

# Software for Creating Scalable Benchmarks from Quantum Algorithms

Noah Siekierski,<sup>1</sup> Stefan Seritan,<sup>1</sup> Neer Patel,<sup>2</sup> Siyuan Niu,<sup>2</sup> Thomas Lubinski,<sup>3,4</sup> Timothy Proctor,<sup>1</sup>

<sup>1</sup>*Quantum Performance Laboratory, Sandia National Laboratories, Livermore, CA 94550, USA*

<sup>2</sup>*University of Central Florida, Orlando, FL 32816, USA*

<sup>3</sup>*QED-C Technical Advisory Committee – Standards, Arlington, VA 22209, USA*

<sup>4</sup>*Quantum Circuits Inc, New Haven, CT 06511, USA*

**Abstract**—Creating scalable, reliable, and well-motivated benchmarks for quantum computers is challenging: straightforward approaches to benchmarking suffer from exponential scaling, are insensitive to important errors, or use poorly-motivated performance metrics. Furthermore, curated benchmarking suites cannot include every interesting quantum circuit or algorithm, which necessitates a tool that enables the easy creation of new benchmarks. In this work, we introduce a software tool for creating scalable and reliable benchmarks that measure a well-motivated performance metric (process fidelity) from user-chosen quantum circuits and algorithms. Our software, called **scarab**, enables the creation of efficient and robust benchmarks even from circuits containing thousands or millions of qubits, by employing efficient fidelity estimation techniques, including mirror circuit fidelity estimation and subcircuit volumetric benchmarking. **scarab** provides a simple interface that enables the creation of reliable benchmarks by users who are not experts in the theory of quantum computer benchmarking or noise. We demonstrate the flexibility and power of **scarab** by using it to turn existing inefficient benchmarks into efficient benchmarks, to create benchmarks that interrogate hardware and algorithmic trade-offs in Hamiltonian simulation, to quantify the in-situ efficacy of approximate circuit compilation, and to create benchmarks that use subcircuits to measure progress towards executing a circuit of interest.

## I. INTRODUCTION

Over the past decade, quantum computing hardware has progressed from few-qubit physics experiments [1] to commercially-available machines with dozens to hundreds of qubits [2]–[7]. These rapid advances have been accompanied by increasing interest in benchmarking the performance of these devices [8]. The most well-developed and widely-used benchmarks for quantum computers, such as the quantum volume benchmark [9] and randomized benchmarking (RB) [10]–[22], quantify a device’s performance on random circuits. However, there has been increasing interest in directly measuring the performance of contemporary quantum computing systems on algorithms and applications [8], as demonstrated by an ever-expanding array of application-based benchmarking suites [6], [23]–[43].

Application-oriented quantum computer benchmarks are designed around quantum algorithms for computational problems [8], such as factoring [44] or Hamiltonian simulation [45]–[54]. A variety of application-oriented benchmarking suites have been developed [6], [23]–[43], such as the

benchmarking suite of the Quantum Economic Development Consortium (QED-C) [55]–[59] and SupermarQ [26]. Unlike benchmarks for individual one- and two-qubit quantum gates (such as one- and two-qubit RB [22]), algorithmic benchmarks are sensitive to errors that only emerge in many-qubit circuits, like many-qubit crosstalk [14]–[18], [60], [61]. Furthermore, unlike benchmarks that use many-qubit random circuits [9], [13], [15], [17], algorithm-oriented benchmarks can quantify the size and impact of errors in circuits that contain the same structures as algorithms. However, many existing application-based benchmarks are not scalable or have technical flaws caused by design decisions taken to enable scaling [8].

Many application-oriented (and other) benchmarks rely on classical computations that scale exponentially in the number of qubits ( $n$ ), e.g., because the benchmark requires classically computing the error-free output distribution of the circuit. Some application-oriented benchmarks use bespoke modifications to the application circuit to make the benchmark efficient [8], [55]. However, this can lead to benchmarks that do not accurately predict performance on the original quantum circuits of interest, e.g., because the altered circuit has different sensitivity to errors [8]. Furthermore, even a robustly-designed algorithmic benchmarking suite must choose which quantum algorithms and circuits to represent—they cannot include every interesting quantum circuit or algorithm. There is therefore a need for a tool that can create efficient and reliable benchmarks from user-chosen algorithms or circuits.

In the last few years, a variety of techniques have been developed that could enable such a user-friendly benchmarking tool, including *mirror circuit fidelity estimation* (MCFE) [17], [62], [63], *full-stack MCFE* [64], *Cliffordization* [65], *accreditation* [66]–[68], and *subcircuit volumetric benchmarking* (SVB) [69]. These techniques are general-purpose tools for efficiently estimating a quantum computer’s performance on some circuit, with complementary properties, e.g., different assumptions about a system’s noise and different costs to implement. They enable almost any algorithm to be turned into a principled benchmark that can be both applied to contemporary systems and scaled to thousands or even millions of qubits. However, deploying these techniques to create new benchmarks has so far required detailed understanding of these methods and bespoke code, because no user-friendly

implementation has been publicly available.

In this paper, we introduce and demonstrate software that implements many of these existing methods, enabling the creation of scalable benchmarks from quantum algorithms. Our software—called `scarab`, for *scalable and robust quantum algorithmic benchmark generator*—is designed to enable the easy creation and implementation of benchmarks from *user-chosen* quantum algorithms or circuits. `scarab`'s use does not require detailed knowledge of the benchmarking techniques it implements, e.g., an understanding of the theory of MCFE. It can therefore be used by quantum algorithms experts or other potential quantum computer users to design reliable and scalable quantum computer benchmarks, or to simply test if a particular quantum computer can successfully run an algorithm or circuit of interest to them. To demonstrate `scarab`, we use it to create benchmarks that interrogate hardware and algorithmic tradeoffs in Hamiltonian simulation, that quantify the in-situ efficacy of approximate circuit compilation for algorithmic circuits, and that measure progress towards executing a (potentially very large) circuit of interest.

## II. PRELIMINARIES

### A. Fidelity

Our software is designed to create efficient benchmarks that measure process fidelity ( $F$ ). We now review the definitions of  $F$  and, for the purposes of comparison with existing benchmarks, classical fidelity. Process fidelity quantifies the accuracy with which a quantum process is implemented by a quantum processor [70], making it a well-motivated success metric, and is defined between two  $n$ -qubit superoperators  $\mathcal{U}$  and  $\Lambda$ . Herein,  $\mathcal{U}$  is the unitary evolution that some  $n$ -qubit circuit  $c$  ideally implements (i.e.,  $\mathcal{U}[\rho] = U\rho U^\dagger$  where  $U \in U(2^n)$ ) and  $\Lambda$  is the superoperator corresponding to a noisy implementation of  $c$ . The process fidelity between such  $\mathcal{U}$  and  $\Lambda$  is [70]

$$F(\mathcal{U}, \Lambda) = \frac{1}{4^n} \text{tr} (\mathcal{U}^\dagger \Lambda). \quad (1)$$

Process fidelity is widely used to quantify how well a circuit  $c$  has been implemented [62]–[65], [69], [70] and to quantify the error in individual gates [70].

Instead of process fidelity, many existing benchmarks compute success metrics that compare a circuit's observed classical outcome distribution ( $\tilde{p}$ ) to its ideal, error-free outcome distribution ( $p$ ). A common metric of this sort is the classical fidelity between  $\tilde{p}$  and  $p$ , given by

$$F_c(p, \tilde{p}) = \left( \sum_x \sqrt{p(x)\tilde{p}(x)} \right)^2, \quad (2)$$

where  $x \in \{0,1\}^n$ ,  $p(x)$  is the probability of obtaining the bitstring  $x$  when the circuit is executed without error, and  $\tilde{p}(x)$  is the probability (or, in practice, an estimate of the probability from observed frequencies) with which the bitstring  $x$  is obtained in a noisy circuit execution. In the case

of the QED-C's benchmarking suite, this fidelity is rescaled to the *normalized* classical fidelity [55]:

$$\bar{F}_c(p, \tilde{p}) = \frac{F_c(\tilde{p}, p) - \frac{1}{2^n} \sum_x \sqrt{p(x)}}{1 - \frac{1}{2^n} \sum_x \sqrt{p(x)}}. \quad (3)$$

Note that computing  $p$  is generally exponentially costly, and that the normalized classical fidelity  $\bar{F}_c$  is ill-behaved if  $p$  is close to the uniform distribution.

### B. Mirror circuit fidelity estimation

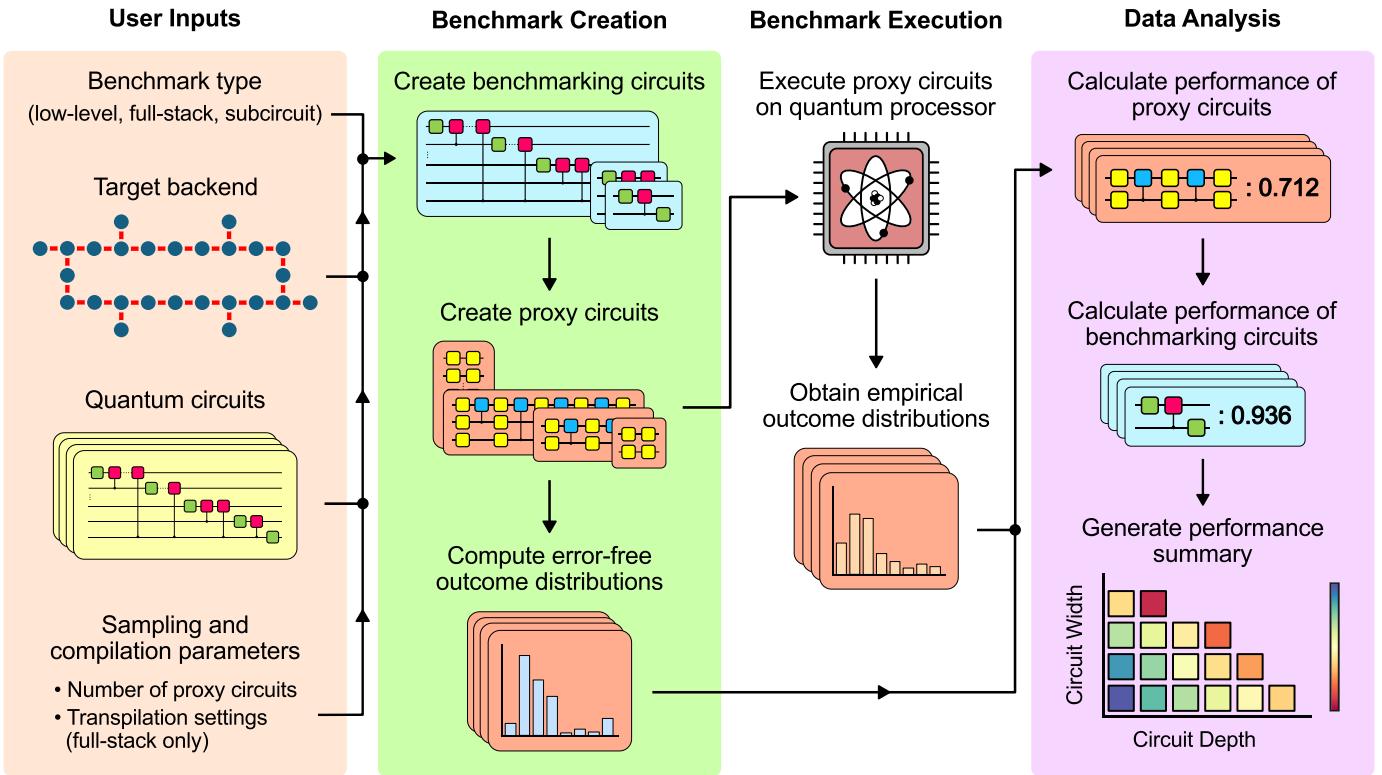
In our benchmarks, we estimate the process fidelity using MCFE [63]. MCFE estimates the process fidelity for  $c$  by running three different kinds of mirror circuits, which we refer to as  $M_1$ ,  $M_2$  and  $M_3$ . The  $M_1$  circuits consist of  $c$  followed by a randomized compilation of its layer-by-layer inverse, surrounded by randomized single-qubit gates that implement randomized state preparation and measurement (SPAM). The  $M_1$  circuits would enable estimating the process fidelity of  $c$  if the inverse and SPAM were error-free, and the  $M_2$  and  $M_3$  circuits enable estimating the fidelity of the inverse circuit and the SPAM, so that  $c$ 's process fidelity can be isolated. The  $M_2$  circuits do this by running a randomized compilation of  $c$  and  $c$ 's layer-by-layer inverse, and the  $M_3$  circuits are simple randomized SPAM circuits.

All three kinds of circuit contain randomized gates, and so when running MCFE is it necessary to decide how many circuits  $|M_1|$ ,  $|M_2|$ , and  $|M_3|$  of each kind to create. Increasing  $|M_1|$ ,  $|M_2|$  and  $|M_3|$  increases the precision of the fidelity estimate, so we report  $|M_1|$ ,  $|M_2|$  and  $|M_3|$  for each benchmark we create. `scarab` computes uncertainties on its process fidelity estimates using a non-parametric bootstrap, enabling the user to discern if an increase in  $|M_1|$ ,  $|M_2|$  and  $|M_3|$  is necessary. We refer the reader to Refs. [17], [62], [63] for a more thorough presentation of MCFE.

## III. SCARAB: AN EFFICIENT BENCHMARK GENERATOR

In this section we introduce `scarab`, which is summarized in Fig. 1. `scarab` is a high-level component within `pyGSTi` [71], an open-source Python package that implements a broad suite of quantum characterization, verification, and validation (QCVV) [70], [72] methods. `scarab` is a benchmark *generator*—it creates a benchmark from user-given inputs. The inputs to `scarab` are:

- *The benchmark type.*—The kind of benchmark to be created. There are three possible kinds of benchmark: *low-level*, *full-stack*, and *subcircuit benchmarks*. The different kinds of benchmark measure different aspects of performance, and they are detailed below.
- *A quantum computer specification.*—A quantum computing system (a.k.a., “backend”), which can be real or hypothetical, for which the benchmark is to be created.
- *Quantum circuits.*—The quantum circuits  $C$  from which to create the benchmark, specified as `qiskit` circuits. These are circuits that contains gates that are (ideally) unitary. The required level of abstraction of the circuits,



**Fig. 1: Scalable benchmarking using `scarab`.** A schematic of `scarab`, which is software for creating efficient benchmarks from interesting quantum circuits on any number of qubits. `scarab` takes user-specified circuits, along with other options which we describe further in the main text, and creates an efficient and robust benchmark from those circuits. The benchmark consists of a set of benchmarking circuits  $B$  whose performance is to be estimated, coupled with a set of proxy circuits  $P$  that enable the efficient performance estimation of each circuit in  $B$ . Each proxy circuit is a mirror circuit, which enables the efficient classical computation of its error-free outcome distribution. The proxy circuits are then executed on the target quantum processor to obtain an empirical outcome distribution for each circuit in  $P$ . The empirical and error-free outcome distributions for the proxy circuits are then passed into the `scarab` data analysis function, which first calculates the performance of each proxy circuit and then uses the performance of the proxy circuits to calculate the performance of the benchmarking circuits. `scarab` also enables the creation of performance summaries including volumetric benchmarking and capability region plots [62].

i.e., whether they are high-level circuits needing compilation or low-level circuits for that system, depends on the benchmark type.

- *Sampling and compilation parameters.*—Parameters that specify the number of circuits used in the fidelity estimation routines that `scarab` uses and (for full-stack benchmarks) compiler parameters. All such parameters have reasonable default options.

The `scarab` API (summarized in Fig. 1) has two parts: circuit creation and data analysis. The circuit creation part takes the above inputs and creates a set of *benchmarking circuits* ( $B$ ) and, from them, creates a set of *proxy circuits* ( $P$ ), to execute. `scarab` efficiently estimates the process fidelity of each benchmarking circuit using data from the proxy circuits. The simplest case is  $B = C$ , i.e., the benchmark created is designed to directly measure the performance of the tested hardware on the input circuits  $C$ , which is the case in `scarab`'s low-level benchmarks. We discuss the form of

$B$  for full-stack and subcircuit benchmarks below.

The benchmarking circuits  $B$  are not the circuits that are output for execution by `scarab` because simply executing the circuits in  $B$  will not enable the efficient estimation of these circuits' fidelities. Instead, `scarab` transforms  $B$  into the set of proxy circuits  $P$  that are to be executed, specified in pyGSTi's circuit format (and which can be converted to OpenQASM or qiskit circuits using pyGSTi's conversion functions). These proxy circuits are designed so that the performance of each circuit in  $P$  can be *efficiently* estimated and the process fidelity of each benchmarking circuit can be estimated from the performance of the proxy circuits. Each circuit in  $P$  is created by applying MCFE [63] to the circuits in  $B$ , and therefore each circuit in  $P$  is a *mirror circuit* [62], [63]. We plan to add the option to instead use Cliffordization [65], an alternative fidelity estimation approach that uses Clifford proxy circuits instead of mirror circuits, in a future version of `scarab`.

The proxy circuits output by  $P$  must be executed by the user

on their target quantum computing system (with no further compilation applied), recording the bit string observed in each execution of each circuit in  $P$ . That data is then processed by the data analysis function of `scarab`, producing an estimate for the process fidelity of each circuit in  $B$  (stored in a pandas DataFrame along with relevant metadata for the circuits) that can be easily manipulated by the user for custom analyses. `scarab` also contains a variety of built-in routines for results analysis and presentation, including *volumetric benchmarking* and *capability region* plots [62].

We now explain the three kinds of benchmark that `scarab` can create:

- *Low-level benchmarks.*—These benchmarks quantify the amount of noise in a set of input *low-level* circuits  $C$ . They do so by efficiently measuring the process fidelity of each circuit in  $C$ . The input circuits  $C$  must all already be compiled for the target system. For this benchmark type, `scarab`'s benchmarking circuits  $B$  are simply the input circuits ( $B = C$ ). The input circuits could be created by passing some high-level algorithmic circuits through a target system's built-in compilation algorithms, or the user may directly define low-level circuits of interest.
- *Full-stack benchmarks.*—These benchmarks efficiently measure the joint performance of a system's compiler and qubits. They do so by taking a set of input *high-level* circuits  $C$ , compiling them to the target system to create the benchmarking circuits  $B$ , and then measuring the process fidelity of those compiled (i.e., transpiled and routed) circuits to the intended, exact unitaries defined by  $C$ . This kind of benchmark can test both exact and approximate compilation algorithms, as demonstrated in Section VII. `scarab`'s full-stack benchmarks currently use the `qiskit` transpiler with user-chosen transpiler flags, but future enhancements may enable flexibility in the compilation software used.
- *Subcircuit benchmarks.*—These benchmarks efficiently measure the performance of varied-shape (i.e., varied in width and depth) subcircuits sampled from the input circuits, enabling benchmarking of contemporary systems with utility-scale circuits [69]. For this type of benchmark, the input circuits  $C$  must be already compiled for the target system, and the benchmarking circuits  $B$  are varied-shape subcircuits from each circuit in  $C$ . Tested circuit shapes can be specified by the user.

#### IV. RUNTIME SCALING

`scarab` is designed to enable the creation of scalable benchmarks, and we demonstrate this scalability by measuring the runtime of `scarab`. In particular, we measured the *classical processing time* ( $t_c$ ), which we define to be the time taken to generate the benchmark's proxy circuits  $P$ , compute any information that is needed to process the data from those circuits (which, in the case of `scarab`'s benchmarks, is a target bit string for each proxy circuit), and perform the data analysis to determine the performance

of both the proxy circuits  $P$  and the benchmarking circuits  $B$ . The time  $t_c$  therefore encompasses the complete classical computational cost of creating and analyzing the results of `scarab`'s benchmarks.

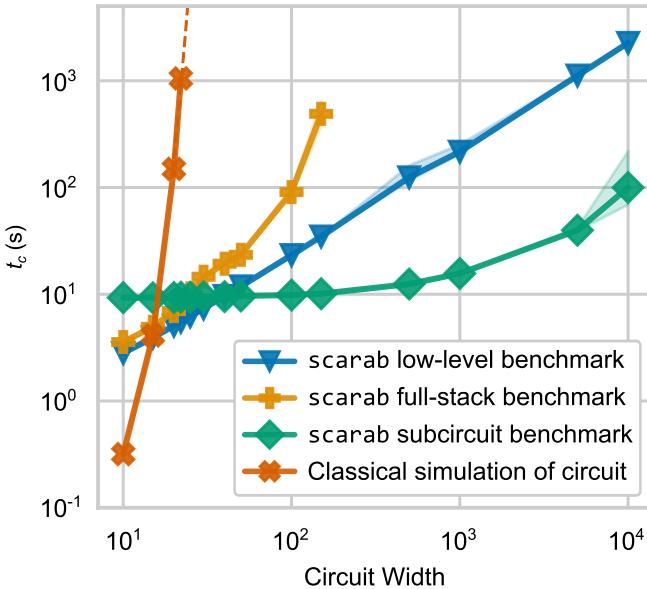
We measured  $t_c$  for the three different kinds of benchmark that we can create with `scarab`: low-level benchmarks, full-stack benchmarks, and subcircuit benchmarks. `scarab` creates a benchmark from a user-given circuit, and so to compute  $t_c$  we must select a circuit or circuit family. We computed  $t_c$  for  $n$ -qubit U3-CZ brickwork circuits [65] of depth 128 with  $n$  varied. We set  $|M_1| = |M_2| = |M_3| = 10$ . For each mirror circuit, we generate fake shot data (in order to compute the data processing time) from a uniform distribution over all  $2^n$  bit strings with 1024 shots. Fig. 2 shows the mean  $t_c$  for each kind of benchmark versus  $n$ . For low-level and subcircuit benchmarks we measured  $t_c$  up to  $n = 10000$ . We observe that  $t_c$  scales linearly for the low-level benchmarks (blue triangles, Fig. 2) and is still practical ( $t_c \approx 5000$  s) even for  $n = 10000$ . For the subcircuit benchmarks, we observe that  $t_c$  scales sublinearly (green diamonds, Fig. 2). This sublinear scaling arises from the fact that, in this test, we are creating benchmarks whose proxy circuits are subcircuits of the input  $n$ -qubit circuit with an  $n$ -independent shape (we create subcircuits of shapes  $\{(w_i, d_i)\} = \{2, 4, 6\} \times \{2, 4, 8\}$ ).

To test the runtime of `scarab`'s full-stack benchmarks, we chose IBM Fez as the target quantum backend and used the `qiskit` transpiler with `optimization_level` set to 3. The qubit count of IBM Fez limits these numerical experiments to 150 qubits. The time  $t_c$  for the full-stack benchmark (orange squares, Fig. 2) is still practical even at  $n = 150$  ( $t_c \approx 800$  s). We also report the time taken to classically simulate the U3-CZ brickwork circuits we used as input to `scarab` (orange crosses, Fig. 2). That simulation, implemented using `qiskit` and which is required by many other benchmarks, has an exponentially growing cost. In contrast, the classical processing cost of `scarab` benchmarks makes them practical even with utility-scale circuits.

#### V. DEMONSTRATING RELIABLE FIDELITY ESTIMATION

We now demonstrate that `scarab` reliably estimates the process fidelity  $F$ , and we explain how this differs from metrics based on classical fidelity used in other benchmarks. To demonstrate that `scarab`'s benchmarks reliably estimate process fidelity, we simulated benchmarks created using `scarab` and compared the estimated process fidelities to the true values of  $F$ . For these demonstrations, the input circuits to `scarab` are circuits from the QED-C's Application-Oriented Benchmarking suite [55] (see the appendices for a summary of that suite). This is therefore also a demonstration of using `scarab` to convert an existing benchmark into a *scalable* benchmark with a well-motivated success metric: process fidelity.

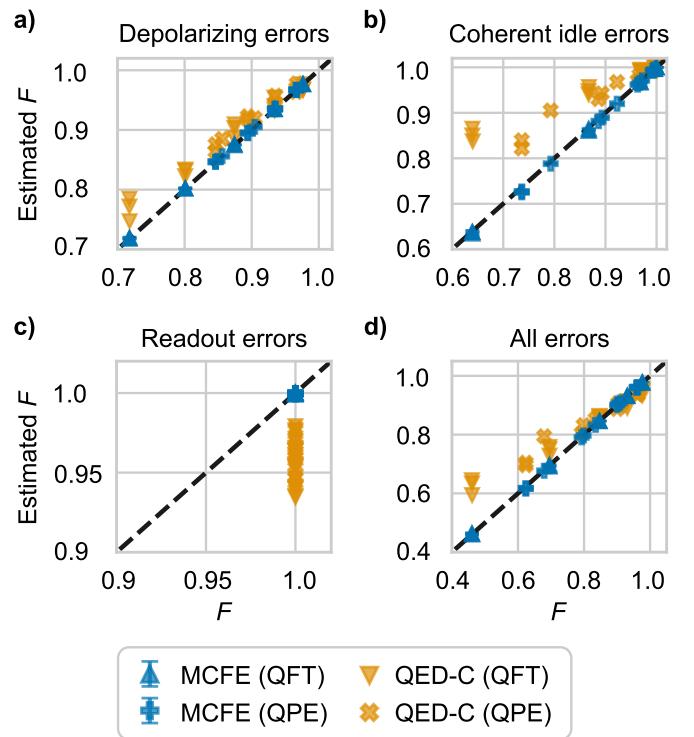
In these simulations, we used  $n$ -qubit circuits with  $n = 3$  to  $n = 6$  from the quantum phase estimation (QPE) and the quantum Fourier transform (QFT) benchmarks of the QED-C's suite. We set  $|M_1| = |M_2| = |M_3| = 1000$ , and we use



**Fig. 2: Classical processing time for `scarab`.** The time ( $t_c$ ) taken by the classical processing in `scarab`, consisting of the time to turn a circuit input into `scarab` into the “proxy circuits” to be run, compute all the information about those circuits needed to analyze data from those circuits (e.g., error-free outcome distributions), and to calculate the performance of the proxy and benchmarking circuits using the `scarab` data analysis function; versus the circuit’s width (number of qubits,  $n$ ). We show the mean  $t_c$  (markers), and the best and worst  $t_c$  (shaded region) over different circuits, for the three kinds of benchmarks created by `scarab`: low-level benchmarks (blue triangles), full-stack benchmarks (yellow pluses), and subcircuit benchmarks (green diamonds). We compare the scaling of  $t_c$  for `scarab` to the time to classically simulate the same quantum circuits using `qiskit_aer` (orange crosses) from which we created efficient benchmarks with `scarab`. This classical simulation is a key step in many other benchmarks, and, unlike the scaling of  $t_c$  for `scarab`, the classical simulation scales exponentially (fit line) and is therefore impractical for many-qubit circuits.

`qiskit_aer` to simulate the noisy circuits. The circuits are compiled to, and the noise model is defined on, the native gate set for many IBM Q systems: the  $\{X, SX, RZ, CZ\}$  gate set. Fig. 3 (blue markers) compares the true process fidelity to the estimate from `scarab`. We make these comparisons for four different noise models (detailed further below), corresponding to the four panels in the figure. For all noise models, the `scarab` process fidelity estimate is in close agreement with the true process fidelity (error bars are 1 standard deviation, and in most cases they are smaller than the data markers). This demonstrates using `scarab` to convert existing unscalable benchmarks into scalable benchmarks, and that `scarab` creates benchmarks that robustly estimate process fidelity.

Many existing benchmarks—including many of the QED-C benchmarks—compute success metrics based on the classical fidelity. Classical fidelity and process fidelity measure



**Fig. 3: Estimating process fidelity with `scarab`.** Simulations demonstrating that benchmarks created with `scarab` reliably estimate a circuit’s process fidelity  $F$  in the presence of complex device errors. We show the  $F$  estimated using `scarab` benchmarks (blue markers) versus the true  $F$  for two classes of circuit (QFT and QPE circuits) under different noise models. Error bars are one standard deviation and calculated using a non-parametric bootstrap. To demonstrate the difference between process fidelity  $F$  and the normalized classical fidelity  $\bar{F}_c$  estimated by other benchmarks, we also plot  $\bar{F}_c$  versus  $F$  (orange markers) for these circuits. We observed that `scarab` benchmarks accurately estimate the process fidelity in the presence of (a) depolarizing errors on gates, (b) coherent errors on gates, (c) readout errors, and (d) all three kinds of errors. The normalized classical fidelity can be significantly larger or smaller than the process fidelity, depending on the details of the noise model.

different aspects of performance. Unlike classical fidelity, process fidelity is sensitive to errors that will affect any input state and final measurement, whereas the classical fidelity is computed for a particular input state and measurement. The process fidelity is therefore more appropriate in cases where a circuit will be used as a subroutine—like QPE and the QFT—and is also arguably more relevant when the circuits being used to create benchmarks are proxies for larger utility-scale circuits (e.g., a small QFT being a proxy for a large QFT used in a useful algorithm). By construction, process fidelity does not quantify SPAM errors, whereas classical fidelity includes a contribution from those errors. We note, however, that `scarab`’s benchmarks do enable estimation of SPAM error (and separating it from circuit error), although `scarab`’s

analysis pipeline does not currently report estimates of SPAM errors.

We demonstrate the difference between classical fidelity and process fidelity in Fig. 3. We show how normalized classical fidelity (orange markers) deviates from process fidelity under different noise models. Fig. 3(a) compares process fidelity and normalized classical fidelity for a noise model with depolarizing noise of strength  $\lambda_{1Q} = 0.0005$  and  $\lambda_{2Q} = 0.005$  on all one- and two-qubit gates, respectively. The normalized classical fidelity overestimates the process fidelity by up to 0.07 or 10%. Fig. 3(b) shows the comparison for a noise model with only readout error, with a rate of  $\epsilon = 0.01$ . The process fidelity is unity in this case, but  $\bar{F}_c$  is sensitive to readout error and therefore underestimates the process fidelity by up to 0.1 or 10%. Fig. 3(c) shows the comparison for a noise model where every idle gate experiences a  $\theta_{\text{idle}} = 0.005$  radian  $Z$  rotation. These coherent idle errors create phase errors before computational basis measurements, which are not captured by classical fidelity. As a result, the classical fidelity overestimates the process fidelity by up to 0.25, or 40%, which is greater than in the case of depolarizing noise. Fig. 3(d) shows the combined effect of all three error sources: depolarizing noise, readout error, and idle rotation. For smaller circuits, the effect of the depolarizing noise and idle rotation is smaller and the readout error is dominant, causing the classical fidelity to underestimate the process fidelity. For larger circuits, the depolarizing errors and idle rotations contribute more to the total error in the circuit, which leads the classical fidelity to overestimate the process fidelity.

## VI. SCALABLE HAMILTONIAN SIMULATION BENCHMARKS

In this and the following two sections, we showcase three applications for `scarab`, using each kind of benchmark that `scarab` can create. We show how `scarab`'s benchmarks can be used to efficiently interrogate the performance of quantum computers and how hardware noise and algorithmic error combine to impact a quantum algorithm. In each case, we demonstrate `scarab` using both simulations and IBM Q experiments.

### A. Hamiltonian simulation

In our first application of `scarab`, we show how it can be used to create low-level benchmarks for Hamiltonian simulation circuits. These benchmarks are designed to enable exploration of the hardware and algorithmic accuracy trade-offs for Hamiltonian simulation, where more accurate algorithms can be used at the cost of more gates—and therefore more noise.

In Hamiltonian simulation algorithms, a core subroutine is the approximate implementation of a unitary  $U$  generated by applying a Hamiltonian  $H$  for some time  $t$ , given by

$$U = \exp(-iHt). \quad (4)$$

Hamiltonian simulation algorithms do not typically apply  $U$  exactly, but instead some unitary  $\tilde{U}$  that approximates  $U$ , i.e.,  $\tilde{U} \approx U$ . The approach to approximating  $U$  that we consider

here is Trotterization. If  $H = \sum_{j=1}^k H_j$ , then the first-order Trotterization is [73]

$$\tilde{U} = \left( \prod_{j=1}^k \exp(-iH_j t/m) \right)^m. \quad (5)$$

The magnitude of the error in this approximation is given by

$$U = \tilde{U} + O\left(\frac{t^2}{m}\right). \quad (6)$$

By increasing the number of Trotter steps ( $m$ ), the discrepancy between  $U$  and  $\tilde{U}$  can be decreased, but at the cost of deeper circuits. In addition to the first-order Trotterization, we will also consider second-order Trotterization [73], [74], where

$$\tilde{U} = \left[ \left( \prod_{j=1}^k \exp\left(\frac{-iH_j t}{2m}\right) \right) \left( \prod_{j=1}^k \exp\left(\frac{-iH_{k+1-j} t}{2m}\right) \right) \right]^m. \quad (7)$$

The magnitude of the error in second-order Trotterization is given by

$$U = \tilde{U} + O\left(\frac{t^3}{m^2}\right). \quad (8)$$

There are three superoperators that are important for quantifying the performance of a noisy implementation of a Trotterization circuit  $c$ :

- The superoperator  $\mathcal{U}$  giving the ideal Hamiltonian evolution with the action  $\mathcal{U}[\rho] = U\rho U^\dagger$ .
- The superoperator  $\tilde{\mathcal{U}}$  for the approximation to  $U$  implemented by an error-free (i.e., noiseless) execution of the Trotterized circuit  $c$ , with the action  $\tilde{\mathcal{U}}[\rho] = \tilde{U}\rho\tilde{U}^\dagger$ .
- The noisy Trotter evolution  $\Upsilon$ , which is the superoperator corresponding to the noisy implementation of the circuit  $c$ .

These superoperators enable us to define three process fidelities that summarize different aspects of the error in a Trotterization circuit:

- The *algorithmic process fidelity*  $F(\mathcal{U}, \tilde{\mathcal{U}})$ , which captures the error due to the Trotter approximation.
- The *noise process fidelity*  $F(\tilde{\mathcal{U}}, \Upsilon)$ , which captures the error in implementing the Trotterized circuit due to hardware noise.
- The *full process fidelity*  $F(\mathcal{U}, \Upsilon)$ , which captures the combine impact of both Trotter approximation error and hardware noise.

`scarab` enables efficiently measuring the noise process fidelity, by creating a low-level benchmark from the Trotter circuit  $c$ , which we demonstrate below with simulations and experimental data. First, however, we consider how we can estimate the full process fidelity. Directly computing the full process fidelity requires access to  $\mathcal{U}$ , which is generally infeasible as it is an exponentially-large matrix (and we also do not have access to  $\Upsilon$  without exponentially-expensive

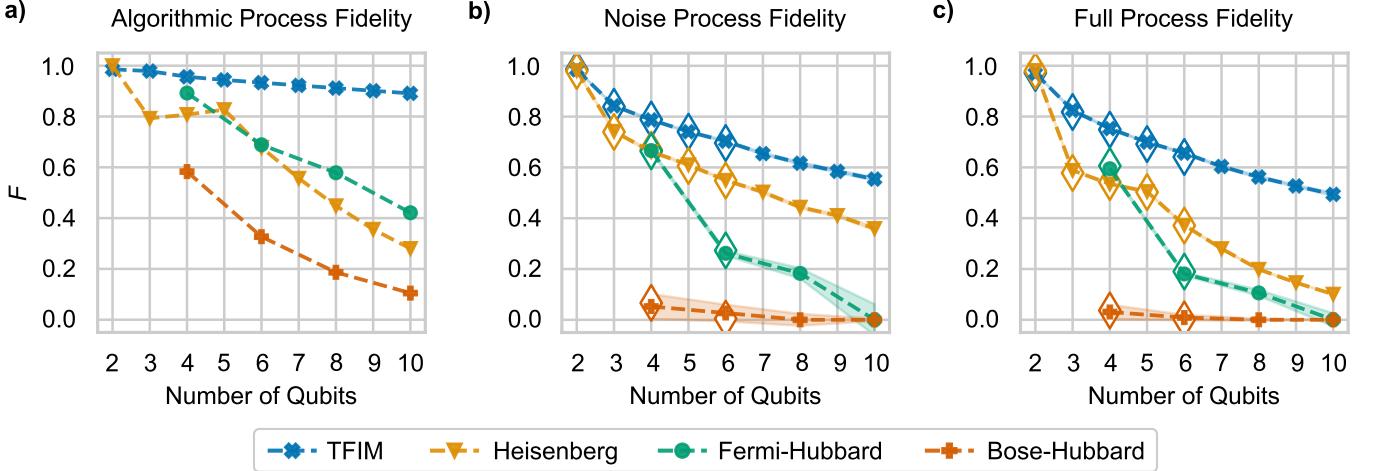


Fig. 4: **Estimating the impact of noise and algorithm approximation with `scarab`'s efficient low-level benchmarks.** Using `scarab`, we created low-level benchmarks from first-order Trotter circuits with four different  $n$ -qubit Hamiltonians—TFIM, Heisenberg, Fermi-Hubbard, and Bose Hubbard Hamiltonians—from HamLib. (a) The algorithmic process fidelity, i.e., the fidelity between the Trotter circuit’s (noise-free) unitary and the ideal unitary evolution for that Hamiltonian, versus  $n$ . (b) The noise process fidelity estimated by the `scarab` benchmarks (solid markers), which is the process fidelity between the noisy Trotter circuit and the noise-free unitary that circuit implements, and its exact value (open diamonds) up to  $n = 6$ . (c) The estimated full process fidelity (solid markers)—i.e., the process fidelity between the ideal unitary evolution and the noisy implementation of the Trotter evolution, approximated as the product of the measured noise process fidelity and the computed algorithmic process fidelity—and its exact value (open diamonds) up to  $n = 6$ . For both the noise and full process fidelities, we observe close agreement between the `scarab` estimate and the true values. Shaded regions around the `scarab` estimates for the process fidelities are 1 standard deviation calculated from a non-parametric bootstrap.

tomography [70]). We therefore propose approximating the full process fidelity as a product of the noise process fidelity and algorithmic process fidelity:

$$F(\mathcal{U}, \Upsilon) \approx F(\mathcal{U}, \tilde{\mathcal{U}}) \cdot F(\tilde{\mathcal{U}}, \Upsilon). \quad (9)$$

Computing the right hand side of Eq. (9) directly requires access to the  $O(2^n)$  unitary matrices  $U$  and  $\tilde{U}$ , to compute  $F(\mathcal{U}, \tilde{\mathcal{U}})$ , which is not scalable. However, Trotter error is well studied [75], [76], and we conjecture that it may be possible to efficiently estimate the algorithmic process fidelity even for large  $n$ .

### B. Demonstration in simulations

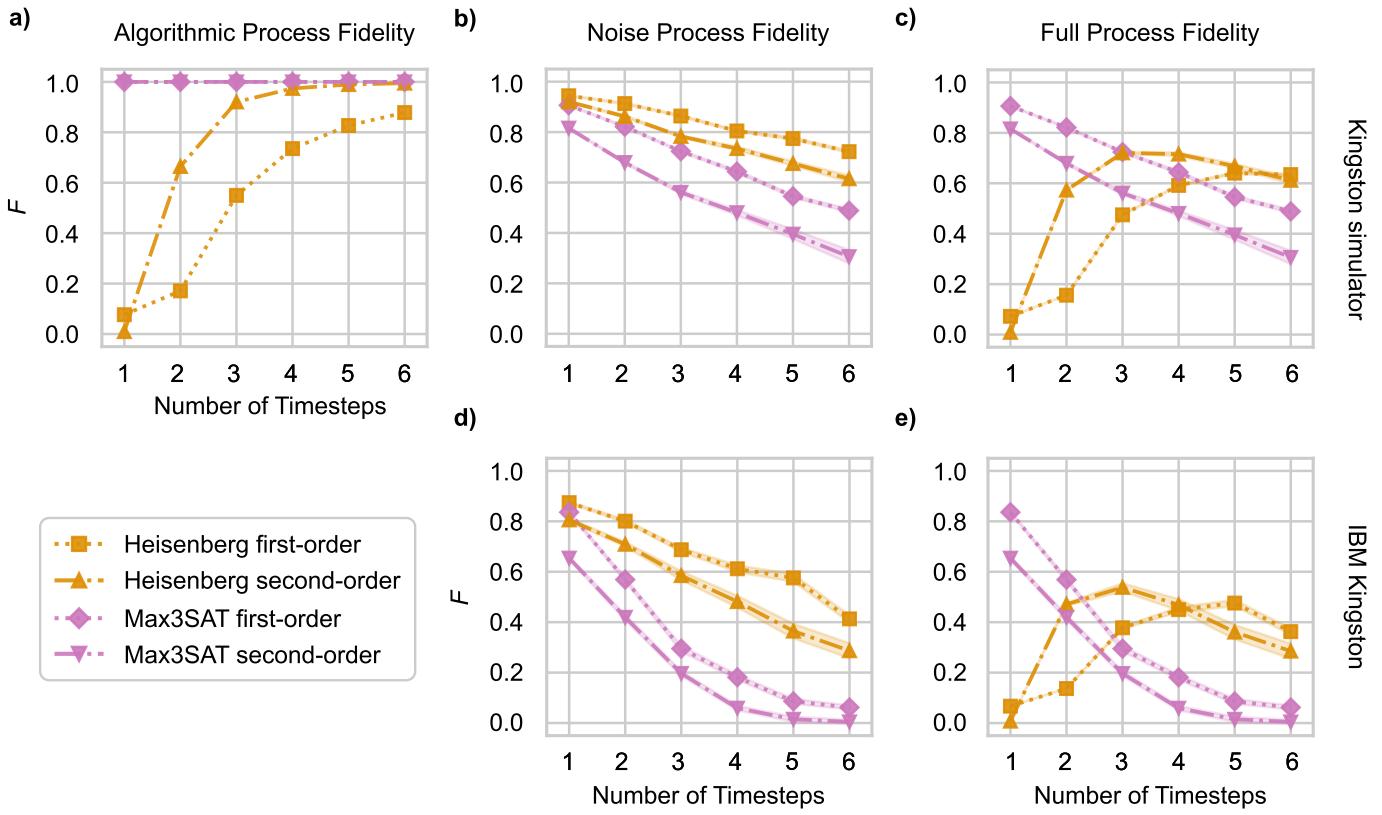
We demonstrate scalable Trotterization benchmarks created with `scarab` and explore the accuracy of Eq. (9) using simulations (all these benchmarks use  $|M_1| = |M_2| = |M_3| = 100$ ). We consider first-order Trotterized circuits for the transverse field Ising model (TFIM), Heisenberg, Bose-Hubbard, and Fermi-Hubbard Hamiltonians on  $n = 2$  to  $n = 10$  qubits, using Hamiltonians from HamLib [77]. More information on these Hamiltonians and the specific choice of parameters for each can be found in the appendices. We used code from the QED-C HamLib benchmark [58] in order to select specific Hamiltonians from the HamLib datasets and generate the first-order Trotter circuits. The circuits are compiled to, and the noise model is defined on, the  $\{X, SX, RZ, CZ\}$  gate set. We simulated a noise model with depolarizing errors on one- and two-qubit gates with depolarizing parameters of  $\lambda_{1Q} = 0.0005$

and  $\lambda_{2Q} = 0.005$ , respectively, with coherent over-rotation errors on the  $X$  and  $SX$  gates of  $\theta_X = \theta_{SX} = 0.01$  radians, and with  $\epsilon = 0.01$  readout error.

In Fig 4, we show the (a) algorithmic, (b) noise, and (c) full process fidelities for these Trotter circuits and this error model. Fig. 4(b) shows the noise process fidelity estimated using `scarab` low-level benchmarks created from these circuits (solid markers) and the exact value of the noise process fidelity for  $n \leq 6$  qubits (open diamonds). As in our simulations of Fig. 3, we observe close agreement between `scarab`'s estimate of the noise process fidelity and its true value. In Fig. 4(c), we show estimates of the full process fidelity from `scarab`'s estimates of the noise process fidelities and the exact algorithmic process fidelities in Fig. 4(a), using Eq. (9). We compare these estimates (solid markers) to the exact values of the full process fidelities (open diamonds), again seeing close agreement.

### C. Demonstration in experiment

We used `scarab` to generate Hamiltonian simulation benchmarks that explore how to maximize the accuracy of the Hamiltonian simulation in-situ, in experiments on IBM Kingston. We explore how to balance the accuracy in the approximation of  $U$  with the effects of hardware error (see Refs. [78]–[80] for similar investigations). We can quantify this trade-off by estimating the full process fidelity, which is sensitive to both Trotter error and hardware noise. We used 5-qubit Heisenberg and Max3SAT Hamiltonians and Trotter-



**Fig. 5: Quantifying noise and algorithmic tradeoffs using scarab benchmarks.** Using scarab, we created low-level benchmarks from first-order and second-order Trotter circuits for two 5-qubit Hamiltonians (Heisenberg and Max3SAT) with varying number of time steps. These benchmarks were created for IBM Kingston, and both run on IBM Fez and IBM’s simulator of IBM Kingston. (a) The algorithmic process fidelity versus the number of time steps for each Hamiltonian and both first- and second-order Trotter circuits. We used scarab to estimate the noise process fidelity with (b) the simulator of IBM Kingston and (d) IBM Kingston. In all cases, we observe that the noise fidelity decreases as the number of time steps increases, due to increasing depth of the circuit. To quantify the optimal trade-off between noise and algorithmic fidelity, we estimated the full process fidelity for (c) the simulation of IBM Kingston and (d) IBM Kingston. For the Heisenberg Hamiltonian, we find that the optimal algorithm parameters are a second-order Trotter circuit with 3 time steps, in both the simulation and the experiment. Shaded regions around the scarab estimates for the process fidelities are 1 standard deviation calculated from a non-parametric bootstrap.

zation at first or second order. We used between 1 and 10 time steps ( $m$ ) for each Trotterization order. We used scarab to create low-level benchmarks ( $|M_1| = |M_2| = 200, |M_3| = 10$ ) from these Trotter circuits, and ran them on IBM Kingston as well as a qiskit\_aer simulation of IBM Kingston.

Figure 5 shows the results of these experiments and simulations. Figure 5(a) shows the algorithmic process fidelity, versus the number of time steps, for each Trotter order and each Hamiltonian. This was calculated exactly using simulations. For the Heisenberg Hamiltonian, we observe that increasing the number of time steps decreases the Trotter error, as expected. For the Max3SAT Hamiltonian, all terms commute and therefore  $\tilde{U} = U$  so the algorithmic process fidelity is unity. Fig. 5(b) and (d) show the noise process fidelity in simulation and experiment, respectively, estimated using scarab.

Increasing the order or the number of time steps decreases the noise process fidelity, because the circuits increasing in

depth. However, increasing the order of the Trotterization or the number of time steps increases the algorithmic process fidelity in most cases (see Fig. 5(a)) for the Heisenberg Hamiltonian, and so an decrease in noise process fidelity could be offset by a larger increase in algorithmic process fidelity. To quantify this, we estimated the full process fidelity (using Eq. (9)), with these estimates shown in Fig. 5(c) and (e). For Max3SAT, the full process fidelity is maximized with 1 time step and first-order Trotterization—because the algorithmic fidelity is unity for all cases. However, for the Heisenberg Hamiltonian, we find that the full process fidelity is maximized with second-order Trotterization with 3 time steps.

## VII. TESTING COMPILERS WITH SCALABLE FULL-STACK BENCHMARKS

We now demonstrate using scarab to create full-stack benchmarks that enable efficient in-situ testing of compiler optimizations that aim to balance the impacts of noise and

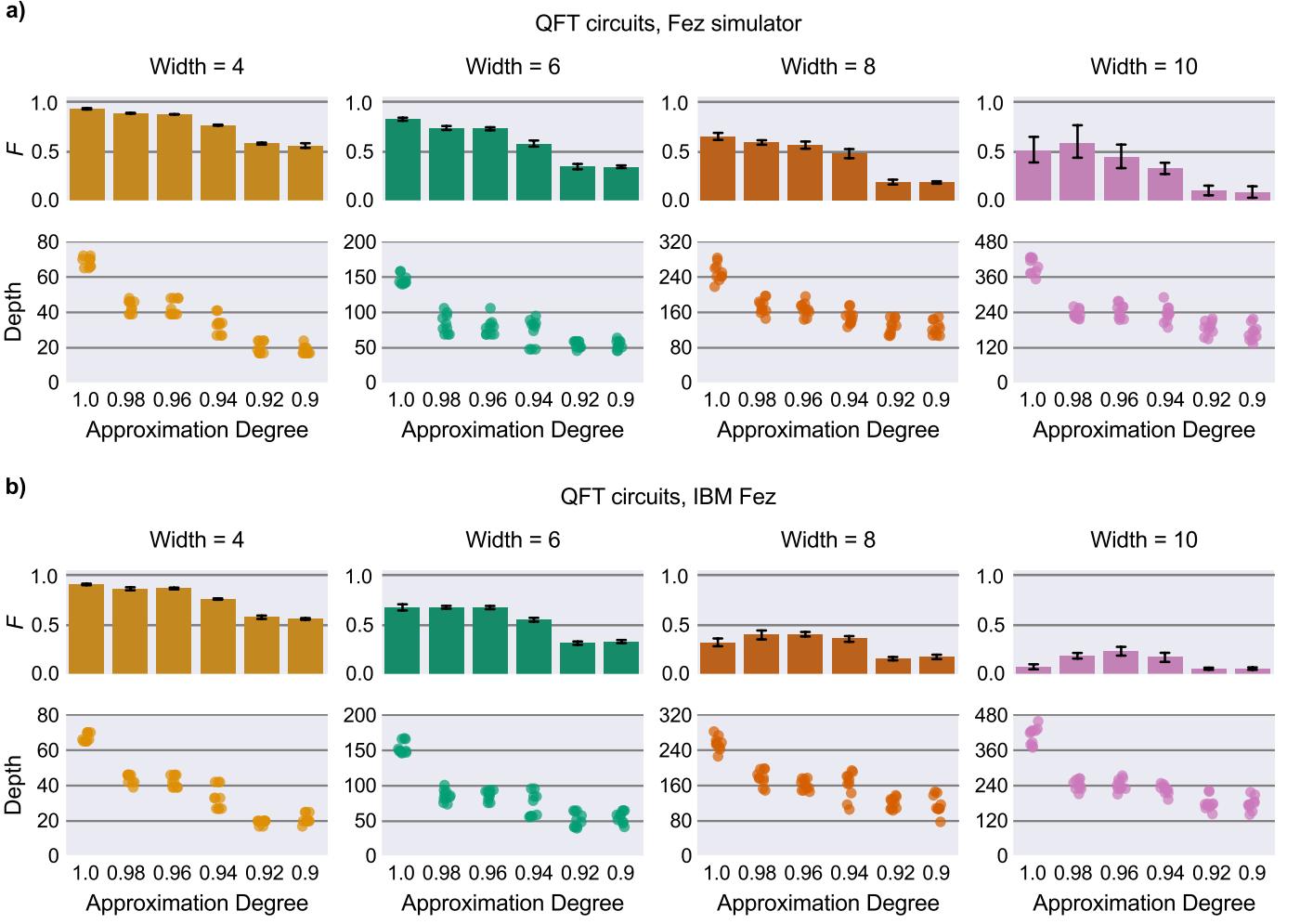
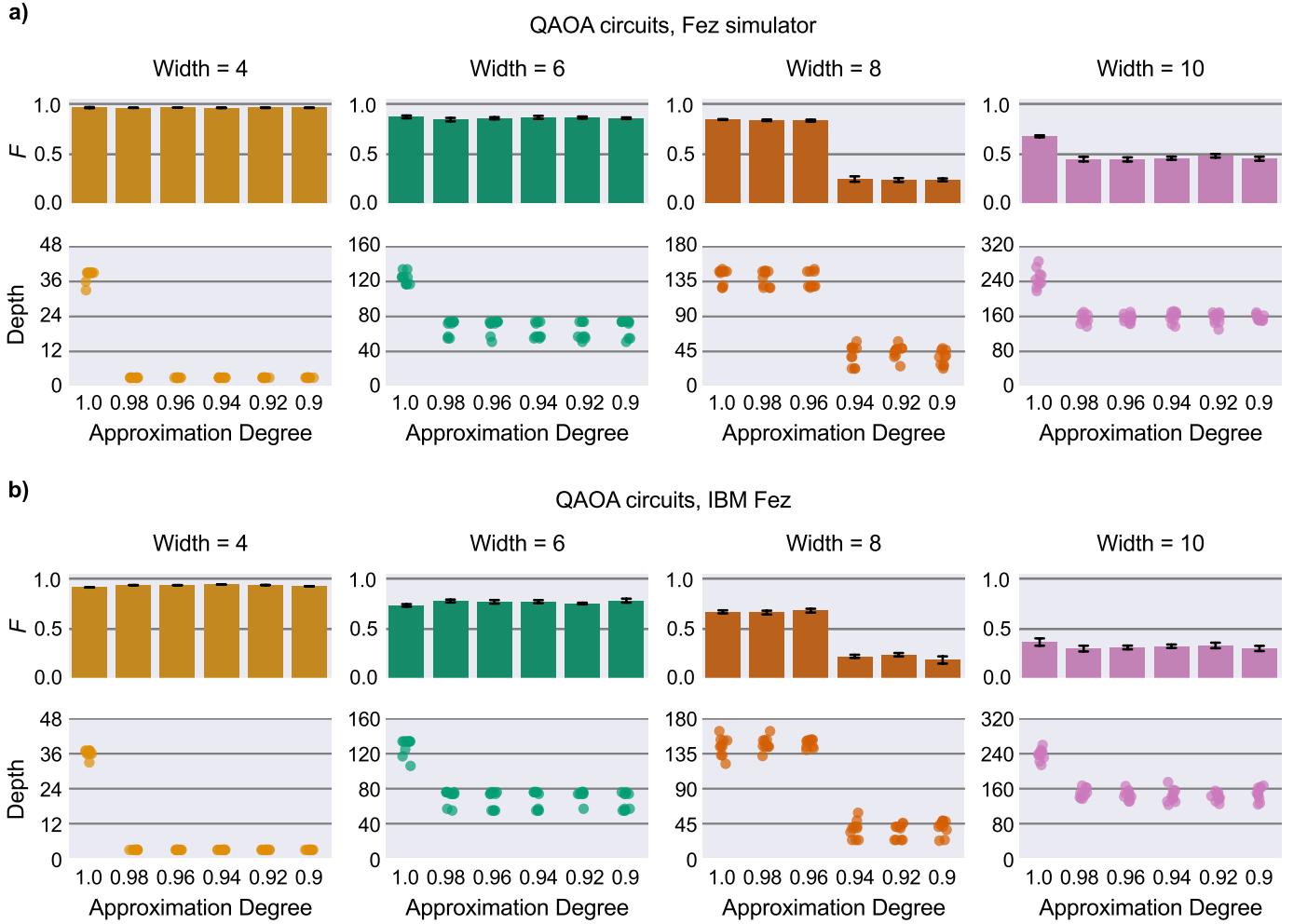


Fig. 6: **Testing approximating compilation algorithms using scarab’s full-stack benchmarks for the QFT.** The efficacy of approximate circuit compilation for QFT circuits, for IBM Fez and a simulation of IBM Fez that uses IBM’s noise model for this system. We show estimates of the process fidelities of compiled QFT circuits, to the ideal, approximation-free unitaries, for  $n = 2$  up to  $n = 10$  qubits and with varying approximation degree in the compilation algorithm (error bars are one standard deviation). We also show the depths of the compiled circuits for each approximation degree. Error bars represent a 95% confidence interval calculated via a nonparametric bootstrap.

algorithmic approximation. Compilation translates a circuit into a system’s native gate set and topology, and compilation algorithms are typically designed with the aim of finding a circuit that will execute with low error (i.e., high fidelity) on that system, e.g., by minimizing circuit depth or the number of two-qubit gates. A compilation of a circuit can either be exact or approximate. An *exact* compilation of a circuit  $c$  creates another circuit  $c'$  that implements the same unitary as  $c$ , i.e.,  $U(c') = U(c)$  where  $U(c)$  and  $U(c')$  are the unitaries implemented by the circuits  $c$  and  $c'$ , respectively. In contrast, *approximate* compilation of a circuit  $c$  creates another circuit  $c'$  that only approximately implements the same unitary as  $c$ , i.e.,  $U(c') \approx U(c)$ . Approximate compilation is a weaker condition, potentially allowing for compiling the input circuit into a low-level circuit that will execute with higher fidelity, e.g., because the circuit is much shallower. This enables trading off intrinsic error—the difference between  $U(c')$  and  $U(c)$ —

and errors due to hardware noise. For instance, a compiler could discard small controlled rotations, or small-angle (e.g.,  $\pi/256$ ) single-qubit rotations, because those gates are expected to cause more error than accrued by not implementing them at all. This tradeoff has been investigated, e.g., in [81].

We explored the in-situ performance of qiskit’s approximate compilation algorithm using scalable full-stack benchmarks ( $|M_1| = |M_2| = |M_3| = 10$ ) created with scarab, designed for and executed on IBM Fez. We created full-stack benchmarks from two kinds of algorithmic circuits: QFT circuits and quantum approximate optimization algorithm (QAOA) circuits (obtained from the qiskit library). For the QAOA circuits, the cost operator we used corresponds to a random  $GNP(n, 2\ln(n)/n)$  graph (we used the default qiskit mixing operator). We used one repetition of each operator and initialize the mixing angles uniformly between 0 and  $\pi$ . For compilation, we used the qiskit



**Fig. 7: Testing approximating compilation algorithms using `scarab`'s full-stack benchmarks for QAOA.** The efficacy of approximate circuit compilation for QAOA circuits, for IBM Fez and a simulation of IBM Fez that uses IBM's noise model for this system. We show estimates of the process fidelities of compiled QAOA circuits, to the ideal, approximation-free unitaries, for  $n = 2$  up to  $n = 10$  qubits and with varying approximation degree in the compilation algorithm (error bars are one standard deviation). We also show the depths of the compiled circuits for each approximation degree. Error bars represent a 95% confidence interval calculated via a nonparametric bootstrap.

transpiler with `optimization_level` set to 3. We varied the `approximation_degree` parameter from 0.9 to 1.0, with 1.0 corresponding to exact compilation. We executed these benchmarks on IBM Fez as well as a `qiskit_aer` simulation of IBM Fez. Since `qiskit`'s transpilation is stochastic (i.e., the same compiled circuit is not always produced for a fixed input circuit) we tested 10 transpilations of each high-level circuit.

Figures 6 and 7 show the results of these experiments and simulations. In most cases, the process fidelity is not improved by using approximate compilation, i.e., the process fidelity for `approximation_degree = 1.0` is the largest or within  $1\sigma$  of the largest estimated process fidelity. However, for the experimental results with the QFT on 8 and 10 qubits, the maximum process fidelity is achieved with approximate compilation. The  $n$ -qubit QFT contains controlled rotations with angles that decrease with increasing  $n$ , and so as  $n$

increases it is possible to improve the noisy QFT's process fidelity by dropping these gates—which, because they are close to the identity gate, has a only small impact on the intrinsic error of the circuit—rather than implementing them imperfectly. These full-stack benchmarks enable quantify this effect in situ, with a system's actual noise processes.

## VIII. PREDICTING ALGORITHM PERFORMANCE WITH SUBCIRCUIT BENCHMARKS

In this section we demonstrate `scarab`'s subcircuit benchmarks and `scarab`'s ability to turn a algorithmic circuit directly into an efficient benchmark. Many interesting circuits are so large that they cannot be implemented with significantly non-zero fidelity on contemporary systems. For example, many of the most promising algorithms for quantum computers appear to require very deep circuits on thousands of qubits [8]. One approach to creating benchmarks that can be run on

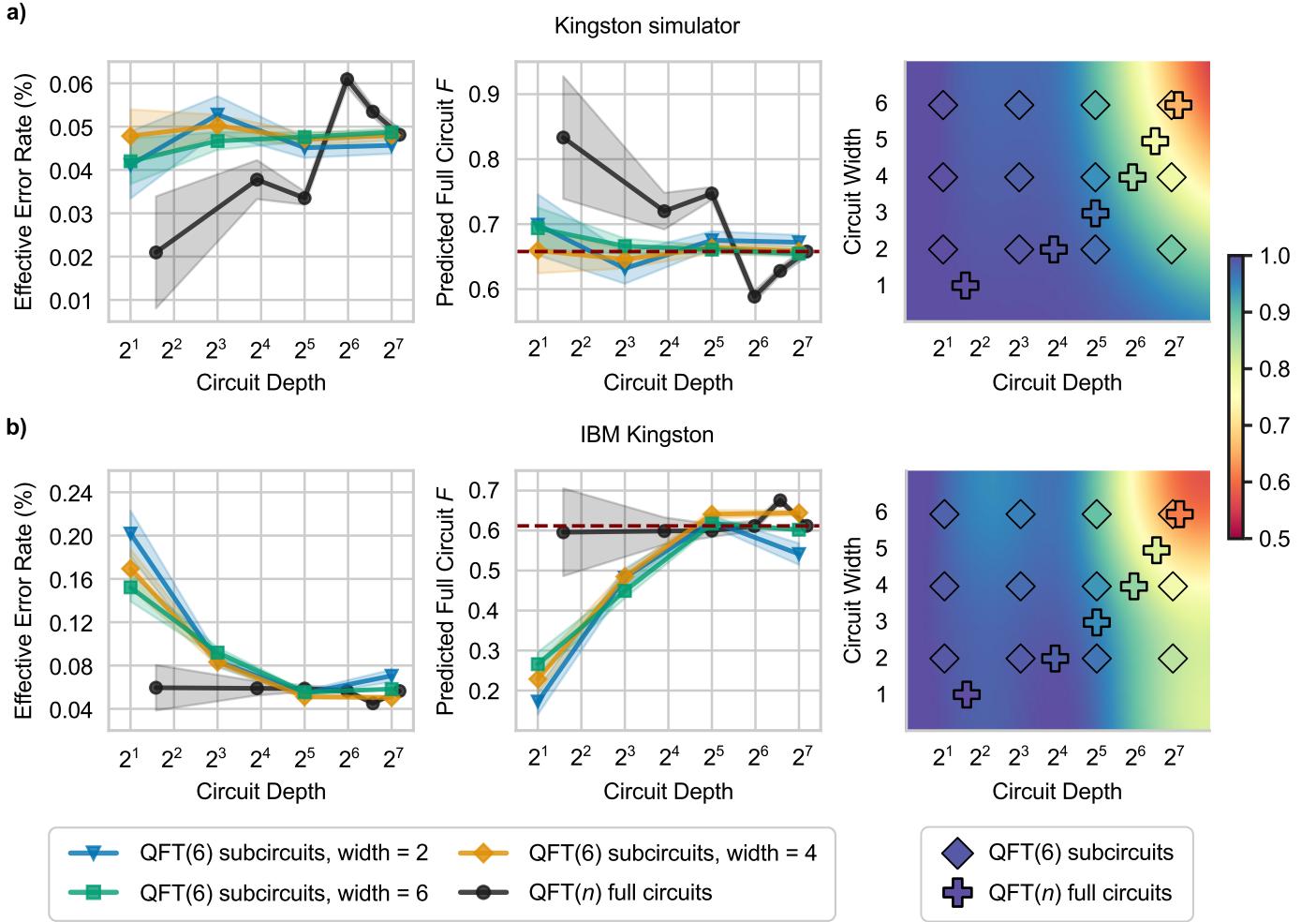


Fig. 8: **Demonstrating scalable subcircuit benchmarking using scarab.** scarab enables the creation of scalable benchmarks from a potentially-large input circuit  $c$  in which the process fidelities of varied-shape subcircuits of  $c$  are estimated. Here we demonstrate this in simulation (upper row) and experiment (lower row) for a small example: a 6-qubit QFT circuit, QFT(6). scarab also enables scalable benchmarks in which the input size of an algorithm is varied, and we also demonstrated this by varying the input size  $n$  of the QFT, QFT( $n$ ). The rightmost plots show the measured process fidelities of subcircuits of QFT(6) of each of 12 different shapes (diamonds) as well as the measured process fidelities of the QFT( $n$ ) circuits (pluses). The heatmap is the predictions for process fidelity obtained by applying Gaussian process regression to the subcircuit data. The leftmost and central plots show effective error rates (see main text) and their predictions for the QFT(6)'s process fidelity, obtained from the subcircuits of each shape (colored points) and the QFT( $n$ ) circuits (black lines). Error bars in the EER and predicted full circuit process fidelity plots are one standard deviation calculated from a non-parametric bootstrapped distribution.

contemporary systems but that are built from potentially very large circuits is to run subcircuits “snipped” out of the given circuit(s) [69]. This method is implemented by scarab’s subcircuit benchmarks. Performance on those circuits can potentially enable the prediction of performance on the input (potentially very large) circuit, and provide a principled approach to measuring progress towards implementing that large circuit with low error. Another approach, possible for circuits with tunable input size or number of qubits  $n$ , is to measure performance as  $n$  is varied. This is also possible to do efficiently, using scarab’s low-level benchmarks.

To demonstrate both approaches to algorithmic benchmarking, we created a subcircuit benchmark from a 6-qubit QFT,

and we also created low-level benchmarks from the  $n$ -qubit QFT with  $n = 2, 3, 4, 5, 6$ . The subcircuit benchmarks used sampling parameters  $|M_1| = |M_2| = 50, |M_3| = 100$  and the  $n$ -qubit QFT benchmarks used  $|M_1| = |M_2| = |M_3| = 100$ . We compiled the  $n$ -qubit QFT for IBM Kingston, used the compiled 6-qubit circuit to create a subcircuit benchmark with scarab, and used the compiled  $n$ -qubit QFTs to create low-level benchmarks with scarab directly from those circuits. We ran these benchmarks on IBM Kingston and also simulated them using the `qiskit_aer` simulation of IBM Kingston. One of the parameters of the subcircuit benchmark type is the circuit shapes to use: we created subcircuits at shapes  $(w, d) \in (2, 4, 6) \times (2^1, 2^3, 2^5, 2^7)$  and sampled 30 subcircuits

of each shape. We chose the 6-qubit QFT because it is a small subroutine that we can directly run on contemporary systems, and this enable us to also estimate the process fidelity of the 6-qubit QFT (using `scarab`) and compare its observed performance to extrapolations from subcircuit benchmarks, with each circuit shape, created from that 6-qubit QFT.

Figure 8 shows the results of these simulations (panel (a)) and experiments (panels (b)). In the rightmost plot of Fig. 8 (a) and (b) we show the estimated mean process fidelities of the subcircuits of each shape (diamonds), as well as the estimated process fidelities of the  $n$ -qubit QFT circuits (pluses), for the simulated and experimental data, respectively. These results are presented on a volumetric plot, i.e., shown on circuit depth  $\times$  circuit width axes. We also show the predictions of a Gaussian process regression (GPR) model fit to the subcircuit data (heat map) [82]. The measured process fidelities versus circuit shape, as well as the GPR-derived heat map, provide a visual summary of these system’s performance on QFT circuits. Furthermore, this GPR model is one approach to predicting the performance of other circuits (e.g., the 6-qubit QFT) from observed process fidelities at a discrete set of circuit shapes [82]. Here, however, we will focus on predicting the 6-qubit QFT’s process fidelity using *effective error rates*.

The effective error rate and its prediction for the full circuit’s process fidelity is as follows. For  $K$  subcircuits of shape  $(w, d)$ , with measured process fidelities  $\{F_{w,d,i}\}_i = 1^K$ , the associated effective error rate is [69]

$$\epsilon_{w,d} = 1 - \left( \prod_{i=1}^K F_{w,d,i} \right)^{1/(wdK)}. \quad (10)$$

This error rate is then used to predict the fidelity of the full circuit as

$$F = (1 - \epsilon_{w,d})^{w_c d_c}, \quad (11)$$

where  $w_c$  and  $d_c$  are the width and depth of the full circuit, respectively.

In the left and central plots of Fig. 8 we show the effective error rates and predicted full circuit fidelity for the subcircuit benchmarks of each circuit shape, as well as for the varied-size QFT circuits, as a function of circuit depth. For the simulated data (upper row), we observe that the subcircuits of each shape (colored lines) accurately predict the full circuit’s fidelity (red dashed lines). In contrast, the predictions of the varied-size QFT circuits (black lines) do not accurately predict the full circuit’s fidelity, for the simulated data, but with decreasing prediction error as the input size of the QFT circuit increases. However, for the experimental data (lower row), we observe the opposite effect: the subcircuits inaccurately estimate the full circuit’s fidelity—with improving accuracy as the circuits get deeper—and the varied-size QFT circuits accurately predict  $F$ . This is, to our knowledge, the first comparison of these two approaches to predicting a circuit’s fidelity from smaller circuits. This comparison was enabled by `scarab`’s flexible interface, and so we anticipate that it will enable more detailed studies of these—and other—benchmarking methods, and a better

understanding of the merits and regime of reliability for different, complementary benchmarking methods.

## IX. DISCUSSION

In this paper we introduced `scarab`, which is a tool for creating benchmarks from user-provided algorithms or circuits. We showed how `scarab` can be used to create efficient and robust benchmarks, and how these benchmarks can enable explorations of different aspects of the performance of contemporary quantum computer. Our demonstrations of `scarab` used circuits defined on physical qubits (i.e., NISQ computations), but `scarab` could also be applied to circuits defined on logical qubits protected by quantum error correction, in fault-tolerant quantum computing (FTQC) architectures. We note, however, that FTQC and NISQ architectures have many important differences, and reliable benchmarking of FTQC systems might require new methods to be developed and added to `scarab`.

We showed how `scarab` can be used to improve existing benchmarks—to make them scalable and robust—and we demonstrated this with exemplar benchmarks from the QED-C’s suite. At its core, `scarab` contains a protocol for efficient estimation of circuit process fidelities, and we showcased how that can be used to estimate different aspects of the performance of an algorithm’s implementation—including the degradation in process fidelity due only to noise, the combined effect of noise and algorithmic approximations, and the in-situ impact of compiler optimizations in the presence of noise. Due to `scarab`’s simple interface and broad applicability, `scarab` could become an important component in benchmark development, handling many of the technical aspects of robust and scalable benchmark design while leaving the selection of interesting candidate circuits and algorithms to the user.

## CODE AVAILABILITY

`scarab` is available within `pyGSTi` (version 0.9.14) and can be found at <https://github.com/sandialabs/pyGSTi>. All the benchmarks created in this work can be reproduced using `scarab` together with `qiskit`, `HamLib`, or the QED-C’s open-source benchmarking suite, which can be found at <https://github.com/SRI-International/QC-App-Oriented-Benchmarks>.

## ACKNOWLEDGEMENTS

We thank Andrew Baczweski for helpful comments on the manuscript. N.S. thanks Aidan Wilber-Gauthier for helpful discussions. This material is based upon work supported by the U.S. Department of Energy, Office of Science (DE-FOA-0002253), National Quantum Information Science Research Centers, Quantum Systems Accelerator. T.P. acknowledges support from an Office of Advanced Scientific Computing Research Early Career Award. Sandia National Laboratories is a multi-program laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA-0003525. All statements of fact, opinion, or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of the U.S. Department of Energy or the U.S. Government. This research used IBM Quantum resources of the Air Force Research Laboratory. The views expressed are those of the authors and do not

reflect the official policy or position of IBM or the IBM Quantum team. The Quantum Economic Development Consortium (QED-C), comprised of industry, government, and academic institutions with NIST support, formed a Technical Advisory Committee (TAC) to assess quantum technology standards and promote economic growth through standardization. The Standards TAC developed the Application-Oriented Performance Benchmarks for Quantum Computing as an open-source initiative with contributions from multiple QED-C quantum computing members. Funding for N.P. was provided by Unitary Foundation and Quantum Computing Data.

## REFERENCES

- [1] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O’Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, “Superconducting quantum circuits at the surface code threshold for fault tolerance,” *Nature*, vol. 508, no. 7497, pp. 500–503, Apr. 2014. [Online]. Available: <http://dx.doi.org/10.1038/nature13171>
- [2] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michelsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, “Quantum supremacy using a programmable superconducting processor,” *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019. [Online]. Available: <http://dx.doi.org/10.1038/s41586-019-1666-5>
- [3] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. B. Ataides, N. Maskara, I. Cong, X. Gao, P. S. Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin, “Logical quantum processor based on reconfigurable atom arrays,” *Nature*, Dec. 2023. [Online]. Available: <http://dx.doi.org/10.1038/s41586-023-06927-3>
- [4] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala, “Evidence for the utility of quantum computing before fault tolerance,” *Nature*, vol. 618, no. 7965, pp. 500–505, Jun. 2023. [Online]. Available: <http://dx.doi.org/10.1038/s41586-023-06096-3>
- [5] S. A. Moses, C. H. Baldwin, M. S. Allman, R. Ancona, L. Ascaruzz, C. Barnes, J. Bartolotta, B. Bjork, P. Blanchard, M. Bohn, J. G. Bohnet, N. C. Brown, N. Q. Burdick, W. C. Burton, S. L. Campbell, J. P. Campora, C. Carron, J. Chambers, J. W. Chan, Y. H. Chen, A. Chernoguzov, E. Chertkov, J. Colina, J. P. Curtis, R. Daniel, M. DeCross, D. Deen, C. Delaney, J. M. Dreiling, C. T. Ertsgaard, J. Esposito, B. Estey, M. Fabrikant, C. Figgatt, C. Folts, M. Foss-Feig, D. Francois, J. P. Gaebler, T. M. Gatterman, C. N. Gilbreth, J. Giles, E. Glynn, A. Hall, A. M. Hankin, A. Hansen, D. Hayes, B. Higashi, I. M. Hoffman, B. Horning, J. J. Hout, R. Jacobs, J. Johansen, L. Jones, J. Karcz, T. Klein, P. Lauria, P. Lee, D. Liefer, S. T. Lu, D. Lucchetti, C. Lytle, A. Malm, M. Matheny, B. Mathewson, K. Mayer, D. B. Miller, M. Mills, B. Neyenhuis, L. Nugent, S. Olson, J. Parks, G. N. Price, Z. Price, M. Pugh, A. Ransford, A. P. Reed, C. Roman, M. Rowe, C. Ryan-Anderson, S. Sanders, J. Sedlacek, P. Shevchuk, P. Siegfried, T. Skripka, B. Spaun, R. T. Sprenkle, R. P. Stutz, M. Swallows, R. I. Tobey, A. Tran, T. Tran, E. Vogt, C. Volin, J. Walker, A. M. Zolot, and J. M. Pino, “A Race-Track Trapped-Ion quantum processor,” *Phys. Rev. X*, vol. 13, no. 4, p. 041052, Dec. 2023. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevX.13.041052>
- [6] J.-S. Chen, E. Nielsen, M. Ebert, V. Inlek, K. Wright, V. Chaplin, A. Maksymov, E. Pérez, A. Poudel, P. Maunz, and J. Gamble, “Benchmarking a trapped-ion quantum computer with 29 algorithmic qubits,” *arXiv preprint arXiv:2308.05071*, 2023.
- [7] Google Quantum AI and Collaborators, R. Acharya, D. A. Abanin, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, J. Atalaya, R. Babbush, D. Bacon, B. Ballard, J. C. Bardin, J. Bausch, A. Bengtsson, A. Bilmes, S. Blackwell, S. Boixo, G. Bortoli, A. Bourassa, J. Bovaard, L. Brill, M. Broughton, D. A. Browne, B. Buechea, B. B. Buckley, D. A. Buell, T. Burger, B. Burkett, N. Bushnell, A. Cabrera, J. Campero, H.-S. Chang, Y. Chen, Z. Chen, B. Chiaro, D. Chik, C. Chou, J. Clae, A. Y. Cleland, J. Cogan, R. Collins, P. Conner, W. Courtney, A. L. Crook, B. Curtin, S. Das, A. Davies, L. De Lorenzo, D. M. Debroy, S. Demura, M. Devoret, A. Di Paolo, P. Donohoe, I. Drozdov, A. Dunsworth, C. Earle, T. Edlich, A. Eickbusch, A. M. Elbag, M. Elzouka, C. Erickson, L. Faoro, E. Farhi, V. S. Ferreira, L. F. Burgos, E. Forati, A. G. Fowler, B. Foxen, S. Ganjam, G. Garcia, R. Gasca, E. Genois, W. Giang, C. Gidney, D. Gilboa, R. Gosula, A. G. Dau, D. Graumann, A. Greene, J. A. Gross, S. Habegger, J. Hall, M. C. Hamilton, M. Hansen, M. P. Harrigan, S. D. Harrington, F. J. H. Heras, S. Heslin, P. Heu, O. Higgott, G. Hill, J. Hilton, G. Holland, S. Hong, H.-Y. Huang, A. Huff, W. J. Huggins, L. B. Ioffe, S. V. Isakov, J. Iveland, E. Jeffrey, Z. Jiang, C. Jones, S. Jordan, C. Joshi, P. Juhas, D. Kafri, H. Kang, A. H. Karamlou, K. Kechedzhi, J. Kelly, T. Khaire, T. Khattar, M. Khezri, S. Kim, P. V. Klimov, A. R. Klots, B. Kobrin, P. Kohli, A. N. Korotkov, F. Kostritsa, R. Kothari, B. Kozlovskii, J. M. Kreikebaum, V. D. Kurilovich, N. Lacroix, D. Landhuis, T. Lange-Dei, B. W. Langley, P. Laptev, K.-M. Lau, L. Le Guevel, J. Ledford, J. Lee, K. Lee, Y. D. Lensky, S. Leon, B. J. Lester, W. Y. Li, Y. Li, A. T. Lill, W. Liu, W. P. Livingston, A. Locharla, E. Lucero, D. Lundahl, A. Lunt, S. Madhuk, F. D. Malone, A. Maloney, S. Mandrà, J. Manyika, L. S. Martin, O. Martin, S. Martin, C. Maxfield, J. R. McClean, M. McEwen, S. Meeks, A. Megrant, X. Mi, K. C. Miao, A. Mieszala, R. Molavi, S. Molina, S. Montazeri, A. Morvan, R. Movassagh, W. Mruczkiewicz, O. Naaman, M. Neeley, C. Neill, A. Nersisyan, H. Neven, M. Newman, J. H. Ng, A. Nguyen, M. Nguyen, C.-H. Ni, M. Y. Niu, T. E. O’Brien, W. D. Oliver, A. Opremcak, K. Ottosson, A. Petukhov, A. Pizzuto, J. Platt, R. Potter, O. Pritchard, L. P. Pryadko, C. Quintana, G. Ramachandran, M. J. Reagor, J. Redding, D. M. Rhodes, G. Roberts, E. Rosenberg, E. Rosenfeld, P. Roushan, N. C. Rubin, N. Saei, D. Sank, K. Sankaragomathi, K. J. Satzinger, H. F. Schurkus, C. Schuster, A. W. Senior, M. J. Shearn, A. Shorter, N. Shutty, V. Shvarts, S. Singh, V. Sivak, J. Skrzyni, S. Small, V. Smelyanskiy, W. C. Smith, R. D. Somma, S. Springer, G. Sterling, D. Strain, J. Suchard, A. Szasz, A. Sztein, D. Thor, A. Torres, M. M. Torunbalci, A. Vaishnav, J. Vargas, S. Vdovichev, G. Vidal, B. Villalonga, C. V. Heidweiller, S. Waltman, S. X. Wang, B. Ware, K. Weber, T. Weidel, T. White, K. Wong, B. W. K. Woo, C. Xing, Z. J. Yao, P. Yeh, B. Ying, J. Yoo, N. Yosri, G. Young, A. Zalcman, Y. Zhang, N. Zhu, and N. Zobrist, “Quantum error correction below the surface code threshold,” *Nature*, vol. 638, no. 8052, pp. 920–926, Feb. 2025. [Online]. Available: <https://www.nature.com/articles/s41586-024-08449-y>
- [8] T. Proctor, K. Young, A. D. Baczewski, and R. Blume-Kohout, “Benchmarking quantum computers,” *Nat. Rev. Phys.*, vol. 7, no. 2, pp. 105–118, Jan. 2025. [Online]. Available: <https://www.nature.com/articles/s42254-024-00796-z>
- [9] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, “Validating quantum computers using randomized model circuits,” *Phys. Rev. A*, vol. 100, no. 3, p. 032328, Sep. 2019. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.100.032328>
- [10] J. Emerson, R. Alicki, and K. Źyczkowski, “Scalable noise estimation with random unitary operators,” *J. Opt. B Quantum Semiclassical Opt.*, vol. 7, no. 10, p. S347, Sep. 2005. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1464-4266/7/10/021/meta>
- [11] J. Emerson, M. Silva, O. Moussa, C. Ryan, M. Laforest, J. Baugh, D. G. Cory, and R. Laflamme, “Symmetrized characterization of noisy quantum processes,” *Science*, vol. 317, no. 5846, pp. 1893–1896, Sep. 2007. [Online]. Available: <http://dx.doi.org/10.1126/science.1145699>
- [12] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, “Randomized benchmarking of quantum gates,” *Phys. Rev. A*, vol. 77, no. 1, p. 012307, Jan. 2008. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.77.012307>
- [13] E. Magesan, J. M. Gambetta, and J. Emerson, “Scalable and robust randomized benchmarking of quantum processes,” *Phys. Rev. Lett.*, vol. 106, no. 18, p. 180504, May 2011. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevLett.106.180504>

- [14] T. J. Proctor, A. Carignan-Dugas, K. Rudinger, E. Nielsen, R. Blume-Kohout, and K. Young, "Direct randomized benchmarking for multiqubit devices," *Phys. Rev. Lett.*, vol. 123, no. 3, p. 030503, Jul. 2019. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevLett.123.030503>
- [15] J. Hines, D. Hothem, R. Blume-Kohout, B. Whaley, and T. Proctor, "Fully scalable randomized benchmarking without motion reversal," *arXiv preprint arXiv:2309.05147*, 2023.
- [16] D. C. McKay, I. Hincks, E. J. Pritchett, M. Carroll, L. C. G. Govia, and S. T. Merkel, "Benchmarking quantum processor performance at scale," *arXiv preprint arXiv:2311.05933*, 2023.
- [17] T. Proctor, S. Seritan, K. Rudinger, E. Nielsen, R. Blume-Kohout, and K. Young, "Scalable randomized benchmarking of quantum computers using mirror circuits," *Phys. Rev. Lett.*, vol. 129, no. 15, p. 150502, Oct. 2022. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevLett.129.150502>
- [18] J. Hines, M. Lu, R. K. Naik, A. Hashim, J.-L. Ville, B. Mitchell, J. M. Kriekbaum, D. I. Santiago, S. Seritan, E. Nielsen, R. Blume-Kohout, K. Young, I. Siddiqi, B. Whaley, and T. Proctor, "Demonstrating scalable randomized benchmarking of universal gate sets," *Phys. Rev. X*, vol. 13, no. 4, p. 041030, Nov. 2023. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevX.13.041030>
- [19] J. Combes, C. Granade, C. Ferrie, and S. T. Flammia, "Logical randomized benchmarking," *arXiv preprint arXiv:1702.03688*, 2017.
- [20] J. Helsen, X. Xue, L. M. K. Vandersypen, and S. Wehner, "A new class of efficient randomized benchmarking protocols," *npj Quantum Information*, vol. 5, no. 1, p. 71, Aug. 2019. [Online]. Available: <https://doi.org/10.1038/s41534-019-0182-7>
- [21] J. Helsen, I. Roth, E. Onorati, A. H. Werner, and J. Eisert, "General framework for randomized benchmarking," *PRX Quantum*, vol. 3, no. 2, p. 020357, Jun. 2022. [Online]. Available: <https://link.aps.org/doi/10.1103/PRXQuantum.3.020357>
- [22] E. Magesan, J. M. Gambetta, B. R. Johnson, C. A. Ryan, J. M. Chow, S. T. Merkel, M. P. da Silva, G. A. Keefe, M. B. Rothwell, T. A. Ohki, M. B. Ketchen, and M. Steffen, "Efficient measurement of quantum gate error by interleaved randomized benchmarking," *Phys. Rev. Lett.*, vol. 109, no. 8, p. 080505, Aug. 2012. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevLett.109.080505>
- [23] K. Chen, W. Fang, J. Guan, X. Hong, M. Huang, J. Liu, Q. Wang, and M. Ying, "VeriQBench: A benchmark for multiple types of quantum circuits," *arXiv preprint arXiv:2206.10880*, 2022.
- [24] N. M. Linke, D. Maslov, M. Roetteler, S. Debnath, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Experimental comparison of two quantum computing architectures," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 114, no. 13, pp. 3305–3310, Mar. 2017. [Online]. Available: <http://dx.doi.org/10.1073/pnas.1618020114>
- [25] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. C. Pisenti, M. Chmielewski, C. Collins, K. M. Hudek, J. Mizrahi, J. D. Wong-Campos, S. Allen, J. Apisdorf, P. Solomon, M. Williams, A. M. Ducore, A. Blinov, S. M. Kreikemeier, V. Chaplin, M. Keesan, C. Monroe, and J. Kim, "Benchmarking an 11-qubit quantum computer," *Nat. Commun.*, vol. 10, no. 1, p. 5464, Nov. 2019. [Online]. Available: <http://dx.doi.org/10.1038/s41467-019-13534-2>
- [26] Tomesh, Gokhale, Omole, Ravi, Smith, Viszlai, Wu, Hardavellas, Martonosi, and Chong, "SupermarQ: A scalable quantum benchmark suite," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, vol. 0, Apr. 2022, pp. 587–603. [Online]. Available: <http://dx.doi.org/10.1109/HPCA53966.2022.00050>
- [27] P. Murali, N. M. Linke, M. Martonosi, A. J. Abhari, N. H. Nguyen, and C. H. Alderete, "Full-stack, real-system quantum computer studies: architectural comparisons and design insights," in *Proceedings of the 46th International Symposium on Computer Architecture*, ser. ISCA '19. New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 527–540. [Online]. Available: <https://doi.org/10.1145/3307650.3322273>
- [28] H. Donkers, K. Mesman, Z. Al-Ars, and M. Möller, "QPack scores: Quantitative performance metrics for application-oriented quantum computer benchmarking," *arXiv preprint arXiv:2205.12142*, 2022.
- [29] J. R. Finžgar, P. Ross, J. Klepsch, and A. Luckow, "QUARK: A framework for quantum computing application benchmarking," *arXiv preprint arXiv:2202.03028*, 2022.
- [30] D. Mills, S. Sivarajah, T. L. Scholten, and R. Duncan, "Application-Motivated, holistic benchmarking of a full quantum computing stack," *arXiv preprint arXiv:2006.01273*, 2020.
- [31] T. Lubinski, S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaise, C. H. Baldwin, K. Mayer, and T. Proctor, "Application-Oriented performance benchmarks for quantum computing," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–32, 2023. [Online]. Available: <http://dx.doi.org/10.1109/TQE.2023.3253761>
- [32] T. Lubinski, J. J. Goings, K. Mayer, S. Johri, N. Reddy, A. Mehta, N. Bhatia, S. Rappaport, D. Mills, C. H. Baldwin, L. Zhao, A. Barbosa, S. Maity, and P. S. Mundada, "Quantum algorithm exploration using Application-Oriented performance benchmarks," *arXiv preprint arXiv:2402.08985*, 2024.
- [33] T. Lubinski, C. Coffrin, C. McGeoch, P. Sathe, J. Apanavicius, and D. E. Bernal Neira, "Optimization applications as quantum performance benchmarks," *arXiv preprint arXiv:2302.02278*, 2023.
- [34] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, "A generative modeling approach for benchmarking and training shallow quantum circuits," *npj Quantum Information*, vol. 5, no. 1, p. 45, May 2019. [Online]. Available: <https://doi.org/10.1038/s41534-019-0157-8>
- [35] A. Li and S. Krishnamoorthy, "QASMBench: A low-level QASM benchmark suite for NISQ evaluation and simulation," *arXiv preprint arXiv:2005.13018*, 2020.
- [36] N. Quetschlich, L. Burgholzer, and R. Wille, "MQT bench: Benchmarking software and design automation tools for quantum computing," *Quantum*, vol. 7, p. 1062, Jul. 2023. [Online]. Available: <https://quantum-journal.org/papers/q-2023-07-20-1062/pdf/>
- [37] Y. Dong and L. Lin, "Random circuit block-encoded matrix and a proposal of quantum LINPACK benchmark," *Phys. Rev. A*, vol. 103, no. 6, p. 062412, Jun. 2021. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.103.062412>
- [38] S. Martiel, T. Ayral, and C. Allouche, "Benchmarking quantum coprocessors in an Application-Centric, Hardware-Agnostic, and scalable way," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–11, 2021. [Online]. Available: <http://dx.doi.org/10.1109/TQE.2021.3090207>
- [39] W. van der Schoot, D. Leermakers, R. Wezeman, N. Neumann, and F. Phillipson, "Evaluating the q-score of quantum annealers," *arXiv preprint arXiv:2208.07633*, 2022.
- [40] W. van der Schoot, R. Wezeman, N. M. P. Neumann, F. Phillipson, and R. Kooij, "Q-score Max-Clique: The first quantum metric evaluation on multiple computational paradigms," *arXiv preprint arXiv:2302.00639*, 2023.
- [41] A. Cornelissen, J. Bausch, and A. Gilyén, "Scalable benchmarks for Gate-Based quantum computers," *arXiv preprint arXiv:2104.10698*, 2021.
- [42] K. Georgopoulos, C. Emary, and P. Zuliani, "Quantum computer benchmarking via quantum algorithms," *arXiv preprint arXiv:2112.09457*, 2021.
- [43] Y. Dong, K. B. Whaley, and L. Lin, "A quantum hamiltonian simulation benchmark," *npj Quantum Information*, vol. 8, no. 1, pp. 1–8, Nov. 2022. [Online]. Available: <https://www.nature.com/articles/s41534-022-00636-x>
- [44] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE, 1994, pp. 124–134. [Online]. Available: <http://dx.doi.org/10.1109/SFCS.1994.365700>
- [45] E. Granet, J. Lykke Jacobsen, and H. Saleur, "Analytical results on the Heisenberg spin chain in a magnetic field," *Journal of Physics A: Mathematical and Theoretical*, vol. 52, no. 25, p. 255302, Jun. 2019. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1751-8121/ab1f97>
- [46] G. H. Low, Y. Su, Y. Tong, and M. C. Tran, "Complexity of Implementing Trotter Steps," *PRX Quantum*, vol. 4, no. 2, p. 020323, May 2023. [Online]. Available: <https://link.aps.org/doi/10.1103/PRXQuantum.4.020323>
- [47] M. Motta and J. E. Rice, "Emerging quantum computing algorithms for quantum chemistry," *WIREs Computational Molecular Science*, vol. 12, no. 3, p. e1580, May 2022. [Online]. Available: <https://wires.onlinelibrary.wiley.com/doi/10.1002/wcms.1580>
- [48] G. H. Low and I. L. Chuang, "Optimal Hamiltonian Simulation by Quantum Signal Processing," *Physical Review Letters*, vol. 118, no. 1, p. 010501, Jan. 2017. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.118.010501>
- [49] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D.

- Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, "Quantum Chemistry in the Age of Quantum Computing," *Chemical Reviews*, vol. 119, no. 19, pp. 10856–10915, Oct. 2019. [Online]. Available: <https://pubs.acs.org/doi/10.1021/acs.chemrev.8b00803>
- [50] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, "Quantum computational chemistry," *Reviews of Modern Physics*, vol. 92, no. 1, p. 015003, Mar. 2020. [Online]. Available: <https://link.aps.org/doi/10.1103/RevModPhys.92.015003>
- [51] R. D. Somma, G. Ortiz, E. H. Knill, and J. Gubernatis, "Quantum simulations of physics problems," E. Donkor, A. R. Pirich, and H. E. Brandt, Eds., Orlando, FL, Aug. 2003, p. 96. [Online]. Available: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.487249>
- [52] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, "Simulation of electronic structure Hamiltonians using quantum computers," *Molecular Physics*, vol. 109, no. 5, pp. 735–750, Mar. 2011. [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/00268976.2011.552441>
- [53] A. M. Childs, D. Maslov, Y. Nam, N. J. Ross, and Y. Su, "Toward the first quantum simulation with quantum speedup," *Proceedings of the National Academy of Sciences*, vol. 115, no. 38, pp. 9456–9461, Sep. 2018. [Online]. Available: <https://pnas.org/doi/full/10.1073/pnas.1801723115>
- [54] S. Lloyd, "Universal Quantum Simulators," *Science*, vol. 273, no. 5278, pp. 1073–1078, Aug. 1996. [Online]. Available: <https://www.science.org/doi/10.1126/science.273.5278.1073>
- [55] T. Lubinski, S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaise, C. H. Baldwin, K. Mayer, and T. Proctor, "Application-Oriented Performance Benchmarks for Quantum Computing," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–32, 2023. [Online]. Available: <https://ieeexplore.ieee.org/document/10061574/>
- [56] T. Lubinski, C. Coffrin, C. McGeoch, P. Sathe, J. Apanavicius, D. Bernal Neira, and Quantum Economic Development Consortium(QED-C) Collaboration, "Optimization Applications as Quantum Performance Benchmarks," *ACM Transactions on Quantum Computing*, vol. 5, no. 3, pp. 1–44, Sep. 2024. [Online]. Available: <https://dl.acm.org/doi/10.1145/3678184>
- [57] T. Lubinski, J. J. Goings, K. Mayer, S. Johri, N. Reddy, A. Mehta, N. Bhatia, S. Rappaport, D. Mills, C. H. Baldwin, L. Zhao, A. Barbosa, S. Maity, and P. S. Mundada, "Quantum Algorithm Exploration using Application-Oriented Performance Benchmarks," 2024, version Number: 1. [Online]. Available: <https://arxiv.org/abs/2402.08985>
- [58] A. Chatterjee, S. Rappaport, A. Giri, S. Johri, T. Proctor, D. E. B. Neira, P. Sathe, and T. Lubinski, "A Comprehensive Cross-Model Framework for Benchmarking the Performance of Quantum Hamiltonian Simulations," *IEEE Transactions on Quantum Engineering*, vol. 6, pp. 1–26, 2025. [Online]. Available: <https://ieeexplore.ieee.org/document/10949677/>
- [59] S. Niu, E. Kökcü, S. Johri, A. Ramesh, A. Chatterjee, D. E. B. Neira, D. Camps, and T. Lubinski, "A Practical Framework for Assessing the Performance of Observable Estimation in Quantum Simulation," 2025, version Number: 1. [Online]. Available: <https://arxiv.org/abs/2504.09813>
- [60] M. Sarovar, T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, "Detecting crosstalk errors in quantum information processors," *Quantum*, vol. 4, no. 321, p. 321, Sep. 2020. [Online]. Available: <https://quantum-journal.org/papers/q-2020-09-11-321/>
- [61] J. M. Gambetta, A. D. Córcoles, S. T. Merkel, B. R. Johnson, J. A. Smolin, J. M. Chow, C. A. Ryan, C. Rigetti, S. Poletto, T. A. Ohki, M. B. Ketchen, and M. Steffen, "Characterization of addressability by simultaneous randomized benchmarking," *Phys. Rev. Lett.*, vol. 109, no. 24, p. 240504, Dec. 2012. [Online]. Available: <http://dx.doi.org/10.1103/PhysRevLett.109.240504>
- [62] T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, "Measuring the capabilities of quantum computers," *Nat. Phys.*, vol. 18, no. 1, pp. 75–79, Dec. 2021. [Online]. Available: <https://www.nature.com/articles/s41567-021-01409-7>
- [63] T. Proctor, S. Seritan, E. Nielsen, K. Rudinger, K. Young, R. Blume-Kohout, and M. Sarovar, "Establishing trust in quantum computations," *arXiv preprint arXiv:2204.07568*, 2022.
- [64] J. Hines and T. Proctor, "Scalable Full-Stack Benchmarks for Quantum Computers," *IEEE Transactions on Quantum Engineering*, vol. 5, pp. 1–12, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/10538040/>
- [65] S. Merkel, T. Proctor, S. Ferracin, J. Hines, S. Barron, L. C. G. Govia, and D. McKay, "When Clifford benchmarks are sufficient; estimating application performance with scalable proxy circuits," 2025, version Number: 2. [Online]. Available: <https://arxiv.org/abs/2503.05943>
- [66] S. Ferracin, T. Kapourniotis, and A. Datta, "Reducing resources for verification of quantum computations," *Physical Review A*, vol. 98, no. 2, p. 022323, Aug. 2018. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.98.022323>
- [67] ———, "Accrediting outputs of noisy intermediate-scale quantum computing devices," *New Journal of Physics*, vol. 21, no. 11, p. 113038, Nov. 2019. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1367-2630/ab4fd6>
- [68] S. Ferracin, S. T. Merkel, D. McKay, and A. Datta, "Experimental accreditation of outputs of noisy quantum computers," *Phys. Rev. A*, vol. 104, no. 4, p. 042603, Oct. 2021. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.104.042603>
- [69] S. K. Seritan, A. Dhumuntarao, A. Q. Wilber-Gauthier, K. M. Rudinger, A. E. Russo, R. Blume-Kohout, A. D. Baczewski, and T. Proctor, "Benchmarking quantum computers with any quantum algorithm," 2025, version Number: 1. [Online]. Available: <https://arxiv.org/abs/2508.05754>
- [70] A. Hashim, L. B. Nguyen, N. Goss, B. Marinelli, R. K. Naik, T. Chistolini, J. Hines, J. P. Marceaux, Y. Kim, P. Gokhale, T. Tomesh, S. Chen, L. Jiang, S. Ferracin, K. Rudinger, T. Proctor, K. C. Young, R. Blume-Kohout, and I. Siddiqi, "A practical introduction to benchmarking and characterization of quantum computers," *arXiv [quant-ph]*, Aug. 2024. [Online]. Available: <https://arxiv.org/abs/2408.12064>
- [71] E. Nielsen, K. Rudinger, T. Proctor, A. Russo, K. Young, and R. Blume-Kohout, "Probing quantum processor performance with pyGSTi," *Quantum Sci. Technol.*, vol. 5, no. 4, p. 044002, Jul. 2020. [Online]. Available: <https://iopscience.iop.org/article/10.1088/2058-9565/ab8aa4>
- [72] R. Blume-Kohout, T. Proctor, and K. Young, "Quantum characterization, verification, and validation," *arXiv [quant-ph]*, Mar. 2025. [Online]. Available: <https://arxiv.org/abs/2503.16383>
- [73] M. Suzuki, "Generalized Trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems," *Communications in Mathematical Physics*, vol. 51, no. 2, pp. 183–190, Jun. 1976. [Online]. Available: <http://link.springer.com/10.1007/BF01609348>
- [74] N. Hatano and M. Suzuki, "Finding Exponential Product Formulas of Higher Orders," in *Quantum Annealing and Other Optimization Methods*, R. Beig, W. Beiglböck, W. Domcke, B.-G. Englert, U. Frisch, P. Hänggi, G. Hasinger, K. Hepp, W. Hillebrandt, D. Imboden, R. L. Jaffe, R. Lipowsky, H. V. Löhneysen, I. Ojima, D. Sornette, S. Theisen, W. Weise, J. Wess, J. Zittartz, A. Das, and B. K. Chakrabarti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, Nov. 2005, vol. 679, pp. 37–68, series Title: Lecture Notes in Physics. [Online]. Available: [http://link.springer.com/10.1007/11526216\\_2](http://link.springer.com/10.1007/11526216_2)
- [75] N. Wiebe, D. Berry, P. Hoyer, and B. C. Sanders, "Higher order decompositions of ordered operator exponentials," *Journal of Physics A: Mathematical and Theoretical*, vol. 43, no. 6, p. 065203, Jan. 2010. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1751-8113/43/6/065203>
- [76] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, "Theory of Trotter Error with Commutator Scaling," *Physical Review X*, vol. 11, no. 1, p. 011020, Feb. 2021. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevX.11.011020>
- [77] N. P. Sawaya, D. Martí-Dafcik, Y. Ho, D. P. Tabor, D. E. B. Neira, A. B. Magann, S. Premaratne, P. Dubey, A. Matsuurra, N. Bishop, W. A. D. Jong, S. Benjamin, O. Parekh, N. Tubman, K. Klymko, and D. Camps, "HamLib: A library of Hamiltonians for benchmarking quantum algorithms and hardware," *Quantum*, vol. 8, p. 1559, Dec. 2024. [Online]. Available: <https://quantum-journal.org/papers/q-2024-12-11-1559/>
- [78] G. C. Knee and W. J. Munro, "Optimal Trotterization in universal quantum simulators under faulty control," *Physical Review A*, vol. 91, no. 5, p. 052327, May 2015. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.91.052327>
- [79] B. D. M. Jones, D. R. White, G. O. O'Brien, J. A. Clark, and E. T. Campbell, "Optimising trotter-suzuki decompositions for quantum simulation using evolutionary strategies," in *Proceedings of the Genetic and Evolutionary Computation Conference*. Prague Czech Republic: ACM, Jul. 2019, pp. 1223–1231. [Online]. Available: <https://dl.acm.org/doi/10.1145/3321707.3321835>
- [80] A. A. Avtandilyan and W. V. Pogosov, "Optimal-order Trotter–Suzuki decomposition for quantum simulation on noisy quantum computers,"

- Quantum Information Processing*, vol. 24, no. 1, p. 8, Dec. 2024. [Online]. Available: <https://link.springer.com/10.1007/s11128-024-04627-z>
- [81] J. Kalloor, L. Kovalsky, M. Weiden, J. Kubiatowicz, E. Younis, C. Iancu, and M. Sarovar, “Application Scale Quantum Circuit Compilation and Optimization,” Oct. 2025, arXiv:2510.18000 [quant-ph]. [Online]. Available: <http://arxiv.org/abs/2510.18000>
- [82] T. Proctor, A. Tran, X. Liu, A. Dhumuntarao, S. Seritan, A. Green, and N. M. Linke, “Featuremetric benchmarking: Quantum computer benchmarks based on circuit features,” 2025, version Number: 1. [Online]. Available: <https://arxiv.org/abs/2504.12575>
- [83] N. Patel, A. Giri, H. P. Patil, N. Siekierski, A. Chatterjee, S. Johri, T. Proctor, T. Lubinski, and S. Niu, “Platform-agnostic modular architecture for quantum benchmarking,” *arXiv [quant-ph]*, Oct. 2025. [Online]. Available: <http://arxiv.org/abs/2510.08469>
- [84] J. Hubbard, “Electron correlations in narrow energy bands,” *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 276, no. 1365, pp. 238–257, Nov. 1963. [Online]. Available: <https://royalsocietypublishing.org/doi/10.1098/rspa.1963.0204>

## APPENDIX

### A. QED-C Application-Oriented Benchmarking Suite

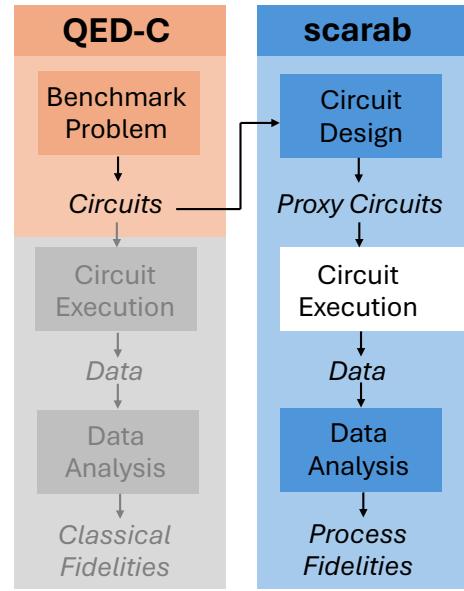
In this appendix we review the QED-C’s application-oriented benchmarking suite and further discuss how it can be interfaced with `scarab` to enable efficient and scalable benchmarks, which we demonstrate with examples in the main text. The QED-C’s application oriented benchmarking suite is an evolving set of over 20 different application-based benchmarks for quantum computers, including algorithms such as phase estimation, Hamiltonian simulation, and the QFT. QED-C suite measures execution quality, runtime costs, and resource requirements for both single-circuit executions and iterative algorithms, using normalized classical (also called Hellinger) fidelity (defined in (2) in the main text) to assess circuit quality under noise. The first implementation of this benchmarking suite integrated all aspects of the benchmarking workflow, including algorithm definition, benchmarking circuit generation, circuit execution, and data analysis. Recently, however, the framework was modularized to enable more flexible interfacing with external benchmarking tools [83], and we leverage this modularization in this work. The updated design separates the workflow into independent stages—problem generation, execution, and analysis—that can be accessed and run independently, as shown in the left column of Fig. 9. This modular workflow enables interfacing the QED-C’s suite with `scarab`. In particular, the circuits defining a QED-C benchmark can be extracted (using a `get_circuits` flag) and then input into `scarab`, as shown in Fig. 9.

The purpose of interfacing the QED-C benchmarking suite with `scarab` is to enable the conversion of the QED-C’s benchmarks into more robust and scalable benchmarks that measure process fidelity. Some of the benchmarks in the QED-C suite use specialized methods to enable them to scale to many qubits, such as picking particular inputs for a circuit that enable efficient classical computation of the expected output. These choices do not always result in a benchmark that will be predictive of the performance of that circuit on other input states. Conversely, some of the QED-C benchmarks are not scalable to many qubits, because they rely on exponentially-scaling classical computations. Interfacing the QED-C suite with `scarab` solves these problems, offloading this aspect of benchmark design to an independent, general-purpose tool.

### B. Hamiltonians

In this appendix, we provide more information on the Hamiltonians obtained from HamLib [77] that we simulate in Section VI-A. These are the transverse field Ising model (TFIM), Heisenberg, Bose-Hubbard, Fermi-Hubbard, and Max3SAT Hamiltonians. The TFIM Hamiltonian is given by

$$H_{\text{TFIM}} = \sum_i h_i X_i + \sum_{\langle i,j \rangle} Z_i Z_j, \quad (12)$$



**Fig. 9: Interfacing the QED-C’s suite with `scarab`.** The modular QED-C benchmarking suite (left column) can be interfaced with `scarab` (right column) to transform any of the QED-C’s benchmarks into efficient, scalable, and robust benchmarks that measure process fidelity. To do so, we can extract the circuits that are used to create a QED-C benchmark and instead input them into `scarab`. Similar interactions with other libraries of computational problems or quantum circuits are straightforward, and some of other examples of these interactions are demonstrated herein.

where the second sum is over the edges  $\langle i, j \rangle$  of the lattice. For our simulations, we consider a 1D lattice with periodic boundary conditions and  $h_i = 2$  for all  $i$ .

The Heisenberg Hamiltonian is given by

$$H_{\text{Heis}} = \sum_{i=1}^N X_i X_{i+1} + Y_i Y_{i+1} + Z_i Z_{i+1} + h_i Z_i. \quad (13)$$

For our simulations, we consider a 1D lattice with periodic boundary conditions and  $h_i = 2$  for all  $i$ .

The Fermi-Hubbard [84] Hamiltonian is given by

$$H_{\text{FH}} = -t \sum_{\langle i,j \rangle, \sigma} (c_{i,\sigma}^\dagger c_{j,\sigma} + c_{j,\sigma}^\dagger c_{i,\sigma}) + U \sum_i n_{i\uparrow} n_{i\downarrow}, \quad (14)$$

where  $\langle i, j \rangle$  are lattice edges,  $\sigma \in \{\uparrow, \downarrow\}$  labels the fermion spin,  $c$  and  $c^\dagger$  are the fermionic creation and annihilation operators, respectively, and  $n_{i\sigma} = c_{i\sigma}^\dagger c_{i\sigma}$  is the number operator associated with spin  $\sigma$  and site  $j$ . The prefactor  $t$  on the first sum is the tunneling strength, and  $U$  is the on-site interaction strength. We set  $t = 1$ ,  $U = 12$  on a 1D lattice with periodic boundary conditions and a Brayvi-Kitaev [49] encoding.

The Bose-Hubbard Hamiltonian is given by

$$H_{\text{BH}} = -t \sum_i (b_{i+1}^\dagger b_i + b_i^\dagger b_{i+1}) + \frac{U}{2} \sum_i n_i (n_i - 1), \quad (15)$$

where  $b_i^\dagger$  and  $b_i$  are creation and annihilation operators respectively,  $n_i = b_i^\dagger b_i$  is the number operator,  $t$  is the tunneling strength, and  $U$  is the site energy. For our simulations, we consider a 1D lattice with non-periodic boundary conditions that uses the Gray encoding, with  $t = 1$  and  $U = 10$ .

The Max3SAT Hamiltonian is a sum of terms that correspond to 3-variable clauses. Each clause has the form

$$(\neg)^{s_i} x_i \vee (\neg)^{s_j} x_j \vee (\neg)^{s_k} x_k, \quad (16)$$

where  $s_j = 1$  if  $x_j$  is negated in the clause and equals 0 otherwise. The corresponding term in the Hamiltonian is

$$I - \frac{1}{8} [I + (-1)^{s_i} Z_i] [I + (-1)^{s_j} Z_j] [I + (-1)^{s_k} Z_k]. \quad (17)$$

The total number of clauses is given by  $rn$ , where  $r$  is the clause ratio and  $n$  is the number of qubits. We set  $r = 2$  for our simulations and set `rinst`, the random instance flag used by HamLib to select a set of clauses, to 02.