into account the special structure of A: the cost of line 8 drops to $4k^2$ flops per step, which yields an overall computational cost of $4n^3/3$ flops.

3.1. Sampling from the special groups

The ideas presented so far can be modified in order to sample matrices with prescribed determinant. Imposing that the determinant be 1 is of particular interest, as it implies sampling from the compact Lie groups $SU(n)$ and $SO(n)$. As discussed in the previous section, the QR factorization of a random matrix A can be used to sample matrices distributed according to the Haar measure over $U(n)$ and $O(n)$. An analogous result holds for the special groups, if the last diagonal entry of R is chosen so that $\det Q = 1$.

Lemma 3.1. *Let $A \sim \mathcal{N}_{\mathbb{C}}^{(n,n)}$ (resp. $A \sim \mathcal{N}_{\mathbb{R}}^{(n,n)}$) and let $A =: QR$ be its QR factorization, where Q and R are chosen so that*

$$r_{ii} \in \{\gamma \in \mathbb{R} : \gamma \geqslant 0\}, \qquad r_{nn} = \det(A) \cdot \left[\prod_{j=1}^{n-1} r_{jj}\right]^{-1},$$

whenever A is nonsingular. Then, Q is distributed according to the Haar measure over $SU(n)$ (resp. $SO(n)$).

Proof. We consider the complex case first. Note that the set of rank deficient matrices has measure zero in $\mathcal{N}_{\mathbb{C}}^{(m,n)}$; therefore, the distribution of the unitary QR factor of such matrices is irrelevant for the distribution under consideration.

When A is nonsingular, fixing the phases of the diagonal entries of R makes the QR factorization unique. Hence, the random variables q_{ij}, for $i, j = 1, \dots, n$ are well-defined. In addition, the choice of the diagonal entries of R ensures that $\det R = \det A$ and thus that $\det Q = 1$.

In order to prove that Q is distributed according to the Haar measure over $U(n)$, we need to show that it has the same distribution as PQ for any constant matrix $P \in SU(n)$. For any such P, the matrix PA has the QR factorization $PA =: (PQ)R$.

Being independent Gaussian random variables, the entries of A are invariant under unitary transformations, thus PA has the same distribution as A. The triangular QR factor of PA is R, which necessarily satisfies the normalization constraints on the diagonal entries. Therefore, PQ has the same distribution as Q.

The proof for the real case is analogous and therefore omitted. □

The above result yields a method for sampling the Haar distribution of the special unitary and orthogonal groups. In the next sections, we discuss how to make this method efficient for sampling the corresponding eigenvalue distribution. The approach we propose can be used for both $U(n)$ and $O(n)$, and $SU(n)$ and $SO(n)$.

Remark 3.2. Note that the Haar distribution of $SU(n)$ coincides with the Haar distribution of $U(n)$ conditioned to the event $\det Q = 1$. This is easily verified by checking that the latter measure is invariant under the action of elements in $SU(n)$. This suggests that the above procedure can be further generalized and used to sample the probability μ_ξ obtained by conditioning μ with $\det Q = \xi$ for some $\xi \in \mathbb{S}^1$, where $\mathbb{S}^1 = \{\xi \in \mathbb{C} : |\xi| = 1\}$ denotes the complex unit circle. If $\xi \neq 1$, these matrices do not form a group, but we can write

$$\{Q \in U(n) \mid \det Q = \xi\} = \{P\widetilde{Q} \mid \widetilde{Q} \in SU(n)\},$$

where P is any constant matrix such that $\det P = \xi$. Sampling the matrices in $SU(n)$ and then multiplying them by any fixed P yields the correct conditional probability distribution. More specifically, by choosing the diagonal matrix $P = \operatorname{diag}(1, \dots, 1, \xi)$ we can readily adapt the algorithm discussed in the next section to sample unitary or orthogonal matrices with determinant ξ.

3.2. The eigenvalue distribution

Given a random matrix sampled according to one of the measures described so far, we are interested in describing the distribution of a generic eigenvalue. This can be computed as a marginal probability by integrating the joint eigenvalue distribution with respect to $n - 1$ variables. For $U(n)$, the latter is known explicitly [2, Chapter 11], and can be used to prove that a generic eigenvalue is uniformly distributed over $\mathbb{S}^1$.

We are unaware of an analogous result for $SU(n)$, and we could not find any references stating the expected distribution for a generic eigenvalue. Nevertheless, using the fact

that the eigenvalue distribution arises from Haar-distributed matrices, we can obtain the partial characterization in the following lemma. The well-known distribution for $U(n)$ is for ease of comparison.

Lemma 3.3. *Let μ and μ_1 be the Haar distributions over $U(n)$ and $SU(n)$, respectively, and let Λ_μ and Λ_{μ_1} be the corresponding distributions for a generic eigenvalue. Then, Λ_μ is the uniform distribution over $\mathbb{S}^1$, and Λ_{μ_1} has a $\frac{2\pi}{n}$-periodic phase, that is,*

$$\Lambda_{\mu_1}(\mathcal{G}) = \Lambda_{\mu_1}\left(e^{\frac{2\pi i}{n}}\mathcal{G}\right), \qquad \text{for any measurable set } \mathcal{G} \subset \mathbb{S}^1. \tag{3.1}$$

Proof. We start by considering the distribution of $U(n)$. Recall that μ is invariant under left multiplication in $U(n)$, and since $\xi I \in U(n)$ for any $\xi \in \mathbb{S}^1$, we have that Λ_μ is invariant under multiplication by $\xi \in \mathbb{S}^1$. Since $\mathbb{S}^1$ is a compact Lie group, Λ_μ must be its Haar measure, which is the uniform distribution.

We can use a similar argument for $SU(n)$. Since $\xi I \in SU(n)$ for any ξ such that $\xi^n = 1$, we have that Λ_{μ_1} is invariant under multiplication by a root of unity, thus must be $\frac{2\pi}{n}$-periodic as in (3.1). □

In Section 6 we will verify this claim experimentally in order to test the correctness of our implementation. In particular, we will find that Λ_μ is the uniform distribution, as expected, and that Λ_{μ_1} has the periodicity predicted by Lemma 3.3.

4. The Hessenberg form of Haar-distributed matrices

As mentioned in the introduction, the unitary QR algorithm of [20] cannot be applied directly to the representation of the upper Hessenberg form of a random matrix. In this section, first we show how to sample a factorization of the upper Hessenberg form of Haar-distributed matrices using only $\mathcal{O}(n)$ random variables, then we explain how to rewrite this factorization in a form that is suitable for computing the eigenvalues with core-chasing algorithms, which we briefly review in Section 5. The main result of this section is the following.

Theorem 4.1. *Let $w_1, \ldots, w_{n-1} \in \mathbb{C}^2$ be independent random vectors such that*

$$w_j = \begin{bmatrix} \alpha_j \\ \beta_j \end{bmatrix}, \qquad \alpha_j \sim \mathcal{N}_{\mathbb{C}}(0,1), \qquad \beta_j^2 \sim \chi^2_{\mathbb{C}}(n-j),$$

and let

$$H = P_1 \ldots P_{n-1} D \in \mathbb{C}^{n\times n} \tag{4.1}$$

be the unitary Hessenberg matrix such that

$$P_j = I - \frac{2}{\|v_j\|_2^2} v_j v_j^*, \quad D = -\operatorname{diag}(\mathrm{e}^{\mathrm{i}\theta_1}, \dots, \mathrm{e}^{\mathrm{i}\theta_n}), \quad v_j = \begin{bmatrix} 0_{j-1} \\ \alpha_j + \mathrm{e}^{\mathrm{i}\theta_j}\|w_j\|_2 \\ \beta_j \\ 0_{n-j-1} \end{bmatrix}, \tag{4.2}$$

where $\theta_j = \operatorname{Arg}\alpha_j$ for $j = 1, \dots, n-1$ and $\theta_n \sim U(-\pi, \pi]$. Then, the joint eigenvalue distribution of H is that of Haar-distributed unitary matrices of $U(n)$.

Proof. In view of the discussion in Section 3, if Q is Haar distributed then

$$Q \sim \widetilde{P}_1 \dots \widetilde{P}_{n-1} D, \tag{4.3}$$

with $\widetilde{P}_j$ defined as follows:

$$\widetilde{P}_j = P(v_j) = I - \frac{2}{\|v_j\|^2} v_j v_j^*, \qquad v_j := \begin{bmatrix} 0_{j-1} \\ u_1^{[j]} + \mathrm{e}^{\mathrm{i}\theta_j}\|u^{[j]}\|_2 \\ u_{2:n}^{[j]} \end{bmatrix}, \qquad u^{[j]} \sim \mathcal{N}_{\mathbb{C}}^{(n-j+1,1)},$$

where $\theta_j = \operatorname{Arg}\left(u_1^{[j]}\right)$, and D is a diagonal matrix defined by $d_{jj} = \mathrm{e}^{\mathrm{i}\theta_j}$. In order to prove the claim we will reduce this matrix to upper Hessenberg form.

More specifically, we prove by induction on n that there exists a unitary matrix U such that $H \sim UQU^*$ is upper Hessenberg, and that $Ue_1 = e_1$. The latter property will be useful in the induction step. Throughout the proof, we will also repeatedly exploit the fact that if W is orthogonal then $WP(v)W^* = P(Wv)$, which can be verified by a direct computation.

If $n = 1$, then Q and H are both diagonal matrices, and there is nothing to prove. If $n = 2$, then Q is already upper Hessenberg, and we may write it as

$$Q = \widetilde{P}_1 D, \qquad \widetilde{P}_1 = I - \frac{2}{\|v_1\|^2} v_1 v_1^*, \qquad v_1 := \begin{bmatrix} \alpha_1 + e^{i\theta_1}\|u^{[1]}\| \\ \beta_1 \end{bmatrix}, \qquad u^{[1]} := \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix}.$$

With the matrix $U = \operatorname{diag}(1, \mathrm{e}^{-\mathrm{i}\theta})$, where $\theta := \operatorname{Arg}(\beta_1)$, we can perform the similarity transformation

$$UQU^* = U\widetilde{P}_1 D U^* = U\widetilde{P}_1 U^* D = P(Uv_1)D, \qquad Uv_1 = \begin{bmatrix} \alpha_1 + e^{i\beta_1}\|u^{[1]}\| \\ |\beta_1| \end{bmatrix}.$$

We now observe that $|\beta_1|^2 \sim \chi^2_{\mathbb{C}}(1)$ and $\|u^{[1]}\|_2 = \left\|\left[\begin{smallmatrix} \alpha_1 \\ |\beta_1| \end{smallmatrix}\right]\right\|_2$, which implies that $P(Uv_1)$ and P_1 have the same distribution. Moreover, the matrix U thus constructed satisfies $Ue_1 = e_1$.

For the inductive step, assume that the statement holds for matrices of order $n-1$, and consider Q as in (4.3). If $\widetilde{P}_1 = I - \frac{2}{\|v_1\|^2} v_1 v_1^*$, we can construct a Householder reflector U_1 such that $U_1 e_1 = e_1$, and

$$U_1 v_1 = U_1 \begin{bmatrix} \alpha_1 + \mathrm{e}^{\mathrm{i}\theta_1} \left\| u^{[1]} \right\| \\ u^{[1]}_{2:n} \end{bmatrix} = \begin{bmatrix} \alpha_1 + \mathrm{e}^{\mathrm{i}\theta_1} \left\| u^{[1]} \right\| \\ \left\| u^{[1]}_{2:n} \right\|_2 \\ 0_{n-2} \end{bmatrix}. \tag{4.4}$$

We note that U_1 can be chosen so to be independent of $\widetilde{P}_2, \dots, \widetilde{P}_{n-1}$, as it depends only on $u^{[1]}$. We can then consider the similarity

$$U_1 Q U_1^* = U_1 \widetilde{P}_1 U_1^* U_1 \widetilde{P}_2 \dots \widetilde{P}_{n-1} D U_1^*,$$

where $U_1 \widetilde{P}_1 U_1^* = I - \frac{2}{\|v_1\|^2} U_1 v_1 (U_1 v_1)^* \sim P_1$ in view (4.4), and $\left\| u^{[1]}_{2:n} \right\|_2^2 \sim \chi^2_{\mathbb{C}}(n-1)$. We now factorize D as

$$D := \begin{bmatrix} 1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} \begin{bmatrix} d_1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} = \widehat{D} D_1,$$

and note that $U_1 D_1 = D_1 U_1$ thanks to $U e_1 = e_1$. Therefore we can write

$$U_1 Q U_1^* \sim P_1 U_1 \widetilde{P}_2 \dots \widetilde{P}_{n-1} \widehat{D} U_1^* D_1 = P_1 \begin{bmatrix} 1 & \\ & \widehat{Q} \end{bmatrix} D_1.$$

We observe that since U_1 is independent of $\widetilde{P}_2 \dots \widetilde{P}_{n-1}$ and $\widehat{D}$, $\widehat{Q}$ is Haar distributed in $U(n-1)$. By inductive hypothesis, there exists an $(n-1) \times (n-1)$ unitary matrix $\widehat{U}$ which satisfies $\widehat{U} e_1 = e_1$ and is such that

$$\widehat{U} \widehat{Q} \widehat{U}^* = \widehat{P}_2 \dots \widehat{P}_{n-1} \widehat{D}, \qquad P_j \sim \begin{bmatrix} 1 & \\ & \widehat{P}_j \end{bmatrix}, \qquad j = 2, \dots, n-1.$$

The property $\widehat{U} e_1 = e_1$ implies that $\left[\begin{smallmatrix} 1 & \\ & \widehat{U} \end{smallmatrix}\right]$ commutes with both P_1 and D_1, and we can write

$$\begin{aligned} \begin{bmatrix} 1 & \\ & \widehat{U} \end{bmatrix} U_1 Q U_1^* \begin{bmatrix} 1 & \\ & \widehat{U}^* \end{bmatrix} &\sim P_1 \begin{bmatrix} 1 & \\ & \widehat{U} \end{bmatrix} \begin{bmatrix} 1 & \\ & \widehat{Q} \end{bmatrix} \begin{bmatrix} 1 & \\ & \widehat{U}^* \end{bmatrix} D_1 \\ &= P_1 P_2 \dots P_{n-1} \widehat{D} D_1 = P_1 P_2 \dots P_{n-1} D, \end{aligned}$$

which concludes the proof. □

The analogue of Theorem 4.1 for real matrices is obtained by replacing the first element of w_j by $\alpha_j \sim \mathcal{N}_{\mathbb{R}}(0,1)$ and by sampling θ_n uniformly from the set $\{0, \pi\}$. The result can be easily modified in order to sample the joint eigenvalues distribution of matrices from the Haar measure over $SU(n)$ and $SO(n)$: setting

$$d_{nn} = (-1)^{n-1} \prod_{i=1}^{n-1} d_{ii}$$

guarantees that $\det H = 1$, while Lemma 3.1 ensures that the matrices are sampled according to the Haar measure.

Remark 4.2. In a similar way, we may set the last diagonal entry of D to obtain $\det Q = \xi$, for any $\xi \in \mathbb{S}^1$. According to Remark 3.2, this procedure samples the Haar distribution conditioned on the event $\det Q = \xi$.

5. Computing the eigenvalues of upper Hessenberg unitary matrices

By Theorem 4.1, any random upper Hessenberg unitary matrix H can be described by $\mathcal{O}(n)$ parameters by using the factored form (4.1).

The computation of the eigenvalues of unitary upper Hessenberg matrices was first considered by Gragg [22], and later investigated by numerous authors, see for instance [33–35]. Here, in particular, we are interested in the approach proposed by Aurentz, Mach, Vandebril, and Watkins [20]. This algorithm, briefly described in Section 5.2, is implemented in `eiscor` [24], a Fortran 90 package for the solution of eigenvalue problems by core-chasing methods available on GitHub.[1] The software computes the eigenvalues of the Hessenberg matrix

$$H = G_1 \cdots G_{n-1} D, \tag{5.1}$$

where the unitary matrices $G_1, \ldots, G_{n-1}$ are plane rotations of the form

$$G_j = \begin{bmatrix} I_{j-1} & & \\ & \widehat{G}_j & \\ & & I_{n-j-1} \end{bmatrix}, \qquad \widehat{G}_j = \begin{bmatrix} c_j & s_j \\ -s_j & \overline{c}_j \end{bmatrix}, \qquad c_j \in \mathbb{C}, \qquad s_j \in \mathbb{R}. \tag{5.2}$$

Because of its special structure, the matrix G_j in (5.2) is said to be "essentially 2×2", and the 2×2 matrix $\widehat{G}_j$ is called a *core block*. In principle, the core chasing algorithm in [20] could be applied to any factorization of H involving only "essentially 2×2" unitary matrices, even though the particular implementation described in [24] involves only the special family of plane rotations. In practice, however, the key operation in the QR algorithm—the so-called *turnover*—has to be implemented with care in order to ensure backward stability. In order to leverage the analysis done for rotations of the form (5.2), it is thus convenient to refactorize H given in the form (4.1) as a product of the form (5.1).

This section is structured as follows. First, we show how to refactorize a representation in terms of 2×2 Householder transformations into one consisting only of plane rotations with real sines. Then we briefly recall the main ideas underlying the unitary QR algorithm implemented in `eiscor`.

[1] https://github.com/eiscor/eiscor.

5.1. Refactorizing core transformations

The assumption that all core blocks in the factorization of the Hessenberg matrix H be plane rotations with real sines is not restrictive, as it is always possible to rewrite H as a product of the form (5.1). This refactorization can be performed by noting that any 2×2 unitary matrix U can be written as

$$U = \begin{bmatrix} c & s \\ -s & \overline{c} \end{bmatrix} \begin{bmatrix} d_1 & \\ & d_2 \end{bmatrix}, \qquad \begin{cases} c = u_{11} e^{\mathrm{i}\theta}, \\ s = -u_{21} e^{\mathrm{i}\theta}, \end{cases} \qquad \theta = \operatorname{Arg}(\overline{u_{21}}). \tag{5.3}$$

where the diagonal entries are given by

$$d_1 = e^{-\mathrm{i}\theta} |u_{11}|^2 + e^{\mathrm{i}\theta} u_{21}^2, \qquad d_2 = e^{\mathrm{i}\theta}(u_{11}u_{22} - u_{21}u_{12}).$$

The procedure above can be performed in a backward stable manner, as it coincide with the computation of the QR decomposition of U [24].

In addition, the product of a plane rotation with real sines G and a unitary diagonal 2×2 matrix D can be refactorized so to swap the order of the two operations. In fact, there exist a unitary 2×2 diagonal matrix $\widetilde{D}$ and a plane rotation with real sines $\widetilde{G}$ such that $GD = \widetilde{D}\widetilde{G}$. This property is easy to verify, and represents the foundation of most core-chasing algorithms [24]. Combining these two observations gives the following.

Lemma 5.1. *Let $H \in \mathbb{C}^{n \times n}$ be a unitary upper Hessenberg matrix factored as in (4.1). Then, there exist $G_1, \ldots, G_{n-1} \in \mathbb{C}^{n \times n}$ unitary plane rotations with real sines and $\widetilde{D} \in \mathbb{C}^{n \times n}$ unitary diagonal such that*

$$H = G_1 \ldots G_{n-1} \widetilde{D}. \tag{5.4}$$

This refactorization can be computed in $\mathcal{O}(n)$ flops.

Proof. The proof is by induction on n. For $n = 2$ we have that $H = P_1 D$, and the refactorization can be performed directly by relying on (5.3). If $n > 2$, then there exist a plane rotation $G_1 \in \mathbb{C}^{n \times n}$ as in (5.2) and a unitary diagonal matrix

$$D_1 := \begin{bmatrix} \alpha & & \\ & \beta & \\ & & I_{n-2} \end{bmatrix}, \qquad \alpha, \beta \in \mathbb{S}^1,$$

such that $P_1 = G_1 D_1$. Since the matrix $\begin{bmatrix} d_{11} & \\ & I_{n-1} \end{bmatrix}$ commutes with $D_1, P_2, \ldots, P_{n-1}$, we can write

$$H = G_1 \begin{bmatrix} \alpha d_{11} & \\ & I_{n-1} \end{bmatrix} \widetilde{P}_2 P_3 \cdots P_{n-1} \widetilde{D}, \qquad \widetilde{P}_2 := \begin{bmatrix} 1 & & \\ & \beta & \\ & & I_{n-2} \end{bmatrix} P_2,$$

where $\widetilde{d}_{ij} = 1$ if $i = j = 1$ and $\widetilde{d}_{ij} = d_{ij}$ otherwise. We note that this refactorization has the form $H = G_1 \begin{bmatrix} \alpha d_1 & \\ & 1 \end{bmatrix} \begin{bmatrix} 1 & \\ & \widetilde{H} \end{bmatrix}$, where $\widetilde{H}$ has the same structure as H but order $n-1$. The inductive hypothesis yields $\begin{bmatrix} 1 & \\ & \widetilde{H} \end{bmatrix} = G_2 \dots G_{n-1} \begin{bmatrix} 1 & \\ & D' \end{bmatrix}$, which gives

$$H = G_1 G_2 \dots G_{n-1} \widetilde{D}, \qquad \widetilde{D} := \begin{bmatrix} \alpha d_{11} & \\ & D' \end{bmatrix}.$$

This procedure provides an algorithm for refactoring H from the form (4.1) to the form (5.4). Noting that each step requires $\mathcal{O}(1)$ flops, for a total of $\mathcal{O}(n)$ flops for the complete refactorization, concludes the proof. □

5.2. Computing the eigenvalues of unitary Hessenberg matrices

We have described how to generate unitary upper Hessenberg matrices whose joint eigenvalue distribution follows the Haar measure, and we have shown how to write it in the factored form (5.4).

In order to compute the spectrum of H in the form (5.4) in $\mathcal{O}(n^2)$ flops, we rely on the method proposed in [20], which belongs to the family of *core-chasing algorithms* [24]. Here we provide a high-level overview of this technique, and refer the interested reader to the original paper [20] for a detailed discussion.

With the term *core transformation* we indicate an essentially 2×2 unitary matrix such as, for example, a plane rotation. The factorization (5.4) is an example of a matrix expressed as product of $n-1$ core transformation and a diagonal matrix. In this particular case, the facotrization can also be used to give a compact representation of H that uses only $\mathcal{O}(n)$ parameters, as opposed to the $\mathcal{O}(n^2)$ that would be necessary if all the entries of H were explicitly stored. Each core transformation acts on a pair of adjacent indices. The matrix G_j in (5.2), for example, acts on the indices j and $j+1$.

The standard single-shift bulge chasing QR algorithm works as follows. Given an upper Hessenberg matrix H, we determine a first core transformation Q_1 acting on the indices 1 and 2 such that $Q_1(H - \rho I)e_1 = \alpha e_1$. The parameter ρ is the *shift*, and has to be carefully chosen in order to ensure fast convergence and robustness of the method [36]. The implementation considered here uses a Wilkinson shift, which is projected onto $\mathbb{S}^1$ as the matrix H is unitary [20].

We use the core transformation Q_1 to compute $Q_1 H Q_1^*$, which is not upper Hessenberg having a nonzero element in position $(3, 1)$. Another core transformation Q_2 acting on the indices 2 and 3 is used to restore the upper Hessenberg structure and obtain $Q_2 Q_1 H Q_1^*$. The similarity $Q_2 Q_1 H Q_1^* Q_2^*$, however, yields a matrix that is not upper Hessenberg because of a nonzero element in position $(4, 2)$, and the process is repeated until the nonzero element, the so-called *bulge*, is eliminated from the last row of the matrix. The focus on the nonzero element that breaks the upper Hessenberg structure and is "chased to the bottom" until it disappears from the matrix, justifies the name *bulge-chasing QR* used for this algorithm.

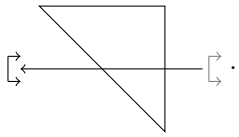
Core-chasing algorithms have a similar formulation, and indeed are mathematically equivalent [24], but do not handle the entries of the matrix directly, as we now explain.

An upper Hessenberg matrix can always be written as $H = QR$, where R is upper triangular and $Q = G_1 \dots G_{n-1}$ is the product of unitary plane rotations. The core-chasing step starts by computing a core transformation Q_1 such that $Q_1(H - \rho I)e_1$ is a scalar multiple of e_1. Keeping $H = QR$ in factored form, the similarity transformation with Q_1 gives

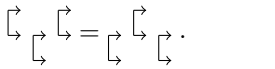
$$Q_1 H Q_1^* = Q_1 \underbrace{G_1 \dots G_{n-1} R}_{H} Q_1^*.$$

We now make the following two key observations.

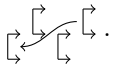
- Given an upper triangular matrix R and a core transformation Q_1, it is always possible to find another upper triangular matrix $\widetilde{R}$ and core transformation $\widetilde{Q}_1$ such that $RQ_1 = \widetilde{Q}_1 \widetilde{R}$. Using the terminology of core-chasing algorithms, the computation of $\widetilde{Q}_1$ and $\widetilde{R}$ from Q_1 and R is a *passthrough* operation, and can be represented pictorially as



- Given the matrices G_i and K_i acting on the ith and $i+1$st indices, and J_{i+1} acting on the $i+1$st and the $i+2$nd indices, the product $G_i J_{i+1} K_i$ can be refactorized as $\widetilde{G}_{i+1} \widetilde{J}_i \widetilde{K}_{i+1}$, where $\widetilde{G}_{i+1}$ and $\widetilde{K}_{i+1}$ act on the $i+1$st and $i+2$nd indices, and $\widetilde{J}_i$ acts on the ith and $i+1$st indices. Similarly, this operation can be represented pictorially as



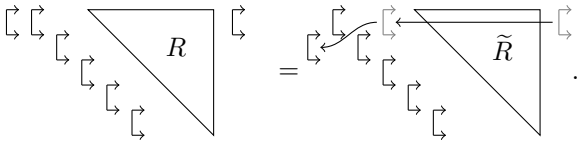
  In the context of core-chasing algorithms, it is more natural to reinterpret the refactorization above as moving the rightmost core transformation to the left, which can be displayed as



  Clearly, all the rotations involved in the step above change, but from the point of view of the structure, it is as if only one rotation had moved. This operation is called *turnover*.

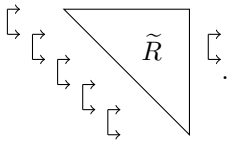
With this notation, we can rephrase the factorization obtained after introducing the core transformation Q_1 as

$$Q_1 H Q_1^* = \quad = \quad . \tag{5.5}$$



In the rightmost factorization, we have first fused the top-left rotations, and then used the *passthrough* and *turnover* to take the rotation that was on the right to the left. If we call this new core transformation Q_2, we can now perform the similarity transformation $Q_2 Q_1 H Q_1^* Q_2^*$ and obtain the matrix

$$Q_2 Q_1 H Q_1^* Q_2^* = \quad .$$



The structure of this matrix is similar to that of $Q_1 H Q_1^*$, in (5.5), but the rightmost core transformation has moved down one step. Indeed, it now acts on indices 2 and 3 instead of 1 and 2. Carrying on this process will move it further down, until it is fused at the bottom. At the very end, the core transformation will hit the bottom rotation in the sequence $G_1 \dots G_{n-1}$, and they will be fused together. This completes the chasing, and is mathematically equivalent to chasing the bulge into the bottom-right corner.

There are a few more technical details to address in order to obtain a complete algorithm. One key point is how to detect deflations, that is, eigenvalues that have converged. In the usual bulge-chasing setting, we monitor subdiagonal elements, setting them to zero as soon as they become "small enough". For core-chasing algorithms, we observe that a subdiagonal element is small if and only if the corresponding rotation in the sequence $G_1 \dots G_{n-1}$ is close to the identity. In fact, this technique is often more accurate than the customary criterion in practice [24].

The main computational step in this process is the refactorization $RQ_1 = \widetilde{Q}_1 \widetilde{R}$, which requires $\mathcal{O}(n)$ flops in general. Therefore, a single core-chasing step requires $\mathcal{O}(n^2)$ flops, and $\mathcal{O}(n^3)$ flops will be necessary to compute the eigenvalues of a generic Hessenberg matrix, if about $\mathcal{O}(n)$ steps are required for the QR iteration to converge. All the other operations require only $\mathcal{O}(1)$ flops, and contribute only a low-order term to the total cost.

However, if H is unitary to begin with, then so is the upper triangular matrix R. As upper triangular unitary matrices must be diagonal, the passthrough operation can be performed in $\mathcal{O}(1)$ flops, making the cost of the QR algorithm quadratic instead of cubic in n. For a more detailed analysis, we refer the reader to the paper where the algorithm was first introduced [20].

Algorithm 5.1: Sample the joint distribution of random matrices.

1 **function** SAMPLEEIGS($n \in \mathbb{N}, \mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}, \xi \in \mathbb{F}$)

Sample eigenvalues of orthogonal (if $\mathbb{F} = \mathbb{R}$) or unitary (if $\mathbb{F} = \mathbb{C}$) matrices. If $\xi \neq 0$, the determinant of the sampled matrices has the same phase as ξ. If $\xi = 0$, the matrices are sampled from $O(n)$ (if $\mathbb{F} = \mathbb{R}$) or $U(n)$ (if $\mathbb{F} = \mathbb{C}$).

2 $\quad \delta \leftarrow 1$
3 $\quad$ **for** $k \leftarrow 1$ **to** $n - 1$ **do**
4 $\quad\quad v_1 \sim \mathcal{N}_{\mathbb{F}}(0, 1)$
5 $\quad\quad v_2 \sim \sqrt{\chi^2_{\mathbb{R}}(n-k)}$
6 $\quad\quad d_k \leftarrow -\mathrm{e}^{\mathrm{i}\operatorname{Arg} v_1}$
7 $\quad\quad v_1 \leftarrow v_1 - d_k \|v\|_2$
8 $\quad\quad U \leftarrow \begin{bmatrix} \delta & \\ & 1 \end{bmatrix} \left(I - \frac{2}{\|v\|_2} v v^*\right)$
9 $\quad\quad \varphi \leftarrow \mathrm{e}^{\mathrm{i}\operatorname{Arg} \overline{u_{21}}}$
10 $\quad\quad c_k \leftarrow \varphi\, u_{11}$
11 $\quad\quad s_k \leftarrow -\varphi\, u_{21}$
12 $\quad\quad d_k \leftarrow d_k(\overline{\varphi}\,|u_{11}|^2 + \varphi\, u_{21}^2)$
13 $\quad\quad \delta \leftarrow \varphi \det U$
14 **if** $\xi \neq 0$ **then**
15 $\quad d_n \leftarrow \mathrm{e}^{\mathrm{i}\operatorname{Arg}\xi - \mathrm{i}\operatorname{Arg}\left(\prod_{j=1}^{n-1} d_j\right)}$
16 **else**
17 $\quad z \leftarrow \mathcal{N}_{\mathbb{F}}(0, 1)$
18 $\quad d_n \leftarrow -\mathrm{e}^{\mathrm{i}\operatorname{Arg} z}$
19 **return** UNITARYQR(c, s, d)

5.3. Sampling the eigenvalues of random unitary and orthogonal matrices

The approach underlying the discussion in Sections 4 and 5 is summarized in Algorithm 5.1.

The function SAMPLEEIGS samples the joint distribution of orthogonal or unitary matrices from a specific distribution determined by the value of third parameter ξ. If ξ is 0, then the function samples the eigenvalues of Haar distributed matrices from the orthogonal group, if $\mathbb{F} = \mathbb{R}$, or from the unitary group, if $\mathbb{F} = \mathbb{C}$. If $\xi \neq 0$, the algorithm samples the eigenvalue distribution of matrices whose determinant has the same phase as ξ. We recall that these matrices form a group only if $\xi = 1$, in which case the algorithm samples the eigenvalue distribution of Haar-distributed matrices from the special orthogonal group $SO(n)$, if $\mathbb{F} = \mathbb{R}$, or special unitary group $SU(n)$, if $\mathbb{F} = \mathbb{C}$.

In order to achieve this, we note that for H in (5.4), we have that $\det H = \det D = \mathrm{e}^{\mathrm{i}\theta}$ for some $\theta \in (-\pi, \pi]$, since the determinant of plane rotations is 1. Therefore, once the first $n-1$ entries of D are chosen, it suffices to choose $d_n = \mathrm{e}^{\mathrm{i}\operatorname{Arg}\xi - \mathrm{i}\operatorname{Arg}\left(\prod_{j=1}^{n-1} d_j\right)}$, which ensures that $\det D = \mathrm{e}^{\mathrm{i}\operatorname{Arg}\xi}$. In the pseudocode, the function UNITARYQR denotes a call to the `eiscor` routine, which computes the eigenvalues of a product of plane rotations of the form (5.2).

The computational cost of the algorithm can be determined by noticing that each step of the for loop on line 3 requires only a constant number of operations, which implies that the whole preprocessing taking place between line 2 and line 18 requires only $\mathcal{O}(n)$ flops.

6. Experimental results

In this section we first validate the new algorithm experimentally, and then compare its performance with that of the naïve method for sampling the joint eigenvalues distribution of Haar-distributed unitary matrices. The experiments were run in MATLAB 9.8.0 (R2020a) Update 4 on a GNU/Linux machine equipped with an Intel Xeon E5-2640 v3 CPU running at 2.60GHz.

In our tests we compare the following implementations.

- `samplemat`, an algorithm that generates a Haar-distributed unitary matrix by calling the function UMULT in Algorithm 3.1 on the identity matrix, and then computes its eigenvalues by using the built-in MATLAB function `eig`.
- `sampleeig`, an implementation of Algorithm 5.1 that exploits the `eiscor` package to run the QR algorithm on the unitary matrix in factored form.

Our implementations of `samplemat` and `sampleeig` are available on GitHub.[2] For reproducibility, the repository also includes the scripts we used to run the tests reported here.

6.1. Unitary matrices

We start by considering the joint distribution of the eigenvalues of Haar-distributed matrices in $U(n)$ and $SU(n)$.

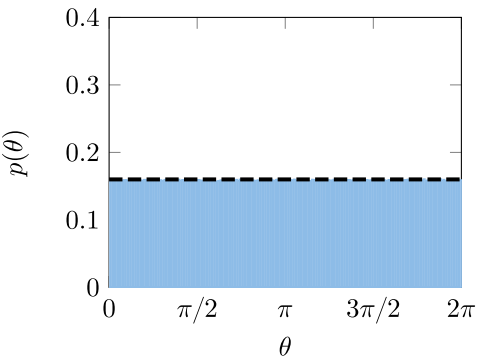
Fig. 1 reports the phase distribution and the spacing of the eigenvalues of 1,000,000 unitary matrices of order 10 sampled from the unitary group (top row) and from the special unitary group (bottom row) using `sampleeig`. The histograms in the four plots are normalized so that the total area of the columns is one. In this way, the histograms can be interpreted as empirical probability densities and can be compared directly with the probability density functions they are expected to match.

Let $e^{i\theta_1}, \ldots, e^{i\theta_n}$ be the eigenvalue of the matrix $A \in U(n)$ normalized so that for i between 1 and n the phase θ_i lies in the interval $[0, 2\pi)$. The histogram in Fig. 1a shows the distribution of the phases of the 10,000,000 sampled eigenvalues, whereas the dashed line indicates the probability density function of the uniform distribution over the interval $[0, 2\pi)$. As the eigenvalues of unitary matrices lie on the unit circle, our results show that the eigenvalues sampled by the procedure are uniformly distributed.

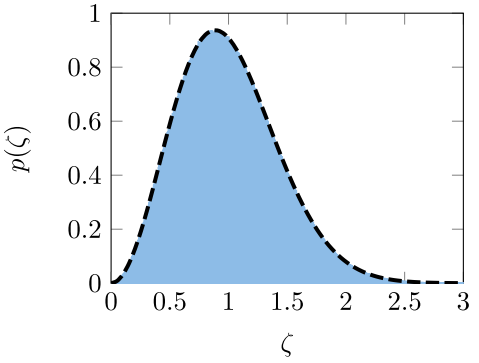
Next we investigate the statistical correlation among the eigenvalues sampled by `sampleeig`. In Fig. 1b we plot the probability density function of the normalized distance between pairs of consecutive eigenvalues, defined by

$$\zeta_i := \frac{n}{2\pi}(\theta_{i+1} - \theta_i), \qquad \theta_{n+1} := \theta_1, \qquad i = 1, \ldots, n,$$

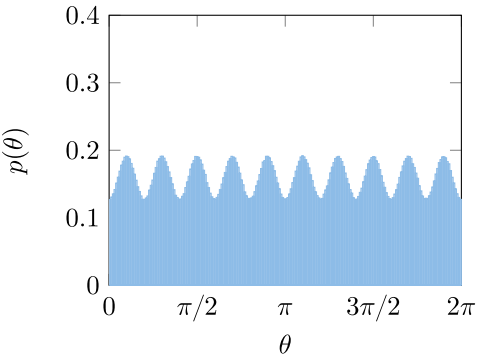
[2] https://github.com/numpi/random-unitary-matrices.



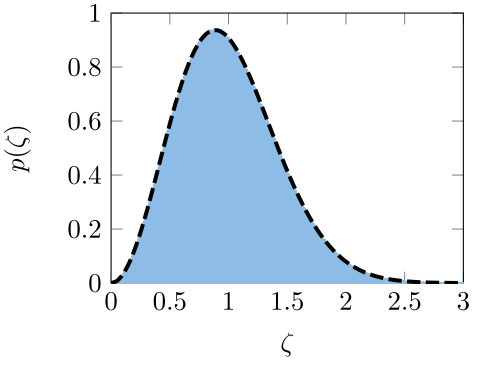
(a) Eigenvalue phase distribution for $U(n)$.



(b) Eigenvalue spacing distribution for $U(n)$.



(c) Eigenvalue phase distribution for $SU(n)$.



(d) Eigenvalue spacing distribution for $SU(n)$.

Fig. 1. Phase (left) and spacing (right) distribution of the eigenvalues of 1,000,000 random unitary matrix of order 10 sampled from the unitary group (top) and from the special unitary group (bottom) using `sampleeig`. The dashed lines represent the uniform distribution over the interval $[0, 2\pi)$ (left) and the Wigner surmise in (6.1) (right).

where the eigenvalues are ordered so that $\theta_1 \leqslant \cdots \leqslant \theta_n$. In this case the dashed line represents the theoretical spacing distribution of Haar-distributed unitary matrices, known as Wigner surmise [2, Section 1.5]

$$p(\zeta) = \frac{\pi\zeta}{2} \exp\left(-\frac{\pi}{4}\zeta^2\right). \tag{6.1}$$

The empirical distribution of the sampled eigenvalues matches closely the theoretical one, confirming that the matrices whose eigenvalues `sampleeig` samples are in fact Haar distributed.

To the best of our knowledge, the probability distribution for the phase and spacing of Haar-distributed matrices in the special unitary group are not known in closed form, but we can use `sampleeig` to obtain a relative frequency distribution based on 10,000,000 samples. The results in Fig. 1c and Fig. 1d suggest that the phase of the eigenvalues of these matrices is not uniformly distributed, but the spacing appears to be the same as

for Haar-distributed matrices in $U(n)$, as the empirical distribution matches the Wigner surmise in (6.1).

We note that the invariance under multiplication by elements in $SU(n)$ implies that Q and $\mathrm{diag}(\xi, \dots, \xi)Q$ must have the same distribution for any ξ such that $\xi^n = 1$, and the phase of the density of the eigenvalue distribution needs to be $2\pi/n$-periodic, as proven in Lemma 3.3. This is clearly visible in Fig. 1c.

6.2. Orthogonal matrices

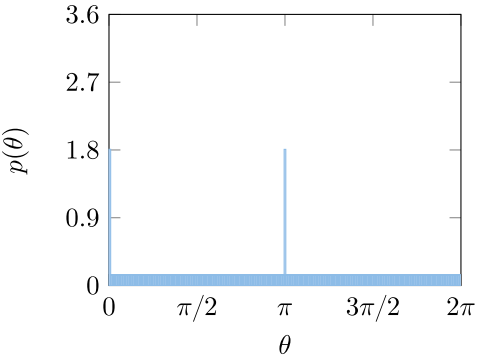
The joint eigenvalue probability density functions for $SO(n)$ and $O^-(n)$ are reported explicitly in [11, Section 2.6] and [37,38]. The corresponding joint eigenvalue distribution for the orthogonal group can be obtained easily, since a matrix in $O(n)$ belongs with equal probability to $SO(n)$ or $O^-(n)$. The eigenvalue distribution for such matrices can be obtained integrating out $n - 1$ variables and are defined in terms of the diagonal correlation kernel of the process [39]. However, excluding the case of unitary matrices, such expressions are not easy to evaluate; in most cases the limiting distribution for large n can be explicitly determined, and is typically the uniform distribution over $\mathbb{S}^1$.

We can use `sampleeig` to get the empirical distribution of the phase and spacing of the eigenvalues of these matrices. In Fig. 2, we report the relative frequency distribution of the phase and spacing of 1,000,000 random matrices of order 10 sampled from the orthogonal group (top row), from the special orthogonal group (middle row), and from the set of orthogonal matrices with determinant -1 (bottom row). In Fig. 3 we report the same data for matrices of order 9, as the behavior of these distributions changes dramatically depending on the parity of n.

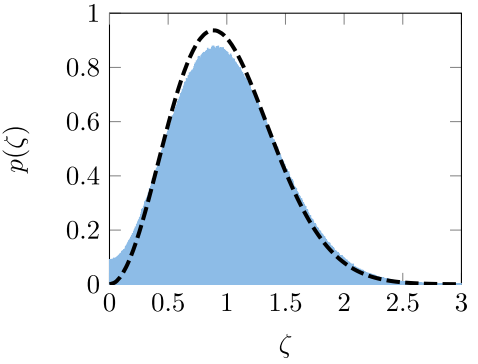
The distribution of phase and spacing for the eigenvalues of matrices sampled from $O(n)$ appears identical for both matrix dimensions we consider. In particular, we note that in Fig. 2a and Fig. 3a there is a mass of probability corresponding to the eigenvalues 1 and -1, which is a consequence of the fact that the eigenvalues of real matrices always appear in conjugate pairs. Therefore, if n is even then matrices with determinant -1 must always have both eigenvalues 1 and -1 (see Fig. 2e), whereas if n is odd then all matrices with determinant 1 must have the eigenvalue 1 (see Fig. 3c) and all those with determinant -1 must have the eigenvalue -1 (see Fig. 3e).

6.3. Timings and computational complexity

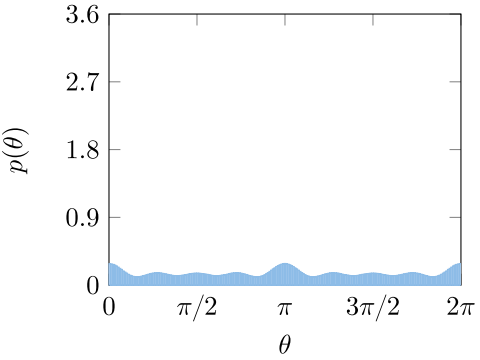
Now we compare the performance of our MATLAB implementations of `sampleeig` and `samplemat`. Fig. 4 shows the time, in seconds, required by the two algorithms to sample the eigenvalues of matrices of order n between 2 and 2^{15}. For matrices of order up to 16, `sampleeig` is slightly slower than `samplemat`; this is due to the fact that normalizing the rotations amounts to a large portion of the overall execution time of the algorithm.



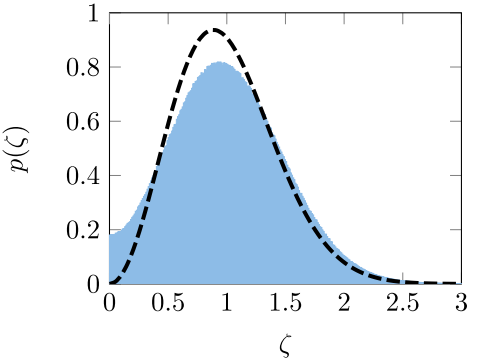
(a) Eigenvalue phase distribution for $O(n)$.



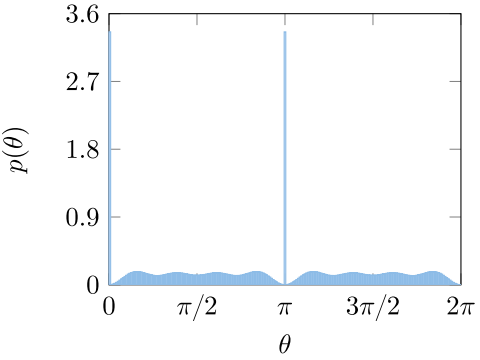
(b) Eigenvalue spacing distribution for $O(n)$.



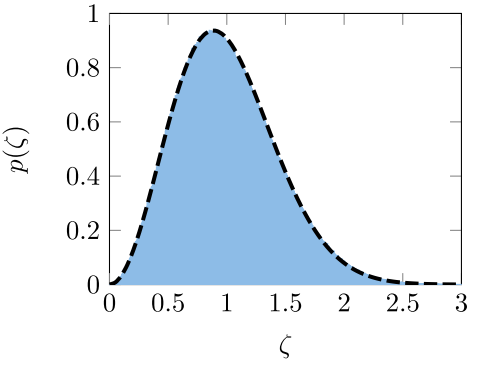
(c) Eigenvalue phase distribution for $SO(n)$.



(d) Eigenvalue spacing distribution for $SO(n)$.

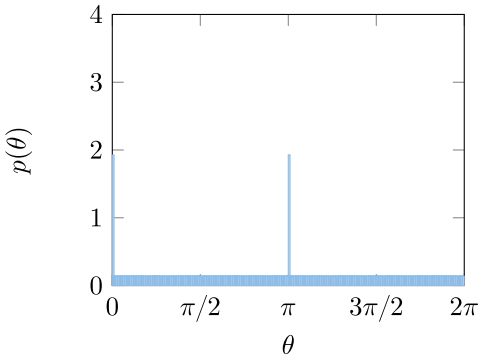


(e) Eigenvalue phase distribution for $O^-(n)$.

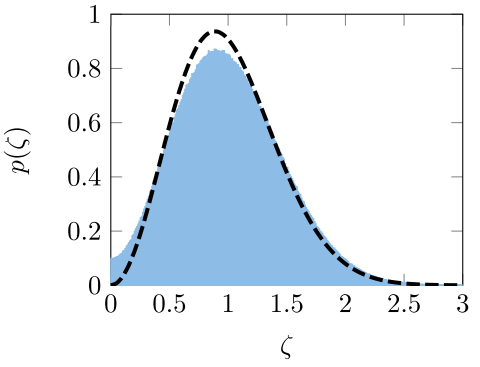


(f) Eigenvalue spacing distribution for $O^-(n)$.

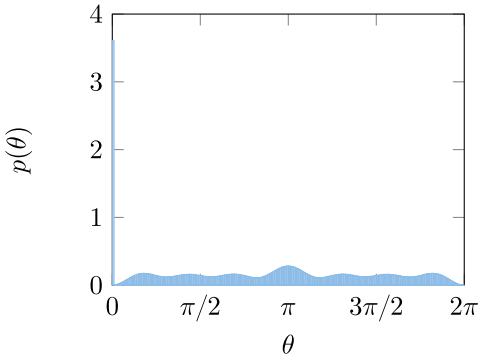
Fig. 2. Phase (left) and spacing (right) distribution of the eigenvalues of 1,000,000 random orthogonal matrix of order 10 sampled from the orthogonal group (top), the special orthogonal group (middle), and the connected component of the orthogonal group that contains only matrices with negative determinant (bottom) using `sampleeig`. The dashed lines in the right column represent the Wigner surmise in (6.1).



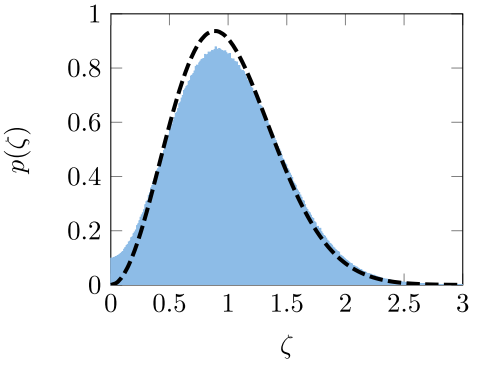
(a) Eigenvalue phase distribution for $O(n)$.



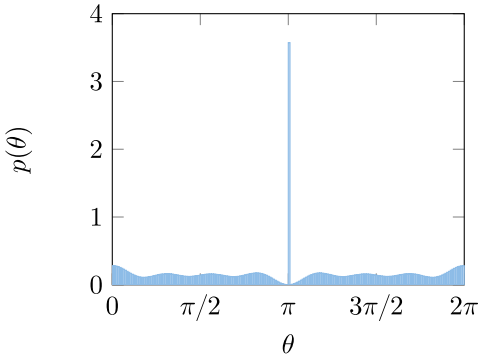
(b) Eigenvalue spacing distribution for $O(n)$.



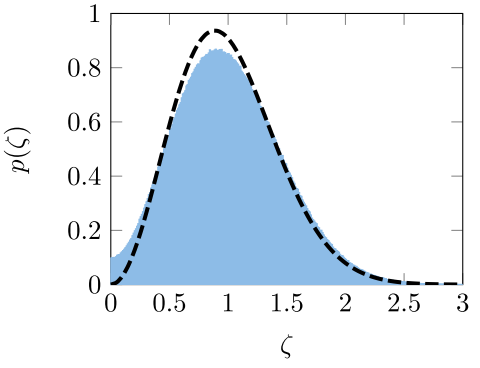
(c) Eigenvalue phase distribution for $SO(n)$.



(d) Eigenvalue spacing distribution for $SO(n)$.



(e) Eigenvalue phase distribution for $O^-(n)$.



(f) Eigenvalue spacing distribution for $O^-(n)$.

Fig. 3. Phase (left) and spacing (right) distribution of the eigenvalues of 1,000,000 random orthogonal matrix of order 9 sampled from the orthogonal group (top), the special orthogonal group (middle), and the connected component of the orthogonal group that contains only matrices with negative determinant (bottom) using `sampleeig`. The dashed lines in the right column represent the Wigner surmise in (6.1).

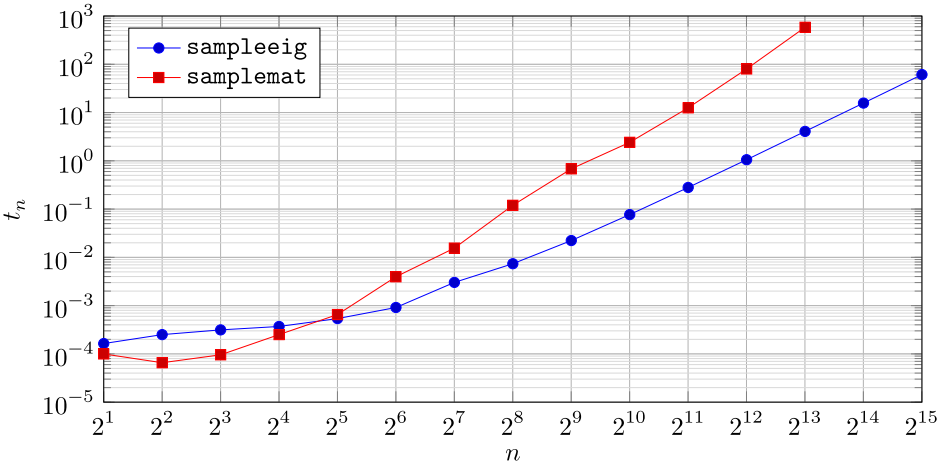


Fig. 4. Time t_n (in seconds) required by `sampleeig` and `samplemat` to sample the eigenvalues of matrices of order n between 2 and 2^{15}. The tests for `samplemat` have been performed only for n up to 2^{13}.

As the computational cost of this operation scales linearly, however, its contribution becomes negligible as n grows: for matrices of order 32 and above the execution time grows much faster for `samplemat` than for `sampleeig`. This is expected, since the two algorithms have cubic and quadratic computational cost, respectively.

7. Conclusions

We have presented a method for sampling the joint distribution of the eigenvalues of Haar-distributed orthogonal and unitary matrices. The two ingredients of our approach are a technique for sampling the upper Hessenberg form of Haar-distributed matrices, and an algorithm for computing the eigenvalues of an $n \times n$ upper Hessenberg unitary or orthogonal matrix in $\mathcal{O}(n^2)$ flops.

Our experimental results show that the new technique is more efficient than the naïve method that first samples a matrix from the Haar distribution and then computes its eigenspectrum numerically. We used this algorithm to investigate experimentally the distribution of the phase and spacing of the eigenvalues of Haar-distributed matrices from $SU(n)$, $O(n)$, $SO(n)$, and $O^-(n)$, groups for which these distributions are not known explicitly.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Preparation of the manuscript was carried out, in part, while the first author was a Visiting Fellow at the University of Pisa. The authors thank Alan Edelman for providing feedback on an early draft of the manuscript.

References

[1] J. Wishart, The generalised product moment distribution in samples from a normal multivariate population, Biometrika 20A (1/2) (1928) 32, https://doi.org/10.2307/2331939.
[2] M.L. Mehta (Ed.), Random Matrices, Pure and Applied Mathematics, vol. 142, Elsevier Academic Press, San Diego, CA, USA, 2004.
[3] Special Issue: Random Matrix Theory, J. Phys. A, Math. Gen. 36 (12) (2003), https://iopscience.iop.org/issue/0305-4470/36/12.
[4] E.P. Wigner, On the statistical distribution of the widths and spacings of nuclear resonance levels, Proc. Camb. Philos. Soc. 47 (4) (1951) 790–798, https://doi.org/10.1017/s0305004100027237.
[5] R. Couillet, M. Debbah, Random Matrix Methods for Wireless Communications, Cambridge University Press, Cambridge, UK, 2011.
[6] K. Rajan, L.F. Abbott, Eigenvalue spectra of random matrices for neural networks, Phys. Rev. Lett. 97 (18) (2006), https://doi.org/10.1103/physrevlett.97.188104.
[7] A. Edelman, N.R. Rao, Random matrix theory, Acta Numer. 14 (2005) 233–297, https://doi.org/10.1017/s0962492904000236.
[8] J. von Neumann, H.H. Goldstine, Numerical inverting of matrices of high order, Bull. Am. Math. Soc. 53 (11) (1947) 1021–1100, https://doi.org/10.1090/s0002-9904-1947-08909-6.
[9] F. Mezzadri, N.C. Snaith (Eds.), Recent Perspectives in Random Matrix Theory and Number Theory, London Mathematical Society Lecture Note Series, Cambridge University Press, 2005.
[10] Random Matrices: the First 90 Years, J. Phys. A, Math. Theor. 52 (43) (2018), https://iopscience.iop.org/journal/1751-8121/page/Random-Matrices.
[11] P.J. Forrester, Log-Gases and Random Matrices, London Mathematical Society Monographs, vol. 34, Princeton University Press, Princeton, NJ, USA, 2010.
[12] A. Edelman, Y. Wang, Random matrix theory and its innovative applications, in: R. Melnik, I.S. Kotsireas (Eds.), Advances in Applied Mathematics, Modeling, and Computational Science, in: Fields Institute Communications, vol. 66, Springer-Verlag, Boston, 2013, pp. 91–116.
[13] Z. Füredi, J. Komlós, The eigenvalues of random symmetric matrices, Combinatorica 1 (3) (1981) 233–241, https://doi.org/10.1007/bf02579329.
[14] H.F. Trotter, Eigenvalue distributions of large Hermitian matrices: Wigner's semi-circle law and a theorem of Kac, Murdock, and Szegö, Adv. Math. 54 (1) (1984) 67–82, https://doi.org/10.1016/0001-8708(84)90037-9.
[15] A. Edelman, B. Sutton, Y. Wang, Random matrix theory, numerical computation and applications, in: Modern Aspects of Random Matrix Theory, in: Proceedings of Symposia in Applied Mathematics, vol. 72, 2014, pp. 53–82.
[16] B.C. Hall, Lie Groups, Lie Algebras, and Representations, Graduate Texts in Mathematics, Springer-Verlag, Cham, 2015.
[17] P.R. Halmos, Measure Theory, Graduate Texts in Mathematics, Springer-Verlag, New York, 1950.
[18] G.W. Stewart, The efficient generation of random orthogonal matrices with an application to condition estimators, SIAM J. Numer. Anal. 17 (3) (1980) 403–409, https://doi.org/10.1137/0717034.
[19] P. Diaconis, M. Shahshahani, The subgroup algorithm for generating uniform random variables, Probab. Eng. Inf. Sci. 1 (1) (1987) 15–32, https://doi.org/10.1017/s0269964800000255.
[20] J.L. Aurentz, T. Mach, R. Vandebril, D.S. Watkins, Fast and stable unitary QR algorithm, Electron. Trans. Numer. Anal. 44 (2015) 327–341, http://etna.mcs.kent.edu/vol.44.2015/pp327-341.dir/pp327-341.pdf.
[21] A. Edelman, N.K. Ure, MIT 18.338 term project: numerical experiments on circular ensembles and Jack polynomials with Julia, http://web.mit.edu/~18.338/www/2013s/projects/ku_report.pdf, 2013.
[22] W.B. Gragg, The QR algorithm for unitary Hessenberg matrices, J. Comput. Appl. Math. 16 (1) (1986) 1–8, https://doi.org/10.1016/0377-0427(86)90169-x.

[23] M.J. Cantero, L. Moral, L. Velázquez, Five-diagonal matrices and zeros of orthogonal polynomials on the unit circle, Linear Algebra Appl. 362 (2003) 29–56, https://doi.org/10.1016/S0024-3795(02)00457-3.
[24] J.L. Aurentz, T. Mach, L. Robol, R. Vandebril, D.S. Watkins, Core-Chasing Algorithms for the Eigenvalue Problem, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2018.
[25] J.L. Aurentz, T. Mach, L. Robol, R. Vandebril, D.S. Watkins, Fast and backward stable computation of roots of polynomials, part II: backward error analysis; companion matrix and companion pencil, SIAM J. Matrix Anal. Appl. 39 (3) (2018) 1245–1269, https://doi.org/10.1137/17m1152802.
[26] R. Bevilacqua, G.M. Del Corso, L. Gemignani, A CMV-based eigensolver for companion matrices, SIAM J. Matrix Anal. Appl. 36 (3) (2015) 1046–1068, https://doi.org/10.1137/140978065.
[27] R. Bevilacqua, G.M. Del Corso, L. Gemignani, Compression of unitary rank-structured matrices to CMV-like shape with an application to polynomial rootfinding, J. Comput. Appl. Math. 278 (2015) 326–335, https://doi.org/10.1016/j.cam.2014.09.023.
[28] L. Gemignani, L. Robol, Fast Hessenberg reduction of some rank structured matrices, SIAM J. Matrix Anal. Appl. 38 (2) (2017) 574–598, https://doi.org/10.1137/16m1107851.
[29] G.H. Golub, C.F. Van Loan, Matrix Computations, 4th edition, Johns Hopkins University Press, Baltimore, MD, USA, 2013.
[30] L.N. Trefethen, D. Bau III, Numerical Linear Algebra, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.
[31] G. Birkhoff, S. Gulati, Isotropic distributions of test matrices, Z. Angew. Math. Phys. 30 (2) (1979) 148–158, https://doi.org/10.1007/bf01601929.
[32] F. Mezzadri, How to generate random matrices from the classical compact groups, Not. Am. Math. Soc. 54 (5) (2007) 592–604, http://www.ams.org/notices/200705/fea-mezzadri-web.pdf.
[33] G.S. Ammar, L. Reichel, D.C. Sorensen, An implementation of a divide and conquer algorithm for the unitary eigen problem, ACM Trans. Math. Softw. 18 (3) (1992) 292–307, https://doi.org/10.1145/131766.131770.
[34] L. Gemignani, A unitary Hessenberg QR-based algorithm via semiseparable matrices, J. Comput. Appl. Math. 184 (2) (2005) 505–517, https://doi.org/10.1016/j.cam.2005.01.024.
[35] W.B. Gragg, L. Reichel, A divide and conquer method for unitary and orthogonal eigenproblems, Numer. Math. 57 (1) (1990) 695–718, https://doi.org/10.1007/bf01386438.
[36] D.S. Watkins, The Matrix Eigenvalue Problem, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.
[37] A. Soshnikov, Determinantal random point fields, Russ. Math. Surv. 55 (5) (2000) 923–975, https://doi.org/10.1070/rm2000v055n05abeh000321.
[38] H. Weyl, The Classical Groups: Their Invariants and Representations, 2nd edition, Princeton University Press, 1946.
[39] P. Diaconis, M. Shahshahani, On the eigenvalues of random matrices, J. Appl. Probab. 31 (1994) 49–62, https://doi.org/10.1017/S0021900200106989.