

Randomness in Quantum Information and Many-Body Physics

Xhek Turkeshi^{*} and Markus Heinrich[†]

Institute for Theoretical Physics, University of Cologne, ZùlpicherstraÙe 77a, 50937 Kùln,
Germany

^{*} xturkesh@uni-koeln.de , [†] markus.heinrich@uni-koeln.de

Abstract

These lecture notes develop a self-contained route from Haar-random unitaries to quantitative diagnostics of randomness in quantum many-body systems. The first part introduces permutation representations, graphical replica calculus, Weingarten integration, and unitary designs. The second turns these tools into classical-shadow estimators and moment-operator bounds for local random circuits. The final part studies anticoncentration, random matrix-product states, magic, entanglement growth, and out-of-time-ordered correlators. The presentation targets Master's students with introductory quantum mechanics and linear analysis as prerequisites; intermediate algebra, regimes of validity, graphical derivations, and graded exercises are included throughout.

Revision convention. Text, formulas, headings, and exercises added or substantially rewritten in this edition are printed in red. Unchanged material from the working manuscript remains black.

Contents

1	About this course	3
1.1	How to use these notes	3
1.2	Global notation	3
2	Quantum Randomness I	4
2.1	What is a Haar measure?	4
2.2	Permutations and their combinatorics	7
2.2.1	Permutations and Cycles	7
2.2.2	The Action of Permutations on Quantum States	13
2.3	Weingarten calculus	17
2.4	Approximating the Haar measure: Unitary designs	20
2.5	The Clifford group	22
2.6	Exercises	24
2.7	Supplementary exercises: permutations and replica calculus	24
2.8	Further reading	26
2.9	Cumulative exercises: Quantum Randomness I	26
3	Measuring Properties using Classical Shadows	28
3.1	Reconstructing quantum states from measurements	28
3.2	Shadow estimation with randomized measurements	30
3.2.1	Clifford measurements	32
3.2.2	Pauli measurements	37
3.2.3	Beyond linear observables: estimating purity	38

permutation π . For instance, $U^{\otimes 2}$ clearly commutes with a swap of the two copies. Hence, the permutation operators R_π are contained in the k -th fold unitary commutant. These permutation operators form what is called a *representation* of the *symmetric group* S_k . A powerful result from representation theory – the *Schur-Weyl duality* – then tells us that the groups $U(d)$ and S_k are *dual to each other* which, among others, implies that every element in the commutant Comm_k can be written as a linear combination of permutation operators R_π :

Theorem 2.2 (Schur-Weyl duality). *The k -fold unitary commutant Comm_k is spanned by $\{R_\sigma \mid \sigma \in S_k\}$. Vice versa, the commutant of $\{R_\sigma \mid \sigma \in S_k\}$ is spanned by $\{U^{\otimes k} \mid U \in U(d)\}$*

We will not seek a deeper understanding of Schur-Weyl duality in this course but merely take this result as a motivation to develop techniques to evaluate $M_k(A)$. In particular, Schur-Weyl duality allows us to express $M_k(A)$ as a linear combination over permutations, with coefficients determined by their combinatorics. This leads to the framework known as *Weingarten calculus*, which enables the exact evaluation of many relevant integrals over the unitary group. As a result, permutations will play a key role throughout this course – not for abstract mathematical reasons, but because they offer a concrete and computable handle on random quantum processes.

2.2 Permutations and their combinatorics

In this section, we will introduce the essential properties of permutations needed for our purposes. While there is a rich and beautiful mathematical structure behind these ideas, we will focus only on the aspects that are directly relevant for our discussion and applications.

2.2.1 Permutations and Cycles

A *permutation* is a reordering of a finite set of elements. In this course, we consider permutations of the set $\{1, 2, \dots, k\}$. We will denote permutations by Greek letters such as π, σ, τ , and so on. The set of all permutations of k elements forms a group under composition, called the *symmetric group*, and denoted by S_k .

Given a permutation $\sigma \in S_k$ and an element $x \in \{1, \dots, k\}$, we write $\sigma(x)$ to indicate the image of x under σ . A common and explicit way to write a permutation is the *two-line notation*, where the first row lists the original elements and the second row gives their images under the permutation:

$$\sigma = \begin{pmatrix} 1 & 2 & \dots & k \\ \sigma(1) & \sigma(2) & \dots & \sigma(k) \end{pmatrix}. \quad (19)$$

Example 2.2: Explicit Notation of Permutations

An example of a permutation of four elements is:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}. \quad (20)$$

This means that the permutation acts as:

$$\sigma(1) = 3, \quad \sigma(2) = 1, \quad \sigma(3) = 4, \quad \sigma(4) = 2. \quad (21)$$

The group operation in S_k is the composition of permutations, denoted by $\sigma \cdot \tau$ for $\sigma, \tau \in S_k$, and defined by

$$(\sigma \cdot \tau)(x) = \sigma(\tau(x)) \quad \text{for all } x \in \{1, \dots, k\}. \quad (22)$$

Note that permutation composition is applied from right to left: τ acts first, followed by σ .

Example 2.3: Product of Permutations

Consider the following two permutations of $k = 4$ elements:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}, \quad \tau = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 3 & 1 \end{pmatrix}. \quad (23)$$

To compute the composition $(\tau \cdot \sigma)(x) = \tau(\sigma(x))$, we apply σ first and then τ :

$$\begin{aligned} \sigma(1) &= 3, & \tau(\sigma(1)) &= \tau(3) = 3, \\ \sigma(2) &= 1, & \tau(\sigma(2)) &= \tau(1) = 4, \\ \sigma(3) &= 4, & \tau(\sigma(3)) &= \tau(4) = 1, \\ \sigma(4) &= 2, & \tau(\sigma(4)) &= \tau(2) = 2. \end{aligned} \quad (24)$$

Putting everything together, we find:

$$\tau \cdot \sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 \end{pmatrix}. \quad (25)$$

It is important to emphasize that for $k \geq 3$, the symmetric group S_k is *non-abelian*, meaning that the order of composition matters – in general, $\sigma \cdot \tau \neq \tau \cdot \sigma$.

Exercise 2.1. Consider the permutations from Example 2.3. Compute the product $\sigma \cdot \tau$, and verify that it differs from $\tau \cdot \sigma$.

The symmetric group contains a special element called the *identity permutation*, denoted by ι , which leaves all elements unchanged:

$$\iota = \begin{pmatrix} 1 & 2 & \dots & k \\ 1 & 2 & \dots & k \end{pmatrix}. \quad (26)$$

By definition, composition with the identity does not change the permutation:

$$\iota \cdot \sigma = \sigma = \sigma \cdot \iota. \quad (27)$$

Moreover, every permutation $\sigma \in S_k$ has an *inverse* σ^{-1} such that:

$$\sigma \cdot \sigma^{-1} = \iota = \sigma^{-1} \cdot \sigma. \quad (28)$$

To compute the inverse of a permutation π , apply the rule

$$\pi^{-1}(\pi(i)) = \iota(i) = i, \quad i \in \{1, 2, \dots, k\}. \quad (29)$$

The resulting list π^{-1} is the inverse permutation.

Example 2.4: Inverse of a Permutation

Consider the following two permutations of $k = 5$ elements:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 1 & 2 & 4 \end{pmatrix}. \quad (30)$$

To compute the inverse σ^{-1} we use the rule $\sigma^{-1}(\sigma(i)) \equiv i$

$$\begin{aligned} \sigma(1) = 3 &\Rightarrow \sigma^{-1}(3) = 1 \\ \sigma(2) = 5 &\Rightarrow \sigma^{-1}(5) = 2 \\ \sigma(3) = 1 &\Rightarrow \sigma^{-1}(1) = 3 \\ \sigma(4) = 2 &\Rightarrow \sigma^{-1}(2) = 4 \\ \sigma(5) = 4 &\Rightarrow \sigma^{-1}(4) = 5 \end{aligned} \quad (31)$$

Reordering the list in terms of the argument, we find

$$\sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 5 & 2 \end{pmatrix}. \quad (32)$$

A *cyclic permutation*, or simply a *cycle*, is a specific type of permutation in which a subset of elements is permuted in a closed loop, while all remaining elements remain fixed. Formally, an l -cycle is a permutation that permutes l elements cyclically and leaves the other $k - l$ elements unchanged. The number l is called the *length of the cycle*.

Concretely, a cycle of length l means that there exists a subset $\{i_1, i_2, \dots, i_l\} \subset \{1, 2, \dots, k\}$ such that

$$\sigma(i_1) = i_2, \quad \sigma(i_2) = i_3, \quad \dots, \quad \sigma(i_{l-1}) = i_l, \quad \sigma(i_l) = i_1, \quad (33)$$

and for all other elements $x \notin \{i_1, \dots, i_l\}$, we have $\sigma(x) = x$.

Exercise 2.2 (Cyclic permutations). *The following are examples of cyclic permutations:*

$$\tau = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix}, \quad \sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 2 & 1 & 3 & 5 \end{pmatrix}. \quad (34)$$

Can you identify the subset $\{a_1, \dots, a_l\}$ that is cyclically permuted in each case? What is the length l of each cycle? Write both permutations in cycle notation, including the fixed points.

One fundamental property of cycles is that any permutation can be decomposed into a product of disjoint cycles. That is, for any $\sigma \in S_k$, there exists a unique set of cycles that act on mutually disjoint subsets of $\{1, 2, \dots, k\}$. This decomposition is unique up to the order in which the cycles are written. This motivates the *cycle notation* of permutations:

$$\sigma = (a_1 a_2 \dots a_{j_1})(a_{j_1+1} a_{j_1+2} \dots a_{j_2}) \dots (a_{j_{r-1}+1} a_{j_{r-1}+2} \dots a_{j_r}). \quad (35)$$

Here, each tuple represents a cycle, and all elements a_m are drawn from $\{1, 2, \dots, k\}$ without repetition. The integer $r = \#(\sigma)$ denotes the *number of disjoint cycles* in the decomposition of σ .

Let us now describe an explicit algorithm to obtain the cycle decomposition of a permutation, as in Eq. (35). The idea is simple: we iteratively follow the action of the permutation until we return to the starting point, keeping track of all visited elements.

- (i) **Start from the smallest unvisited element.** Initially, this is $x = 1$. If 1 has already been included in a previous cycle, move to the next smallest unvisited element.

- (ii) **Construct a cycle by iterating the permutation.** Begin by writing down x . Then repeatedly apply the permutation π to generate the sequence

$$x, \pi(x), \pi(\pi(x)), \pi(\pi(\pi(x))), \dots$$

Continue this process until you return to the starting point x . The list of elements

$$(x \ \pi(x) \ \pi^2(x) \ \dots \ \pi^{j-1}(x)) \quad (36)$$

forms a cycle of length j . Mark all of these elements as visited.

- (iii) **Repeat the process.** Find the next smallest unvisited element and return to step (ii). Continue until all elements have been visited. The full cycle decomposition of π is then obtained by combining the individual cycles found in each iteration.

Example 2.5: Cycle decomposition

Consider the permutation

$$\tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 3 & 1 & 2 \end{pmatrix}. \quad (37)$$

We apply the cycle decomposition algorithm step by step:

- Start with 1:

$$1 \xrightarrow{\tau} 4 \xrightarrow{\tau} 1.$$

This gives the first cycle: $(1 \ 4)$. Mark 1 and 4 as visited.

- Next smallest unvisited element is 2:

$$2 \xrightarrow{\tau} 5 \xrightarrow{\tau} 2.$$

This gives the second cycle: $(2 \ 5)$. Mark 2 and 5 as visited.

- The last unvisited element is 3, and since

$$3 \xrightarrow{\tau} 3,$$

this is a fixed point (a 1-cycle), written as (3) .

Combining the above, the full cycle decomposition is:

$$\tau = (1 \ 4)(2 \ 5)(3). \quad (38)$$

Note that the *order* in which disjoint cycles are written is irrelevant. For instance, in Example 2.5, all of the following represent the same permutation:

$$\tau = (1 \ 4)(2 \ 5)(3) = (2 \ 5)(1 \ 4)(3) = (3)(1 \ 4)(2 \ 5).$$

When the total number of elements k is clear from the context (e.g., $k = 5$ in this case), it is common to omit one-cycles, also denoted *fixed points*, because these elements are understood to remain unchanged under the permutation. Using this convention, the permutation in Example 2.5 is simply written as:

$$\tau = (1 \ 4)(2 \ 5). \quad (39)$$

With this notation, the identity permutation is denoted by $\iota = ()$, which is shorthand for $\iota = (1)(2)\cdots(k)$ – that is, all elements are fixed.

The *cycle structure* of a permutation, denoted by $\lambda(\pi)$, is the list of the lengths of its disjoint cycles. For example, the permutation $\tau = (1\ 4)(2\ 5)$ has cycle structure $\lambda(\tau) = (2, 2, 1)$. The number of disjoint cycles is then given by the length of this list:

$$\#(\pi) = |\lambda(\pi)|, \quad \text{for any } \pi \in S_k. \quad (40)$$

Example 2.6: Cycle Structure

Consider the following permutations of 6 elements

$$\tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 4 & 1 & 6 & 3 & 2 \end{pmatrix}, \quad \sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 2 & 6 & 4 & 1 & 3 \end{pmatrix}. \quad (41)$$

It is a simple exercise to show that their cycle decomposition is $\tau = (1, 5, 3)(2, 4, 6)$ and $\sigma = (1, 5)(3, 6)$ [which is a shorthand notation for $\sigma = (1, 5)(3, 6)(2)(4)$]. The permutation τ has two cycles of length 3, hence the cycle structure is $\lambda(\tau) = (3, 3)$. Instead, σ is composed of two 2-cycles and two 1-cycles, so the cycle structure is $\lambda(\sigma) = (2, 2, 1, 1)$.

Exercise 2.3. Show that conjugation preserves the cycle structure of a permutation. That is, for any $\sigma, \pi \in S_k$, prove that

$$\lambda(\pi\sigma\pi^{-1}) = \lambda(\sigma).$$

Transpositions, also known as *swaps*, are a special class of permutations that exchange exactly two elements and leave all others unchanged. A transposition has the form:

$$\sigma = (i\ j) = \begin{pmatrix} 1 & \cdots & i & \cdots & j & \cdots & k \\ 1 & \cdots & j & \cdots & i & \cdots & k \end{pmatrix}. \quad (42)$$

It is straightforward to verify that *any permutation* can be written as a product of transpositions. However, unlike the decomposition into disjoint cycles, this representation is *not unique* – a given permutation can be written in many different ways as a product of transpositions. While this makes the transposition decomposition less suitable for labeling permutations, it is extremely useful in algebraic manipulations. For instance, it turns out that the number of transpositions in any representation is always odd or always even, which justifies the definition of the *sign of a permutation*:

$$\text{sgn}(\sigma) = (-1)^{\#\text{transpositions in } \sigma}. \quad (43)$$

The sign function is important for the construction of representations of S_k and plays an important role in multilinear algebra, in particular in the definition of the *determinant of a matrix*.

We conclude this section by reviewing some structural aspects of the symmetric groups S_k for varying values of k . A key property is that the group S_k naturally embeds into S_{k+1} : that is, every permutation of k elements can be viewed as a permutation of $k + 1$ elements that leaves the $(k + 1)$ -th element fixed. More formally, we can write

$$S_{k+1} = S_k \sqcup \{(j\ k+1) \cdot \sigma : \sigma \in S_k, j = 1, 2, \dots, k\}, \quad (44)$$

where the union is disjoint and the second term represents all permutations obtained by composing an element of S_k with a transposition that swaps $k + 1$ with one of the first k elements.

This recursive structure is useful for establishing many properties of permutations by induction on k . A simple but important example is the total number of elements in the symmetric group.

Theorem 2.3 (Counting of Permutations). *Given $k \geq 1$, the total number of permutations in S_k is*

$$|S_k| = k!. \quad (45)$$

Proof. The total number of permutations is given by the factorial $k! = k(k-1) \cdots 2 \cdot 1$, with the convention that $0! = 1$. To prove this, we use induction.

For $k = 1$, the symmetric group $S_1 = \{\iota\}$ consists only of the identity permutation, so $|S_1| = 1 = 1!$.

Assume now that $|S_k| = k!$ for some $k \geq 1$. From Eq. (44), the next symmetric group can be written as

$$S_{k+1} = S_k \sqcup \{(j \ k+1) \cdot \sigma : \sigma \in S_k, j = 1, \dots, k\}. \quad (46)$$

There are k possible values of j , and for each j , σ runs over all $k!$ permutations in S_k . Hence,

$$|S_{k+1}| = |S_k| + k \cdot |S_k| = k! + k \cdot k! = (k+1) \cdot k! = (k+1)!. \quad (47)$$

This completes the proof by induction. $\square$

Consider now the following permutations of six elements:

$$\sigma_1 = (1 \ 2 \ 3)(5 \ 6), \quad \sigma_2 = (2 \ 4 \ 6 \ 1), \quad \sigma_3 = (1 \ 2)(3 \ 4)(5 \ 6). \quad (48)$$

While their cycle structures differ, all three permutations have exactly three disjoint cycles. This illustrates that simply counting the number of cycles is a coarser classification than specifying the full *cycle structure*.

In many computations throughout this course, we will be interested in the number of permutations in S_k with a fixed number of cycles $r = \#(\sigma)$. This quantity is given by the *unsigned Stirling numbers of the first kind*, denoted by $c(k, r)$. These numbers satisfy the recursive relation:

$$c(k+1, r) = k \cdot c(k, r) + c(k, r-1), \quad (49)$$

This recurrence computes the numbers inductively without listing all permutations. They form a triangle analogous to Pascal's triangle; the values through $k = 10$ are tabulated in Appendix A.1.

Starting with the base case $k = 1$, where $S_1 = \{()\}$, we find:

$$c(1, 0) = 0, \quad c(1, 1) = 1. \quad (50)$$

Using the recurrence, the next values for $k = 2$ are:

$$c(2, 0) = 0, \quad c(2, 1) = 1, \quad c(2, 2) = 1. \quad (51)$$

Exercise 2.4. Compute the values $c(k, r)$ for $1 \leq r \leq k$ when $k = 3$ and $k = 4$. Verify that:

$$\sum_{r=1}^k c(k, r) = k!, \quad \sum_{r=1}^k c(k, r) x^r = x(x+1) \cdots (x+k-1). \quad (52)$$

The first identity reflects the fact that summing over all permutations with a fixed number of cycles r recovers the total number of permutations in S_k , while the second gives another combinatorial interpretation of $c(k, r)$ as the coefficients in the power series of the 'rising factorial'.

2.2.2 The Action of Permutations on Quantum States

Throughout this course, we work with a d -dimensional Hilbert space $\mathcal{H} = \mathbb{C}^d$, with orthonormal basis $\{|x\rangle\}_{x=0}^{d-1}$. Its k -fold tensor power $\mathcal{H}^{\otimes k}$ is called the *replica space*. A permutation $\sigma \in S_k$ acts by moving the content of slot j to slot $\sigma(j)$:

$$R_\sigma |x_1, \dots, x_k\rangle = |x_{\sigma^{-1}(1)}, \dots, x_{\sigma^{-1}(k)}\rangle. \quad (53)$$

The inverse is essential. For example, if $\sigma = (1\ 4\ 3)(2)$, then $1 \mapsto 4$, $4 \mapsto 3$, and $3 \mapsto 1$; therefore

$$R_\sigma |x_1, x_2, x_3, x_4\rangle = |x_3, x_2, x_4, x_1\rangle. \quad (54)$$

With the usual convention $(\sigma\pi)(j) = \sigma(\pi(j))$, Eq. (53) defines a unitary representation:

$$R_{\sigma\pi} = R_\sigma R_\pi, \quad R_i = \mathbb{1}, \quad R_{\sigma^{-1}} = R_\sigma^{-1} = R_\sigma^\dagger. \quad (55)$$

Indeed,

$$R_\sigma R_\pi |x_1, \dots, x_k\rangle = |x_{\pi^{-1}(\sigma^{-1}(1))}, \dots, x_{\pi^{-1}(\sigma^{-1}(k))}\rangle \quad (56)$$

$$= |x_{(\sigma\pi)^{-1}(1)}, \dots, x_{(\sigma\pi)^{-1}(k)}\rangle = R_{\sigma\pi} |x_1, \dots, x_k\rangle. \quad (57)$$

This calculation also fixes the convention used in every later replica formula.

We leave the verification of the other axioms – such as unitarity and inverse consistency – as an exercise.

Exercise 2.5. Verify that R defines a unitary representation of the symmetric group S_k , i.e., check Eq. (55).

Exercise 2.6. Show that permutations act on tensor products of operators as follows:

$$R_\sigma (A_1 \otimes A_2 \otimes \dots \otimes A_k) R_\sigma^\dagger = A_{\sigma^{-1}(1)} \otimes A_{\sigma^{-1}(2)} \otimes \dots \otimes A_{\sigma^{-1}(k)}. \quad (58)$$

Hint: Apply both sides to a product basis state.

Composite Systems In practice, we often work with *multi-qudit systems* described by a Hilbert space of the form $\mathcal{H} = (\mathbb{C}^q)^{\otimes n}$, corresponding to n qudits of local dimension q . In this setting, the k -fold copy of the system is given by the tensor product

$$((\mathbb{C}^q)^{\otimes n})^{\otimes k}, \quad (59)$$

which can be naturally visualized as a $k \times n$ grid of qudits (see Fig. 1). Each row represents one replica of the full system, and each column represents the k copies of a single local qudit.

Quantum operations such as global unitaries typically act *row-wise*, that is, identically and independently on each copy. Such operations are of the form

$$U^{\otimes k}, \quad \text{where } U \in U((\mathbb{C}^q)^{\otimes n}), \quad (60)$$

meaning that the unitary acts in parallel across the k rows.

In contrast, permutations act by permuting the *rows*, i.e., the k copies of each local qudit. This operation is performed *column-wise*, and can be implemented by applying the same permutation operator to each column in parallel. This leads to a convenient factorized structure: if we reinterpret the total space via the isomorphism

$$((\mathbb{C}^q)^{\otimes n})^{\otimes k} \simeq ((\mathbb{C}^q)^{\otimes k})^{\otimes n}, \quad (61)$$

then the permutation operator R_π acting on the full system decomposes as

$$R_\pi = r_\pi^{\otimes n}, \quad (62)$$

Here r_π permutes the k replicas of one local qudit: it acts on $(\mathbb{C}^q)^{\otimes k}$, a space of dimension q^k (not dimension k). This “horizontal” factorization is exactly what is depicted in Fig. 1 and will be essential in constructing efficient representations of randomized operations throughout the course.