Benchmarking challenge for Quantum Circuit Tensor Network Simulators

Pablo Andrés Martínez
Quantinuum, Cambridge, UK

April 30, 2025

1 Motivation

There is a lack of benchmarking literature on quantum circuit simulators based on tensor networks. Two of the reasons are:

1. There is no standard benchmarking suite of circuits of practical interest that are challenging for these simulators.
2. Expertise is required to tune the parameters of approximate simulation methods for maximum performance. Even when two simulators have the same available parameters (e.g. maximum virtual bond dimension), the trade-offs they adjust can drastically depend on the algorithm used, making fair comparisons difficult.

To address the first point, we provide a set of circuits with varied simulation hardness. These circuits were provided by Quantinuum's research teams and are application oriented. To address the second point, we ask developers of simulators to submit their own results to this challenge, as they are the ones that can showcase the best possible performance of their algorithms.

This benchmarking challenge will help Quantinuum identify the best quantum circuit simulators for our research teams to use. We believe these circuits are of general interest, but we acknowledge that a more diverse and extensive benchmarking suite would be required to represent the full range of applications of the quantum industry. The results of this benchmarking task will be advertised to our research teams, who may use this information to choose who to partner with for their simulation needs. Moreover, the identified leaders may be invited to collaborate with us to integrate their solutions in our stack.

2 Instructions for participation

Participants are required to approximately simulate the circuits from the benchmarking suite attached (see section 2.6). The simulation algorithm used must support sampling from the resulting state at the end of the circuit, as well as calculation of expectation values. Information regarding the fidelity of the final state must be provided, see section 3. Furthermore, participants are required to calculate certain expectation values of Pauli observables for each circuit; these will be used to cross-validate results. The contents of a submission are detailed in section 2.3.

- Participants can send us as many submissions as they wish. We suggest participants send us two submissions per simulation algorithm: one where they attempt to reach the highest fidelity possible for every circuit, and one where they attempt to reach the best time to solution (within a reasonable level of fidelity, see section 2.4).
- All of the results reported in the same submission must be obtained using the same simulation algorithm.
- Simulation parameters (such as bond dimension, truncation cutoff, etc) *can* be tuned differently for different circuits in the same submission, with the goal of achieving a desired performance trade-off for each circuit.
- Participants may apply automated circuit optimisation and analysis prior to simulation, but they must include the time spent on this in their submission, see section 3.

2.1 Important dates

- 15^{th} of April, 2025. Potential participants are invited to join the challenge and receive this document. Participants have two weeks to provide feedback on the challenge and confirm participation.
- 1^{st} of May, 2025. The challenge starts. The files listed in section 2.6 will be sent to the participants on this date. They will be given two months to complete the challenge.
- 1^{st} of July, 2025. The challenge ends. All participants must have sent their submissions by this date.
- 15^{th} of July, 2025. The participants are informed of their place in the rankings they participated in (see section 2.2) and the data from the public tier is made available publicly.

2.2 How will the data be used?

The submissions from each participant will be evaluated by Quantinuum and rankings will be automatically generated, following the criteria discussed in section 2.4. Two different tiers of result sharing are offered: one public and another private and anonymous. Separate rankings will be generated for each tier, including only the participants within the tier.

- *Public tier*. The list of participants in the public tier will be made publicly available. All of the public tier participants' submissions will be included in a public repository after the challenge has ended. The scripts to generate the rankings (following the criteria discussed in section 2.4) will be included in the public repository. Participants are free to advertise the results of the rankings and reference the repository as proof.

- *Anonymous tier*. The list of participants in the anonymous tier will only be known to Quantinuum employees. Only Quantinuum employees will have access to the participants' submissions. Once the submissions are processed, the participants will be informed of their position in the rankings of the anonymous tier, but they will not have access to the ranking lists themselves. Since neither the names of the participants nor the number of participants in the anonymous tier will be revealed at any point, no two participants will be able to asses how they compare to each other. The participants agree not to publicly advertise their position in the anonymous tier ranking, they also agree not to cite Quantinuum as an external endorser of the quality of their simulator. Quantinuum will not make any public statement on the results from the anonymous tier.

2.3 Contents of a submission

Templates for the documents to be included in each submission are attached along this document and listed in section 2.6. A single submission consists of:

- A copy of the appropriate `PUB_AGREEMENT.txt` and/or `ANON_AGREEMENT.txt` document(s) from the files attached. Including either (or both) implies you have read and agreed to its contents. This is how you opt-in to participate in the public and/or anonymous tier.

- A `METRICS.csv` file with the results, containing the same columns and rows as the `METRICS.csv` file attached.
  - Rows correspond to the different circuits in the benchmarking suite, see section 4.
  - Columns correspond to the benchmark metrics used to evaluate performance, see section 3.
  - If a particular circuit could not be simulated, or the metric is not applicable, leave the corresponding cells blank.

- A `SHOTS.txt` plain text file with the outcome of 100 sample bitstrings obtained by measuring (in the computational basis) all qubits of the final state of the circuit. Write one line per shot, use strings of n characters `'0'` and `'1'` for each shot (where n is the number of qubits). The ordering of the bits must follow the increasing lexicographic order of the qubit names: `a[0]` $<$ `q[0]` $<$ `q[1]`.

- An `EXP_VAL.json` file with the calculated expectation values and the same structure as the `EXP_VAL.json` file attached.
  - The json file contains a nested dictionary where the top level keys are the names of the circuit files provided.
  - The keys of the children dictionaries are the observables requested. These are provided as strings of n characters in the set {`I`,`X`,`Y`,`Z`} (where n is the number of qubits) following the increasing lexicographic order of the qubit names.
  - The values of the children dictionaries should be replaced with the expectation value of the corresponding observable on the final state of the circuit.
- A `README.txt` (or `.pdf`, `.doc`...) providing the following details:
  - The specifications of the hardware used to run the simulation.
  - A brief description of the simulation algorithm used and the features that may distinguish it from others.
  - Which of the circuit formats listed in section 2.6 was used, or if you had to do some conversion. Indicate if your algorithm supports n-qubit gates with $n > 2$ and, if not, how did you deal with them.
  - A brief description of the trade-off explored in this solution: e.g. maximise fidelity or minimise resources.
  - Information about the parameters the algorithm admits and which values were chosen for this submission (which may be different for different circuits).
  - Information on any additional source of inaccuracies that may not be accounted for in the fidelity metrics from section 3.
  - Information on how we may access your simulation algorithm in the future, e.g. open sourced, licensed, cloud, on-prem, etc.
- You are welcome to include any extra information and metrics that are not included in the `METRICS.csv` file as *additional files* within the submission.

Send us a `.zip` folder containing the files listed above to the email address `pablo.andresmartinez@quantinuum.com` with subject "Simulator benchmarking challenge". **One `.zip` file per submission**.

Reproducibility. We expect participants to take the appropriate measures to guarantee reproducibility of their results. In the case of outliers being identified in the submissions, we may ask participants to give us further details on how to reproduce them. We suggest as best practice to create a standalone repository (with one branch per submission) where all necessary scripts, parameter values, version information and instructions for setup are provided. Public tier participants are encouraged to make this repository public and include a link to it in their `README.txt`. Anonymous tier participants would keep the repository private and consider giving read access to Quantinuum employees upon request.

2.4 How will the data be evaluated?

Fidelity-band rankings. Submissions will be arranged in three "fidelity bands" depending on the achieved *mirror fidelity* (see section 3). The submissions within the same fidelity band will be ordered by *total runtime* (see section 3). There will be a separate ranking for each distinct tuple of *fidelity band*, *circuit* and *tier* (see section 2.2). Anonymous tier participants will receive their position in the ranking for each circuit and each fidelity band they qualify for. The three fidelity bands are:

- *Accurate*: mirror fidelity ≥ 0.9.
- *High signal*: mirror fidelity ≥ 0.6.
- *Low signal*: mirror fidelity ≥ 0.2.

Notice that these bands can overlap. For instance, if a participant provides a submission that qualifies for the *high signal* band, that submission is immediately included in the *low signal* band as well. In the case the same participant has multiple submissions qualifying for the same ranking, only the best submission (i.e. the one with lowest runtime) will be considered. We encourage participants to provide two submissions: one that qualifies for the highest fidelity band they can achieve for each circuit, and another where they attempt to get the fastest runtime possible on the immediately lower fidelity band.

Expectation value cross-validation. The expectation values obtained in submissions within the *accurate* and *high signal* fidelity bands above will be compared to each other. Outliers will be reported to the corresponding participant as a potential bug.

Expert opinion. All data submitted will be made available to our research teams, who may be looking for specific trade-offs. These assessments will be kept internal within Quantinuum, but may encourage collaboration efforts with the participants in the future.

2.5 Hardware disadvantage

If your simulator is open-sourced and you believe that you are at a disadvantage due to the hardware you have access to, we offer to run the simulation on a machine with CPU AMD EPYC 7763 (512GB RAM) and NVIDIA A100 (40GB VRAM). These resources are provided via a NERSC grant and, as such, their use is constrained to open-sourced software. To be considered for this you must:

- Host all of the code on public repositories, including parameter values and detailed instructions for us to run the simulations.
- Agree to participate in the public tier (see section 2.2).

- Make a standard submission, as described in section 2.3 with the results obtained using the hardware available to you, for a parameter configuration that makes simulation feasible on your hardware. This is so that we can verify that your scripts run and results are reproducible.

We will only offer this to participants whose simulation algorithm we believe has strong potential and would become competitive when run on better hardware. Send an email to `pablo.andresmartinez@quantinuum.com` to start a discussion.

2.6 Files attached

- `circuit_suite.zip`. All of the circuits in the benchmarking suite. Three subdirectories are included; the three of them contain the same circuits but in different formats and gatesets:
  - `circuit_suite/pytket_orig` contains the circuits in pytket json format, see [6]. These are the original circuits provided by our research teams and include n-qubit gates with $n > 2$.
  - `circuit_suite/pytket_decomp` contains the circuits in pytket json format, see [6]. All n-qubit gates with $n > 2$ gates have been decomposed, so that the circuits use a standard gateset.[1]
  - `circuit_suite/qasm` contains the circuits in OpenQASM 2 format. These circuits are identical to those in `pytket_decomp`, using the same gateset converted to the standard OpenQASM library `qelib1.inc` [4].
- `dagger_circuits.zip`. Same as above, but containing the dagger of each of the circuits with the same file name. These are meant to be used in the calculation of the mirror fidelity, see section 3.
- `metadata.csv`. A table including metadata information for each of the circuits in the suite; the columns are explained below.
  - *Family.* A label identifying the origin of the circuit, see section 4.
  - *Hardness.* A rough estimate of the simulation hardness, obtained from attempts at simulating the circuit using our own simulators. Values are either: *easy*, *medium*, *hard* or *unreasonable*.[2]
  - *Qubits.* The number of qubits in the circuit.
  - *Gate counts.* One column per gate in the common gateset (see [5] for gate definitions) used by the circuits from `circuit_suite/pytket_orig`, each indicates the number of gates of this kind in the original circuit.

[1]The gateset is {`Rz, Rx, ZZPhase, XXPhase, CX, H, S, Sdg, T`}. The definition of each of these gates is provided in [5]; notice that `ZZPhase` corresponds to `Rzz` and `XXPhase` corresponds to `Rxx`.

[2]We do not expect circuits labelled as *unreasonable* to be simulated with enough fidelity to appear in any of the fidelity bands from section 2.4, but we are keen to be proven wrong!

- `METRICS.csv`. Template of the metrics file to be completed by the participants, see section 3.
- `EXP_VAL.json`. Template of the expectation value dictionary to be completed by the participants, see section 2.3.
- `PUB_AGREEMENT.txt` and `ANON_AGREEMENT.txt`. These contain the agreement to be included by participants in their submission to participate on either the public tier or anonymous tier (or both). The contents reproduce the bullet points from section 2.2.

3 Benchmark metrics

Each metric listed below corresponds to a column in the `METRICS.csv` file that the participants must complete. The essential metrics that every participant must provide are: **mirror fidelity** and **total runtime**. Other metrics may be left blank if not applicable. We encourage participants to provide results for all of the metrics if possible.

- **Runtime metrics (s)**. There is a column for each of these in `METRICS.csv`. Each of the runtime metrics reported here must be disjoint from each other, i.e. with no interval of time counted more than once, with the exception of *total runtime*.
  - **Total runtime**. The total time taken by the simulation framework since it first sees the circuit until it produces the measurement bitstrings from 100 shots. This value must be larger or equal to the sum of *simulation time*, *preprocessing time* and *shot time*.
  - **Simulation time**. The time the simulator took to evolve $|0\rangle^{\otimes n}$ to $C|0\rangle^{\otimes n}$ for the corresponding circuit C of n qubits.
  - **Preprocessing time**. Include here any process that produces an equivalent circuit to be simulated or analyses the circuit to increase performance of the simulator (e.g. qubit mapping to MPS sites). Optimisations that are done in tandem with the simulator (e.g. such as contracting gates into an MPO) should be included in *simulation time* rather than here.
  - **Shot time**. The time taken to produce 100 sample bitstrings by measuring all qubits in the computational basis on the already calculated final state.
  - **Expectation value time**. The total time taken to calculate the expectation values requested in `EXP_VAL.json` from the already calculated final state.
  - **Other time**. Any computation time spent on the circuit that does not qualify as any of the above. Please include details on what this time was spent on in your `README.txt` document.

- **Mirror fidelity**. For a given circuit C, simulate $|\psi\rangle = C|0\rangle$; then, simulate $|\phi\rangle = C^\dagger|\psi\rangle$ and calculate the value of $|\langle 0|\phi\rangle|^2$ for the resulting approximation $|\phi\rangle$ by computing the inner product explicitly. The dagger of every circuit attached can be found in `daggers_circuits.zip`. It is essential that $|\psi\rangle$ is obtained first, and then $C^\dagger$ is applied to it, so that there is no cancellation of gates between C and $C^\dagger$. Please include in your `README.txt` file a brief description of the approach you used to guarantee there was no such gate cancellation.

- **Fidelity estimate**. An estimate of the final state fidelity $|\langle\psi|\psi'\rangle|^2$ where $|\psi\rangle$ is the ideal final state of the circuit and $|\psi'\rangle$ is the approximation obtained by the simulator. Please include in your `README.txt` file a brief description of the method used to calculate this estimate.

- **Final state memory (MB)**. The memory required to represent the state at the end of the circuit.

4 Circuits

Here we discuss the origin of the circuits in the benchmarking suite. The *family* column in the `metadata.csv` file attached refers to this classification.

- **condensed_matter**. Circuits for Trotterised Hamiltonian simulation of 2D lattices from our condensed matter team.

- **chemistry_uccsd**. UCCSD ansatz state preparation circuits for different molecules. The parameters of the ansatz are initialised to values obtained by an unconverged CCSD over 50 iterations. They are comprised of a sequence of Pauli exponentials $e^{\theta\vec{\sigma}}$. The circuits were generated using InQuanto [8].

- **mvsp**. Circuits for multivariate state preparation, see paper for details [7].

- **qmci**. Circuits obtained using our Quantum Monte Carlo Integration engine, see [1]. The circuits included compute the integrals governing the interaction between certain particles, see paper for details [10].

- **qec_non_ft**. Circuits encoded with QEC codes with up to 30 physical qubits per codeblock. Different codes are used, exploring the trade-off between encoding rate and code distance. The state preparation in the circuits are not fault tolerant, and no QEC syndrome extraction is applied.[3] Further details on the circuits and the QEC codes used are provided in appendix A.

[3]We do not expect the inclusion of such measurement gadgets would increase the hardness of the simulation. Hence, we choose to omit them so that the inverse of the circuits are readily available for the calculation of mirror fidelity.

- **bonus_qec**. These circuits contain magic state injections, which require mid-circuit measurements. Hence, their inverse circuit has not been provided. Please leave the entry in `METRICS.csv` for mirror fidelity blank, since it does not apply. This family of circuits will not be included in the rankings described in section 2.4. Simulating this family is considered an optional bonus task.

5 Bonus task

If your simulator supports application of noise channels, you are welcome to make *a separate submission* where you report the results from running the circuits using a noise model. The noise model parameters that would be relevant for us are documented in our webpage [2, 9]. No ranking will be generated for this bonus task; this is only meant as an opportunity for you to showcase your simulator's features.

To avoid confusion with standard submissions, please add a `BONUS.txt` file to the root of your submission's `.zip` folder, including any information that you would like Quantinuum researchers to know about the features you are showcasing. You are welcome to showcase any feature you please, not just noise models. However, keep in mind that these will not be ranked nor there is any commitment from the research teams to review these. The most likely scenario is that only simulators that already perform highly in the standard submission will be explored.

References

[1] Ismail Yunus Akhalwaya et al. *A Modular Engine for Quantum Monte Carlo Integration*. 2023. arXiv: `2308.06081 [quant-ph]`. URL: `https://arxiv.org/abs/2308.06081`.

[2] Quantinuum's noise model parameter glossary. URL: `https://docs.quantinuum.com/systems/user_guide/emulator_user_guide/noise_model.html`.

[3] MQT's Quantum Error Correction library. URL: `https://mqt.readthedocs.io/projects/qecc/en/latest/`.

[4] OpenQASM qelib1 library. URL: `https://github.com/Qiskit/qiskit/blob/main/qiskit/qasm/libs/qelib1.inc`.

[5] Pytket OpType. URL: `https://docs.quantinuum.com/tket/api-docs/optype.html`.

[6] Loading json circuits to pytket. URL: `https://docs.quantinuum.com/tket/api-docs/circuit_class.html#pytket.circuit.Circuit.from_dict`.

[7] Matthias Rosenkranz et al. "Quantum state preparation for multivariate functions". In: *Quantum* 9 (Apr. 2025), p. 1703. ISSN: 2521-327X. DOI: 10.22331/q-2025-04-09-1703. URL: https://doi.org/10.22331/q-2025-04-09-1703.

[8] Andrew Tranter et al. *InQuanto: Quantum Computational Chemistry*. 2022. URL: https://www.quantinuum.com/products-solutions/inquanto.

[9] Quantinuum's H2 noise model parameter values. URL: https://docs.quantinuum.com/systems/user_guide/emulator_user_guide/emulators/h2_emulators.html.

[10] Ifan Williams and Mathieu Pellen. *A general approach to quantum integration of cross sections in high-energy physics*. 2025. arXiv: 2502.14647 [quant-ph]. URL: https://arxiv.org/abs/2502.14647.

A Details on the QEC circuit families

The circuits in the family `qec_non_ft` and `bonus_qec` were custom made for the purpose of this challenge.[4] There are three types of circuits whose high level structure is detailed in the figures included at the end of this document; each wire corresponds to a codeblock of the appropriate number of physical qubits, and gates are transversally applied across all physical qubits in the code blocks. For $[[n, k, d]]$ codes where $k = 1$, the transversal gates implement the corresponding gate at the logical level, since all codes are self-dual CSS codes. However, this is not the case for codes where $k > 1$, where these transversal gates may be implementing arbitrary Clifford operations at the logical level. State preparation and measurement are not done fault tolerantly. There are no syndrome extraction gadgets in the circuits.

There are five different QEC codes used, all of them CSS selfdual codes. The details of the code can be found in the webpages indicated below, where the check matrix of the code is given in the H entry.

- Steane code $[[7, 1, 3]]$:
  `https://qecdb.org/codes/6705224a4315521360862353`
- Distance 5 colour code $[[17, 1, 5]]$:
  `https://qecdb.org/codes/672e4f2a4f33a38b84d76b7c`
- High distance code $[[23, 1, 7]]$:
  `https://qecdb.org/codes/67052a628271751a9ef4e1d7`
- High encoding rate code $[[23, 13, 3]]$:
  `https://qecdb.org/codes/6735e7081dac5c4a5964ac58`

[4]We are aware that extended stabiliser simulators may be better suited than tensor network approaches for the task of simulating these. We have tried to make them relevant for this challenge by including a non-trivial number of non-Clifford gates on each of them.

- Balanced trade-off between distance and encoding rate $[[28, 4, 5]]$: `https://qecdb.org/codes/67a38f559d65c7b4098268e9`

The encoding gadgets for non fault-tolerant state preparation and measurement were generated using MQT's QECC library [3].

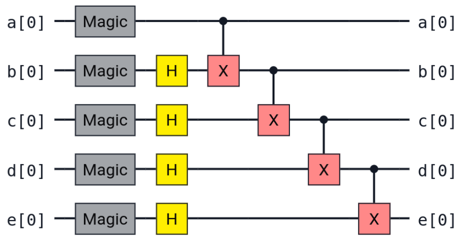

Figure 1: High-level structure of *linear* circuits from the `qec_non_ft` family. Each wire corresponds to a codeblock, gates are applied transversally on all physical qubits of the block. The *Magic* box indicates non fault-tolerant magic state preparation.

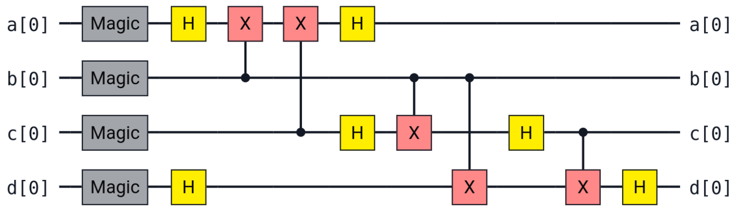

Figure 2: High-level structure of *diamond* circuits from the `qec_non_ft` family. Each wire corresponds to a codeblock, gates are applied transversally on all physical qubits of the block. The *Magic* box indicates non fault-tolerant magic state preparation.

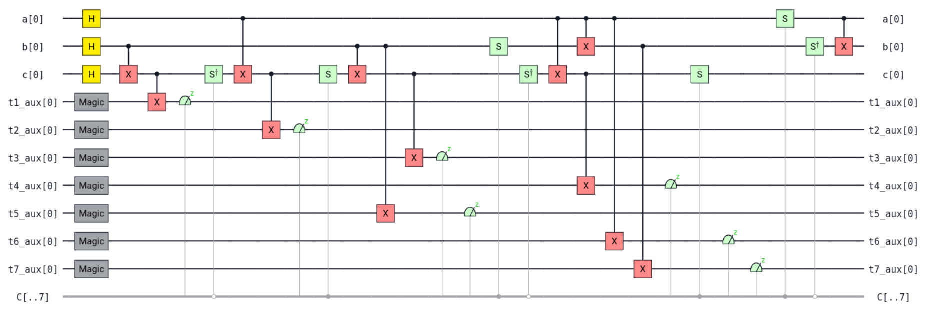

Figure 3: High-level structure of *t_injections* circuits from the `bonus_qec` family. Each wire corresponds to a codeblock, gates are applied transversally on all physical qubits of the block. The *Magic* box indicates non fault-tolerant magic state preparation. Measurements are not fault-tolerant. For codes where $k = 1$, this circuit implements a logical CCZ gate applied to a logical $|+\rangle|+\rangle|+\rangle$ state.