

RandomMeas.jl: A Julia Package for Randomized Measurements in Quantum Devices

Andreas Elben^{1,2} and Benoît Vermersch^{3,4}

¹PSI Center for Scientific Computing, Theory and Data, Paul Scherrer Institute, 5232 Villigen PSI, Switzerland

²ETH Zürich - PSI Quantum Computing Hub, Paul Scherrer Institute, 5232 Villigen PSI, Switzerland

³Univ. Grenoble Alpes, CNRS, LPMCC, 38000 Grenoble, France

⁴Quobly, 38000 Grenoble, France

We introduce `RandomMeas.jl`, a modular and high-performance open-source software package written in Julia for implementing and analyzing randomized measurement protocols in quantum computing. Randomized measurements provide a powerful framework for extracting properties of quantum states and processes such as expectation values, entanglement, and fidelities using simple experimental procedures combined with classical post-processing, most prominently via the classical shadow formalism. `RandomMeas.jl` covers the full randomized measurement workflow, from the generation of measurement settings for use on a quantum computer, the optional classical simulation of randomized measurements with tensor networks, to a suite of estimators for physical properties based on classical shadows. The package includes advanced features such as robust and shallow shadow techniques, batch estimators, and built-in statistical uncertainty estimation. Its unified, composable design enables the scalable application and further development of randomized measurements protocols across theoretical and experimental contexts.

1 Introduction

Randomized measurement (RM) protocols have emerged as a widely used experimental technique for extracting expectation values, entanglement properties, fidelities, and even full process information from quantum systems [1]. They offer a unifying strategy for characterizing many-qubit states using modest hardware resources [2]

and have been demonstrated across various platforms including trapped ions [3, 4, 5, 6], photonic devices [7, 8], and superconducting circuits [9, 10, 11, 12, 13, 14].

An RM experiment consists of sampling a set of random unitaries $\{U_\alpha\}_{\alpha=1}^{N_U}$ from a prescribed ensemble, applying each unitary to the quantum state of interest, and performing a projective measurement in the computational basis to obtain a bitstring $\mathbf{s} \in \{0, 1\}^N$ [15, 16, 17, 18]. Repeating this process N_M times per unitary results in a dataset of $N_U \times N_M$ bitstrings. These data are then analyzed classically to extract the physical properties of interest, often using the classical shadow formalism [2]. Crucially, the measurement protocol is independent of the final observable, enabling the guiding principle of RM: “*measure first, ask questions later*” [1].

While the data acquisition procedure is deliberately simple, efficiently processing the resulting large datasets—particularly for intermediate- and large-scale quantum systems—requires dedicated classical software. In parallel, recent years have witnessed a surge of theoretical advances in classical shadows and RM techniques, greatly enhancing the capabilities of RM-based methods. However, their classical software implementations are often developed independently, resulting in incompatible interfaces and code structures.

`RandomMeas.jl` addresses these challenges by offering a unified, modular, and high-performance JULIA package that supports the entire RM workflow, and takes advantage of the ITensors library `ITensors.jl` for performing tensor-network operations. The package includes sampling measurement settings, optionally simulating the measurement process via tensor networks (with decoherence), constructing various types of

classical shadows, applying robust error mitigation routines, and computing a range of statistical estimators with built-in uncertainty quantification. The package emphasizes standardization and modularity, making RM methods—including recent theoretical innovations—more accessible, reusable, and extensible for the broader quantum research community. The current release represents a first step, with further features planned in future updates, as outlined in the outlook section 5.

The package is modular by design. Its public API revolves around three core object types—*measurement settings*, *measurement data*, and *classical shadows*—that can be flexibly combined to implement and extend RM protocols. Advanced features such as robust shadows [19, 20, 10], shallow shadows [21, 22, 23], batch shadow estimation [24], and common randomized measurements [25] can be enabled via simple keyword switches.

Comparison to existing software. Several software libraries support aspects of randomized measurements (RM), each serving different goals and user groups. The `povm-toolbox` [26] for Qiskit enables local and global POVMs, including arbitrary informationally complete measurements, and provides basic classical shadow routines. PennyLane offers a tutorial-level implementation [27] and a `ClassicalShadow` class for random Pauli measurements, integrated into its broader quantum machine-learning framework and suited for small- to moderate-scale demonstrations. Mitiq incorporates primitives for robust classical shadows with calibration-based error mitigation and supports multiple quantum backends, making it a flexible platform for exploring error-mitigated RM, though simulations are generally restricted to $\lesssim 20$ qubits [28]. Other toolkits, such as `TensorCircuit` (a general-purpose tensor-network simulator) [29] and `Paddle Quantum` (educational demonstrations) [30], include basic classical shadow capabilities within broader simulation or teaching-focused contexts.

In the small-system regime where state-vector simulation is feasible (typically $\lesssim 20$ qubits), these packages overlap in functionality for standard Pauli classical shadows (and closely related local measurement schemes), and perfor-

mance depends strongly on the backend, batching strategy, and estimator workload. Beyond this, `RandomMeas.jl` also supports a broader class of RM protocols, including arbitrary local random-unitary (RU) ensembles, global RU measurements, and circuit-based shallow shadows. `RandomMeas.jl` is designed to support the *full RM workflow in a scalable, modular, and high-performance setting*. It provides tensor-network-based simulation and post-processing for RM protocols, enabling the simulation and analysis of large-scale systems beyond the reach of state-vector methods. It supports multiple classical shadow variants—including error-mitigated robust, shallow, and batch shadows—and provides efficient estimators for both linear and polynomial functionals with built-in statistical error quantification via resampling techniques. `RandomMeas.jl` is designed for seamless integration into both actual QPU experiments and large-scale classical simulations. Its emphasis on modularity, composability, and reproducibility makes it uniquely suited for developing, testing, and deploying advanced RM protocols in practice.

To facilitate practical tool selection, we provide benchmark scripts in the repository (for representative RM workloads) that record the execution environment (Julia version and package versions) and can be rerun across machines and problem parameters. In particular, `benchmarks/scaling_benchmarks.ipynb` and `benchmarks/example_benchmarks.jl` cover representative simulation and post-processing workloads (dense vs. factorized shadow construction, local Pauli expectation values, and reduced-density-matrix moments) and report both run time and memory footprints.

Article overview. This article is organized as follows. Section 2 introduces the core concepts behind randomized measurements and outlines the software architecture. Section 3 walks through a complete RM workflow, from setting generation to observable estimation. Subsequent sections present advanced features, including robust and shallow shadows, batch shadow estimation, and polynomial functional estimation, illustrated through a series of curated JUPYTER notebooks provided in the package’s `examples` folder. These notebooks are described in more detail in appendix A.

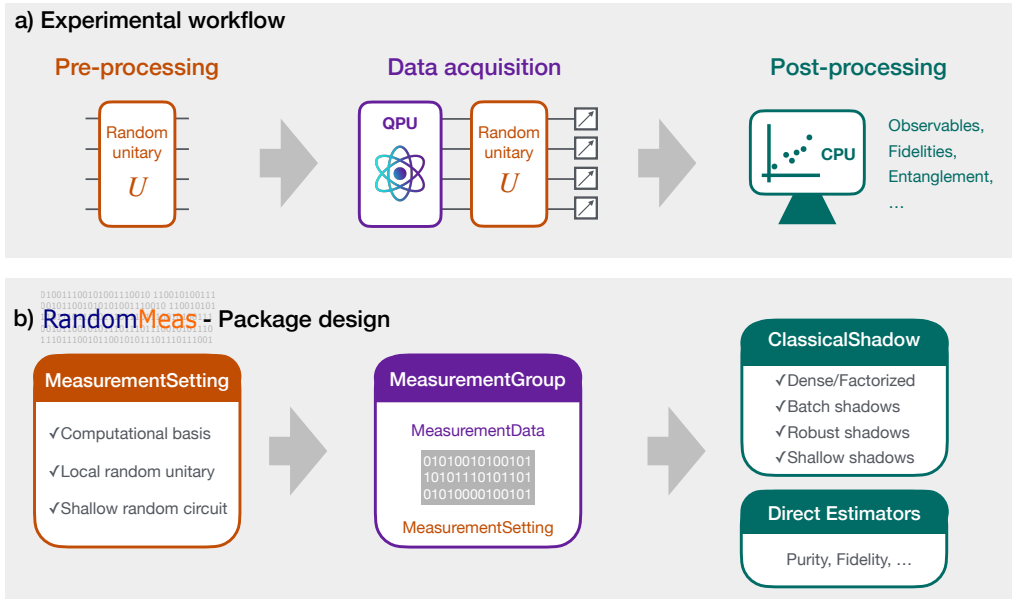


Figure 1: *Experimental RM workflow and corresponding organization of the package RandomMeas.jl.* (a) Pre-processing consists in sampling random unitaries, specifying randomized measurement settings, from a prescribed ensemble. During data acquisition random unitaries are implemented on the state (or process) of interest and projective measurements are performed. Classical post-processing predicts a wide range of quantum state (or quantum process) properties. (b) Within RandomMeas.jl, pre-processing, data acquisition and post-processing are separated in three independent sets of data structures. Data acquisition can be performed with a quantum device (quantum processing unit, QPU), or, alternatively, simulated classically on a CPU. For the latter, RandomMeas.jl provides tensor network routines based on ITensors.jl.

RandomMeas.jl is released under the Apache 2.0 license and available at <https://github.com/bvermersch/RandomMeas.jl>. The documentation can be found here <https://bvermersch.github.io/RandomMeas.jl/dev/>. The specific version corresponding to this paper is tagged with `v0.3.1_quantum_publication` and archived on Zenodo as Ref. [31].

2 Randomized-measurement workflow and package architecture

2.1 Randomized-measurement workflow

Randomized-measurement (RM) protocols allow one to access many observables and entropic quantities of a quantum state ρ while requiring only projective measurements in random bases [1]. A convenient way to describe an RM experiment, and, correspondingly, the design philosophy of RandomMeas.jl, is to split the procedure into three stages (see Fig. 1):

(i) Pre-processing. Measurement settings are specified by a random unitary U_{set} sampled from

a prescribed ensemble. The most common choice is a product of single-qubit rotations, $U_1 \otimes \cdots \otimes U_N$, but shallow multi-qubit circuits can be advantageous in specific scenarios. The total number of sampled settings is denoted N_U .

(ii) Data acquisition. On the quantum processor each unitary U_{set} is applied, followed by a computational-basis measurement. Repeating the measurement N_M times produces $N_U \times N_M$ bit strings $\mathbf{s} \in \{0,1\}^N$. The collection of outcomes and employed settings is called a *measurement group*.

(iii) Post-processing. The recorded bit strings, together with the employed measurement settings, are processed classically to estimate properties of the quantum state of interest. To this end, we can use the classical shadow formalism to define classical snapshots $\hat{\rho}$ that can, for instance, be used to estimate expectation values of observables O via $\hat{O} = \text{Tr}[O\hat{\rho}]$. The precise definition of classical shadows depends on the ensemble of measurement settings, typically one chooses them to be unbiased estimators of the density matrix, i.e., $\mathbb{E}[\hat{\rho}] = \rho$, where $\mathbb{E}$ is the expectation

value over both the random ensemble of measurement settings and the quantum mechanical average. With local measurement settings where each single qubit unitary U_i is sampled from a unitary 3-design [1]. Then, the classical shadow is defined as [2]

$$\hat{\rho} = \bigotimes_{i=1}^N (3 U_i^\dagger |s_i\rangle\langle s_i| U_i - \mathbb{1}). \quad (1)$$

where $\mathbf{s} = (s_1, \dots, s_N)$ is the measurement outcomes of the computational-basis measurement after applying U .

In special cases, for instance purity estimation $\text{tr}(\rho^2)$ [17] or cross-entropy fidelity estimation [32], one can bypass the shadow step and apply analytic estimators directly to the measurement group.

2.2 Package design

`RandomMeas.jl` mirrors the conceptual workflow directly in its software architecture by providing three families of data structures:

Measurement settings. The abstract type `MeasurementSetting` represents a single measurement setting. Its most frequently used subclass, `LocalMeasurementSetting`, covers the case of tensor-product unitaries, with concrete implementations `LocalUnitaryMeasurementSetting` (independently sampled single-qubit rotation per site from the Haar measure on $U(2)$ or X, Y, Z rotations) and `ComputationalBasisMeasurementSetting` (the identity on every site). The package also provides `ShallowUnitaryMeasurementSetting`, which stores an ordered list of one- and two-qubit gates that realize a shallow depth circuit to implement the measurement setting.

Measurement data. A `MeasurementData` instance holds the bit strings obtained from projective measurements performed in a single setting, together with the setting itself. Several such objects, collected over N_U different settings, are gathered into a `MeasurementGroup`. The library offers simulators that, given a tensor-network representation of a quantum state ρ , simulate the entire acquisition process and return a `MeasurementGroup` identical in form to experimental data.

Classical shadows. Post-processing routines evaluate `MeasurementGroup` objects, and evolve around the abstract type `AbstractShadow`. Three concrete formats for classical shadows are available. `FactorizedShadow` is memory-efficient and scales to essentially arbitrary N by storing vectors of 2×2 single-qubit classical shadows per site; `DenseShadow` stores a classical shadow as a full $2^N \times 2^N$ object. Since memory and basic linear-algebra costs scale exponentially in N (i.e. with 4^N matrix entries), this representation is primarily intended for small systems or small subsystems (typically N or $N_A \sim 10$ – 12 , depending on hardware and how many shadows/batches are kept in memory), where it can be faster than factorized shadows; finally `ShallowShadow` is tailored to measurement settings generated by shallow circuits. In the context of dense representations, `MeasurementProbability` objects contain, for each setting, the empirical Born distribution (as a 2^N dimensional normalized vector) extracted either from the raw bit strings (`MeasurementData`) or via simulations. A suite of estimator functions then acts on classical shadows or directly on `MeasurementGroup` data structures to compute fidelities, purities, overlaps, Rényi entropies, *etc.*

All tensor algebra is delegated to the highly optimised `ITensors.jl` library [33], which enables both efficient classical simulation of randomized measurements in large quantum systems and convenient manipulation of high-rank tensors that arise during post-processing.

2.3 Installing and usage

The `RandomMeas.jl` package can be easily installed using the Julia package manager. To install, simply execute the following command in the Julia REPL:

```
1 import Pkg
2 Pkg.add(RandomMeas)
```

To use `RandomMeas.jl`, one can then import the package and check its version.

```
1 using RandomMeas
2 RandomMeas.version()
```

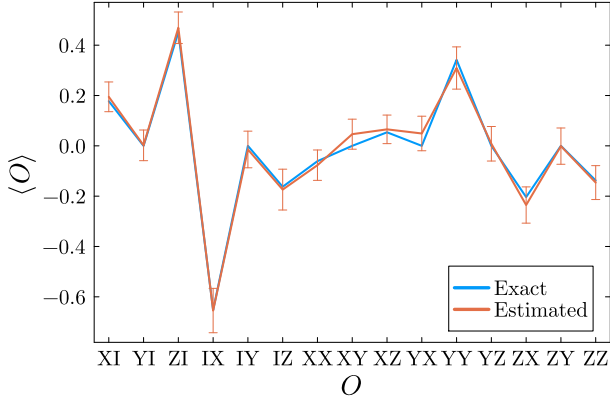


Figure 2: *Estimating expectation values with RandomMeas.jl.* Using a random MPS of $N = 50$ qubits, bond dimension 2, and randomized measurements with $N_U = 200$, $N_M = 100$, we extract all Pauli observables with support on sites 1 or 4, as described in Sec. 3. The errors bars are obtained following Sec. 3.4, and correspond throughout this manuscript to ± 2 estimated standard deviations.

3 Basic usage

This section (and the corresponding notebook) walks through one complete workflow of the package – from sampling random measurement settings, through collecting measurement data, to building classical shadows and estimating observables and non-linear functionals. In the last subsection, we show how to estimate the statistical accuracy via the Jackknife procedure.

3.1 Data acquisition

Randomized measurement settings. We start by importing the package and define the (total) system size of interest N . We now sample randomized measurement settings. Here, we parametrize them through single-qubit (local) random unitaries which are drawn for each qubit independently from the Haar ensemble on $U(2)$. We draw $N_U = 200$ settings.

```
1 N = 50 # Number of qubits
2 NU = 200 # Number of measurement settings
3 measurement_settings =
4     [LocalUnitaryMeasurementSetting(N) for _ in 1:
      NU]
```

Interfacing RandomMeas.jl with quantum experiments. As a next step, randomized measurements can be performed on actual quantum hardware. On the one hand, RandomMeas.jl

provides routines for exporting measurement settings. This includes interfaces to `qiskit` and, for `LocalUnitaryMeasurementSetting`, an OpenQASM export routine (QASM 2.0). These workflows are demonstrated in the example notebooks (see also App. A) and illustrate how to deploy the generated measurement circuits using external software frameworks. On the other hand, `qiskit` (or any other native quantum programming language) can be used to directly sample N_U random measurement settings and express them as parameterized quantum circuits (see the dedicated `qiskit` example notebooks, App. A).

After performing randomized measurements on quantum hardware, the measured bitstrings and the corresponding random unitaries can be post-processed using RandomMeas.jl, for instance by constructing a `MeasurementGroup` as:

```
1 basis_transfo = npzread("data/qiskit_u.npy")
2 meas = 2 .- npzread("data/qiskit_data.npy")
3 measurement_group = MeasurementGroup(meas,
    basis_transfo)
```

as detailed in the example notebooks (see also App. A).

Classical simulation of randomized measurements. Alternatively, one can simulate the entire RM protocol. Here, we choose a weakly entangled matrix product state (MPS) $|\psi\rangle$ with bond dimension 2.

```
1 site_indices = siteinds("Qubit",N)
2 ψ = random_mps(site_indices, linkdims=2) #
    Quantum state of interest
```

For each setting, we perform $N_M = 100$ projective measurements, yielding a “MeasurementGroup” container that stores all measurement outcomes and measurement settings used.

```
1 NM=100 # Number of projective measurements per
    setting
2 measurement_group = MeasurementGroup(ψ ,
    measurement_settings,NM)
```

This constructor also supports input states provided either as an MPS (pure state) or as an MPO (density operator). The optional keyword `mode` selects tensor-network sampling (default) or dense-array sampling (when feasible).

3.2 Classical shadows and estimation of expectation values

Building classical shadows The first steps of post-processing consist usually in building classi-

cal shadows [2]. Here, we first use "factorized" classical shadows that are memory efficient objects (stored as $N \times 2 \times 2$ arrays) suitable for large system sizes.

```
1 classical_shadows = get_factorized_shadows(
    measurement_group)
```

Expectation values. For estimating expectation values, we construct an observable

$$O = Z_1 \otimes I_2 \otimes I_3 \otimes X_4 \otimes I_5 \otimes \cdots \otimes I_N.$$

```
1 ops = ["I" for _ in 1:N]
2 ops[1] = "Z"
3 ops[4] = "X"
4 O=MPQ(site_indices,ops)
```

We compute the shadow estimate $\langle \widehat{O} \rangle$ and its standard error, and the exact value $\langle O \rangle = \langle \psi | O | \psi \rangle$.

```
1 estimate_val = get_expect_shadow(0,
    classical_shadows)
2 exact_val = inner(ψ',0,ψ)
```

Dense shadows for small subsystems.

When the observable acts only on a few qubits it is efficient to *restrict* the data to that subsystem and keep each shadow in its full matrix form. For a subsystem of N_A qubits a dense classical shadow is stored as a full $2^{N_A} \times 2^{N_A}$ object (i.e. 4^{N_A} entries). In practice, dense shadows are typically advantageous for small N_A (roughly $N_A \sim 10$ – 12 on a workstation, depending on batching and how many shadows are stored/processed at once); for larger N_A , factorized shadows are usually preferable due to the exponential memory *and* arithmetic cost. A reproducible timing/memory comparison underlying this guideline is provided in `benchmarks/scaling_benchmarks.ipynb`.

In our example the operator O has support on qubits 1 and 4 only. The three-step workflow is therefore: reduce the measurement data to the sites $\{1,4\}$; build dense shadows for that two-qubit subsystem; estimate $\langle O \rangle$ with those shadows.

```
1 reduced_group = reduce_to_subsystem(
    measurement_group, [1, 4])
2 dense_shadows = get_dense_shadows(reduced_group)
3 estimate_val = get_expect_shadow(
    reduce_to_subsystem(0,[1,4]), dense_shadows)
```

Both routes - factorized and dense - yield the same estimated value for $\langle O \rangle$; the dense-shadow routines are merely faster on smaller (sub-) systems.

An example of estimation of multiple Pauli strings from the same RM dataset is shown in Fig. 2.

3.3 Estimating non-linear functionals

Many physically relevant quantities—such as purities $p_2 = \text{tr}(\rho^2)$ or higher-order Rényi entropies—are polynomial functionals of the density matrix. Classical shadows provide a natural way to estimate such polynomial functionals via the so-called *U-statistics* formalism [2, 34]: to estimate, for instance, $\text{tr}(\rho^k)$, one replaces each factor in the trace product by a statistically independent classical shadow (obtained from independently sampled random unitaries) and averages over all distinct combinations [2, 34]. For $N_M = 1$, this yields

$$\hat{p}_k = \frac{(N_U - k)!}{N_U!} \sum_{\substack{j_1, \dots, j_k \\ \text{all distinct}}} \text{tr}[\hat{\rho}_{j_1} \hat{\rho}_{j_2} \cdots \hat{\rho}_{j_k}], \quad (2)$$

which is an unbiased estimator of the k -th moment $p_k = \text{tr}(\rho^k)$ for $k > 1$ integer.

For general $N_M > 1$, the combinatorics of the U-statistic become more involved, but `RandomMeas.jl` handles this general case. Thus, users can directly call `get_trace_moments(shadows, collect(2:5))` for arbitrary N_U and N_M without worrying about the underlying formulas.

Evaluating the U-statistic exactly scales as $\frac{(N_U - k)!}{N_U!} N_M^k$. It therefore becomes quickly impractical when N_U , N_M and k grow. A crucial practical simplification is obtained by introducing *batch shadows* [24]. Here the N_U unitaries are partitioned into N_B disjoint batches of size N_U/N_B , and one averages within each batch (including over all N_M outcomes), defining $\bar{\rho}_b = \frac{1}{(N_U/N_B)N_M} \sum_{j \in b} \sum_{l=1}^{N_M} \hat{\rho}_{j,l}$ where $\hat{\rho}_{j,l}$ is the classical shadow constructed from measurement l after the application of the random unitary j . The resulting N_B batch shadows are then combined according to the same U-statistic formula (2), but with N_U replaced by N_B , reducing the number of terms to $\frac{(N_U - k)!}{N_U!}$ while retaining the leading $1/N_U$ variance in the practically relevant regime $N_B \ll N_U$ [24].

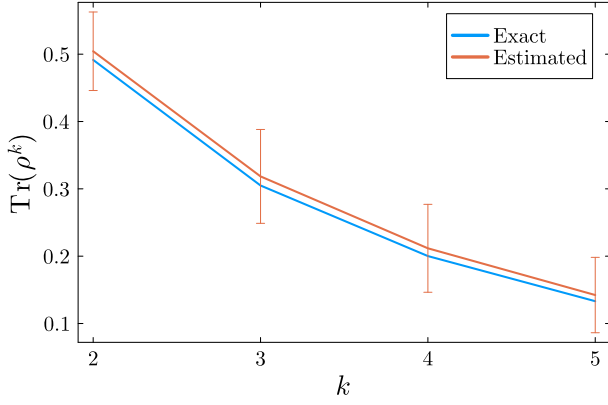


Figure 3: Estimation of trace moments $p_k = \text{tr}(\rho^k)$ using the same RM data as in Fig. 2, and using 8 batch shadows.

In `RandomMeas.jl`, this batching workflow is accomplished in two lines:

```
1 batch_shadows = get_dense_shadows(
    measurement_data; n_ru_batches = 8)
2 moments = get_trace_moments(batch_shadows,
    collect(2:5))
```

The first call forms $N_B = 8$ batch-averaged dense shadows, while the second evaluates the reduced U-statistic on them.

Another simplification of the post-processing consists in using estimation formulas that do not require building classical shadows but work directly with the obtained measurement data. This is the case in particular for the purity [17, 3], and cross-platform fidelity [35]. Our package provides the two functions `get_purity`, and `get_overlap` that take `MeasurementGroup` objects as inputs and return the purity and the state overlap $\text{tr}(\rho_1 \rho_2)$.

3.4 Quantifying statistical accuracy

Randomized-measurement protocols infer physical quantities from finite samples of random bases and finite numbers of projective read-outs. Any numerical estimate produced by `RandomMeas.jl` must therefore be accompanied by an *uncertainty* that reflects two independent sources of fluctuations, *ensemble noise* arising from the limited number N_U of sampled measurement settings; and *shot noise* originating from the finite number N_M of projective measurements per setting.

Linear functionals. For estimates that are *linear* in the state—e.g. an observable expectation

value $\text{tr}(O\rho)$ —the estimator is the sample mean of N_B (batch-)shadow evaluations. The routine `get_expect_shadow` (and analogues) accepts the keyword `compute_sem=true` and returns the value together with its standard error $\sigma = \sqrt{\text{Var}[\hat{O}]/N_B}$.

Polynomial functionals. Non-linear quantities such as purity $\text{tr}(\rho^2)$ of a quantum state ρ or higher trace moments admit unbiased *U-statistic* estimators but lack simple analytic variance formulae. `RandomMeas.jl` therefore implements the leave-one-out *jackknife* [36] to estimate the variance of the corresponding estimate:

1. From N_B (batch) shadows compute the U-statistic $\hat{\theta}$.
2. For each $i = 1, \dots, N_B$ omit shadow i and recompute $\hat{\theta}_{(i)}$ from the remaining $N_B - 1$ shadows.
3. The jackknife bias-corrected point estimate is $\theta_{\text{JK}} = N_B \hat{\theta} - (N_B - 1) \bar{\hat{\theta}}_{(\cdot)}$, with variance $\sigma_{\text{JK}}^2 = \frac{N_B - 1}{N_B} \sum_i (\hat{\theta}_{(i)} - \bar{\hat{\theta}}_{(\cdot)})^2$.

A direct implementation requires $\mathcal{O}(N_B)$ evaluations of the costly U-statistic. `RandomMeas.jl` avoids this overhead by caching all terms that enter the full estimator and re-aggregating them to obtain every $\hat{\theta}_{(i)}$ essentially for free; the run time therefore remains dominated by the single computation of the full U-statistic. It also provides functionality to estimate the full covariance matrix of multiple trace moments via jackknife.

Practical guidelines. Reliable uncertainties require a minimum of roughly ten (batch) shadows ($N_B \gtrsim 10$) [24]. Users are encouraged to check convergence of both the point estimate and the jackknife error bar as a function of N_B . We note that bootstrap resampling is another natural option for uncertainty quantification, but it is not currently implemented in `RandomMeas.jl`; we focus on jackknife as a deterministic and computationally efficient default.

All post-processing routines expose the keywords `compute_sem` / `compute_cov` to activate these statistical diagnostics; by default only the point estimate is returned.

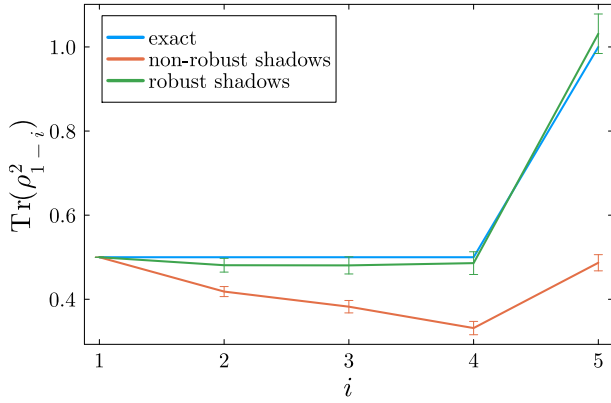


Figure 4: Robust shadow estimation of the purity of (sub-)systems of a 5 qubit GHZ state as a function of the subsystem size i . The measurement is affected by local depolarization noise of random strength of mean value 0.1 and standard deviation 0.02. We used $N_U = 2000$ unitaries for the calibration experiment and for the measurement ($N_M = 100$ for both).

4 Advanced usage of the package

In the section, we present examples of advanced usage of the toolbox. First we present two illustrative examples for performing classical shadow tomography: In presence of measurement noise (Sec. 4.1), and using shallow quantum circuits as measurements settings (Sec. 4.2). Then, in Sec. 4.4, we present a list of other possible advance uses of the toolbox, and connect them to jupyter notebooks that are available online (see also App. A).

4.1 Robust shadow tomography

Randomized-measurement protocols are appealing precisely because they are *hardware friendly*: in their simplest version, only local basis rotations and computational basis measurements are needed. In practice, however, during the basis rotations, read-out crosstalk and errors degrade the quality of the recorded bit strings. Classical shadow estimators that neglect those imperfections can be severely biased.

Twirling and effective noise. A key observation—formalised in Refs. [19, 20]—is that the ensemble average over random unitaries *twirls* the native noise channel into a simple, diagonal form¹. For local single-qubit rotations

¹Under the assumption of gate-independent, time-stationary and Markovian noise.

the effective noise on each qubit is fully characterized by a single *depolarization parameter* G_i (or, equivalently, an error probability p_i). If these parameters are known, one can *undo* the bias at the level of the shadow map² and recover an unbiased estimator.

Calibration from a reference experiment.

To measure *depolarization parameters*, we start from a known product state $|\psi_0\rangle$ (typically $|\psi_0\rangle = |0\rangle$) and acquire a measurement group `calibration_group`, which will be used to build robust shadows for a subsequent experiment performed on a unknown quantum state ρ .

Based on this data, `RandomMeas.jl` first constructs a calibration vector G [10] containing the per-qubit depolarization parameters, which can be then passed to any shadow-construction routine for the post-processing of the actual measurement data. (`get_factorized_shadows`, `get_dense_shadows`, ...).

```
1 G_e = get_calibration_vector( $\psi_0$ ,
                             calibration_group)
2 robust_shadows = get_factorized_shadows(
    measurement_data; G)
```

Because the correction is inserted at the earliest possible point (the definition of the classical shadow itself) `RandomMeas.jl` offers robust versions of *every* estimator implemented in the library.

Visual illustration. Figure 4 presents the standard and the robust estimators for the purity of a noisy GHZ state, as in the experimental work [10], see also the corresponding jupyter notebook. While standard, i.e. non-robust, classical shadows, underestimate the values of the purity, the robust version rescales the estimator to remove the effect of measurement errors.

4.2 Shallow shadows

Classical shadows constructed from local measurement settings are hardware-friendly and efficient for estimating local observables, but may yield large statistical errors when estimating

²All mitigation is performed *a posteriori* at the expense of a possibly enlarged sample complexity: no additional pulses are inserted into the experimental sequence.

high-weight operators³. An alternative, known as *shallow shadows*, replaces product random unitaries with low-depth random quantum circuits. This approach can substantially reduce estimator variance for high-weight observables [21, 22, 37, 23, 11].

Learning the inverse shadow map. In `RandomMeas.jl` shallow shadows are implemented using uni-dimensional random quantum circuits composed of single-qubit and nearest-neighbor two-qubit gates of fixed depth. In contrast to local-unitary schemes, the corresponding shadow map is not analytically known and must be learned numerically from training data. This is done during a training phase using a vector of `ShallowMeasurementSetting` instances.

```
1 NU_training = 2000 # Number of unitaries
2 N = 10
3 site_indices_i = siteinds("Qubit", N; addtags="input")
4 site_indices_o = siteinds("Qubit", N; addtags="output")
5 depth = 2
6 settings = [ShallowUnitaryMeasurementSetting(N,
    depth; site_indices=site_indices_o) for _ in 1:NU_training];
```

Here we use two instances of site indices, `site_indices_i`, `site_indices_o`, in order to describe the action of quantum channels on input and output states, respectively.

From the resulting data, the effective measurement channel is reconstructed as

$$\mathcal{M}(\rho) = \mathbb{E}_U \left[\sum_{\mathbf{s}} \langle \mathbf{s} | U \rho U^\dagger | \mathbf{s} \rangle U^\dagger | \mathbf{s} \rangle \langle \mathbf{s} | U \right], \quad (3)$$

where the average is taken over the shallow circuit ensemble.

Internally, $\mathcal{M}$ is represented as a depolarizing channel parametrized by a real-valued MPS. Using the sets of N_U unitaries defined through the variable `settings`, this MPS is obtained through a tensor-network fitting procedure [38], using the numerically evaluated Born probabilities generated from a fixed state $\rho_0 = |0\rangle\langle 0|$. In a second step, we compute the inverse map $\mathcal{M}^{-1}$ via automatic differentiation. More theoretical details are given in the corresponding Jupyter notebook.

³The number of measurements scale exponentially with the weight of the observables of interest [2].

Observable estimation. Once the inverse map is learned, it can be used to construct classical shadows from randomized measurements on unknown quantum states. Here, we simulate these randomized measurements classically using the built-in functionality of `RandomMeas.jl`. A new measurement group is collected using the same circuit depth:

```
1 measurement_group = MeasurementGroup(ψ, NU, NM;
    setting_type=
    ShallowUnitaryMeasurementSetting, depth=depth
    , mode=Dense);
```

Each measurement outcome $\mathbf{s}$, obtained from a shallow circuit U , is then mapped to a classical shadow via

$$\hat{\rho} = \mathcal{M}^{-1}(U^\dagger | \mathbf{s} \rangle \langle \mathbf{s} | U), \quad (4)$$

enabling unbiased estimation of expectation values. For instance, an observable O_j can be evaluated as

```
1 shadow = get_shallow_shadows(measurement_group,
    inverse_shallow_map, site_indices_i,
    site_indices_o)
2 estimate = real(get_expect_shadow(observable[j],
    shadow))
```

This procedure supports scalable estimation of many observables, and can significantly reduce variance for non-local observables compared to standard local-measurement shadows [37].

4.3 Benchmarking of quantum devices

Randomized measurement protocols also enable practical and scalable benchmarking of quantum processors. `RandomMeas.jl` includes built-in support for two widely used protocols: cross-entropy benchmarking (XEB) [39, 40] for comparing a (noisy) QPU against an ideal classical simulation and cross-platform fidelity estimation for comparing two QPUs [35]. These routines exploit efficient tensor-network simulation to benchmark realistic circuits and noisy dynamics at scale.

Cross-entropy benchmarking (XEB) and self-XEB. Cross-entropy benchmarking [39, 40] quantifies hardware fidelity by comparing the (empirical) probabilities of experimentally measured bitstrings to the ideal output probabilities $p(\mathbf{s}) = |\langle \mathbf{s} | U | \mathbf{0} \rangle|^2$ of a random quantum circuit U . Given a dataset of bitstrings $\{\mathbf{s}\}$, the XEB fidelity is computed as

$$\mathcal{F}_{\text{XEB}} = 2^N \cdot \mathbb{E}_{\mathbf{s} \sim \text{exp.}}[p(\mathbf{s})] - 1, \quad (5)$$

where the expectation is over experimental bit-strings and $p(\mathbf{s})$ is computed for each experimental measurement outcome $\mathbf{s}$ using a classical simulation of the ideal circuit. For shallow circuits, the output distribution can deviate significantly from the Porter–Thomas form, leading to a biased XEB estimator. The *self-XEB* protocol [13] (see also Refs. [41, 42]) corrects for this by normalizing with probabilities obtained directly from the reference circuit, yielding

$$\mathcal{F}_{\text{self-XEB}} = \frac{\mathbb{E}_{\mathbf{s} \sim \text{exp.}}[p(\mathbf{s})] - \bar{p}}{\mathbb{E}_{\mathbf{s} \sim \text{ref.}}[p(\mathbf{s})] - \bar{p}},$$

where $\bar{p} = 1/2^N$ is the uniform baseline.

Both XEB and self-XEB are supported directly from `MeasurementData` objects.

```
1 XEB = get_XEB(ψ, measurement_data)      #
   standard XEB
2 selfXEB = get_selfXEB(ψ)                # self-
   XEB reference
3 XEB_corrected = XEB / selfXEB            # bias-
   corrected estimate
```

Here, `get_XEB` computes the standard XEB estimator with measurement data from computational basis measurements on a noisy version of ψ . The second routine evaluates the self-XEB normalization for ψ . In this context, the package can simulate ideal and noisy dynamics using matrix-product-state and quantum-trajectory methods [43].

4.4 Jupyter notebooks for advanced usage

To illustrate the use of `RandomMeas.jl` in realistic scenarios, we provide a collection of 14 Jupyter notebooks, listed in App. A. They cover a broad range of protocols and use cases, from classical-shadow tomography (including robust and shallow-circuit variants) and quantum-device benchmarking (e.g., cross-entropy benchmarking, cross-platform fidelity estimation) to entanglement-entropy measurements and topological-order detection. Several notebooks reproduce or reanalyse experimental data, such as the ion-trap experiment of Ref. [3], while others simulate noisy quantum circuits or demonstrate error-mitigation techniques such as virtual distillation. Several notebooks cover importing and exporting measurement settings and results to interface with quantum computing experiments using languages such as `qiskit` or `QASM`. Each notebook contains background material, references,

and executable code illustrating data acquisition and post-processing workflows, serving as a practical complement to the examples presented in this article. Reproducible benchmark scripts are included in the repository under `benchmarks`.

5 Conclusion & Outlook

We have introduced `RandomMeas.jl`, a scalable and modular Julia package for developing, testing and implementing randomized measurement (RM) protocols. Its architecture separates pre-processing, data acquisition and post-processing, enabling flexible integration with different experimental setups and simulation backends. The package supports the full RM workflow, from measurement setting generation to advanced estimators, and includes a tensor-network-based implementation of classical shadows suitable for large quantum systems. It offers robust routines for a variety of RM variants, built-in statistical error estimation, and a unified interface for both experimental and theoretical applications. These features make `RandomMeas.jl` well suited for current experiments and large-scale simulations, while providing a flexible foundation for future extensions.

Looking ahead, several development directions naturally emerge:

Machine learning and scalable tomography. Efficient and robust shadow-construction routines can be combined with classical machine learning techniques to design kernel methods for state classification [44, 45], or to implement process shadow tomography for Liouvillian learning [46, 47]. Scalable randomized measurements approaches to quantum state tomography, including methods based on MPOs [14], or on neural network quantum states [48, 49] (cf. the Julia package `PastaQ`), are another promising extension. Sampling strategies during pre-processing may be further optimized via importance sampling [50, 51] or derandomization techniques [52], incorporating classical prior knowledge about the state or observable of interest [17, 53, 54, 55].

Error mitigation. Randomized measurements enable several powerful quantum error mitigation techniques. `RandomMeas.jl` already supports robust shadows and can be naturally

extended to include virtual distillation [56] and probabilistic error cancellation (PEC) [57, 58, 59]. Such extensions could further leverage efficient tensor-network inversion techniques [60] and symmetry restoration ideas based on symmetry-resolved shadows [61, 6, 53, 54, 55], broadening applicability to realistic quantum devices.

Performance optimization with GPUs. While tensor-network methods in `RandomMeas.jl` already enable large-scale simulations, significant runtime gains could be achieved by leveraging GPU acceleration. In particular, GPU-based contraction of `ITensors.jl` objects would directly benefit the most computationally intensive post-processing routines.

By combining a standardized interface, modular design, and a focus on scalability and reproducibility, `RandomMeas.jl` provides a versatile platform for advancing the state of the art in randomized measurements. We invite contributions from the community to expand its functionality, integrate new protocols, and explore emerging applications at the interface of quantum information, simulation, and machine learning.

Acknowledgements

We thank P. Zoller, A. Rath, V. Vitale, R. Kueng, H. Y. Huang and J. Preskill for previous collaborations that triggered the development of this code. We thank L. Nie, A. Raj, A. Vijay and J. Kudler-Flam for discussions, in particular related to the estimation of trace moments. BV thanks M. Votto and D. Lanier for feedback on the code.

Funding information Work in Grenoble is funded by the French National Research Agency via the research programs Plan France 2030 EPIQ (ANR-22-PETQ-0007), QUBITAF (ANR-22-PETQ-0004) and HQI (ANR-22-PNCQ-0002).

References

- [1] Andreas Elben, Steven T. Flammia, Hsin-Yuan Huang, Richard Kueng, John Preskill, Benoît Vermersch, and Peter Zoller. “The randomized measurement toolbox”. *Nat. Rev. Phys.* **5**, 9–24 (2022).
- [2] Hsin-Yuan Huang, Richard Kueng, and John Preskill. “Predicting Many Properties of a Quantum System from Very Few Measurements”. *Nat. Phys.* **16**, 1050–1057 (2020).
- [3] Tiff Brydges, Andreas Elben, Petar Jurcevic, Benoît Vermersch, Christine Maier, Ben P. Lanyon, Peter Zoller, Rainer Blatt, and Christian F. Roos. “Probing entanglement entropy via randomized measurements”. *Science* **364**, 260–263 (2019).
- [4] Manoj K. Joshi, Andreas Elben, Benoît Vermersch, Tiff Brydges, Christine Maier, Peter Zoller, Rainer Blatt, and Christian F. Roos. “Quantum Information Scrambling in a Trapped-Ion Quantum Simulator with Tunable Range Interactions”. *Phys. Rev. Lett.* **124**, 240505 (2020).
- [5] D. Zhu, Z. P. Ciani, C. Noel, A. Risinger, D. Biswas, L. Egan, Y. Zhu, A. M. Green, C. Huerta Alderete, N. H. Nguyen, Q. Wang, A. Maksymov, Y. Nam, M. Cetina, N. M. Linke, M. Hafezi, and C. Monroe. “Cross-platform comparison of arbitrary quantum states”. *Nature Communications* **13** (2022).
- [6] Lata Kh. Joshi, Johannes Franke, Aniket Rath, Filiberto Ares, Sara Murciano, Florian Kranzl, Rainer Blatt, Peter Zoller, Benoît Vermersch, Pasquale Calabrese, Christian F. Roos, and Manoj K. Joshi. “Observing the Quantum Mpemba Effect in Quantum Simulations”. *Phys. Rev. Lett.* **133**, 010402 (2024).
- [7] Ting Zhang, Jinzhao Sun, Xiao-Xu Fang, Xiao-Ming Zhang, Xiao Yuan, and He Lu. “Experimental quantum state measurement with classical shadows”. *Phys. Rev. Lett.* **127**, 200501 (2021).
- [8] G.I. Struchalin, Ya. A. Zagorovskii, E.V. Kovlakov, S.S. Straupe, and S.P. Kulik. “Experimental estimation of quantum state properties from classical shadows”. *PRX Quantum* **2** (2021).
- [9] K. J. et al Satzinger. “Realizing topologically ordered states on a quantum processor”. *Science* **374**, 1237–1241 (2021).
- [10] Vittorio Vitale, Aniket Rath, Petar Jurcevic, Andreas Elben, Cyril Branciard, and Benoît Vermersch. “Robust Estimation of the Quan-

- tum Fisher Information on a Quantum Processor”. *PRX Quantum* **5**, 030338 (2024).
- [11] Hong-Ye Hu, Andi Gu, Swarnadeep Majumder, Hang Ren, Yipei Zhang, Derek S. Wang, Yi-Zhuang You, Zlatko Mineev, Susanne F. Yelin, and Alireza Seif. “Demonstration of robust and efficient quantum property learning with shallow shadows”. *Nat. Comm.* **16**, 2943 (2025).
 - [12] Hang Dong, Pengfei Zhang, Ceren B. Dağ, Yu Gao, Ning Wang, Jinfeng Deng, Xu Zhang, Jiachen Chen, Shibo Xu, Ke Wang, Yaozu Wu, Chuanyu Zhang, Feitong Jin, Xuhao Zhu, Aosai Zhang, Yiren Zou, Ziqi Tan, Zhengyi Cui, Zitian Zhu, Fanhao Shen, Tingting Li, Jiarun Zhong, Zehang Bao, Hekang Li, Zhen Wang, Qiujiang Guo, Chao Song, Fangli Liu, Amos Chan, Lei Ying, and H. Wang. “Measuring the spectral form factor in many-body chaotic and localized phases of quantum processors”. *Physical Review Letters* **134** (2025).
 - [13] T. I. et al Andersen. “Thermalization and criticality on an analogue–digital quantum simulator”. *Nature* **638**, 79–85 (2025).
 - [14] Matteo Votto, Marko Ljubotina, Cécilia Lancien, J. Ignacio Cirac, Peter Zoller, Maksym Serbyn, Lorenzo Piroli, and Benoît Vermersch. “Learning Mixed Quantum States in Large-Scale Experiments”. *Physical Review Letters* **136**, 090801 (2026).
 - [15] S. J. van Enk and C. W J Beenakker. “Measuring $\text{Tr}\rho^n$ on Single Copies of ρ Using Random Measurements”. *Phys. Rev. Lett.* **108**, 110503 (2012).
 - [16] M Ohliger, V Nesme, and J Eisert. “Efficient and feasible state tomography of quantum many-body systems”. *New J. Phys.* **15**, 015024 (2013).
 - [17] Andreas Elben, Benoît Vermersch, Marcello Dalmonte, J. Ignacio Cirac, and Peter Zoller. “Rényi Entropies from Random Quenches in Atomic Hubbard and Spin Models”. *Phys. Rev. Lett.* **120**, 050406 (2018).
 - [18] Andreas Elben, Benoît Vermersch, Christian F. Roos, and Peter Zoller. “Statistical correlations between locally randomized measurements: a toolbox for probing entanglement in many-body quantum states”. *Phys. Rev. A* **99**, 052323 (2019).
 - [19] Senrui Chen, Wenjun Yu, Pei Zeng, and Steven T. Flammia. “Robust shadow estimation”. *PRX Quantum* **2**, 030348 (2021).
 - [20] Dax Enshan Koh and Sabee Grewal. “Classical Shadows With Noise”. *Quantum* **6**, 776 (2022).
 - [21] Hong-Ye Hu, Soonwon Choi, and Yi-Zhuang You. “Classical shadow tomography with locally scrambled quantum dynamics”. *Phys. Rev. Research* **5**, 023027 (2023).
 - [22] Ahmed A. Akhtar, Hong-Ye Hu, and Yi-Zhuang You. “Scalable and Flexible Classical Shadow Tomography with Tensor Networks”. *Quantum* **7**, 1026 (2023).
 - [23] Christian Berton, Jonas Haferkamp, Marcel Hinsche, Marios Ioannou, Jens Eisert, and Hakop Pashayan. “Shallow shadows: Expectation estimation using low-depth random clifford circuits”. *Phys. Rev. Lett.* **133**, 020602 (2024).
 - [24] Aniket Rath, Vittorio Vitale, Sara Murciano, Matteo Votto, Jérôme Dubail, Richard Kueng, Cyril Branciard, Pasquale Calabrese, and Benoît Vermersch. “Entanglement barrier and its symmetry resolution: theory and experiment”. *PRX Quantum* **4**, 010318 (2023).
 - [25] Benoît Vermersch, Aniket Rath, Bharathan Sundar, Cyril Branciard, John Preskill, and Andreas Elben. “Enhanced estimation of quantum properties with common randomized measurements”. *PRX Quantum* **5**, 010352 (2024).
 - [26] Laurin E. Fischer, Timothée Dao, Ivano Tavernelli, and Francesco Tacchino. “Dual-frame optimization for informationally complete quantum measurements”. *Phys. Rev. A* **109**, 062415 (2024).
 - [27] Roeland Wiersema and Brian Doolittle. “Classical shadows”. https://pennylane.ai/qml/demos/tutorial_classical_shadows (2021). Date Accessed: 2025-07-02.
 - [28] Ryan LaRose, Andrea Mari, Sarah Kaiser, Peter J. Karalekas, Andre A. Alves, Piotr Czarnik, Mohamed El Mandouh, Max H.

- Gordon, Yousef Hindy, Aaron Robertson, Purva Thakre, Misty Wahl, Danny Samuel, Rahul Mistri, Maxime Tremblay, Nick Gardner, Nathaniel T. Stemen, Nathan Shammah, and William J. Zeng. “Mitiq: A software package for error mitigation on noisy quantum computers”. *Quantum* **6**, 774 (2022).
- [29] Shi-Xin Zhang, Jonathan Allcock, Zhou-Quan Wan, Shuo Liu, Jiace Sun, Hao Yu, Xing-Han Yang, Jiezhong Qiu, Zhaofeng Ye, Yu-Qin Chen, Chee-Kong Lee, Yi-Cong Zheng, Shao-Kai Jian, Hong Yao, Chang-Yu Hsieh, and Shengyu Zhang. “TensorCircuit: a Quantum Software Framework for the NISQ Era”. *Quantum* **7**, 912 (2023).
- [30] “Paddle Quantum” (2020).
- [31] Andreas Elben and Benoît Vermersch. “Randommeas.jl: The randomized measurement toolbox in julia”. *Zenodo* (2026).
- [32] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. “Characterizing quantum supremacy in near-term devices”. *Nat. Phys.* **14**, 595–600 (2018).
- [33] Matthew Fishman, Steven White, and Edwin Miles Stoudenmire. “The ITensor Software Library for Tensor Network Calculations”. *SciPost Physics Codebases* **Page 004** (2022).
- [34] Andreas Elben, Richard Kueng, Hsin-Yuan (Robert) Huang, Rick van Bijnen, Christian Kokail, Marcello Dalmonte, Pasquale Calabrese, Barbara Kraus, John Preskill, Peter Zoller, and Benoît Vermersch. “Mixed-State Entanglement from Local Randomized Measurements”. *Phys. Rev. Lett.* **125**, 200501 (2020).
- [35] Andreas Elben, Benoît Vermersch, Rick van Bijnen, Christian Kokail, Tiff Brydges, Christine Maier, Manoj Joshi, Rainer Blatt, Christian F. Roos, and Peter Zoller. “Cross-Platform Verification of Intermediate Scale Quantum Devices”. *Phys. Rev. Lett.* **124**, 010504 (2020).
- [36] Chien-Fu Jeff Wu. “Jackknife, bootstrap and other resampling methods in regression analysis”. *Ann. Stat.* **14**, 1261–1295 (1986).
- [37] Matteo Ippoliti, Yaodong Li, Tibor Rakovszky, and Vedika Khemani. “Operator relaxation and the optimal depth of classical shadows”. *Physical Review Letters* **130** (2023).
- [38] T. Baumgratz, A. Nüßeler, M. Cramer, and M. B. Plenio. “A scalable maximum likelihood method for quantum state tomography”. *New J. Phys.* **15**, 125004 (2013).
- [39] Sergio Boixo, Sergei V. Isakov, Vadim N. Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J. Bremner, John M. Martinis, and Hartmut Neven. “Characterizing quantum supremacy in near-term devices”. *Nature Physics* **14**, 595–600 (2018).
- [40] Frank et al Arute. “Quantum supremacy using a programmable superconducting processor”. *Nature* **574**, 505–510 (2019).
- [41] Daniel K. Mark, Joonhee Choi, Adam L. Shaw, Manuel Endres, and Soonwon Choi. “Benchmarking quantum simulators using ergodic quantum dynamics”. *Physical Review Letters* **131** (2023).
- [42] Adam L. Shaw, Zhuo Chen, Joonhee Choi, Daniel K. Mark, Pascal Scholl, Ran Finkelstein, Andreas Elben, Soonwon Choi, and Manuel Endres. “Benchmarking highly entangled states on a 60-atom analogue quantum simulator”. *Nature* **628**, 71–77 (2024).
- [43] Daniel Jaschke, Simone Montangero, and Lincoln D. Carr. “One-dimensional many-body entangled open quantum systems with tensor network methods”. *Quantum Science and Technology* **4**, 013001 (2018).
- [44] Hsin-Yuan Huang, Richard Kueng, Giacomo Torlai, Victor V. Albert, and John Preskill. “Provably efficient machine learning for quantum many-body problems”. *Science* **377**, eabk3333 (2022).
- [45] Laura Lewis, Hsin-Yuan Huang, Viet T. Tran, Sebastian Lehner, Richard Kueng, and John Preskill. “Improved machine learning algorithm for predicting ground state properties”. *Nat. Comm.* **15**, 895 (2024).
- [46] Jonathan Kunjummen, Minh C. Tran, Daniel Carney, and Jacob M. Taylor. “Shadow process tomography of quantum channels”. *Phys. Rev.A* **107**, 042403 (2023).

- [47] Ryan Levy, Di Luo, and Bryan K. Clark. “Classical Shadows for Quantum Process Tomography on Near-term Quantum Computers”. *Phys. Rev. Research* **6**, 013029 (2024).
- [48] Giacomo Torlai, Christopher J. Wood, Atithi Acharya, Giuseppe Carleo, Juan Carrasquilla, and Leandro Aolita. “Quantum process tomography with unsupervised learning and tensor networks”. *Nat. Comm.* **14**, 2858 (2023).
- [49] Hsin-Yuan Huang, John Preskill, and Mehdi Soleimanifar. “Certifying almost all quantum states with few single-qubit measurements” (2024) [arXiv:2404.07281](https://arxiv.org/abs/2404.07281).
- [50] Charles Hadfield, Sergey Bravyi, Rudy Raymond, and Antonio Mezzacapo. “Measurements of quantum hamiltonians with locally-biased classical shadows” (2020). [arXiv:2006.15788](https://arxiv.org/abs/2006.15788).
- [51] Aniket Rath, Rick van Bijnen, Andreas Elben, Peter Zoller, and Benoît Vermersch. “Importance sampling of randomized measurements for probing entanglement”. *Phys. Rev. Lett.* **127**, 200503 (2021).
- [52] Hsin-Yuan Huang, Richard Kueng, and John Preskill. “Efficient estimation of pauli observables by derandomization”. *Phys. Rev. Lett.* **127**, 030503 (2021).
- [53] Jacob Bringewatt, Jonathan Kunjummen, and Niklas Mueller. “Randomized measurement protocols for lattice gauge theories”. *Quantum* **8**, 1300 (2024).
- [54] Andrew Zhao and Akimasa Miyake. “Group-theoretic error mitigation enabled by classical shadows and symmetries”. *npj Quantum Information* **10** (2024).
- [55] Frederic Sauvage and Martin Larocca. “Classical shadows with symmetries” (2024). [arXiv:2408.05279](https://arxiv.org/abs/2408.05279).
- [56] Alireza Seif, Ze-Pei Cui, Sisi Zhou, Senrui Chen, and Liang Jiang. “Shadow distillation: Quantum error mitigation with classical shadows for near-term quantum processors”. *PRX Quantum* **4**, 010303 (2023).
- [57] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. “Error mitigation for short-depth quantum circuits”. *Phys. Rev. Lett.* **119** (2017).
- [58] Suguru Endo, Simon C. Benjamin, and Ying Li. “Practical quantum error mitigation for near-future applications”. *Phys. Rev. X* **8**, 031027 (2018).
- [59] Hamza Jnane, Jonathan Steinberg, Zhenyu Cai, H. Chau Nguyen, and Bálint Koczor. “Quantum error mitigated classical shadows”. *PRX Quantum* **5**, 010324 (2024).
- [60] Sergei Filippov, Matea Leahy, Matteo A. C. Rossi, and Guillermo García-Pérez. “Scalable tensor-network error mitigation for near-term quantum computing” (2023) [arXiv:2307.11740](https://arxiv.org/abs/2307.11740).
- [61] Vittorio Vitale, Andreas Elben, Richard Kueng, Antoine Neven, Jose Carrasco, Barbara Kraus, Peter Zoller, Pasquale Calabrese, Benoît Vermersch, and Marcello Dalmonte. “Symmetry-resolved dynamical purification in synthetic quantum matter”. *SciPost Physics* **12**, 106 (2022).
- [62] Christian Kokail, Christine Maier, Rick van Bijnen, Tiff Brydges, Manoj K. Joshi, Petar Jurcevic, Christine A. Muschik, Pietro Silvi, Rainer Blatt, Christian F. Roos, and Peter Zoller. “Self-Verifying Variational Quantum Simulation of the Lattice Schwinger Model”. *Nature* **569**, 355–360 (2019).
- [63] Tiff Brydges, Andreas Elben, Petar Jurcevic, Benoît Vermersch, Christine Maier, Ben P. Lanyon, Peter Zoller, Rainer Blatt, and Christian F. Roos. “Probing Rényi Entanglement Entropies via Randomized Measurements”. <https://zenodo.org/records/2527010> (2018).

A Jupyter notebooks for advanced usage

This appendix provides the complete list of Jupyter notebooks accompanying `RandomMeas.jl`. They can be found in the project repository, each with background material, references, and executable code. Many notebooks reproduce or reanalyse experimental results, while others are purely numerical and illustrate specific randomized-measurement protocols.

A.1 Classical shadows

1. *Energy measurements with classical shadows* — Estimation of energies and energy variances, e.g. for verifying ground-state preparation in quantum simulation [62, 2].
2. *Robust shadow tomography* — Construction of classical shadows robust to measurement errors via calibration [19, 20, 25, 10].
3. *Process shadow tomography* — Illustration of quantum-gate fidelity estimation using common randomized measurements [46, 47].
4. *Classical shadows with shallow circuits* — Calibration and use of shallow-circuit measurement settings for improved estimation of nonlocal observables [21, 22, 23, 11].
5. *Virtual distillation* — Quantum error-mitigation technique based on classical shadows [56].

A.2 Quantum benchmarking

1. *Cross-entropy and self-cross-entropy benchmarking* — Fidelity estimation for random circuits using classical simulation [40, 13].
2. *Fidelities from common randomized measurements* — Estimation of $\langle \psi | \rho | \psi \rangle$ with reduced statistical errors [25].
3. *Cross-platform verification* — Mixed-state fidelity estimation between states prepared on two quantum devices [35].

A.3 Entanglement measurements

1. *Entanglement entropy of pure states and the Page curve* — Purity and Rényi-entropy estimation via randomized measurements.
2. *Analysis of Brydges et al. (Science 2019) data* — Re-analysis of the first experimental randomized-measurement dataset [3, 63], including higher-order Rényi entropies and batch shadows [24].
3. *Mixed-state entanglement detection* — Experimental detection of entanglement via the p_3 -PPT condition [34].
4. *Topological entanglement entropy* — Simulation of purity measurements for the toric code and extraction of the topological contribution [9, 17, 3].

A.4 Import/Export and interface with experiments

1. *Qiskit hardware interface (example workflow)* — `RandomizedMeasurementsQiskit.ipynb` generates randomized measurement circuits and executes them in Qiskit. `RandomizedMeasurementsQiskitPostprocessing.ipynb` imports the exported unitaries and shots into `RandomMeas.jl` and performs the classical post-processing by constructing a `MeasurementGroup`.

2. *Exporting measurement circuits to OpenQASM* — Demonstration of the OpenQASM (QASM 2.0) export routine for `LocalUnitaryMeasurementSetting` and `ComputationalBasisMeasurementSetting`, enabling straightforward use of the corresponding randomized-measurement circuits in external SDKs (e.g. Qiskit, Cirq).

A.5 Miscellaneous

1. *Noisy circuit simulations with tensor networks* — Simulation of noisy dynamics using density-matrix propagation or quantum-trajectory methods [43].
2. *Estimating statistical error bars via Jackknife resampling* — Resampling-based uncertainty estimates from a single dataset.