A MODULAR ENGINE FOR QUANTUM MONTE CARLO INTEGRATION

ISMAIL YUNUS AKHALWAYA $^{1,3},$ ADAM CONNOLLY 1, ROLAND GUICHARD $^{1,\dagger},$ STEVEN HERBERT $^{1,4},$ CAHIT KARGI 1, ALEXANDRE KRAJENBRINK 1, MICHAEL LUBASCH 2, CONOR MC KEEVER 2, JULIEN SORCI 1, MICHAEL SPRANGER 1, IFAN WILLIAMS 1

ABSTRACT. We present the Quantum Monte Carlo Integration (QMCI) engine developed by Quantinuum. It is a quantum computational tool for evaluating multi-dimensional integrals that arise in various fields of science and engineering such as finance. This white paper presents a detailed description of the architecture of the QMCI engine, including a variety of distribution-loading methods, a novel quantum amplitude estimation method that improves the statistical robustness of QMCI calculations, and a library of statistical quantities that can be estimated. The QMCI engine is designed with modularity in mind, allowing for the continuous development of new quantum algorithms tailored in particular to financial applications. Additionally, the engine features a resource mode, which provides a precise resource quantification for the quantum circuits generated. The paper also includes extensive benchmarks that showcase the engine's performance, with a focus on the evaluation of various financial instruments.

Contents

List of Tables		
Part	I. Introduction and background	<u> </u>
1.	Introduction	<u> </u>
2.	Preliminaries	(
	2.1. Probability and statistics	(
	2.2. Quantum states as probability distributions	10
	2.3. Quantum Monte Carlo integration	11
	2.4. Basics of mathematical finance	12
3.	Summary of our previous works	14
	3.1 The problem with Grover-Rudolph state preparation	1/

E-mail address: steven.herbert@quantinuum.com & alexandre.krajenbrink@quantinuum.com. Date: August 14, 2023.

¹ Quantinuum, Terrington House, 13–15 Hills Road, Cambridge CB2 1NL, United Kingdom

² Quantinuum, Partnership House, Carlisle Place, London SW1P 1BX, United Kingdom

³School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa

⁴Department of Computer Science and Technology, University of Cambridge, United Kingdom

[†] Current address: PASQAL, 7 Rue Léonard de Vinci, 91300 Massy, France.

	3.2.	Sampling with reversible circuits	14
	3.3.	Fourier quantum Monte Carlo integration	15
	3.4.	Noise-aware quantum amplitude estimation	17
Part	TT	Principal novel methods	19
4.		anced P-builder	19
1.		Enhanced P-builder functions	19
	4.2.	Example: look-back options	21
	4.3.	Common financial calculations encompassed by the enhanced P-builder	21
	4.4.	Using the enhanced P-builder to construct (geometric) Brownian motion	21
	4.5.	Classes for common financial instruments	23
5.		stically robust quantum amplitude estimation	$\frac{1}{25}$
	5.1.	Statistical robustness	25
		Linear combination of unitaries QAE	27
	5.3.	QAE robustness benchmarks	32
Part	III.	Full technical specification	35
6.		view of the QMCI engine architecture	35
7.		ribution loader	37
•	7.1.	Quantum walk	38
		Partial differential equations	42
		Parameterised quantum circuits	43
	7.4.	Benchmarks	47
8.		stical quantities estimated by the QMCI engine	50
	8.1.	Estimating the mean	51
	8.2.	Estimating the conditional expectation	52
	8.3.	Estimating the variance & second moment	52
	8.4.	Estimating the expectation of the random variable exponentiated	52
	8.5.	Estimating the conditional expectation of the random variable exponentiated	53
	8.6.	Estimating a single qubit as sampling a Bernoulli random variable	54
	8.7.	Applying the function to part of the support	54
	8.8.	Summary of QMCI convergences	55
9.		ntum amplitude estimation	56
	9.1.	LCU QAE	57
		MLQAE	57
		IQAE	57
		Prepare and measure	58
	9.5.	Summary of QAE convergences	58
10.	Res	ource mode	59
	10.1.	NISQ resource mode	60
	10.2.	·	60
	10.3.	Tightening the fault-tolerant resource estimate	62
	10.4.		63
Part	IV.	Benchmarks and Discussion	64
		chmarks	64
		Modelling the financial time-series	64
	11.2.		65
		Benchmark results	65

12. Discussion		69	
	12.1. The QMCI engine as an R&D tool	69	
	12.2. From theoretical quantum advantage to useful quantum advantage	69	
	12.3. Further applications of the QMCI engine	71	
Pa	art V. End matter and appendices	72	
-	Acknowledgments	72	
	References	73	
	Appendix A. QAE statistical robustness benchmarks	77	
	Appendix B. RMSE convergence plots	81	
-	Appendix C. Circuits for benchmarks	84	
	List of Tables		
1	Enhanced P -builder: arithmetic operations	20	
2 Enhanced P -builder: logical operations		20	
3 Common option payoff calculations using the enhanced P -builder		23	
4	4 Performance of Gaussian distribution loaders		
5	5 Performance of lognormal distribution loaders		
6	6 Convergence of QMCI calculations		
7	7 Convergence of QAE methods		
8	NISQ resource quantification for the benchmark problems		
9	Fault-tolerant resource quantification for the benchmark problems	68	

Part I. Introduction and background

1. Introduction

For high-dimensional integrals with no analytic solution *Monte Carlo integration* (MCI) is invariably the most efficient method of (classical) numerical integration, and for this there is a proven quadratic quantum advantage. In particular, quantum Monte Carlo integration (QMCI) maps the integral value to the amplitude of a qubit, and then uses quantum amplitude estimation (QAE) to estimate this value. QAE is such that the estimator's root mean-squared error (RMSE) is proportional to the reciprocal of the number of samples, whereas for (classical) MCI the RMSE is proportional to the reciprocal of the square root of the number of samples. Over the past few years, a number of breakthroughs have fuelled the hope that QMCI will be an application of quantum computing that enjoys quantum advantage at a relatively early stage. Various works have shown how QAE can be performed without requiring the computationally costly subroutine of quantum phase estimation (QPE) [1–4] (hereafter these are generally referred to as QPE-free QAE algorithms). Furthermore it has been shown that the Monte Carlo integral can be decomposed as a Fourier series, such that each harmonic can be estimated using only a shallow quantum circuit, whilst maintaining the full quantum advantage [5].

In this paper we present Quantinuum's new end-to-end QMCI engine, which is principally tailored to applications in mathematical (quantitative) finance. Whilst MCI touches on almost every area of science, technology, operations research and business, there has been specific enthusiasm for quantum solutions within the financial sector [6–18]. Many of the problems within mathematical finance are naturally posed in a way that makes them amenable to quantisation, and thus able to benefit from the theoretical quantum advantage. In particular, financial applications often have an appropriate blend of simplicity and complexity. For example, when considering derivative pricing, standard random processes such as geometric Brownian motion may typically be used to model some underlying asset price¹. We can therefore design efficient ways to load these relatively simple random processes as quantum states. However, the actual Monte Carlo integral will itself be complex (or, more precisely, high-dimensional) when the derivative is path-dependent, and hence in principle can benefit from QMCI. Furthermore, when it comes to path-dependent derivative instruments, there is potential for almost infinite variation. In particular, various thresholds (e.g., in barrier options), sums (e.g., to calculate the average as required in Asian options) and maxima (e.g., in look-back options) of random variables pertaining to time-slices of the time-series may be taken². We are thus motivated to provide as key components of the QMCI engine: (i) a library of quantum circuits encoding models of financial time-series; and (ii) a generic way to construct computations that are commonplace in mathematical finance. In the former case, we include some simple explicit circuits that are required for our benchmark results, and we also comprehensively detail a number of generic methods for constructing quantum circuits that efficiently encode probability distributions. In the latter case, we introduce the enhanced P-builder³ – a module that allows efficient construction of quantum circuits encoding a wide variety of financial derivative payoff and portfolio risk calculations.

The quantum circuit encoding the (financial) quantity of interest is then inputted into the Monte Carlo integrator, and for this we use the aforementioned Fourier series decomposition of the Monte

¹At least in textbook models – in time we will supplement these with more advanced models for financial time-series.

²See Table 3 for more details on these instruments.

³"P" has become the standard notation for the quantum circuit that loads a probability distribution – and this is how we use it here.

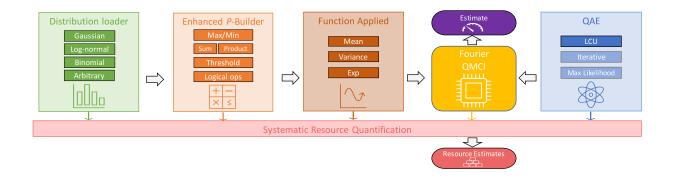


FIGURE 1. A graphical illustration of the end-to-end QMCI engine.

Carlo integral. The Fourier series decomposition is possible when taking expectations of finitely supported probability distributions (as is any probability distribution represented on a digital computer – quantum or classical – even if it actually approximates an infinitely supported distribution) owing to the fact that a periodic piecewise function can be constructed to match the desired function applied to the random variable over the support, and joined up to be smooth elsewhere. Of particular importance are periodic piecewise functions that are (i) linear; and (ii) exponentially increasing in the piece corresponding to the support of the probability distribution, as these enable computation of the mean of the random variable and the mean of its exponential, as required when working in price and return space, respectively. For these statistical quantities, along with the second moment (which we include as another statistical quantity which is often of interest) the QMCI engine includes optimised periodic piecewise functions and we give an explicit analytic upper-bound on the RMSE convergence, which is demonstrated with a simple numerical example.

This upper-bound on QMCI convergence itself relies on a complete characterisation of the underlying QAE subroutine and, whilst the existing literature is reasonably comprehensive, we have identified a significant lacuna that requires attention before such an end-to-end QMCI engine could be proposed. Specifically, the quadratic quantum advantage in RMSE convergence of QAE algorithms masks the fact that the estimators are not statistically robust – in that they obtain the claimed accuracy whilst being skewed and heavy-tailed. This is a potentially significant deficiency of QMCI as, even though the convergence is slower, classical MCI estimators are distributed as unbiased Gaussian random variables. To resolve the issue of statistical robustness, we propose linear combination of unitaries quantum amplitude estimation (LCU QAE) which achieves comparable speed of convergence to existing QAE algorithms, whilst outputting an estimator which is (to an acceptably-close approximation) an unbiased Gaussian random variable.

As well as executing quantum circuits to compute small-scale QMCI problems, the QMCI engine also includes a resource mode, which precisely quantifies the quantum resources (numbers of qubits and gates) required to run any QMCI problem (i.e., including full-scale problems, whose size exceeds what can be executed or simulated on present-day quantum hardware and simulators). Using resource mode, we present a set of benchmarks, corresponding to simple, but not trivial, financial derivative instruments. Thus, the inclusion of resource mode enables the QMCI engine to bridge between the small-scale problems that can be executed on today's quantum hardware and simulators, and the full-scale QMCI problems that will eventually fulfil the promised useful quantum advantage. On this subject, we expect applications to enjoy useful quantum advantage

if they are such that (i) MCI is the best approach classically; (ii) precision is crucial – and it's hard to achieve adequate precision by classical means; and (iii) there is a realistic prospect of complementary quantum advantage in distribution loading. (The third of these is actually optional, and refers to the fact that, if the random process modelling the financial time-series in question can be sampled more efficiently quantumly than classically – which we refer to as "complementary quantum advantage in distribution loading" – then the overall quantum advantage may arise even sooner.) In the context of mathematical finance, we expect calculating the Greeks for financial derivatives, and portfolio risk / optimisation to be the principal beneficiaries. Fig. 1 illustrates the core components and operation of the QMCI engine.

2. Preliminaries

2.1. Probability and statistics.

2.1.1. Random variables. A random variable, X, is realised as some actual value according to its probability distribution. The probability distribution is typically described by its probability density function (PDF) in the case of a continuous random variable, or a probability mass function (PMF) in the case of a discrete random variable. Mathematically, given a d-dimensional domain Ω , we define a PDF f(x) as

$$f_X: \ \Omega \subset \mathbb{R}^d \to \mathbb{R}_+$$

$$x \mapsto f_X(x)$$

$$s.t. \ \int_{\Omega} f_X(x) \, \mathrm{d}x = 1$$

$$(2.1)$$

For our purposes, the random variables always take real numerical values, and the support of the random variable is a single interval of the real line outside of which the PDF or PMF is always zero which means that if the domain Ω is not bounded, the support of the distribution f(x) will be truncated to a bounded space.

The probability that the random variable, X, takes value, x is typically denoted p(X = x) and, slightly overloading notation for simplicity, in the case of discrete random variables we also use p(x) to denote the PMF. In the case of continuously distributed random variables, such a conflation of PDF and probability would be an abuse too far, and so instead we denote the PDF $f_X(x)$, such that in one dimension:

$$p(a \leqslant X \leqslant b) = \int_{a}^{b} f_X(x) dx \tag{2.2}$$

We present in Example 2.1 two standard distributions commonly encountered in mathematical finance.

Example 2.1 (Standard distributions). Two common univariate probability distributions are:

(1) The Gaussian (normal) distribution $X \sim \mathcal{N}(\mu, \sigma^2)$ where

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$
 (2.3)

(2) The log-normal distribution $X \sim \text{Lognormal}(\mu, \sigma^2)$ where

$$f_X(x) = \frac{1}{\sigma x \sqrt{2\pi}} e^{-\frac{1}{2}(\frac{\log x - \mu}{\sigma})^2}$$
 (2.4)

Equivalently, we have $\log X \sim \mathcal{N}(\mu, \sigma^2)$.

2.1.2. Expectation. The expectation of a continuous random variable is defined as

$$\mathbb{E}(X) = \int_{-\infty}^{\infty} x f_X(x) dx \tag{2.5}$$

and the expectation of a discrete random variable is defined:

$$\mathbb{E}(X) = \sum_{x \in \Omega} x p(x) \tag{2.6}$$

where the sum is taken over the entire support of $x \in \Omega$. It is worth noting that expectation is a linear function, and therefore is such that should an affine transform be applied to the random variable, say Y = aX + b, then we have:

$$\mathbb{E}(Y) = a\mathbb{E}(X) + b \tag{2.7}$$

It turns out that this basic property is very useful when it comes to QMCI.

Some function, $q(\cdot)$, may be applied to the random variable prior to the expectation, in which case we have for continuous random variables:

$$\mathbb{E}_{g(x)}(X) = \mathbb{E}(g(X)) = \int_{-\infty}^{\infty} g(x) f_X(x) dx, \qquad (2.8)$$

and for discrete random variables:

$$\mathbb{E}_{g(x)}(X) = \mathbb{E}(g(X)) = \sum_{x \in \Omega} g(x)p(x). \tag{2.9}$$

For instance, the (non-centralised) second moment is such that $g(X) = X^2$.

2.1.3. Indicator functions and "conditional expectation". Indicator functions are defined

$$\mathbb{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$
(2.10)

where A is a set. For our purposes, A is always the subset of the support of a random variable. We use indicator functions to "filter out" certain points of probability mass when computing expectations. Such calculations have the general form:

$$\mathbb{E}_{g(x)}(X|A) = \sum_{x \in \Omega} g(x)p(x)\mathbb{1}_A(x)$$
(2.11)

where $\mathbb{E}_{q(x)}(X|A)$ is a quantity that we (somewhat informally) refer to as the "conditional expectation" (given the subset of the support, A).

2.1.4. Multivariate distributions and marginalisation. Let X and Y be two random variables. If they are independent, then their distributions factorise, i.e.

$$\forall (x,y) : p(X = x, Y = y) = p(X = x)p(Y = y)$$
(2.12)

However, in general this equivalence need not hold, in which case they are dependent random variables that are realised as actual values according to a joint probability distribution. The joint probability distribution is typically described by a multivariate PDF $(f_{X,Y}(x,y))$ in the continuous case; or a multivariate PMF (p(x,y)) in the discrete case. The multivariate setting extends naturally to any number of dimensions, and from a joint distribution it is possible obtain the marginal distribution of any particular random variable. For instance, if some d-dimensional multivariate PDF is $f_{X_1,...X_d}(x_1,...x_d)$, then the random variable pertaining to the first dimension is distributed according to the PDF obtained by marginalisation:

$$f_{X_1}(x) = \int_{x_2...x_d} f_{X_1, X_2, ..., X_d}(x, ..., x_d) dx_2 ... dx_d$$
(2.13)

and similarly with sums in place of integrals for the discrete case (note this applies to any dimension, the first has simply been selected to keep the notation light). As the marginal distribution is the distribution of the first random variable (obtaining by averaging over all of the other random variables), we can also define the expectation:

$$\mathbb{E}_{g(x)}(X_1) = \int g(x) f_{X_1}(x) dx = \int_{x_1 \dots x_d} g(x_1) f_{X_1, \dots, X_d}(x_1, x_2, \dots, x_d) dx_1 \dots dx_d$$
 (2.14)

once again, similarly with sums in place of integrals for the discrete case.

2.1.5. Bayes' law. PDFs and PMFs are typically parametric in nature; as well as taking x as an argument, they contain parameters that dictate the instance of some family of functions. For example, the PDF of the Gaussian distribution, given in Eq. (2.3) and denoted $\mathcal{N}(\mu, \sigma^2)$, is described by two parameters: μ , the mean, and σ^2 , the variance.

Collectively, the parameters of a PDF / PMF are often denoted Θ , and one common task in applied statistics is to infer the parameter values from some data, \mathcal{D} . This data is generally sampled from the random process (i.e., as specified by the PDF / PMF in question) or some variant thereof (for instance, the sample with some function applied). Thus the aforementioned task requires evaluating the *conditional* probability of the parameters *given* the observed data denoted $p(\Theta|\mathcal{D})$. Conditional probabilities are related to joint probabilities by

$$p(\Theta|\mathcal{D})p(\mathcal{D}) = p(\Theta, \mathcal{D}) = p(\mathcal{D}|\Theta)p(\Theta)$$
(2.15)

From this, Bayes' law follows as an easy corollary

$$p(\Theta|\mathcal{D}) = \frac{p(\mathcal{D}|\Theta)p(\Theta)}{p(\mathcal{D})}$$
 (2.16)

which is useful as the probability of the data given the parameters, $p(\mathcal{D}|\Theta)$, is simple to ascertain (this is either the PDF / PMF itself, or some simple function thereof). In Bayesian statistics $p(\mathcal{D}|\Theta)$ is known as the likelihood (of the data), $p(\Theta)$ is the prior belief about the parameter values, $p(\Theta|\mathcal{D})$ is known as the posterior (which is generally what is sought), and $p(\mathcal{D})$ is treated as a normalising constant.

2.1.6. Estimators. The final step in the "common task" introduced above is to extract some estimate, $\hat{\Theta}$ of the parameters from the posterior distribution. Suppose that some random process is parameterised by a single parameter, θ (as it indeed is in QAE): one important figure of merit is the mean-squared error (MSE) of the estimator $\hat{\theta}$, (itself a random variable whose probability distribution, $p(\hat{\theta})$, depends on data and the estimator type), with respect to the true θ

$$MSE = \mathbb{E}((\hat{\theta} - \theta)^2) = \int (\hat{\theta} - \theta)^2 p(\hat{\theta}) d\hat{\theta}$$
 (2.17)

RMSE =
$$\sqrt{\mathbb{E}((\hat{\theta} - \theta)^2)} = \sqrt{\int (\hat{\theta} - \theta)^2 p(\hat{\theta}) d\hat{\theta}}$$
 (2.18)

where we also formally define the RMSE.

By treating the MSE as the key figure of merit, one important estimator immediately follows, namely the minimum mean-squared error (MMSE) estimator. However, even though the aim is to choose a value of $\hat{\theta}$ that suppresses the MSE, this cannot be done directly from Eq. (2.17) as the true parameter value, θ , is in general unknown (that is after all what is being estimated). Instead,

the probability of θ given the empirical data is used:

$$\hat{\theta}_{\text{MMSE}} = \arg\min_{\hat{\theta}} \left(\left(\mathbb{E}((\hat{\theta} - \theta)^2) \right) = \arg\min_{\hat{\theta}} \left(\int (\hat{\theta} - \theta)^2 p(\theta | \mathcal{D}) d\theta \right)$$
 (2.19)

and similarly with a sum in place of the integral for the discrete case.

Another important estimator is the maximum a-posteriori (MAP) estimator:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\hat{\theta}} (p(\hat{\theta}|\mathcal{D})) \tag{2.20}$$

In the case of a uniform prior, the MAP estimator is equal to the maximum likelihood estimator:

$$\hat{\theta}_{\text{MLE}} = \underset{\hat{\theta}}{\arg\max}(p(\mathcal{D}|\hat{\theta})) \tag{2.21}$$

In each of these cases (MAP estimator and MLE), the PDF / PMF is supported on a range of θ .

2.1.7. Monte Carlo integration. Suppose that there is some random process which we can sample (termed "sample access"), but for which we cannot necessarily query the probability mass / probability density at any point in the support (termed "query access") – then one way to estimate the mean of the distribution is by MCI:

$$\mathbb{E}(X) \approx \frac{1}{q} \sum_{i=1}^{q} X_i \tag{2.22}$$

where X_i are independent, identically distributed (iid) samples from the random process (of which there are q in total). Furthermore, this also applies to the case where a function is applied to the random variables:

$$\mathbb{E}_{g(x)}(X) \approx \frac{1}{q} \sum_{i=1}^{q} g(X_i)$$
(2.23)

If we let Y be the random variable Y = g(X) (with special case g(X) = X such that Y = X), then we can express the convergence of the estimator for the mean, $\hat{\mu}_Y$, as approximated by MCI, in terms of RMSE:⁴

$$\sqrt{\mathbb{E}((\hat{\mu}_Y - \mathbb{E}(Y))^2)} = \frac{\sigma_Y}{\sqrt{q}}$$
 (2.24)

where σ_Y is the standard deviation of Y. If we further let $\max(Y)$ and $\min(Y)$ be the maximum and minimum points of non-zero probability density / mass for the random variable Y (i.e., the extrema of its support), then we further get:

$$\sqrt{\mathbb{E}((\hat{\mu}_Y - \mathbb{E}(Y))^2)} \leqslant \frac{\max(Y) - \min(Y)}{2\sqrt{q}}$$
 (2.25)

with the bound saturated when half of the probability mass is concentrated at each of the extrema.

 $^{^4}$ Note that in the case that the samples are not iid, a similar expression with a different exponent of q may hold. Note also that for some (heavy-tailed) distributions the standard deviation could be infinite, in which case this is not a meaningful estimator.

- 2.1.8. Implicitly-defined random variables and marginal Monte Carlo integrals. Suppose some random variable, Y, is defined as a function of other random variables, $\{X_i\}$, and it is not possible to analytically express the PDF / PMF thereof. In this case, we have a joint distribution over $Y \cap \{X_i\}$, and even if the desired statistical quantity is some expectation only of Y, in general (for sufficiently large number of random variables, i.e., large $|\{X_i\}|$) the most efficient method will still be to sample from the joint distribution and estimate the expectation of Y as a marginal variable accordingly. This is a manifestation of the essential property that MCI is the most efficient means of estimating expectations of high-dimensional random processes, and for applications of QMCI it is of paramount importance to appreciate that this essential principle holds even when only the marginal expectation of a single, implicitly-defined random variable, is desired.
- 2.2. Quantum states as probability distributions. In this section we assume that readers are familiar with the essentials of quantum computation⁵, and focus in on the specifics of how quantum circuits are used to represent statistical quantities.
- 2.2.1. Qubit bundles as registers. Suppose we have a computational basis state over some n qubits, then by treating the first qubit as the most significant bit, we can read the computational basis state, $|b_1b_2...b_n\rangle$, as the bitstring $b_1...b_n$. We refer to blocks of qubits that we treat as representing some such binary values as registers and, in the absence of any additional information, treat x as a binary value.
- 2.2.2. Quantum states as probability distributions over binary values. If we have some superposition of n qubits (for arbitrary relative phases, φ_i)

$$|p\rangle = \sum_{x_i \in \{0,1\}^n} e^{2\pi \mathbf{i}\varphi_i} \sqrt{p_i} |x_i\rangle$$
 (2.26)

then measurement in the computational basis will have the effect of sampling a random variable distributed such that the binary value x_i occurs with probability p_i . The nomenclature " $|p\rangle$ " is deliberately suggestive of probability, and we denote a circuit that prepares this state from the all-0 state P, that is: $|p\rangle = P|0^n\rangle$. Building a circuit P to efficiently encode financial random processes is one of the central topics we address in this paper, in particular we propose our enhanced P-builder in Section 4.

- 2.2.3. Quantum states as multivariate probability distributions over real numbers. In general, we do not wish simply to sample binary values, but real numbers corresponding to (regularly spaced) discrete points on the real line. Furthermore, as MCI is only the best solution (classically) for high dimensional integrals, it is necessary to use quantum states to encode multivariate distributions. Both of these can easily be achieved, taking a lead from the notation in Ref. [5], any quantum circuit with some n qubits may be interpreted as preparing a d-dimensional multivariate distribution where each dimension is represented by n_i qubits, also called wires hereafter, where i indexes the dimension, and is such that $n = \sum_{i=1}^{d} n_i$. Furthermore, for each dimension, the binary numbers supported are interpreted as real numbers starting at some position $x_{\ell}^{(i)}$ and with equal spacing $\Delta^{(i)}$. This is sketched in Fig. 2.
- 2.2.4. Implicit marginalisation. The Born rule for evaluating the measurement outcome probabilities when a subset of the qubits are measured in the computational basis is such that marginalisation occurs implicitly. That is, if the qubits corresponding to the i^{th} dimension are measured in the computational basis, and all of the other qubits are left alone, then the measurement outcomes will sample from the marginal distribution of the i^{th} random variable. This is critical, as the settings

⁵We recommend Ref. [19] if this is not the case.

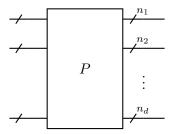


FIGURE 2. An illustration of how a general circuit, P, can be interpreted as loading a multivariate probability distribution as a quantum state.

we consider for QMCI are those covered in Section 2.1.8, where only the marginal expectation is of interest.

2.3. Quantum Monte Carlo integration. Suppose we wish to perform (marginal) MCI for some implicitly-defined random variable, X, where sample access to the corresponding joint distribution has been encoded in a circuit, P. We let d be the number of dimensions in the joint distribution, and to simplify the following notation and diagrams, we let the dimension in question be the final, d^{th} , dimension. We now perform an affine transformation on the support of the $d^{(th)}$ dimension,

$$\tilde{X}^{(d)} = aX^{(d)} + b \tag{2.27}$$

such that the support of $\tilde{X}^{(d)}$ is [0, 1], which simply amounts to changing the description of how samples from the d^{th} dimension are interpreted as real numbers (we later perform the inverse affine transform in classical post-processing to recover the mean of the actual random variable with this shifting and re-scaling).

A circuit, denoted R and termed the "quantum arithmetic circuit", is then executed to (approximately) reversibly apply the function $\arcsin(\sqrt{\tilde{x}^{(d)}})$, such that the result is placed in a new qubit register (of appropriate size). This register is then used to control a bank of R_y rotation gates, such that the state⁷

$$\cos(\arcsin\sqrt{\tilde{x}^{(d)}}) |\Phi_0\rangle |0\rangle + \sin(\arcsin\sqrt{\tilde{x}^{(d)}}) |\Phi_1\rangle |1\rangle = \sqrt{1 - \tilde{x}^{(d)}} |\Phi_0\rangle |0\rangle + \sqrt{\tilde{x}^{(d)}} |\Phi_1\rangle |1\rangle \quad (2.28)$$

is prepared. Here $|\Phi_0\rangle$ and $|\Phi_1\rangle$ are quantum states whose exact expression is not of concern (but, slightly departing from the usual convention, they are each considered to have an explicit global phase – otherwise this expression is not sufficiently general). In general, this will actually happen for a superposition of computational basis states, as dictated by the circuit, P, which thus gives:

$$|\psi\rangle = \sum_{i} \sqrt{p_i} \left(\sqrt{1 - \tilde{x}_i^{(d)}} |\Phi_0\rangle |0\rangle + \sqrt{\tilde{x}_i^{(d)}} |\Phi_1\rangle |1\rangle \right)$$
 (2.29)

and thus, by the Born rule, the probability of measuring 1 on the final qubit is

$$\sum_{i} \left(\sqrt{p_i \tilde{x}_i^{(d)}} \right)^2 = \mathbb{E}(\tilde{x}^{(d)}) \tag{2.30}$$

as desired. QAE can then be used to estimate this quantity, with RMSE convergence proportional to inverse of the number of samples (number of uses of the circuit P). Defining $\hat{\mu}$ as the estimate,

⁶Here we assume that the desired expectation is $\mathbb{E}(X)$ and any applied functions (other than g(X) = X) have been included in the circuit P.

⁷This bank of controlled rotation gates is of the same form as that shown in Ref. [5, Fig. 1] – that is, such that the rotation angle is equal to the value in the register.

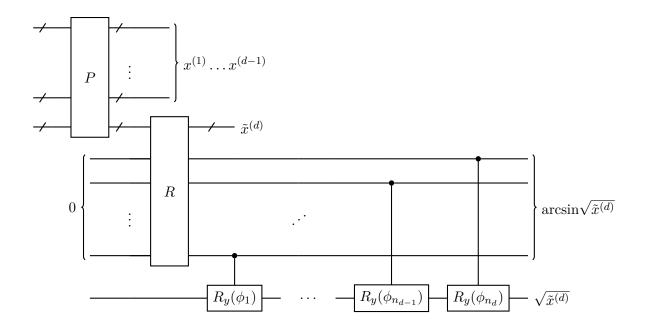


FIGURE 3. An illustration of how a probability distribution loading circuit, P, can be supplemented with a quantum arithmetic circuit, R, such that an expectation value of interest is encoded in the amplitude of a qubit. In order for the bank of controlled rotations to implement the sine, angles $\{\phi_1, \ldots, \phi_{n_d}\}$ will be some constant multiplied by a 2 raised to incrementally increasing (integer) powers. Note that for illustrative purposes we have used a mixed convention in the qubit labelling in this figure: $x^{(1)} \ldots x^{(d-1)}$, $\tilde{x}^{(d)}$, the label 0 to the left of R and $\arcsin \sqrt{\tilde{x}^{(d)}}$ all represent values encoded in binary registers, whereas $\sqrt{\tilde{x}^{(d)}}$ on the bottom-most qubit represent the amplitude of this qubit (such that the outcome one is measured with probability $\tilde{x}^{(d)}$.

 c_{QAE} as the constant of proportionality, and applying the inverse affine transform, we get overall convergence:

$$RMSE \leqslant c_{QAE} \frac{x_u^{(d)} - x_\ell^{(d)}}{q}$$
(2.31)

where $x_u^{(d)}$ is the largest value in the support of the d^{th} random variable (this represents the scaling of the error, when the original affine transform to map the support to [0,1] is reversed). Equation (2.31) therefore represents a quadratic advantage over classical MCI (i.e., in Eq. (2.25)). A sketch of how the circuits P and R are composed is given in Fig. 3.

2.4. Basics of mathematical finance. Mathematical finance is a rich and wide-ranging field in its own right, and so it is therefore only possible to address a relatively simple abstraction – even in a paper such as this, which primarily focuses on applications of QMCI to mathematical finance. For instance, we largely gloss over the important general distinction between the "P" and "Q" worlds. The latter includes the pricing of derivative instruments which covers most of the examples we include – although we do mention how the same computational methods may be applied to risk and portfolio management calculations (which belong to the P world). This conflation of the two worlds is perhaps made a little more palatable (and mathematically sound) by the fact that we set

the risk-free rate to zero (for simplicity) in the examples and benchmarks we give⁸. Even within the Q world, our examples are somewhat specialised, and we essentially focus on equities – even though the methods apply equally to other assets such as FX instruments. For the reader who wishes to delve deeper into mathematical finance, we recommend Ref. [20].

- 2.4.1. Assets. An asset is a resource with economic value, such as a stock in a listed company. For our purposes, we will assume that all assets have a well-defined price (given by some market) which varies with time.
- 2.4.2. *Portfolios*. A portfolio is a set of one or more assets, which in general may be correlated. For theoretical purposes a portfolio is any set of assets, whether or not it represents a sensible real-world example thereof.
- 2.4.3. Financial instruments: derivatives & options. Rather than simply buying an asset, one may purchase a derivative instrument (or simply "derivative") such as an option, where one has the option but not obligation to buy (termed a "call option") or sell (termed a "put option") a certain asset (termed the "underlying") on a certain date at a pre-defined price (termed the "strike price"). Such an option is clearly advantageous for the holder as it need only be exercised if the market conditions are favourable and so it follows that options are themselves assets, and indeed the "underlying" in an option contract need not be a stock, but could itself be some derivative instrument.
- 2.4.4. Pricing derivatives. When two counterparties enter into a derivative they need to agree on a price. According to basic theory, the price they should (optimally) agree on is known as the "fair value" which depends on the expected trajectory of the price of the underlying and the strike price (if applicable); for instance a call option with a low strike price on an asset whose value is expected to increase should be relatively expensive to purchase. Option prices can be calculated from the expected payoff. As an example, consider a European call option with strike price, S; in this case the payoff is

$$Payoff = \max(X - S, 0) \tag{2.32}$$

where X is a random variable modelling the price of the underlying at the expiry time of the contract. Typically the *expected* payoff – averaging over X – is what needs to be calculated (or estimated) to price a derivative. Here the "max" condition captures the fact that the option will only be exercised if the underlying has reached a price exceeding the strike price, otherwise the option will not be exercised and the payoff will be zero.

2.4.5. Path dependency. There are myriad types of financial derivatives, however an important distinction is whether a derivative is path independent, where the expected payoff only relies on the underlying's price at the derivative's expiry time; or path dependent, where the expected payoff depends on the price of the underlying throughout the lifetime of the derivative (and thus depending on the "path" that the asset price has taken).

Discretised versions of path-dependent options are high-dimensional random processes (represented by multivariate probability distributions with a dimension for each time-slice), and it follows that MCI (and variants thereof) is usually the most effective means of computing the expected payoff classically, and hence there is an opportunity for quantum advantage by using QMCI.

⁸Setting the risk-free rate to zero is a simplification that we make for convenience, and does not limit the applicability – or quantum advantage – of the techniques we use in this paper.

 $^{^9}$ Derivatives of FX and interest rate instruments are also traded – but this distinction is not required in what follows.

Conversely, path-independent options are generally low-dimensional random processes, and hence no such opportunity exists – MCI is unlikely to be the best approach classically, unless high-dimensionality is introduced in another way, such as in basket options where the underlying is a group of (non-trivially correlated) assets.

- 2.4.6. Price and return space. In mathematical finance, estimates are made given statistical models for how the price of an asset or a portfolio will evolve in the future. If this is modelled directly, this is termed working in "price space"; however, in practise it is often beneficial to work with the (natural) logarithm of the price; this is termed working in "return space".
- 2.4.7. Volatility. How much an asset's price is expected to vary is measured by its volatility. For the purposes of this paper, we need only consider a simple textbook model where the price varies according to geometric Brownian motion, and hence in return space the price is modelled by a Gaussian random variable. The total volatility is then the standard deviation of this Gaussian random variable at the final time in the time series.
- 2.4.8. Use of Monte Carlo integration in mathematical finance. MCI is generally the most efficient means of numerically evaluating high-dimensional integrals, other methods such as Quasi-Monte Carlo [21] suffer the "curse of dimensionality" meaning that the computational cost grows exponentially with the number of dimensions. Alternative approaches such as solving stochastic differential equations (and so bypassing numerical integration altogether) also only work well in the low-dimensional setting.

As already mentioned, within mathematical finance, path-dependent derivatives are typically modelled as high-dimensional random processes, as are portfolios of correlated assets (whose risk may need to be MCI in this paper – as it is general purpose – there exist a host of techniques that have been tailored to various settings, and may represent (possibly marginal) classical improvements for some of the examples we give, see e.g. Refs [22–24].

3. Summary of our previous works

- 3.1. The problem with Grover-Rudolph state preparation. In the overview of QMCI in Section 2.3, the number of classical samples is equated with the number of uses of the quantum circuit encoding the sample access, P. This equivalence assumes that whenever it is possible to classically sample, then a corresponding circuit P can be constructed. The Grover-Rudolph method of state preparation has frequently been cited as evidence of this equivalence [25], however in an earlier work we showed that, owing to the reliance on (classical) MCI in the Grover-Rudolph method, this represents a false economy [26]. Specifically, as the desired accuracy of the estimator increases, so does the circuit depth of the Grover-Rudolph method, meaning that the overall convergence is no better than classical MCI. For this reason, we do not rely on Grover-Rudolph as a means to deliver quantum advantage in our QMCI engine.
- 3.2. Sampling with reversible circuits. Whilst the Grover-Rudolph method was shown to be inadequate as a state preparation method for QMCI, we showed that simply encoding the classical sampling circuit itself as a reversible circuit is sufficient to prove the equivalence between classical samples and quantum uses of P [27] To do so, we noted that classical samples are obtained by mapping uniform (pseudo) randomness to samples from the required distribution, and this "classical sampling circuit" can be used directly (in its reversible form) as the circuit P. As this construction leaves any intermediate results as further dimensions of a joint distribution, its utility relies directly on the implicit marginalisation, discussed in Section 2.2.4. Encoding probability distributions into quantum states using reversible circuits constitutes one of the methods mentioned in Section 7.

3.3. Fourier quantum Monte Carlo integration. The seminal observation amongst our earlier works is that the Monte Carlo integral can always be decomposed as a Fourier series which means that the computationally-expensive quantum sub-circuit, R (as in Section 2.3), which performs a quantum arithmetic operation using quantum operations can be omitted and furthermore, that the full quadratic quantum advantage can still be achieved in this way¹⁰.

The central idea is that, any function, g(.) (including the simple case g(x) = x) applied to a random variable can be altered outside of the support of the random variable, without affecting the expectation value. So it follows that a periodic piecewise function, g(x), can be constructed such that g(x) = g(x) within the support of x and elsewhere adheres to certain smoothness conditions¹¹. As periodic functions always have a Fourier series, it follows that the Monte Carlo integral can be evaluated by performing a weighted sum of terms of the form $\mathbb{E}_{\sin(m\omega x)}(x)$ and $\mathbb{E}_{\cos(m\omega x)}(x)$ and that such quantities can be calculated "naturally" in a quantum circuit by simply applying the bank of controlled rotation gates to the random variable in question. To see how this works in more detail, we begin by formally and generally defining a QAE algorithm:

Definition 3.1 (Quantum amplitude estimation algorithm). A QAE algorithm takes as an input an n+1 qubit circuit A where $A|0^{n+1}\rangle = \cos\theta |\Phi_0\rangle |0\rangle + \sin\theta |\Phi_1\rangle |1\rangle$ (where $|\Phi_0\rangle$ and $|\Phi_1\rangle$ are arbitrary n-qubit states), and returns an estimate, \hat{a} , of $a = \sin^2\theta$. The allowed number of uses, denoted "q", of the circuit A is given as an input to the QAE algorithm.

From this definition, we can write any QAE algorithm as a function in pseudocode form:

$$\hat{a} = QAE(A, q) \tag{3.1}$$

such that convergence of any QAE algorithm can be expressed as:

$$RMSE \leqslant c_{OAE}q^{-\lambda/2} \tag{3.2}$$

for some constant c_{QAE} , and where $1 \leq \lambda \leq 2$.

Fourier QMCI uses parameterised circuits of the form, $A(P, i, \beta, m, \omega)$, where:

- P is a circuit that loads a multivariate probability distribution;
- *i* is a positive integer specifying the index of the dimension for which the marginal expectation will be estimated:
- β is either 0 or $\pi/2$, and enables respectively either the cosine or sine of the value in the qubit register to be computed;
- m is a positive integer that corresponds to the index of a Fourier series harmonic 12 ;
- ω is the angular frequency of a periodic piecewise function that applies a desired function over the support of the random variable.

Figure 4 details how the circuit A is built from these parameters as well as: the value of the first point of non-zero probability mass, x_{ℓ} , and spacing between points of probability mass, Δ , for the dimension of interest. From this, we get:

Proposition 3.2. [5, Proposition 2] Let QAE(A,q) be a QAE algorithm in the sense of Definition 3.1, which has RMSE convergence parameterised by λ as in Eq. (3.2).

¹⁰The subject invention described in the article is US patent pending and titled "Quantum Computing System and Method" with Publication Number US2023/0036827.

¹¹Note that f(.) and f(.) were used in place of g(.) and g(.) in Ref. [5].

 $^{^{12}}$ In Ref. [5] this was n, and has been changed as n is used throughout this paper to denote the number of qubits.

i. $1 - 2QAE(A(P, i, 0, m, \omega), q)$ is an estimate of:

$$\sum_{x^{(i)} \in \Omega} p(x^{(i)}) \cos(m\omega x^{(i)}) \tag{3.3}$$

with $RMSE \leq 2c_{QAE}q^{-\lambda/2}$.

ii. $1-2QAE(A(P,i,\pi/2,m,\omega),q)$ is an estimate of:

$$\sum_{x^{(i)} \in \Omega} p(x^{(i)}) \sin(m\omega x^{(i)}) \tag{3.4}$$

with $RMSE \leq 2c_{QAE}q^{-\lambda/2}$.

where Ω is the set over which the variable x has been discretised¹³.

This proposition then allows us to bypass the circuit R: first we build a periodic piecewise function g(x) such that g(x) = g(x) over the support of p(x). For example, some g(x) of the following form, which repeats with period $x_{\tilde{u}} - x_{\ell}$, is suitable in general:

$$g(x) = \begin{cases} g(x) & \text{if } x_{\ell} \leqslant x < x_{u} \\ \tilde{g}(x) & \text{if } x_{u} \leqslant x < x_{\tilde{u}} \end{cases}$$
 (3.5)

where $x_{\tilde{u}} \ge x_u$ and $\tilde{g}(x)$ is itself sufficiently smooth and chosen such that the pieces join sufficiently smoothly. As g(x) is periodic, it has a Fourier series:

$$g(x) = a_0 + \sum_{m=1}^{\infty} a_m \cos(m\omega x) + b_m \sin(m\omega x)$$
(3.6)

where $\omega = 2\pi/T$ and T is the period of the periodic piecewise function.

We define μ as the expectation value we wish to approximate, and as g(x) = g(x) whenever $p(x^{(i)}) \neq 0$, we can express:

$$\mu = \sum_{x^{(i)} \in \Omega} p(x^{(i)}) g(x^{(i)})
= \sum_{x^{(i)} \in \Omega} p(x^{(i)}) g(x^{(i)})
= \sum_{x^{(i)} \in \Omega} p(x^{(i)}) \left(\sum_{m=1}^{\infty} \left(a_m \cos(m\omega x^{(i)}) + b_m \sin(m\omega x^{(i)}) \right) + a_0 \right)
= a_0 + \sum_{m=1}^{\infty} a_m \left(\sum_{x^{(i)} \in \Omega} p(x^{(i)}) \cos(m\omega x^{(i)}) \right) + b_m \left(\sum_{x^{(i)} \in \Omega} p(x^{(i)}) \sin(m\omega x^{(i)}) \right)$$
(3.7)

To estimate μ , we can thus estimate each of the parenthesised sums in the final line of (3.7) individually using the result of Proposition 3.2. Upon this, the central result is based:

Theorem 3.3. [5, Theorem 3]. The quantity, $\mu = \mathbb{E}(g(X))$, where $X \sim p(x^{(i)})$ and g is a function that is continuous in value and first derivative and whose second and third derivatives are piecewise-continuous and bounded, can be estimated with $RMSE \leq \tilde{c}_g q^{-\lambda/2}$, where q is the number of uses of a circuit preparing the quantum state $|p\rangle$, \tilde{c}_g is a constant that depends on $g(\cdot)$ and c_{QAE} , and λ

¹³Note that we take the discretised random variable as the defined input and so are not concerned here with the work needed to actually do the discretisation.

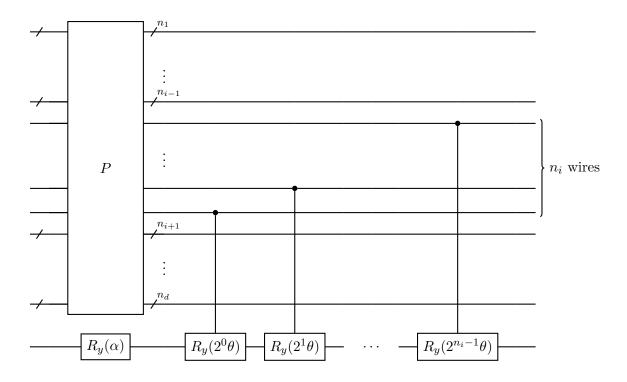


FIGURE 4. Circuit diagram of $A(P, i, \beta, m, \omega)$, where $\alpha = m\omega x_{\ell} - \beta$ and $\theta = m\omega \Delta$, in which x_{ℓ} and Δ are the first point of probability mass and spacing (as defined in Section 2.2) for the i^{th} dimension of P.

is the convergence rate of some QAE subroutine (i.e., as defined in Eq. (3.2)) which operates on circuits of the form defined in Fig. 4.

Here, the smoothness conditions suffice to guarantee that the quadratic quantum advantage is always retained. This Fourier series decomposition approach to QMCI is extremely flexible in terms of the functions that can be applied, and a key element of the design of the QMCI engine is the construction of suitable periodic piecewise functions for statistical quantities of interest.

3.4. Noise-aware quantum amplitude estimation. QAE circuits – in particular circuits for QPE-free QAE – all have the same regular structure. They all consist of repeated instances of the same sub-circuit, Q, followed by the measurement of a single qubit. In our previous work, we showed how these properties can be exploited to model the inclement noise as a Gaussian random variable, regardless of the actual underlying noise characteristics – and furthermore, that such a random variable can be accounted for as if it were an additional component of variation in the parameter estimation [28]. This we named noise-aware QAE, and we showed how the noise parameter could be co-learned alongside the parameter (amplitude of interest), such that the additional component of variation introduced by the hardware noise can be exactly compensated for with additional shots of the circuit.

For practical purposes, one way to think about noise-aware QAE is as a means of application-specific error-mitigation, which allows the maximum executable circuit depth to be extended by additional sampling. For instance, observe that a simple depolarising noise model for the machine IBMQ *Rome* running circuit A_1 has $\tilde{p}_{coh} = 0.9276$ according to Ref. [28, Table 1], and a simple

calculation therefore shows that (according to definition of \tilde{p}_{coh} in Ref. [28, Eq. (6)]) after $1/(1-0.9276) \approx 14$ Grover iterates, it is expected that the state has decohered. However, in Ref. [28, Fig. 3 Rome] it can be seen that there is still a significantly higher than 0.5 probability of measuring 1 (as would be the case for a fully decohered state) when there are many more Grover iterates than 14. Noise-aware QAE therefore effectively allows the "signal-to-noise ratio" to be amplified in such cases in a manner that naturally combines with the outputted amplitude estimate. As such, for experimental purposes we can leverage noise-aware QAE to run deeper QMCI circuits than one may expect from simple "rules of thumb".

Part II. Principal novel methods

4. Enhanced P-builder

In principle it is always possible to offload the entire work of building the circuit P (as defined in Section 2.2) onto the user (that is, they should either construct and submit their own circuit, or call a circuit from the standard library of distributions, as detailed in Section 7). However, our central philosophy is that the QMCI engine should allow users to build complex financial instruments from simple building blocks, and to this end, the circuit P can be enhanced by the following:

- (1) Taking simple functions of the random variables sampled from one or two dimensions of a multivariate probability distribution to define new random variables (which can therefore be represented in a multivariate distribution with the number of dimensions increased accordingly). One such function is "summing", and as it is easy to keep track of the number of random variables being summed, this operation also encompasses averaging, which is needed, for example, in pricing Asian options.
- (2) Comparing the value of some random variable (i.e., the value sampled from some dimension of the multivariate probability distribution) to (i) that from another dimension or; (ii) some pre-defined threshold. These comparisons are set up as Boolean statements, and *indicator* qubits are then added to represent the truth of these statements. Thresholds are required in, for example, pricing barrier options.
- (3) Indicator qubits may be combined in further Boolean expressions to give additional indicator qubits. For financial instruments whose payoff relies in a non-trivial way on various thresholds / comparisons, this allows the payoff to be conditioned on any logical function of these.

In Section 4.2 we give an explicit example of how some of these functions are used to construct *look-back options*; and in Table 3 we sketch how these operations can be used to compute the payoff for some of the most widely-used financial derivatives, as well as enabling common portfolio risk metrics to be calculated¹⁴.

4.1. Enhanced P-builder functions. From user-specified commands, the enhanced P-builder automatically constructs efficient reversible circuits that accomplish the required function. In particular, by making mild (and practically reasonable) restrictions about the way that the binary registers encode numerical values, it is possible to use (reversible forms of) binary operations directly, hence avoiding the costly full arithmetic circuitry. Table 1 summarises the functions that generate new random variables from existing ones (that is, those functions pertaining to the first item in the preceding enumeration). In each case the function takes either one or two random variables as its input, and it is worth noting that there is no loss in generality in this, as all of the operations are associative, and so it is trivial to extend to higher numbers of random variables by repetition.

The second and third items in the preceding enumeration may be grouped as logical rather than binary operations, and prepare single qubits whose truth value is determined by (and therefore correlated with) the values in the registers representing the relevant dimensions of the multivariate probability distribution. These qubits therefore encode Bernoulli random variables when measured in the computational basis, whose probability is equal to the probability that the encoded logical statement is true. Table 2 summarises the logical operations included in the enhanced P-builder. In the case where the user wishes to add a further indicator qubit as a function of the existing indicator qubits, then this logical expression must be explicitly given in the form of an exclusive sum of products (ESOP - see [29] and references therein) of the input indicator qubits (and/or their

¹⁴The enhanced P-builder is the subject of patent applications GB2210997.9 and GB2301481.4.

Function	Inputs	Output
Sum	$x^{(i)}, x^{(j)}$ $x^{(i)}, c$	$x^{(d+1)} = x^{(i)} + x^{(j)}$ $x^{(d+1)} = x^{(i)} + c$
Product	$x^{(i)}, x^{(j)}$ $x^{(i)}, c$	$x^{(d+1)} = x^{(i)} \times x^{(j)}$ $x^{(d+1)} = x^{(i)} \times c$
Maximum	$x^{(i)}, x^{(j)}$ $x^{(i)}, c$	$x^{(d+1)} = \max(x^{(i)}, x^{(j)})$ $x^{(d+1)} = \max(x^{(i)}, c)$
Minimum	$x^{(i)}, x^{(j)}$ $x^{(i)}, c$	$x^{(d+1)} = \min(x^{(i)}, x^{(j)})$ $x^{(d+1)} = \min(x^{(i)}, c)$

Table 1. Operations combining random variables contained in the registers pertaining to the dimensions of a d-dimensional multivariate probability distribution (encoded in a quantum state). For all i it is necessary that $\Delta^{(i)} = 2^{\alpha}$ for some (positive or negative) integer α . Strictly speaking this requirement is stronger than necessary for certain operations, however it is sufficient for all, and moreover it provides closure – in the sense that the new dimension will certainly have Δ which is a integer power of 2. In each case the operation may combine two random variables, or a random variable and a classical constant, c, which is taken as an input. Observe that each operation prepares a new (d+1)th dimension of the probability distribution, and the reversible nature means that the original d dimensions are also returned unaltered.

Function	Inputs	Output	Notes
Compare	$x^{(i)}, x^{(j)}$	$\mathrm{IF}(x^{(i)} \geqslant x^{(j)})$	Require: $\Delta^{(i)} = 2^{\alpha} \Delta^{(j)}$
Threshold	$x^{(i)}, c, \text{TYPE}$	$ IF(x^{(i)} \ge c) IF(x^{(i)} < c) $	When $TYPE = lower$ When $TYPE = upper$
ESOP	$\{B\}$	$ESOP(\{B\}), ESOP$	_

TABLE 2. Operations preparing indicator qubits from the values in the registers corresponding to the dimensions of the multivariate probability distribution (first two operations); or existing indicator qubits (third operation). α is a positive or negative integer and $\{B\}$ is a set of Boolean variables; and for the third operation the specific ESOP is explicitly given by the user.

negations). There is no loss of generality in so doing, as every logical expression is equivalent to some ESOP, and in specifying this format it is easy to construct the corresponding quantum circuit.

Figure 5 gives an illustration of how the enhanced circuit P differs compared to that which was input. Notably, any of the output indicator qubits can be used to control the function applied as detailed in Section 8, and in this way the enhanced P-builder provides a general framework for constructing quantities of interest in mathematical finance.

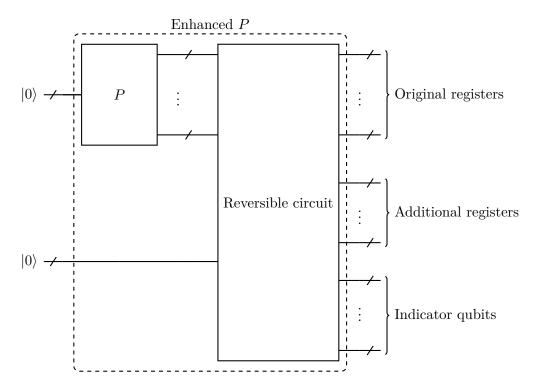
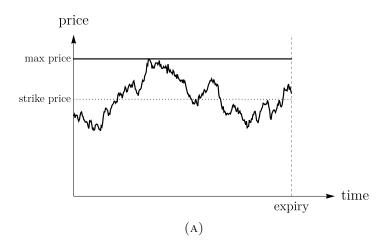


FIGURE 5. A general illustration of how the enhanced P-builder augments an input circuit P with further multivariate probability distribution dimensions and indicator qubits.

- 4.2. **Example: look-back options.** An example of the sort of calculation for which the enhanced P-builder can be used is look-back option payoffs. We can model a look-back call option as taking the maximum asset value over all of the time slices, and choosing to exercise the option if this maximum value exceeds the strike price. We can therefore use the max function of the enhanced P-builder, followed by a conditional expectation (see Section 8), with the condition being that the maximum price exceeds the strike price (thus constituting a threshold operation). The instructions to programme the enhanced P-builder to build such a circuit are given in Fig. 6.
- 4.3. Common financial calculations encompassed by the enhanced P-builder. Let a suitable circuit, P, encode some financial time-series model for a number of (possibly correlated) assets that has been sliced appropriately such that a multivariate distribution is prepared with each dimension representing a unique asset / time pair. If such a P is taken as an input, as is the case for example of look-back options given in Section 4.2, then the enhanced P-builder provides an extremely flexible framework to construct an appropriate circuit to compute the payoff for many of the financial instruments that are commonly traded. In Table 3 some of these are summarised, with high-level descriptions of how each instrument's payoff is programmed. Additionally, certain financial risk calculations can be programmed with the enhanced P-builder, in particular when the input circuit P encodes a model for the prices of a large number of correlated assets composing a portfolio. For example, CVaR is essentially a European option payoff (on the total value of the portfolio) with the VaR value (once found) as strike price.
- 4.4. Using the enhanced P-builder to construct (geometric) Brownian motion. The functions contained within the enhanced P-builder involve the addition of dimensions to multivariate



```
# The enhanced P-builder operates on a circuit encoding 4 time-slices stored as
# four dimensions of a multivariate distribution

description operations = [Max(1,2), Max(3,4), Max(5,6)]
# These max operations create new dimensions 5, 6 and 7 respectively

thresholds = [Threshold(dimension=7, value=1, type=BoundType.Lower),]
# The threshold is taken on the max of all 4 time-slices (encoded in dimension 7)

(B)
```

FIGURE 6. (A) An illustrative sketch of a financial time-series, with strike price shown, and a look-back option payoff being (i) the maximum price reached in the lifetime of the option if this value is greater than the strike price; (ii) zero otherwise. (B) An example of how the enhanced P-builder can be used to construct a quantum circuit that encodes the payoff of a look-back option when the continuously varying time series is modelled as a discrete random process at four time slices. The threshold operation prepares an indicator qubit which can then be used to control the expectation, as detailed in Section 8.

probability distributions, and this feature can be put to additional use for constructing simple models of financial time-series.

If some time-series modelled by Brownian motion (as is a textbook model for financial time-series in return space) and is time-sliced at regular intervals, then the joint distribution between the random variables at the various time slices is a multivariate Gaussian distribution. However, the fact that the motion is Brownian motion means that the correlations in the multivariate Gaussian are heavily structured, and hence a general purpose multivariate distribution loading circuit may incur unnecessary operations. Instead, we can see that using the enhanced P-builder, it is possible to load a circuit P consisting of iid univariate Gaussians, and then to sum these random variables using the enhanced P-builder functionality, to construct additional dimensions corresponding to the Brownian motion. Similarly, if iid lognormal distributions are loaded then multiplication can be applied to yield geometric Brownian motion (the corresponding textbook model when working in price space). Figure 7 gives the instructions that programme the enhanced P-builder to construct Brownian

Option	Operations	Thresholds	Payoff conditions
European	-	Value = strike price; dimension = final time-slice	Threshold is met
Asian	Sum time slices	Value = strike price × number of time- steps; dimension = sum	Threshold is met
Look-back	Max time slices	Value = strike price; dimension = max	Threshold is met
Barrier	_	 (i) Value = knock in / out value; dimension = all time-slices; (ii) value = strike price; dimension = final time-slice 	All thresholds met
Chooser	-	, ()	and strike price is exceeded; OR put is chosen and strike price is not

TABLE 3. Some of the common option payoff calculations that can be constructed with the enhanced P-builder. In the "Thresholds" column "sum" is the dimension containing the sum of the random variables for all the time-slices and "max" is the dimension containing the maximum of the random variables for all the time-slices. Note that the Asian option requires working in price space and " \times number of time-steps" is required as the strike price applies to the average. We can easily elaborate to also encompass baskets of the included options, compound combinations of these options and options where the payoff is either binary or the actual final price.

motion, with geometric Brownian motion prepared analogously for an input of iid lognormals and Sum replaced by Product.

- 4.5. Classes for common financial instruments. Although the main purpose of the enhanced P-builder is to provide users with a programming environment to construct arbitrary financial instruments, to aid usability we also provide classes covering simple financial instruments on single assets modelled by (geometric) Brownian motion. Noting that European and chooser options are easy to price when applied to single assets (as the former is path independent and the latter only depends on a single intermediate point in the path) we omit these, and so provide such classes for:
 - (1) barrier options;

```
1  # The enhanced P-builder operates on a circuit encoding 4 iid. Normals
2
3  operations = [Sum(4, 1), Sum(5, 2), Sum(6, 3)]
4  # The Brownian motion is therefore in dimensions 4,5,6,7
```

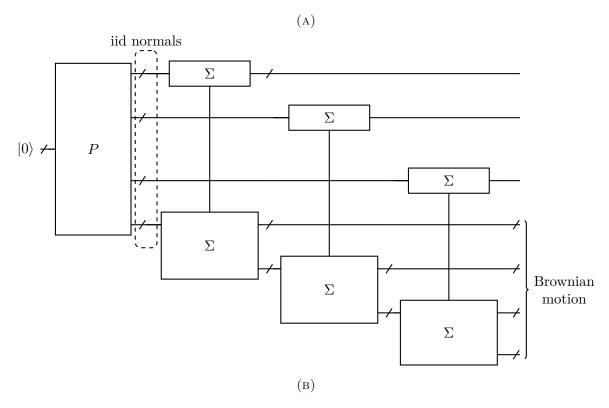


FIGURE 7. (A) Instructions to programme the enhanced P-builder to construct Brownian motion; (B) an illustration of the circuit generated. The Σ boxes correspond to the summation of the input iid univariate Gaussian random variables, where the straight lines connecting boxes represent joint entangling operations.

- (2) look-back options;
- (3) autocallables (see Section 11 for details).

For each of these, the classes: (i) build the circuit P by composing a suitable number of copies of a quantum circuit encoding an iid (log)normal distribution (which is taken as an input); (ii) use the enhanced P-builder to construct (geometric) Brownian motion; (iii) further use the enhanced P-builder to encode the expected payoff in the value of a register, controlled by a single indicator qubit (in the case of autocallables this is a little more complicated, as the whole of QMCI is run a number of times – however we postpone discussion of this until Section 11). In particular, a further layer of abstraction is introduced, such that the expected payoff is computed according to user inputs¹⁵:

- whether to work in price or return space;
- the number of time-slices;

 $^{^{15}}$ In time we will supplement these with other inputs, such the risk-free rate and the ability to discount according to risk.

- the stock volatility:
- whether the option is a call or put, and the strike price;
- knock-in / knock-out barrier conditions (when applicable);
- autocallable conditions (when applicable);
- whether the payoff is binary or the asset's value;
- \bullet the desired RMSE / allowed number of uses of the input circuit P (the quantum analogue of a limit on the number of random samples).

5. Statistically robust quantum amplitude estimation

The original form of QAE [30] requires the quantum Fourier transform to perform QPE, and whilst this does not pose a problem in terms of asymptotic performance, it is likely to be prohibitively computationally costly in the near term; this has led to the development of alternative algorithms such as amplitude estimation without phase estimation [1] (which we hereafter abbreviate as "MLQAE" – owing to its alternative name "maximum likelihood QAE") and other related proposals [2,3], all of which rely on classical post-processing to return the amplitude estimate. The headline quantum speed-up in QAE is in the RMSE convergence of the estimator as a function of the number of quantum queries / number of classical samples (q) – for QAE the RMSE scales proportional to q^{-1} compared to classical Monte Carlo methods, for which the RMSE scales proportional to $q^{-1/2}$.

Whilst RMSE captures the overall convergence, it does not fully describe the statistical properties and robustness of the estimator. In particular, the kurtosis – a measure of the 'tailedness' of the probability distribution for the estimator, and therefore for the propensity to produce outliers – is an important property for potential applications of QAE that are sensitive to non-bulk performance of the estimator, such as in finance where rare events can carry great importance [31]. In practice the excess kurtosis (relative kurtosis as compared to a unit Gaussian) is often studied, which is justified in this case because the classical MCI estimator has a Gaussian distribution. In addition, other related properties such as the skewness of the distribution for the estimator are important for gaining a general appreciation of the statistical character of the estimate returned by QAE.

We now define the quantities which together characterise the quality and robustness of any estimator. In the context of QAE, these should be of high quality for every amplitude, and to be conservative it is necessary to consider the worst case across the full range of amplitudes. This is because, when QAE is used for MCI, the quantity of interest gets mapped to a certain (fixed) amplitude which is then estimated. So it follows, that if for some calculation this happens to be to an amplitude with relatively poor statistical properties, it is still necessary that the returned estimate adheres to the promised quality. This critical point has not always been given proper attention in the literature, and on occasion the RMSE convergence has either been aggregated across all amplitudes, or indeed is only given for some arbitrary specific amplitudes.

5.1. Statistical robustness. Let $\hat{\theta}$ be the statistical estimator of a quantity θ . Its central moments μ_n are defined as

$$\mu_n(\widehat{\theta}) = \mathbb{E}\left(\left(\widehat{\theta} - \mathbb{E}(\widehat{\theta})\right)^n\right) \tag{5.1}$$

To measure the quality of this estimator, we use four metrics.

5.1.1. Bias. The bias of an estimator is given by

$$\operatorname{Bias}(\widehat{\theta}) = \mathbb{E}(\widehat{\theta} - \theta) \tag{5.2}$$

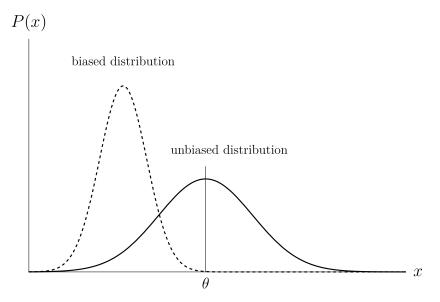


FIGURE 8. Example of biased and unbiased distributions.

and is therefore the amount that the mean of the estimator differs from the actual parameter. This is illustrated in Fig. 8. Bias gives an indication of how much better some estimator could be – in general, it should be possible to quantify and account for a bias, hence giving an unbiased estimator with better RMSE.

5.1.2. (Root) mean-squared error. As previously defined in Section 2.1, the mean-squared error of the estimator is given by

$$MSE(\widehat{\theta}) = \mathbb{E}((\widehat{\theta} - \theta)^2)$$
 (5.3)

and the RMSE is the square-root of this value. The headline quantum advantage is given as an improved scaling of RMSE with the number of samples, and this is typically all that has been characterised in existing QAE algorithms.

5.1.3. Skewness. The skewness of the estimator is

$$Skew(\widehat{\theta}) = \frac{\mu_3(\widehat{\theta})}{\mu_2(\widehat{\theta})^{3/2}}$$
 (5.4)

See Fig. 9 for an illustration of skewed distributions.

5.1.4. Kurtosis and excess kurtosis. The kurtosis of the estimator is

$$Kurt(\widehat{\theta}) = \frac{\mu_4(\widehat{\theta})}{\mu_2(\widehat{\theta})^2}$$
 (5.5)

As the kurtosis of a Gaussian is equal to 3, the excess kurtosis is defined:

$$\operatorname{Ex-Kurt}(\widehat{\theta}) = \frac{\mu_4(\widehat{\theta})}{\mu_2(\widehat{\theta})^2} - 3 \tag{5.6}$$

See Fig 10 for an illustration of distributions with different kurtosis.

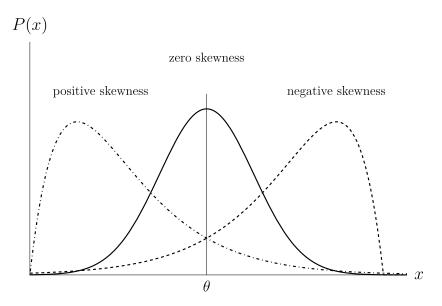


FIGURE 9. Examples of positive, negative and zero skewness.

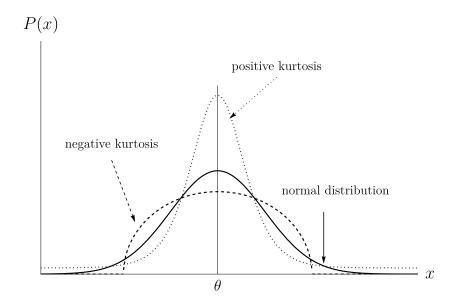


FIGURE 10. Examples of positive, negative and zero excess kurtosis.

5.2. Linear combination of unitaries QAE. We conducted initial studies suggesting that existing forms of QPE-free QAE suffer from poor statistical robustness (a point substantiated in detail in our QAE statistical robustness benchmarks in Section 5.3), which thus motivates us to propose a novel method for modifying any QPE-free QAE algorithm to improve the robustness by making use of a linear combination of unitary (LCU) operations [32]. Whilst the method itself is generic, we give an explicit form of LCU QAE based on the exponentially increasing sequence (EIS) of MLQAE [1] that not only performs competitively when considering RMSE convergence, but also appears asymptotically unbiased and exhibits near-Gaussian kurtosis and skewness. This demonstrates the robustness of the general procedure as compared to standard QPE-free QAE implementations. As far as we are aware, this represents the first time that the statistical properties

and robustness of a QAE algorithm have been analysed in this manner ¹⁶.

To understand the operation of LCU QAE, and its improved statistical robustness, it is first necessary to introduce QAE more generally. QAE makes use of quantum amplitude amplification – a generalisation of *Grover's search algorithm* [33] – to provide quantum advantage in estimation convergence. Restating Def. 3.1, this is formulated by considering a unitary operator A that prepares an initial (n+1)-qubit state

$$|\psi\rangle \equiv A |0^{n+1}\rangle = \sqrt{1-a} |\Phi_0\rangle |0\rangle + \sqrt{a} |\Phi_1\rangle |1\rangle \tag{5.7}$$

where a represents the amplitude of the state (the parameter to be estimated) and $|\Phi_0\rangle$ and $|\Phi_1\rangle$ are n-qubit states. Amplitude amplification then consists of amplifying the probability of measuring one on the final qubit by applying the following operator to $|\psi\rangle$

$$Q \equiv -AS_0 A^{\dagger} S_{\chi} \tag{5.8}$$

where S_{χ} is a unitary operator which acts as

$$S_{\chi} |\Phi_{1}\rangle |1\rangle = -|\Phi_{1}\rangle |1\rangle$$

$$S_{\chi} |\Phi_{0}\rangle |0\rangle = |\Phi_{0}\rangle |0\rangle$$
(5.9)

and S_0 is a simple unitary operator that does not depend on A.

Defining the parameter $\theta \in [0, \pi/2]$ such that $\sin^2 \theta = a$, then

$$|\psi\rangle = \cos\theta \,|\Phi_0\rangle \,|0\rangle + \sin\theta \,|\Phi_1\rangle \,|1\rangle \tag{5.10}$$

so it follows that applying the operator A can be considered as a rotation by an angle θ in the two-dimensional invariant subspace spanned by $|\Phi_1\rangle|1\rangle$ and $|\Phi_0\rangle|0\rangle$. Similarly, it can be shown that applying Q^m (that is, applying Q a total of m successive times) to $|\psi\rangle$ yields

$$Q^{m} |\psi\rangle = \cos((2m+1)\theta) |\Phi_{0}\rangle |0\rangle + \sin((2m+1)\theta) |\Phi_{1}\rangle |1\rangle$$
(5.11)

such that each application of Q further rotates the state by an additional 2θ . Thus the state can be rotated to any odd integer multiple of θ in this manner, providing the amplitude amplification.

However, the condition that the initial rotation angle is exactly θ limits the overall variation in the angles rotated to by the circuit $Q^m A$, and preliminary studies have shown that this leads to poor statistical robustness. In particular, for certain values of θ the rotation always results in a state that lies close to the principal axes of the invariant subspace, which leads to particularly poor performance. By contrast, further preliminary studies, in which we created the artificial situation where the initial starting angle could be set to a variety of values spread between 0 and $\pi/2$, regardless of θ , remedied these problems. Such an artificial situation cannot, of course, be exactly replicated in an actual quantum circuit – however our proposed method for modifying any QPE-free QAE algorithm constructs the initial state using a linear combination of unitaries, and this does allow for some variation in the starting angle. Moreover, we show that statistical robustness can be obtained without drastically reducing the performance in terms of RMSE convergence. Specifically, we use LCU to initialise a new state $|\tilde{\psi}\rangle$ with a rotation to an angle α in the same subspace

$$|\tilde{\psi}\rangle = \cos\alpha |\Phi_0\rangle |0\rangle + \sin\alpha |\Phi_1\rangle |1\rangle$$
 (5.12)

by exploiting the fact that using Eq. (5.9) it is easy to prepare

$$S_{\chi}A|0^{n+1}\rangle = \cos\theta |\Phi_0\rangle |0\rangle - \sin\theta |\Phi_1\rangle |1\rangle$$
(5.13)

 $^{^{16}}$ This is the subject of patent applications GB2301482.2 and GB2211165.2.

such that the state is instead rotated by an angle $-\theta$. That is, the states in Eq. (5.10) and Eq. (5.13) lie in the same invariant subspace, and so a linear combination of the unitary operations that prepares these can be used to prepare a state of the form given in Eq. (5.12).

Additionally, to further enhance the variety of possible angles for the initially prepared state, we make use of the operator

$$\tilde{A} \equiv (I_{2^n} \otimes X)A \tag{5.14}$$

which thus has the effect of applying a single Pauli-X rotation to the $(n+1)^{\text{th}}$ qubit, such that state is rotated by the angle $\tilde{\theta} = \pi/2 - \theta$ in a different invariant subspace spanned by $|\Phi_0\rangle |1\rangle$ and $|\Phi_1\rangle |0\rangle$. Following the above rationale, $S_{\chi}\tilde{A}$ thus rotates the state by an angle $-\tilde{\theta}$. Much of the following analysis applies to each case (rotated by θ or $\tilde{\theta}$) and so we introduce θ to refer generally to either θ or $\tilde{\theta}$; and $|\Phi_{0/1}\rangle$ and $|\Phi_{1/0}\rangle$ refer to states that are $|\Phi_0\rangle$ or $|\Phi_1\rangle$ (when $|\Phi_{0/1}\rangle$ and $|\Phi_{1/0}\rangle$ are used as a pair, the implication is that one is $|\Phi_0\rangle$ and the other $|\Phi_1\rangle$). Figure 11 demonstrates the various initial angles of states that can be prepared using these simple operators.

A LCU – corresponding to weighted linear sums of either A and $S_{\chi}A$ or \tilde{A} and $S_{\chi}\tilde{A}$ – can thus be constructed, which when applied to the state $|0^{n+1}\rangle$ results in an initial state of the (un-normalised) form

$$|\tilde{\psi}\rangle = \cos(\boldsymbol{\theta}) |\Phi_{0/1}\rangle |0\rangle + F\sin(\boldsymbol{\theta}) |\Phi_{1/0}\rangle |1\rangle$$
 (5.15)

where $-1 \le F \le 1$ and the definition of $|\tilde{\psi}\rangle$, from Eq. (5.12), has now been extended to correspond to states prepared in either the original and secondary subspaces. It thus follows that α in Eq. (5.12) is given by:

$$\alpha = \tan^{-1} \left(F \tan(\boldsymbol{\theta}) \right) \tag{5.16}$$

Figure 12 demonstrates the form of the quantum circuit used to perform LCU state preparation 17 , where U_a and U_b are either the pair $\{A, S_{\chi}A\}$ or the pair $\{\tilde{A}, S_{\chi}\tilde{A}\}$. Such a circuit results in the state

$$|0\rangle |\omega\rangle \rightarrow \left[\left(\cos^2(\beta/2)U_a + \sin^2(\beta/2)U_b\right)\right] |0\rangle |\omega\rangle + \left[\frac{1}{2}\sin(\beta)(U_b - U_a)\right] |1\rangle |\omega\rangle$$
 (5.17)

which is thus such that, if the measurement outcome (for the first qubit) is zero, then the state collapses to $\left(\cos^2(\beta/2)U_a + \sin^2(\beta/2)U_b\right)|\omega\rangle$.

Our proposed method for modifying any QPE-free QAE algorithm thus utilises different LCU state-preparation circuits to prepare initial states with a variety of different starting angles. An important caveat of the LCU state preparation is that there is a probability that the LCU state preparation will fail:

$$p_{\text{fail}} = \frac{1}{4} \sin^2(\beta) ||U_b - U_a||^2 = \sin^2(\beta)$$
 (5.18)

which thus tends to one as β tends to $\pi/2$ – i.e., when the LCU is designed to prepare an equal mixture of the two unitaries such that the prepared state is $|\Phi_{0/1}\rangle|0\rangle$. In order to avoid certain and highly probably failures, $p_{\rm fail}$ can be upper-bounded by some $p_{\rm max-fail}$, which is set by constraining the range of the rotation parameter β as $0 \le \beta \le \sin^{-1}(\sqrt{p_{\rm max-fail}})$. This corresponds to the weighting of the two terms U_a and U_b in Eq. (5.17), defining a bound (either upper or lower) on the factor F in Eq. (5.15), $F_{\rm bound}$, and thus indirectly the rotated angle of the initial state, which is then bounded (equivalently either upper or lower) by $\theta_{\rm bound} = \tan^{-1}\left(\sqrt{1-p_{\rm max-fail}}\tan(\theta)\right)$.

 $^{^{17}}$ Note that the procedure requires an additional ancilla qubit, as is standard for LCU state preparation.

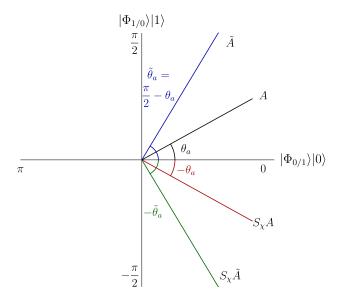


FIGURE 11. Initial rotation angles in the two-dimensional invariant subspace spanned by either $\{|\Phi_1\rangle|1\rangle$, $|\Phi_0\rangle|0\rangle\}$ or $\{|\Phi_0\rangle|1\rangle$, $|\Phi_1\rangle|0\rangle\}$, corresponding to applying the unitary operators A and $S_{\chi}A$ or \tilde{A} and $S_{\chi}\tilde{A}$, respectively.

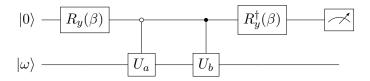


FIGURE 12. Quantum circuit for (non-deterministically) preparing a LCU state. The desired state is prepared when the measurement outcome of the ancilla qubit is post-selected to be zero (equivalently, in practice the procedure can just be repeated until successful).

To use the variety of starting angles that LCU provides to enhance the statistical robustness of QAE, we first note that there are four possible categories of LCU initial states that can be prepared (referred to as LCU_{1-4}), each corresponding to rotating the state to within a different angular range bounded by some θ_{bound} :

LCU₁(
$$\beta$$
): $\cos^{2}(\beta/2)A + \sin^{2}(\beta/2)S_{\chi}A$, $\theta_{\text{bound}} \leq \alpha \leq \theta$
LCU₂(β): $\cos^{2}(\beta/2)S_{\chi}A + \sin^{2}(\beta/2)A$, $-\theta \leq \alpha \leq -\theta_{\text{bound}}$
LCU₃(β): $\cos^{2}(\beta/2)\tilde{A} + \sin^{2}(\beta/2)S_{\chi}\tilde{A}$, $\tilde{\theta}_{\text{bound}} \leq \alpha \leq \tilde{\theta}$
LCU₄(β): $\cos^{2}(\beta/2)S_{\chi}\tilde{A} + \sin^{2}(\beta/2)\tilde{A}$, $-\tilde{\theta} \leq \alpha \leq -\tilde{\theta}_{\text{bound}}$ (5.19)

where $\beta = \cos^{-1}(F)$.

The circuits preparing these possible starting angles are then used in place of $A|0^{n+1}\rangle$ in QAE circuits, which thereafter proceed as usual. Figure 13 demonstrates the combined quantum circuit – consisting of LCU state-preparation circuit and the amplitude amplification circuit, Q^m , – required to realise this procedure. As the operations Q^m only occur when the LCU post-selection

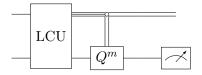


FIGURE 13. Quantum circuit for performing QAE using LCU state preparation. The LCU block stands for the circuit given in Figure 12, and the top wire is the post-measurement ancilla bit of this circuit.

Algorithm 1 Generic algorithm for any QPE-free QAE.

Require: Quantum circuit A; probability distribution $p(\theta)$ initialised as the prior; Stopping Criterion

- 1: Set m, n_{shots} , (optional) additional parameters
- 2: Prepare and measure n_{shots} of $Q^m A |0\rangle$
- 3: Update $p(\theta)$ using measurement outcomes based on standard probabilistic procedures
- 4: if Stopping Criterion then
- 5: Return \hat{a} (or $\hat{\theta}$)
- 6: **else** Update m, n_{shots} ; Goto Line 2
- 7: end if

is successful, LCU QAE can be seen to "fail fast" – in the sense that the majority of the circuit need only be executed when the initial state preparation has been successful. This is the intended implication of the classical control on Q^m , although the principle of deferred measurement can be invoked to give an equivalent circuit where these measurements are performed in the final layer, if desired.

Our proposed method is a general method for modifying any QPE-free QAE algorithm by introducing LCU state preparation to improve the statistical properties and robustness of the estimator for the amplitude (or equivalently for θ). Any QPE-free algorithm can be formulated as an instance of the generic framework given in Algorithm 1. (Note that Algorithm 1 requires a classical probability distribution, $p(\theta)$ to be updated: we found that a sufficiently dense grid of point masses sufficed for this purpose, but more advanced techniques such as particle filtering [34] may alternatively be used.) Going into further detail, Stopping Criterion could be, for example, the desired accuracy of the estimate, total wall-clock time, total number of uses of A etc; the estimator \hat{a} could be any estimator, for example, ML, MMSE etc (see Section 2.1 for definitions of these). Optionally, following each shot $p(\theta)$ could be updated and the Stopping Criterion could also be assessed.

This general framework can be enhanced to include LCU state preparation, as shown in Algorithm 2. In the general description, no preferred designation into the four regions (corresponding to LCU₁₋₄) is given, however there is good reason to divide these equally (or approximately equally if n_{shots} is not a multiple of 4). This is because the way that the restriction on starting angle dictated by the choice of $p_{\text{max-fail}}$ works is such that, if LCU₁ and LCU₂ only provide a small range of different angles, then LCU₃ and LCU₄ will provide a large range. This is sketched in Fig. 14 and in this way, even without knowing θ (that is, after all, what we are trying to discover) a large variety of initial angles is guaranteed – and this is the key ingredient for good statistical robustness. It is worth noting that, in Algorithm 2, the n_{shots} where m = 0 remain unchanged, as LCU state preparation is most useful when the state is subsequently rotated by amplitude amplification steps (i.e., Q^m for m > 0).

Algorithm 2 Generic algorithm for LCU QPE-free QAE.

Require: Quantum circuit A; probability distribution $p(\theta)$ initialised as the prior; Stopping Criterion; $p_{\text{max-fail}}$

- 1: Initialise m = 0, n_{shots}
- 2: Prepare and measure n_{shots} of $Q^m A |0\rangle$
- 3: Set m, n_{shots} , (optional) additional parameters
- 4: for it=1: n_{shots} do
- 5: Select $y \in \{1, 2, 3, 4\}$
- 6: Select β (choice restricted by $p_{\text{max-fail}}$)
- 7: Prepare and measure $Q^m[LCU_y(\beta)]|0\rangle$ (waiting for a successful LCU preparation)
- 8: Update $p(\theta)$ using measurement outcomes based on standard probabilistic procedures
- 9: end for
- 10: if Stopping Criterion then
- 11: Return \hat{a} (or $\hat{\theta}$)
- 12: **else** Goto Line 3
- 13: end if

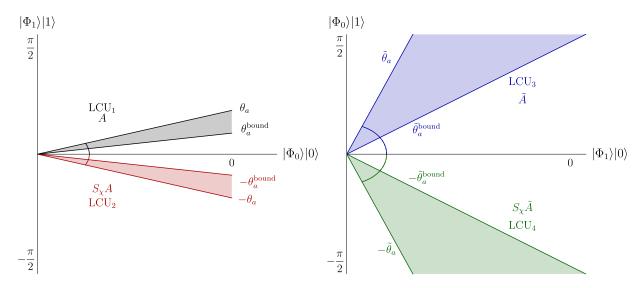


FIGURE 14. Range of rotation angles in (left) the two-dimensional invariant subspace spanned by $\{|\Phi_1\rangle|1\rangle, |\Phi_0\rangle|0\rangle\}$ for LCUs of the operators A and $S_\chi A$, corresponding to LCU categories (black) LCU₁ and (red) LCU₂, and (right) the two-dimensional invariant subspace spanned by $\{|\Phi_0\rangle|1\rangle, |\Phi_1\rangle|0\rangle\}$ for LCUs of the operators \tilde{A} and $S_\chi \tilde{A}$, corresponding to LCU categories (blue) LCU₃ and (green) LCU₄, prepared given the specific algorithm described in this section. For illustrative purposes it is assumed that F^{bound} and the true value of θ_a are such that $\theta_a^{\text{bound}} = 0.5\theta_a$. The bold lines correspond to the boundaries of the sectors of the corresponding LCU categories. Note that if one subspace has a tight range of angles, the other will have a large range.

5.3. **QAE robustness benchmarks.** In order to benchmark the performance of LCU QAE, we define an explicit algorithm, based on the EIS $(m \in \{0, 1, 2, 4, 8, 16, ...\})$ of Ref. [1], with the following settings:

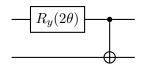


FIGURE 15. The circuit used to prepare various amplitudes used to benchmark QAE. In particular, the parameterised angle is used to prepare the various amplitudes.

- For m = 0, $n_{\text{shots}} = 66$; for all other m, $n_{\text{shots}} = 44$. The 44 shots are divided equally between the four LCU segments (11 into each), and a range of initial starting angles are chosen within each LCU segment.
- q, as selected by the user, is taken as the number of uses in circuits with *successful* LCU post-selections. The overhead of failed LCU post-selections is then accounted for when reporting the overall converge as a function of total uses (failed and successful).
- The returned estimate is a MMSE estimate.

For each given value of q, a circuit P whose form is given in Fig. 15 is used (by varying θ) to prepare states with 49 equally-spaced amplitude values in the range [0.02, 0.98]. QAE is then run using a state-vector simulator, and the corresponding amplitude estimate stored. This process is repeated 10000 times for each configuration (a configuration being a particular q and amplitude value). The (absolute) bias, MSE/RMSE, (absolute) skewness, and excess kurtosis are then calculated for each configuration. Bootstrapping, implemented via the bias-corrected and accelerated bootstrap interval [35], is used to determine a 68% confidence interval on each metric, from which (asymmetric) uncertainties on the data points are determined. These values are then also averaged across all amplitude values, where the uncertainties are propagated to the average using the methods discussed in Ref. [36]. For comparison, we also performed this process for two other prominent forms of QPE-free QAE, namely Iterative QAE (IQAE) [2] and MLQAE itself.

Figure 16 shows the statistical benchmarks quantities for the various versions of QAE averaged over the 49 amplitudes. Even from this relatively coarse display of the results, it can be seen that LCU QAE has significantly better statistical robustness than the other two methods (which, in fairness, were not designed with this in mind). Crucially, this advantage is maintained when the results are viewed for each amplitude, as shown in Figs. 34-37 – notably the statistical robustness is still good even in the worst case (that we must conservatively assume) confirming the superiority of LCU QAE in this regard. To give a little more detail, excess kurtosis is the single most important figure of merit when assessing whether extreme events occur with significantly greater (or smaller) probability than for a Gaussian random variable with the same mean and variance, and although there is no consensus about exactly what value constitutes sufficient approximate Gaussianity – although excess kurtosis between -2 and 2 is somewhat commonly taken as indicative of Gaussianity (see e.g., Ref. [37]). In our results we take the relatively aggressive stance of treating 0.3 as an acceptable value of excess kurtosis (i.e., 10% of the kurtosis of a Gaussian) – and we can see from Fig. 37 where the value of 0.3 is illustrated by the horizontal dashed line that, apart from the odd outlier, this is met by virtually all of the LCU QAE estimators, but never the IQAE and MLQAE estimators, across the various amplitudes and numbers of uses analysed. Indeed, IQAE and ML QAE would often not even meet the relaxed value of 2 as acceptable excess kurtosis. (Note we plot the excess kurtosis on a logarithmic scale – which is made possible by the fact that the empirically-found values were always positive.)

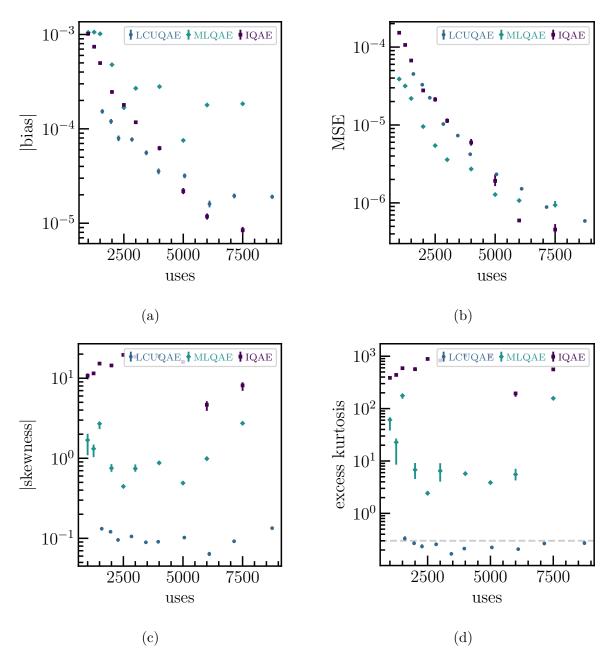


FIGURE 16. The statistical performance of the various forms of QAE averaged over the 49 amplitudes.

Part III. Full technical specification

6. Overview of the QMCI engine architecture

In this Section we present an overview of the modular architecture of Quantinuum's Quantum Monte Carlo Integration Engine. Fig. 17 gives a high-level illustration, where the following six modules are included:

- (1) Distribution circuit: the QMCI engine takes as an input the quantum circuit encoding the probability distribution of interest for the MCI. The precise architecture of this module is further broken down in Fig. 18.
- (2) Quantity to estimate: the QMCI engine takes as an input the function to apply to the random variables sampled in the MCI, i.e., the statistical quantity to estimate.
- (3) Quantum amplitude estimation: the user can select which type of QAE algorithm is used.
- (4) Backend call: this compiles the required quantum circuits for the chosen backend before executing it.
- (5) Cloud call: this executes the quantum circuits required by QMCI on a simulator, emulator or quantum hardware.
- (6) QMCI estimate: this returns the desired numerical MCI quantity and / or the resources needed to run the estimate.

Throughout the engine, careful attention is taken to ensure that the computational terms are adequately described. This is particularly well exemplified in the case of the additional data attached to the quantum circuit objects in order for them to be interpreted as preparing probability distributions (detailed in Section 2.2). Fig. 18 gives full details of how such distribution circuits are built:

- (1) Distribution: this is a library of probability distributions of interest for the simulation coming e.g. from parametrised distributions (such as a Gaussian distribution) or stochastic models. This will eventually also contain multivariate distributions.
- (2) State preparation: this is a library of quantum algorithms which can prepare the distribution chosen in Distribution. This library includes methods based on quantum walks and solving partial differential equations (PDEs) as well as pre-trained parameterised quantum circuits (PQCs).
- (3) Enhanced *P*-builder: as detailed in Section 4, this module interacts with the distribution circuit to enhance it with circuitry related to basic operations on distributions such as sums, products, maxima / minimima and thresholds.
- (4) Distribution circuit: this serves as an input of the QMCI engine and includes the quantum circuit to generate the distribution, the number of wires per dimension of the distribution and also the support and spacing of each dimension; as well as information about any indicator qubits introduced by the enhanced P-builder.

The modular architecture has been designed to be extensible; for instance it is easy to add new forms of QAE without disturbing the essential operation of the QMCI engine. Fig. 33 gives pseudocode for how the user can programme all of the options associated with these modules.

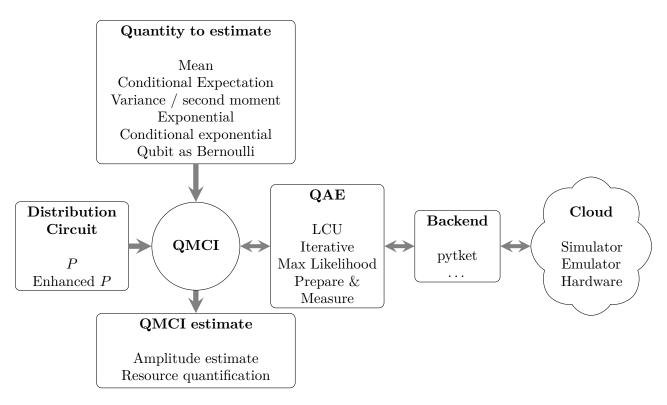


FIGURE 17. Schematic showing the architecture of Quantinuum's Quantum Monte Carlo Integration Engine. The Distribution Circuit module is described explicitly in Fig. 18.

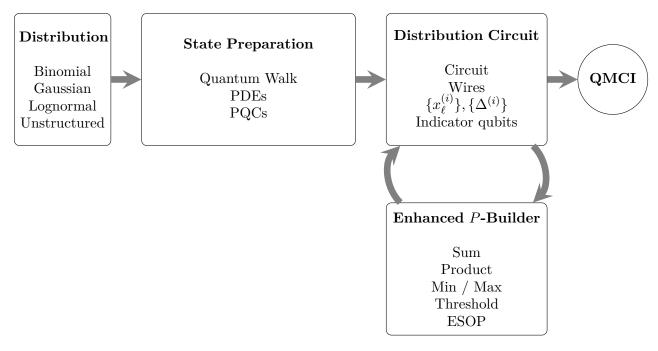


FIGURE 18. Flow of the Distribution Circuit module.

7. Distribution loader

This Section describes the distribution loader¹⁸ module of the Quantum Monte Carlo Integration engine. The design has versatility and extensibility at its heart: it encodes both univariate and multivariate probability distributions, and it is such that it will be easy to integrate future distribution loading methods. In general we may be interested in problems that are continuous or discrete in nature, and in the former case we require a suitable discretisation (as one does for classical computation). Indeed for a random continuous process described by a random variable X with a PDF $f_X(x)$, to encode the probability distribution on a register, we first introduce its discretisation over a uniform grid of 2^n points $\{x_1, \ldots, x_{2^n}\}$ where points are at a distance Δ from each other, labelled by the binary encoding

$$\{x_1 \equiv \underbrace{00\dots 00}_n, \ x_2 \equiv 00\dots 01, \ x_{2^n} \equiv 11\dots 11\}$$
 (7.1)

The probability distribution is then evaluated on a discrete set Ω as in Proposition 3.2, where the points are at distance Δ such that

$$p_i \approx f_X(x_i)\Delta, \quad \forall i = 1, \dots, 2^n$$
 (7.2)

and following Eq. (2.26), we prepare the following quantum state

$$|p\rangle = P|0^n\rangle = \sum_{x_i \in \{0,1\}^n} e^{2\pi \mathbf{i}\varphi_i} \sqrt{p_i} |x_i\rangle$$
(7.3)

Notably the relative phases, $\{\varphi_i\}$, are arbitrary as we are only concerned with probabilities of computational basis state measurement outcomes. We envisage that the additional degrees of freedom that this property offers may help us to efficiently prepare a state encoding the required distribution in some cases.

To recap Section 6, the following information must be provided by the distribution loader in order for the downstream modules of the QMCI engine to correctly interpret the prepared state as an encoding of a probability distribution:

- (1) the circuit P preparing the quantum state;
- (2) the list of the qubits encoding particular dimensions of the multivariate probability distribution (treating univariate distributions as a special case);
- (3) the left endpoint of each dimension of the support of the distribution, denoted $x_{\ell}^{(i)}$ for the i^{th} dimensions;
- (4) the spacing between the points, denoted $\Delta^{(i)}$ for the i^{th} dimension;
- (5) the presence or one or more indicator qubits (as optionally included by using the enhanced *P*-builder).

This information forms the attributes of the module Distribution Circuit, as presented in Fig. 18.

In general there are three types of distribution loader:

- (i) classical methods to prepare some specific distribution;
- (ii) quantum approaches that aim to use actual quantum-mechanical phenomena to prepare some specific distribution;
- (iii) unstructured methods which consider 2^n positive real numbers that sum to one and prepare the corresponding PMF.

¹⁸For our purposes, "data loading" and "state preparation" may be treated as synonyms of "distribution loading".

Regarding "classical methods", Ref. [27] shows that every classical sampling algorithm can be compiled as a Toffoli circuit applied to an initial uniformly random bitstring, and so every classical sampling algorithm can be turned into an appropriate circuit P with about the same number of gates. This result was proposed principally to demonstrate that the quadratic reduction in sample complexity (of QMCI over classical MCI) can always be made to manifest as at least a quadratic reduction in computational complexity. However, using the q-marginal construction of Ref. [27] may also prove to be practically beneficial in certain circumstances. In particular, we envisage this approach potentially bearing fruit when preparing complicated random processes, that do not readily map into quantum circuits in any other manner. In the long-run we will supplement the enhanced P-builder to include all of the usual arithmetic operations needed to encode classical sampling algorithms into quantum circuits. For now, though, this is low priority and we have instead initially focused on included low-depth distribution-loading methods and the enhanced P-builder just has the functionality we need to construct simple (but still interesting) random processes (i.e., for Brownian and geometric Brownian motion) – as well as the functionality to construct financial instruments.

"Quantum approaches" is perhaps the most exciting of these, and certainly it is the one which has the most potential to complement the quantum advantage in QAE with a further advantage in the distribution loading. On this, we propose two approaches: firstly, using quantum walk to prepare a binomial distribution, detailed in Section 7.1; and secondly PDE-based methods to prepare Gaussian distributions, detailed in Section 7.2.

Whilst both of the above concern the situation where some analytically-defined probability distribution or random process is encoded in a quantum circuit, "unstructured methods" assumes no such structure (as its name implies). For this reason, it is the most general – any analyticallydefined distribution or process can be approximated as a suitable set of probability mass values – however, it is also the least scalable: an n-qubit circuit requires 2ⁿ classical values to be specified. For this reason, this method is only viable when relatively small resource states are required as inputs – and are then turned into more complicated (multidimensional) random processes using the enhanced P-builder. This is exactly the setting that we are concerned with when constructing Brownian motion / geometric Brownian motion from iid Gaussian / lognormal probability distributions. For this reason, we have dedicated significant effort to unstructured data-loaders including implementing a simplification of the Grover-Rudolph method [25] and developing a method based on Ref. [38]. In each of these cases, we found that the circuits were far too deep to be viable for near- and medium-term use. We do, however, find that training PQCs as machine learning (generative) models to sample the desired probability distributions suffices as a means of unstructured data loading for the example benchmark problems we address in Section 11, and this is the focus of Section 7.3.

7.1. Quantum walk. The first method we present uses continuous quantum walks to prepare binomial distributions. The core of this method can be implemented using standard Hamiltonian simulation techniques [39–41].

The binomial distribution is defined

$$p_i = \binom{n}{i} p^i (1-p)^{n-i} \tag{7.4}$$

This method relies on the construction of a continuous quantum walk on a graph \mathcal{G} whose transition matrix takes an initial state to a desired quantum state at a given time. In general if \mathcal{G}



FIGURE 19. Simplest complete graph with two vertices K_2

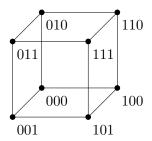


FIGURE 20. Representation of the hypercube Q_3 which is the 3-fold Cartesian product of K_2 .

is a graph with adjacency matrix A, then the continuous-time quantum walk on \mathcal{G} is the quantum walk with transition matrix [42,43]

$$U_{\mathcal{G}}(t) := e^{\mathbf{i}tA} \tag{7.5}$$

To prepare the state of interest (7.3) we therefore require

$$P \equiv U_{\mathcal{G}}(t) \tag{7.6}$$

We begin by preparing the simplest probability distribution from a quantum walk on a complete graph with two vertices, which we name K_2 and represent in Fig. 19. The adjacency matrix for this graph is the Pauli X matrix (7.7) and the transition matrix for the walk at time t, the matrix $U_{K_2}(t)$, can be written as

$$A = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad U_{K_2}(t) = \begin{pmatrix} \cos(t) & \mathbf{i}\sin(t) \\ \mathbf{i}\sin(t) & \cos(t) \end{pmatrix}$$
(7.7)

Running the quantum walk until a time $\tau = \arccos \sqrt{p}$ induces the following transition matrix

$$U_{K_2}(\tau) = \begin{pmatrix} \sqrt{p} & \mathbf{i}\sqrt{1-p} \\ \mathbf{i}\sqrt{1-p} & \sqrt{p} \end{pmatrix}$$
 (7.8)

This matrix acts on a 1-qubit state as a simple $R_x(.)$ rotation, i.e.

$$U_{K_2}(\tau)|0\rangle = \sqrt{p}|0\rangle + \mathbf{i}\sqrt{1-p}|1\rangle \tag{7.9}$$

producing the Bernoulli distribution, which is the n = 1 case of the Binomial distribution displayed in Eq. (7.4).¹⁹ We now use this simple result to build a graph which prepares the Binomial distribution.

Definition 7.1. If \mathcal{G}_1 and \mathcal{G}_2 are graphs with adjacency matrices $A(\mathcal{G}_1)$ and $A(\mathcal{G}_2)$, respectively, then the Cartesian product of \mathcal{G}_1 and \mathcal{G}_2 is the graph $\mathcal{G}_1 \boxtimes \mathcal{G}_2$ defined by the adjacency matrix $A(\mathcal{G}_1 \boxtimes \mathcal{G}_2) := A(\mathcal{G}_1) \otimes I + I \otimes A(\mathcal{G}_2)$. The following relationship between the transition matrices of \mathcal{G}_1 , \mathcal{G}_2 and $\mathcal{G}_1 \boxtimes \mathcal{G}_2$ is given by (see [44, Eqs. (34-35)] or [45, Lemma 4.2])

$$U_{\mathcal{G}_1 \boxtimes \mathcal{G}_2}(t) = U_{\mathcal{G}_1}(t) \otimes U_{\mathcal{G}_2}(t) \tag{7.10}$$

¹⁹The Bernoulli distribution can, of course, be prepared by a single R_y gate, and so this is purely a motivational result for what will follow.

The *n*-fold Cartesian product $K_2 \boxtimes K_2 \boxtimes \cdots \boxtimes K_2 := K_2^{\boxtimes n}$ is equal to the *n*-dimensional hypercube Q_n (see Fig. 20 for an example with n=3), which has vertex set $\{0,1\}^n$, and two vertices are adjacent if and only if their Hamming distance is 1. (The Hamming distance between two bit strings of equal length is the number of different elements at the same position.) From Eqs. (7.10) and (7.9), we deduce the following result, which is a direct consequence of the tensor product structure. Let u be a vertex in Q_n of weight n-k, then we have the inner product

$$\left| \langle u | U_{Q_n}(\tau) | 0^n \rangle \right|^2 = p^k (1 - p)^{n - k} \tag{7.11}$$

This construction is a product of Bernoulli states but not a Binomial state at this stage – further work is needed to obtain the combinatorial factor $\binom{n}{k}$.

Definition 7.2. For a given graph \mathcal{G} , a partition of \mathcal{G} is a partition of its vertex set. We say a partition $\pi = \{C_0, \ldots, C_n\}$ of \mathcal{G} is equitable if for any two classes C_i , $C_j \in \pi$, the number of neighbours of $v \in C_i$ in C_j is a constant b_{ij} which depends only on i and j.

For the *n*-dimensional hypercube Q_n , the partition $\pi = \{C_0, \ldots, C_n\}$ where C_k is the set of vectors in $\{0,1\}^n$ of weight k, is equitable [46]. The integers b_{ij} are computed as

$$b_{ij} = \begin{cases} n-i & j=i+1\\ i & j=i-1\\ 0 & \text{otherwise} \end{cases}$$
 (7.12)

Definition 7.3. Let \hat{S} be the matrix with

- rows labelled by the vertices of \mathcal{G} ,
- columns labelled by the classes of π ,
- and whose (v, C_j) entry is $1/\sqrt{|C_j|}$ if $v \in C_j$ and 0 otherwise.

We call the graph with adjacency matrix $\hat{A} := \hat{S}^{\mathsf{T}} A \hat{S}$ the symmetrized quotient of \mathcal{G} . It has rows and columns labelled by the classes of π . The (C_i, C_j) -entry of $\hat{S}^{\mathsf{T}} A \hat{S}$ is $\sqrt{b_{ij} b_{ji}}$.

Let $\hat{U}(t) := \exp(\mathbf{i}t\hat{A})$ be the transition matrix of the symmetric quotient of \mathcal{G} for some equitable partition π . Following Ref. [47, Theorem 1], if π contains a class C_0 with only one vertex, say $C_0 = \{a\}$, then for any class $C_i \in \pi$ and any vertex $b \in C_i$, we have

$$\langle C_i | \hat{U}(t) | C_0 \rangle = \sqrt{|C_i|} \langle b | U(t) | a \rangle \tag{7.13}$$

The overall construction can be seen as a generalisation of Ref. [44]. We denote the symmetrised quotient of Q_n by this partition as \hat{Q}_n , whose adjacency matrix of \hat{A} has entries

$$\hat{A}_{i,j} = \begin{cases} \sqrt{(n-i)(i+1)} & j = i+1\\ \sqrt{(i-1)(n-i+1)} & j = i-1\\ 0 & \text{otherwise} \end{cases}$$
 (7.14)

for $0 \le i, j \le n$. Note that \hat{A} is the adjacency matrix of a weighted path. Since the class C_k of π has size $|C_k| = \binom{n}{k}$ and considering $C_0 = \{|0^n\rangle\}$, we have our final result presented in Theorem 7.4.

Theorem 7.4. Given $\hat{U}(t)$, the transition matrix of \hat{Q}_n , we have at time $\tau = \arccos \sqrt{p}$

$$\left| \langle C_k | \hat{U}(\tau) | C_0 \rangle \right|^2 = \left| \langle C_k | \hat{U}(\tau) | 0 \rangle \right|^2$$

$$= \binom{n}{k} p^k (1-p)^{n-k}$$
(7.15)

for any $0 \le k \le n$. That is, the graph \hat{Q}_n prepares the Binomial distribution $\mathcal{B}(n,p)$.

The implementation of such procedure is then done by encoding a Hamiltonian representing the adjacency matrix

$$\mathcal{H}_{\mathcal{B}} := \hat{A} \tag{7.16}$$

and running Hamiltonian evolution. Motivated by the general result of Theorem 7.4, we derive explicitly in Example 7.5 the construction for n = 3: this is special case of the above theorem that is small enough to follow explicitly, but large enough to be non-trivial.

Example 7.5 (Example with Q_3 , n=3). Let us study in details the case of the hypercube Q_3 depicted in Fig. 20. The vertex set reads

$$\{(000), (001), (010), (100), (110), (101), (011), (111)\}\$$
 (7.17)

We denote this vertex set in parenthesis because these are intermediate objects not representing our final qubits. We then partition the vertex set by Hamming weight in four classes as

$$C_0 = \{(000)\}, C_1 = \{(001), (010), (100)\}, C_2 = \{(110), (101), (011)\}, C_3 = \{(111)\}$$
 (7.18)

Each class of this partition will thereafter represent a different quantum state.

Note that the size of each subset is given by $|C_i| = \binom{3}{i}$. To confirm that this gives an equitable partition, we check that the number of neighbours of a vertex $v \in C_i$ in the class C_j is a constant b_{ij} which only depends on $\{i, j\}$, and not on the vertex we choose from C_i .

For example, let's look at C_1, C_2 : for any vertex v we choose from C_1 , it will have 2 neighbours in C_2 . For example, (001) has neighbours (101) and (011) in C_2 ; (100) has neighbours (110) and (101) in C_2 . We can check this for any two classes C_i and C_j and see that the same property holds. From considering C_1, C_2 , we can see that $b_{1,2} = 2$. In general, we get the constants:

$$b_{0,1} = 3, \ b_{1,2} = 2, \ b_{2,3} = 1, \ b_{3,2} = 3, \ b_{2,1} = 2, \ b_{1,0} = 1$$
 (7.19)

and 0 otherwise. The matrix \hat{A} whose entries are computed by $\sqrt{b_{ij}b_{ji}}$ therefore reads

$$\hat{A} = \begin{pmatrix} 0 & \sqrt{3} & 0 & 0\\ \sqrt{3} & 0 & 2 & 0\\ 0 & 2 & 0 & \sqrt{3}\\ 0 & 0 & \sqrt{3} & 0 \end{pmatrix}$$
 (7.20)

We choose to label the rows and columns of \hat{A} by new basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$, which corresponds to the classes C_0 , C_1 , C_2 , C_3 , respectively. From Eq. (7.13), the coefficients of the transition matrix $\hat{U}(t) := \exp(it\hat{A})$ are given by

$$\langle i|\hat{U}(t)|00\rangle = \sqrt{|C_i|}\langle b_i|U(t)|(000)\rangle \tag{7.21}$$

for any element b_i in the class C_i , and where we used the mapping

$$\{C_0, C_1, C_2, C_3\} \mapsto \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$$
 (7.22)

We can write the basis states on the left side in binary representation to make it clearer how to interpret this on the quantum computer:

$$\langle 00|\hat{U}(t)|00\rangle = \sqrt{\binom{3}{0}} \langle (000)|U(t)|(000)\rangle, \quad \langle 01|\hat{U}(t)|00\rangle = \sqrt{\binom{3}{1}} \langle (001)|U(t)|(000)\rangle,$$

$$\langle 10|\hat{U}(t)|00\rangle = \sqrt{\binom{3}{2}} \langle (011)|U(t)|(000)\rangle, \quad \langle 11|\hat{U}(t)|00\rangle = \sqrt{\binom{3}{3}} \langle (111)|U(t)|(000)\rangle$$

$$(7.23)$$

We already showed in Eq. (7.11) that $|\langle j|U(t)|(000)\rangle|^2 = p^k(1-p)^{n-k}$ when j is a vertex that has weight k, allowing us to obtain the the binomial distribution.

We have studied here a particular graph – the Cartesian product of K_2 – to generate the Binomial distribution. More generally, we expect continuous-time quantum walk to be an important way of preparing quantum states. In particular, the fact that this method uses the intrinsically quantum-mechanical nature of quantum walk to prepare a classically-defined probability distribution suggests that it may be an effective means of preparing large quantum states, where classical (Toffoli circuit) approaches prove to be expensive. In the immediate future, however, the reliance on Hamiltonian simulation puts it out of reach for all but the most trivial probability distributions.

7.2. Partial differential equations. The second method we present is based on the solution of PDEs. Indeed, one way to generate a probability distribution is to solve a PDE called the Fokker-Planck equation for stochastic systems [48]. As a simple example let us consider the following heat equation in 1+1 dimensions for a field f(x,t) depending on one spatial coordinate $x \in \mathbb{R}$ and one time coordinate t > 0:

$$\partial_t f = \partial_x^2 f. \tag{7.24}$$

If one starts from the initial condition $f(x, t = 0) = \delta(x)$ where δ is the Dirac delta function, then the standard normalised solution is given by the heat kernel

$$f(x,t) = \frac{1}{\sqrt{4\pi t}} e^{-\frac{x^2}{4t}}, \quad \int_{\mathbb{R}} f(x,t) \, \mathrm{d}x = 1, \quad \forall t > 0$$
 (7.25)

It is then possible to obtain a Gaussian with an arbitrary variance depending on how long we let the heat equation evolve. We can change the mean of the Gaussian by inducing a translation in the initial condition.

More generally, the Fokker-Planck equation can generate arbitrary probability distributions that arise from stochastic dynamics such as the following one-dimensional stochastic differential equation

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t \tag{7.26}$$

Here μ is the drift, σ defines the so-called diffusion coefficient $D = \sigma^2/2$ and W_t is a standard Wiener process [49]. The PDF, $f_X(x,t)$, of observing the process X_t at position x and time t then obeys the PDE

$$\partial_t f = -\partial_x [\mu(x,t)f] + \partial_x^2 [D(x,t)f]. \tag{7.27}$$

Note that equations (7.26) and (7.27) can be extended to higher-order spatial dimensions allowing a potential encoding of multivariate distributions.

There exist quantum algorithms which can solve linear and nonlinear PDEs and encode their solution in the amplitudes of a wave function [50–65]. Common approaches to solving linear differential equations, such as the Fokker-Planck equation, are Hamiltonian simulation [50,66] and the quantum linear systems algorithm (widely known as "HHL" – the initials of the authors) [67]. In the context of Hamiltonian simulation it is convenient to rewrite the differential equation considered as a Schrödinger equation and then use quantum algorithms for simulating that Schrödinger equation. Based on that idea a versatile framework, so-called *Schrödingerisation*, for differential equations was proposed recently [64,65]. HHL is a quantum algorithm for solving linear equations and can be applied to time-dependent differential equations by discretising space and time and expressing the entire evolution as a single system of linear equations [54]. For the Fokker-Planck equation it was argued that HHL has advantages over Hamiltonian simulation via Schrödingerisation [68]. Both HHL and Hamiltonian simulation applied to the Fokker-Planck equation are, in

principle, efficient in the number of variables and, therefore, should not suffer from the curse of dimensionality encountered by classical solvers. However, these quantum algorithms lead to deep circuits in general and can require a large number of ancilla qubits.

One possibility to reduce the quantum hardware requirements of solving PDEs is to make use of variational quantum algorithms such as [58,69] or to classically optimize quantum circuits as in [70,71]. Additional variational approaches to state preparation are presented in the following section.

7.3. Parameterised quantum circuits. The third state preparation method we present here uses PQCs [72]. We explore this method to prepare Gaussian and lognormal distributions. In general PQCs are trained to minimise the L_{α} norm for some α . Specifically, for a target state $|\psi_{\text{target}}\rangle$ and an approximate version thereof prepared by a trained PQC, $|\psi_{\text{PQC}}\rangle$, the L_{α} norm is defined:

$$|| |\psi_{\text{target}}\rangle - |\psi_{\text{PQC}}\rangle ||_{\alpha} = \left(\sum_{i=1}^{N} \left(\left[|\psi_{\text{target}}\rangle - |\psi_{\text{PQC}}\rangle \right]_{i} \right)^{n} \right)^{1/\alpha}$$
(7.28)

where $|\psi_{\text{target}}\rangle$ and $|\psi_{\text{PQC}}\rangle$ are N-element vectors. L_1 , L_2 and L_{∞} norms are most commonly used as the distances to be minimised in PQC training, and in the last case this amounts to the greatest absolute discrepancy (for any element) between the target and prepared states. We now consider two classes of PQC ansatz.

7.3.1. Hardware-efficient ansatz. In the following we consider a PQC ansatz $U(\vec{\theta})$ composed of L+1 layers for a register of n qubits which overall defines $n \times (L+1)$ variational parameters according to

$$U(\vec{\theta}) = U_R(\vec{\theta}^{L+1}) \underbrace{U_{\text{ENT}} U_R(\vec{\theta}^L) \dots U_{\text{ENT}} U_R(\vec{\theta}^1)}_{L-\text{times}}$$
(7.29)

where the free parameters are the angles of the $R_y(.)$ rotations, i.e.

$$U_R(\vec{\theta}^k) = \bigotimes_{i=0}^{n-1} R_y(\theta_i^k) \tag{7.30}$$

and the entanglement is applied by a (fixed) set of gates $U_{\rm ENT}$. For clarity, we summarize our different notations:

- (1) $\vec{\theta}$ stands for the vectors of all parameters
- (2) $\vec{\theta}^k$ stands for the vector of parameters in the k-th layer of the PQC
- (3) θ_i^k is the scalar angle of rotation of the *i*-th qubit in the *k*-th layer of the PQC.

Starting from the state $|0^n\rangle$, the state obtained using our ansatz is then

$$|\psi_{\text{PQC}}\rangle = |\psi(\vec{\theta})\rangle = U(\vec{\theta})|0^n\rangle$$
 (7.31)

This type of ansatz is part of the class of hardware-efficient (HWE) ansatze [79], as the entangling gates U_{ENT} can be chosen differently for different physical devices. In order to train and optimise the rotation angles we minimise the L_1 , L_2 and L_{∞} norms²⁰. Other cost functions which are physics-inspired can also be introduced [13].

²⁰The L_{∞} is also useful for bounding the discrepancy between continuous PDFs and the discretised version thereof.

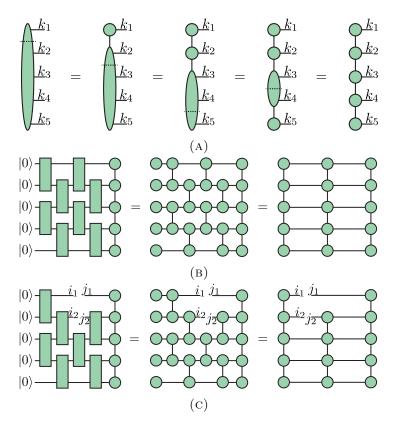


FIGURE 21. Five-qubit quantum state preparation using the TN approach in Ref. [58]. (A) The vector representing $\sqrt{p(x_k)}$ for a probability distribution p(x) evaluated on an equidistant grid of 2^5 grid points, x_k , is written as a tensor with 5 indices $(k_1, k_2, k_3, k_4, k_5) = \text{binary}(k)$. An exact matrix product state (MPS) [73–75] representation is obtained by performing several QR or polar decompositions at the dotted lines. An approximate MPS representation can be obtained by using truncated SVD instead of the QR or polar decompositions. (B) The quantity $\mathcal{F} = \langle \psi_{\text{MPS}} | \psi_{\text{PQC}} \rangle$ for a variational quantum brickwall circuit preparing the state denoted $|\psi_{POC}\rangle$ of circuit depth 4 is written as a TN. To achieve this, the vector $|0\rangle$ is written as a tensor with one index and every two-qubit gate is QR- or polar-decomposed into two tensors with three indices each. Finally we multiply adjacent tensors to turn \mathcal{F} into a TN consisting of MPS and a matrix product operator (MPO) [76,77]. (C) The TN required for the update of a variational two-qubit gate is obtained by removing that gate from the \mathcal{F} TN. Then the same steps as in (B) are used to transform the resulting TN into one containing only MPS and MPO. Contracting the resulting TN gives a matrix $A_{(i_1,i_2),(j_1,j_2)}$ with row-multi-index (i_1,i_2) and column-multi-index (j_1, j_2) . The optimal unitary \tilde{U} maximizing $\mathcal{F} = \operatorname{tr}(A\tilde{U})$, where $\operatorname{tr}(\cdot)$ denotes the trace over (\cdot) , follows from the SVD of A = USV: $\tilde{U} = V^{\dagger}U^{\dagger}$ [78]. We loop over all of the variational two-qubit gates and for each compute the optimal gate and insert it in the TN. This procedure is iterated until the cost function (here the L_2 norm) converges or until a set maximum number of iterations is exhausted.

7.3.2. Tensor network methods with brickwall ansatz. Another approach is to use a brickwall ansatz circuit composed of generic two-qubit gates. This is then used for quantum state preparation based on tensor network (TN) methods [73–75] as was used in Ref. [58]. In this case, the L_2 norm is

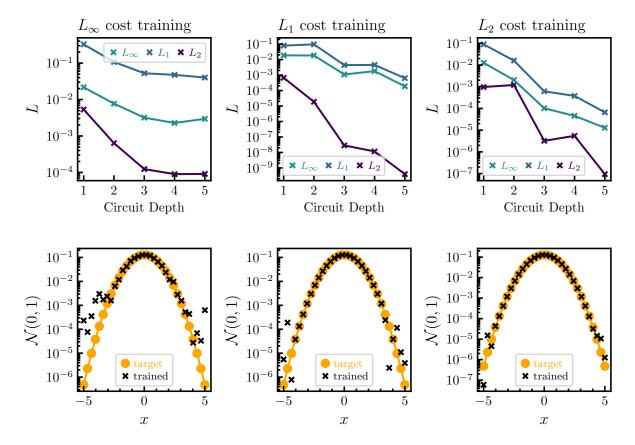


FIGURE 22. Training of a Gaussian random variable $\mathcal{N}(0,1)$ with respect to three cost functions (L_{∞}, L_1, L_2) as a function of the circuit depth for a HWE ansatz. The second row shows the prepared 32-point probability mass functions versus the target unit Gaussian for the maximum circuit depth of five layers. The quality of the training is measured by the L norm discrepancy of the prepared amplitudes with respect to the ground truth. We observe that the L_2 cost function provides the best training performance for the various metrics measured.

chosen to measure the distance between the target and trained state, and the training uses singular value decomposition (SVD) of the environment TN [78], as detailed in Fig. 21.

7.3.3. Training a PQC to prepare a Gaussian distribution. We first train a PQC to prepare a Gaussian random variable with 0 mean and variance 1, i.e.

$$X \sim \mathcal{N}(0,1) \tag{7.32}$$

which is known as a "unit Gaussian". By shifting and re-scaling this variable, we can then obtain samples from any $\mathcal{N}(\mu, \sigma^2)$. In terms of the distribution loader this means that the circuit is untouched, and the classical data describing how to interpret the quantum circuit as preparing an encoding of a probability distribution (namely x_{ℓ} and Δ) is updated accordingly. Hence the training only has to be done for the unit Gaussian. Figure 22 gives the results when a HWE-ansatz PQC with 5 qubits is used to prepare a $2^5 = 32$ point discrete approximation of the Gaussian distribution over the range [-5, 5]. In particular, it shows results for the cases where each of the L_{∞} , L_1 and L_2 norms are minimised, and we observe that overall L_2 norm provides the best performance. Henceforth we therefore only train to minimise the L_2 norm, and focus on deeper ansätze to improve

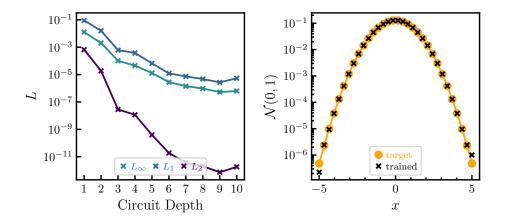


FIGURE 23. Training of a Gaussian random variable $\mathcal{N}(0,1)$ with respect to the L_2 norm cost function for a HWE ansatz. We can see in the left-hand subfigure the benefit of adding additional layers, and the right-hand subfigure shows the prepared 32-point probability mass functions versus the target unit Gaussian for the maximum circuit depth of nine layers (i.e., the number of layers with best performance).

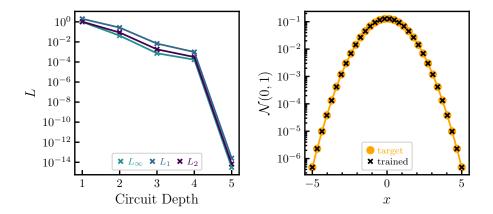


FIGURE 24. Training of a Gaussian random variable $\mathcal{N}(0,1)$ with respect to the L_2 norm cost function for TN training with a brickwall ansatz. We can see in the left-hand subfigure the benefit of adding additional layers, and the right-hand subfigure shows the prepared 32-point probability mass functions versus the target unit Gaussian for the maximum circuit depth of five layers (i.e., the number of layers with best performance).

the overall performance. In particular, Fig. 23 shows a 5-qubit approximation of the unit Gaussian with nine layers. We also obtain good results for preparing a 32-point discrete approximation of the unit Gaussian using the TN training with the brickwall ansatz, as shown in Fig. 24.

7.3.4. Training a PQC to prepare a lognormal distribution. Unlike for the Gaussian distribution, the lognormal distribution does not enjoy the location-scale property. This means that it is not possible to simply train a generic Lognormal, $\mathcal{LN}(\mu_0, \sigma_0^2)$ for a fixed set of parameters $\{\mu_0, \sigma_0^2\}$ to obtain the samples for any Lognormal. More precisely, if σ^2 is fixed, then it is possible to obtain

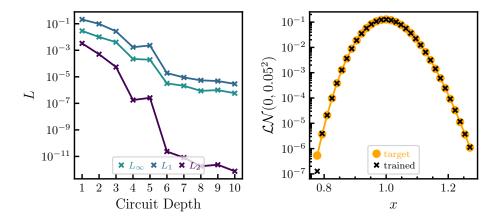


FIGURE 25. Training of a five-qubit approximation of the lognormal distribution with parameters $\sigma = 0.05$ and $\mu = 0$ with respect to L_2 cost function using a HWE ansatz.

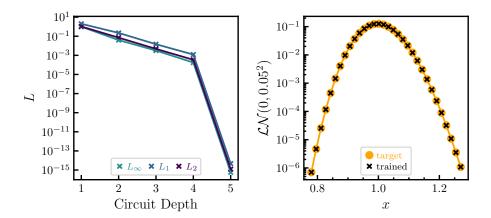


FIGURE 26. Training of a five-qubit approximation of the lognormal distribution with parameters $\sigma = 0.05$ and $\mu = 0$ with respect to L_2 cost function using a brickwall ansatz with TN training.

samples for any μ by re-scaling, and hence we focus on Lognormals of the form $\mathcal{LN}(0, \sigma^2)$. This still means that we must select σ , and here we set $\sigma = 0.05$ – which matches the low-volatility setting we address in the benchmarks in Section 11. For HWE and brickwall (with TN training) ansatz respectively, we show in Figs. 25 and 26 the training of the quantum circuit using 5 qubits with the L_2 cost function for the lognormal distribution with these parameters.

7.4. **Benchmarks.** For every quantum circuit that encodes a probability distribution, we must ask two crucial questions: how well do samples from the quantum state prepared match those from the desired probability distribution, and what quantum resources are needed to achieve this. In general there will be a trade-off between these two desiderata – higher quality approximations will require more quantum resources. In this section, we define a set of metrics to measure how well a certain distribution loading method performs, according to these two criteria. We then evaluate these for five- and six-qubit Gaussian and lognormal distributions prepared with the two PQC

```
state_preparation = StatePreparation()
  state_preparation_parameters = StatePreparationParameters(
      distribution=Normal(mean=0, variance=1), # Load the univariate unit Gaussian
      n_qubits=6,
4
5
      x1=-5,
6
      delta=10/63,
      ansatz="HWE",
                                                  # (or "TN") Specify the ansatz/loader
8
      n_layers=6,
                                                  # Number of ansatz layers
9
 distribution_circuit = state_preparation.get_circuit(state_preparation_parameters)
```

FIGURE 27. Example pseudocode for distribution loading.

training methods described in Section 7.3. Pseudocode giving an example to illustrate how these circuits can be loaded to prepare the desired probability distribution is given in Fig. 27.

7.4.1. Statistical figures of merit to assess the quality of a distribution loading routine. We introduce common metrics used in statistics to compare two discrete probability distributions P (resp. Q) with probability masses $\{p_i\}_{i\in\Omega}$ (resp. $\{q_i\}_{i\in\Omega}$). The first is the Kullback-Leibler (KL) divergence, or relative entropy²¹

$$D_{\mathrm{KL}}\left(\mathbf{P}||\mathbf{Q}\right) = \sum_{i \in \mathbf{Q}} p_i \log\left(\frac{p_i}{q_i}\right) \tag{7.33}$$

The second is the Jensen-Shannon (JS) divergence, which is a symmetrised version of the KL divergence

$$D_{\rm JS}(\mathbf{P}||\mathbf{Q}) = \frac{1}{2} \left(D_{\rm KL} \left(\mathbf{P}||\frac{\mathbf{P}}{2} + \frac{\mathbf{Q}}{2} \right) + D_{\rm KL} \left(\mathbf{Q}||\frac{\mathbf{P}}{2} + \frac{\mathbf{Q}}{2} \right) \right) \tag{7.34}$$

The third is the total-variation distance

$$\delta(\mathbf{P}, \mathbf{Q}) = \frac{1}{2} \sum_{i \in \Omega} |p_i - q_i| \tag{7.35}$$

The fourth is the infidelity, which is one minus the fidelity, defined as

$$F(P,Q) = \left(\sum_{i \in \Omega} \sqrt{p_i q_i}\right)^2 \tag{7.36}$$

The L_1 , L_2 and L_{∞} norms also give measures on the closeness of fit, and in particular the L_{∞} norm is useful for upper-bounding how close the final Monte Carlo estimate can be to the true solution.

7.4.2. Quantum resource figures of merit. In order to assess the resources required for some distribution loading method, we first need to compile the circuit into a standard form. Compiling to the gateset {TK1, CNOT}, where

$$TK1 = R_z(\alpha)R_x(\beta)R_z(\gamma). \tag{7.37}$$

(for some α , β , γ) captures all single-qubit unitaries is a suitably generic choice for this. We then count the following resources: the number qubits in the prepared state ("state qubits") and ancilla qubits required; the circuit, CNOT and TK1 depths; as well as the number of total gate, CNOT and TK1 gate counts as obtained from the tket methods n_gates_of_type(OpType.CX) and depth_by_type(OpType.CX) (and similarly by replacing CX by TK1).

 $^{^{21}}$ Here we use the natural logarithm in the definition – even though \log_2 is more conventional in information theoretic circles.

	HWE	TN	HWE	TN
Number of state qubits	5	5	6	6
Number of ancilla qubits	0	0	0	0
Circuit depth	18	21	22	37
CNOT depth	12	10	15	18
TK1 depth	6	11	7	19
Number of gates	50	65	72	142
Number of CNOT gates	20	20	30	46
Number of TK1 gates	30	45	42	96
KL divergence(target, trained)	1.85×10^{-5}	1.25×10^{-14}	1.14×10^{-4}	7.65×10^{-15}
KL divergence(trained, target)	2.84×10^{-5}	1.25×10^{-14}	2.14×10^{-4}	6.86×10^{-15}
JS divergence	5.30×10^{-6}	1.38×10^{-17}	2.96×10^{-5}	1.19×10^{-16}
Total-variation distance	3.31×10^{-5}	1.26×10^{-14}	2.17×10^{-4}	3.10×10^{-9}
L_{∞}	1.29×10^{-5}	3.14×10^{-15}	2.58×10^{-5}	7.37×10^{-10}

Table 4. Numerical data for the circuits preparing a Gaussian random variable $\mathcal{N}(0,1)$ on the interval [-5,5].

	$_{ m HWE}$	TN	HWE	TN
Number of state qubits	5	5	6	6
Number of ancilla qubits	0	0	0	0
Circuit depth	21	21	34	37
CNOT depth	14	10	23	18
TK1 depth	7	11	11	19
Number of gates	59	64	116	142
Number of CNOT gates	24	20	50	46
Number of TK1 gates	35	44	66	96
KL divergence(target, trained)	1.08×10^{-5}	4.12×10^{-15}	9.39×10^{-6}	9.04×10^{-11}
KL divergence(trained, target)	2.12×10^{-6}	4.10×10^{-15}	1.54×10^{-5}	9.05×10^{-11}
JS divergence	6.85×10^{-7}	3.34×10^{-17}	1.67×10^{-3}	2.26×10^{-11}
Total-variation distance	9.57×10^{-6}	3.97×10^{-15}	2.78×10^{-6}	4.74×10^{-7}
L_{∞}	3.20×10^{-6}	1.86×10^{-15}	3.96×10^{-6}	6.68×10^{-8}

TABLE 5. Numerical data for the circuits creating a lognormal random variable with mean 0.0 and variance 0.05^2 on the interval [49/63, 81/63].

7.4.3. Numerical experiments. We now evaluate these figures of merit for trained PQCs encoding five- and six-qubit Gaussian and lognormal distributions (for both straightforward L_2 -norm minimisation with a HWE ansatz and MPS training with a brickwall ansatz). (Note that each of these have been studied before in the literature, for the Gaussian distribution, see Refs. [13,80–85], and for the lognormal distribution, see Refs. [83–85].) The six-qubit Gaussian and lognormal distributions encoded in HWE are subsequently used in the benchmarks detailed in Section 11, and the full circuits are included in Appendix C. The results are displayed in Tables 4 and 5²².

²²All circuits were rebased using RebaseTket and were then optimised using the FullPeepholeOptimise of pytket, the documentation of pytket can be found at https://cqcl.github.io/tket/pytket/api/.

For the purposes of our benchmarks (see Section 11), achieving L_{∞} of around $10^{-5}-10^{-6}$ is sufficient (as in this case the error incurred by the training is significantly less than that of the QMCI itself) and we can see that the HWE PQC method achieves this with relatively modest quantum resources (note that the most important resource quantity is the CNOT gate count). For this reason, we find that PQC methods are suitable, at least for simple financial calculations and this corroborates the findings of Ref. [13]. We also note that in the TN PQC benchmarks, where we allowed more quantum resources to be used, extremely close fit was achieved – further validating the PQC methods in general. It is, however, worth observing that PQC methods are likely to suffer during fault-tolerant operation, where continuously-parameterised rotation gates are typically approximately synthesised by gates from a finite set. Furthermore, it is only by training generic "resource" states, that can be re-used, that we can reasonably omit mention of the training cost itself. Should it be the case that a bespoke trained PQC were needed for a single problem, then this would likely negate the quantum advantage once the overall cost (including training) is accounted.

8. Statistical quantities estimated by the QMCI engine

One of the strengths of the Fourier series decomposition of the Monte Carlo integral, as in Ref. [5], is that it is extremely flexible in terms of allowing different functions to be applied to the random variables prior to taking the expectation. For the purposes of the QMCI engine, we need only consider a relatively modest collection of statistical quantities to be estimated (i.e., functions to be applied), although it is easy to bring further functions online as needed.

To apply a function, g, to a random variable X, it is necessary to construct a periodic (in general piecewise) function such that the piece covering the support of the random variable corresponds to the desired function, and the overall function is sufficiently smooth to achieve the required convergence condition. A further important requirement, at least from an operational perspective, is to have a clear characterisation of the overall convergence of the returned estimate. This can be achieved by noting that the output is a straightforward affine combination of estimated amplitudes, where each amplitude has RMSE proportional to $c_{\rm QAE}/q$ (here q is the number of uses assigned to that harmonic). In particular, a general form of the Fourier series approximation of the Monte Carlo integral value is (from Eq. 3.7)):

$$\mu = a_0 + \sum_{m=1}^{\infty} a_m \left(\sum_{x^{(i)} \in \Omega} p(x^{(i)}) \cos(m\omega x^{(i)}) \right) + b_m \left(\sum_{x^{(i)} \in \Omega} p(x^{(i)}) \sin(m\omega x^{(i)}) \right)$$
(8.1)

so it follows that, according to Proposition 3.2, each of the parenthesised terms can be estimated using QAE operating on circuits of the form shown in Fig. 4 Furthermore, letting $q_m^{(a)}$ and $q_m^{(b)}$ be respectively the uses assigned to estimate the cosine and sine terms of the m^{th} harmonic, then the overall estimate has RMSE given by:

$$RMSE(\hat{\mu}) = 2c_{QAE} \sqrt{\sum_{m=1}^{\infty} a_m^2 \left(q_m^{(a)}\right)^{-2} + b_m^2 \left(q_m^{(b)}\right)^{-2}}$$
(8.2)

which follows directly from Proposition 3.2.

With knowledge of the specific Fourier series, as well as the way that a given number of uses is spread amongst the various harmonics, the RMSE of the estimate can always be expressed in the

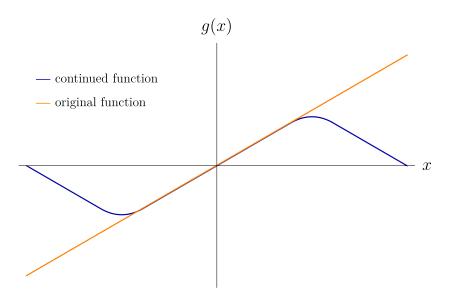


FIGURE 28. A sketch of the periodic piecewise function for estimating the mean. The location-scale property of the expectation is used to transform the support to the piece $x \in [-1, 1]$, as shown here, and the remainder of the function is constructed such that it is sufficiently smooth and has even symmetry.

following form:

$$\sqrt{\mathbb{E}((\widehat{\mathbb{E}_{g(x)}(X)} - \mathbb{E}_{g(x)}(X))^2)} \leqslant c_g c_{\text{QAE}} \frac{\max_{x \in [x_\ell, x_u]} (g(x)) - \min_{x \in [x_\ell, x_u]} (g(x))}{q}$$
(8.3)

where c_q is a constant that depends only on the function being applied.

With these requirements in mind, we include in the QMCI engine Fourier series decompositions that allow the following six quantities to be calculated:

- (1) the mean i.e., g(x) = x;
- (2) the conditional expectation (defined in Eq. (8.10));
- (3) the variance which uses the mean, as above, and the second moment, i.e., $g(x) = x^2$;
- (4) expectation of the random variable exponentiated i.e., $q(x) = \exp(x)$;
- (5) the conditional expectation of the random variable exponentiated (defined in Section 8.5);
- (6) the expectation of a single qubit treated such that (computational basis) measurements thereof are samples of a Bernoulli random variable.
- 8.1. Estimating the mean. To estimate the mean, we use the location-scale property of the mean to always assume that the random variable is supported over $x \in [-1, 1]$ within the QMCI engine, then shifting and re-scaling the returned estimate accordingly. This means that we always use exactly the same periodic piecewise function to construct the Fourier series, as sketched in Fig. 28, and in this way we obtain overall convergence:

$$\sqrt{\mathbb{E}((\widehat{\mathbb{E}_x(X)} - \mathbb{E}_x(X))^2)} = 1.68c_{\text{QAE}} \frac{x_u - x_\ell}{q}$$
(8.4)

i.e., $c_x = 1.68$ as in this case $\max_{x \in [x_\ell, x_u]} (f(x)) - \min_{x \in [x_\ell, x_u]} (f(x)) = x_u - x_\ell$.

8.2. Estimating the conditional expectation. The enhanced P-builder allows *indicator* qubits to be included, which encode a superposition of Boolean values, whose truth varies with each point supporting the multivariate probability mass function. The indicator qubit is therefore (as the name implies) an indicator function of the random variable, $\mathbb{1}(X)$, from which we obtain:

$$\mathbb{E}_{x|\mathbb{1}(x)}(X) = \sum_{X} X \mathbb{1}(X) p(X) + x^* \sum_{X} (1 - \mathbb{1}(X)) p(X)$$
(8.5)

where x^* is a user-defined value such that $x_{\ell} \leq x^* \leq x_u$. By default, if no value is given by the user:

$$x^* = \begin{cases} 0 & \text{if } x_{\ell} \leqslant 0 \leqslant x_u \\ x_{\ell} & \text{otherwise} \end{cases}$$
 (8.6)

In all of these cases, the convergence for the conditional expectation is equal to that given in Eq. (8.4).

Calculating the payoff of a European call option gives a (very) simple example of how we may use indicator qubits in this way. In this case, the threshold conditioning in the enhanced P-builder is set-up such that $\mathbbm{1}(X)$ returns the value 1 for the subset of prices above the strike price. More generally (for path-dependent derivatives), the indicator qubit may "gather up" conditioning on various logical statements throughout the history of the derivative instrument.

8.3. Estimating the variance & second moment. The variance of a random variable is defined as $\mathbb{E}[(X - \mathbb{E}[X])^2]$, which is estimated by first estimating the mean, and (classically) subtracting this from the support of X to "centralise". The second moment of this centralised version of X is then estimated. For this, we require $g(X) = X^2$, and use the fact that zero will now certainly be contained within the support to treat the support as $-x_b \leq x \leq x_b$, where $x_b = \max(x_u, -x_\ell)$. To construct the corresponding piecewise periodic function, the scale property of the second moment is used such that the support of the random variable is treated as [-1,1], with x_b then used to re-scale the returned estimate. Once again, in this way we can use the same periodic piecewise function every time the second moment is to be estimated, regardless of the support of the random variable. In particular, we design a periodic piecewise function which is equal to x^2 in the range [-1,1], as sketched in Fig. 29, and in this way we obtain overall convergence:

$$\sqrt{\mathbb{E}((\widehat{\mathbb{E}_{x^2}(X)} - \mathbb{E}_{x^2}(X))^2)} = 2.82c_{\text{QAE}} \frac{(\max(x_u, -x_\ell))^2}{q}$$
(8.7)

i.e., $c_{x^2} = 2.82$ as in this case $\max_{x \in [x_\ell, x_u]}(g(x)) - \min_{x \in [x_\ell, x_u]}(g(x)) = (\max(x_u, -x_\ell))^2$.

The QMCI engine allows to user to directly request the variance, in which case the mean and second moment are each estimated – in such a way that the overall desired accuracy is obtained with minimal total computational load.

8.4. Estimating the expectation of the random variable exponentiated. Estimating $\mathbb{E}(\exp X)$ is especially relevant when working in return (log-price) space in financial applications. In the case of the exponential, we can shift the random variable in a manner that can be accounted for according to:

$$\mathbb{E}_{\exp(x)}(X) = \exp(a)\mathbb{E}_{\exp(x)}(X - a) \tag{8.8}$$

however we cannot re-scale the random variable. This means that we cannot use exactly the same periodic piecewise function, regardless of the user-defined support, in the way we can for the mean and second moment. Our approach is to optimise the design of the periodic piecewise function by shifting the random variable, as shown in Fig. 30, however it is unavoidably the case that a

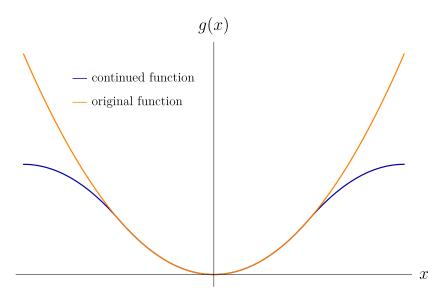


FIGURE 29. A sketch of the periodic piecewise function for estimating the second moment. The scale property of the expectation is used to transform the support to the piece $x \in [-1, 1]$, as shown here, and the remainder of the function is constructed such that it is sufficiently smooth and has even symmetry.

different Fourier series approximates the estimated quantity as the range of the support varies. This property, it turns out, means that when the RMSE of the estimator is divided by the number of uses, q, and the range of the support $\max_{x \in [x_\ell, x_u]}(g(x)) - \min_{x \in [x_\ell, x_u]}(g(x)) = \exp(x_u) - \exp(x_\ell)$, the result is not constant (although it does not depend on $\exp(x_u)$ and $\exp(x_\ell)$ individually, just $\exp(x_u) - \exp(x_\ell)$). In order to adhere to the canonical form given in Eqn. (8.3), it is therefore necessary to specify the support of x. For this, we choose to focus on the low-volatility setting²³, specifically, the standard deviation, $\sigma = 0.1$, which means that for five standard deviations²⁴ we have to set $x_\ell = -0.5$, $x_u = 0.5$, thus giving a total range of 1. With this in mind, we tailor the periodic piecewise function such that the RMSE of the estimator is divided by the number of uses, q, and the range of the support $\exp(x_u) - \exp(x_\ell)$ is as shown in Fig. 31, and from this we get that $c_{\text{exp}} \leq 2.59$, i.e.,

$$\sqrt{\mathbb{E}((\widehat{\mathbb{E}_{\exp(x)}(x)} - \mathbb{E}_{\exp(x)}(x))^2)} \leqslant 2.59c_{\text{QAE}} \frac{\exp(x_u) - \exp(x_\ell)}{q}$$
(8.9)

8.5. Estimating the conditional expectation of the random variable exponentiated. As was the case for the mean, it is also important to be able to exponentiate a random variable subject to some condition given by an indicator function. In this case, we again allow the user to input some value $\exp(x_{\ell}) \leq \exp(x^*) \leq \exp(x_u)$ (set to $\exp(x_{\ell})$ by default), and then estimate

$$\mathbb{E}_{\exp(x)|\mathbb{1}(x)}(X) = \sum_{X} \exp(X)\mathbb{1}(X)p(X) + \exp(x^*) \sum_{X} (1 - \mathbb{1}(X))p(X)$$
(8.10)

which has the same convergence properties as the straightforward exponential, given in Eqn. (8.9).

 $^{^{23}}$ It is possible to design an alternative periodic piecewise extension of $\exp(x)$ optimised for greater ranges of support, should high-volatility applications be needed, however this is not our focus for the benchmarks we give in this paper.

²⁴Stretching out to five standard deviations is probably unnecessarily conservative in most applications, however we follow the precedent set in Ref. [13].

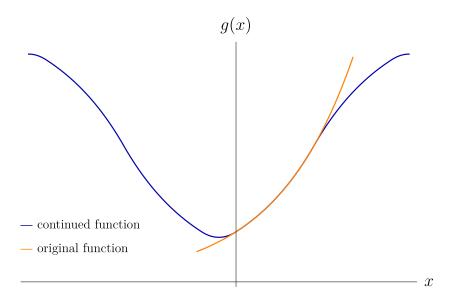


FIGURE 30. A sketch of the periodic piecewise function for estimating the expectation of a random variable exponentiated. Here, the highlighted piece is $\exp(x)$ with the support shifted accordingly, and the remainder of the function is constructed such that it is sufficiently smooth and has even symmetry.

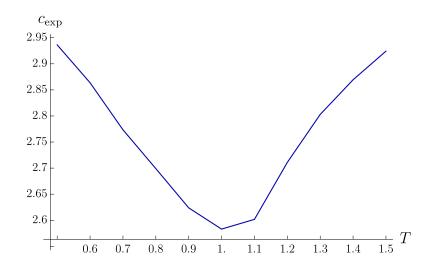


FIGURE 31. An illustration of c_{\exp} as a function of $T \equiv \exp(x_u) - \exp(x_\ell)$, for the periodic piecewise function that has a minimum around $\exp(x_u) - \exp(x_\ell) = 1$.

8.6. Estimating a single qubit as sampling a Bernoulli random variable. By definition, this is just a restatement of QAE itself, and so we get convergence:

$$\sqrt{\mathbb{E}((\mathbb{E}_{\text{Bernoulli}}(X) - \mathbb{E}_{\text{Bernoulli}}(X))^2)} = c_{\text{QAE}} \frac{1}{q}$$
(8.11)

i.e., $c_{\text{Bernoulli}} = 1$ and in this case $\max_{x \in [x_\ell, x_u]}(g(x)) - \min_{x \in [x_\ell, x_u]}(g(x)) = 1$ $(x_\ell = 0; \Delta = 1)$.

8.7. Applying the function to part of the support. As each dimension of the multivariate probability distribution has explicitly defined support, by default the periodic piecewise function

Quantity	c_f	Crossover (LCU QAE)	Crossover (ML QAE)	Crossover (IQAE)
Mean & Conditional expectation	1.68	1.12%	1.86%	1.03%
Second moment	2.82	0.67%	1.11%	0.61%
Exponential expectation & Conditional exponential expectation	2.59	0.72%	1.20%	0.67%
Single qubit (Bernoulli)	1	1.88%	3.12%	1.73%

Table 6. Comparison between (upper-bounds on) classical and quantum convergences for various statistical quantity estimates in terms of the number of samples, q. "Crossover" is the error (RMSE), given as a percentage of the range of the support, at which it becomes more efficient to use the quantum algorithm.

is such that the desired function is applied to the entire support (i.e., $x_{\ell} \dots x_{u}$), however the user can optionally choose for this section to cover only part of the support. This is useful if, for example, it is known that the entire probability mass is concentrated in this region. In this case, the convergence would be correspondingly improved, as the numerator in Eq. (8.3) would now only take the maximum and minimum function evaluation over the restricted range.

8.8. Summary of QMCI convergences. Values of $c_{\rm QAE}$ for the three forms of QAE included in the QMCI engine are given in Table 7 which can thus be used to perform a direct comparison between QMCI and classical MCI in terms of convergence rates, using also the bounds on the convergence of classical MCI estimators, in Section 2.1:

$$\sqrt{\mathbb{E}((\widehat{\mathbb{E}_f(x)} - \mathbb{E}_f(x))^2)} \leqslant \begin{cases} \frac{(\max(x_u, -x_\ell))^2}{2\sqrt{q}} & \text{if } f = x^2\\ \frac{f(x_u) - f(x_\ell)}{2\sqrt{q}} & \text{otherwise} \end{cases}$$
(8.12)

Here the condition that $x_{\ell} \leq 0 \leq x_u$ for the second moment estimation has again been used, as well as the property that all of the other functions are monotonically increasing. Table 6 compares the classical and quantum MCI convergences in terms of the number of samples, however this is a fair (or even conservative) way to do so, given that a classical sample is at least as hard to prepare as a quantum sample [27], and with Fourier QMCI we have little circuitry other than that encoding the sampling circuit [5]. (Note that hardness of preparing a sample is equivalent to sampling speed.)

In order to illustrate the validity of these bounds in practice, we give explicit examples for the case where the quantity is being calculated for a 32 point (5 qubit) discrete approximation of a Gaussian random variable distributed as $\mathcal{N}(0,0.1^2)$. For the conditional expectation / conditional exponentiated expectation we set $x^* = 0$, and in all cases MLQAE was used. Figure 32 shows that the upper-bounds hold for all five quantities when compared to numerical data obtained by running the calculations 10000 times on a state-vector simulator (notice that the upper-bound on qubit as Bernoulli follows directly from the extensive analysis on QAE in Sections 5 and 9, and hence we omit repeating this here).

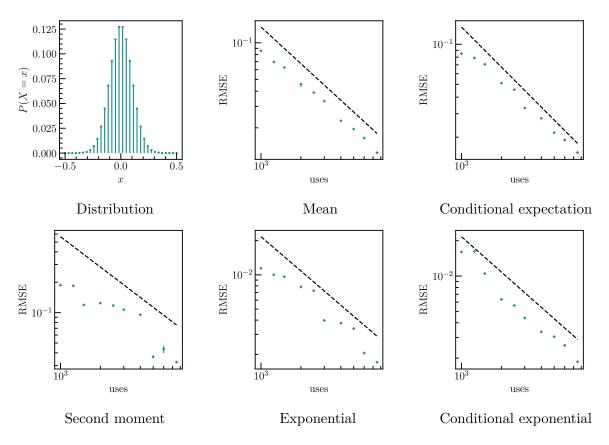


FIGURE 32. Theoretical upper-bounds (dashed lines) on QMCI convergence for the five quantities of interest, against numerically simulated data (single points). Thus we can see that the theoretical upper-bounds indeed hold in all cases.

9. QUANTUM AMPLITUDE ESTIMATION

The QMCI engine includes three QAE subroutines, our LCU QAE, MLQAE and IQAE; additionally we include prepare and measure (PAM), in which the state $A | 0 \rangle$ is simply prepared and measured the specified number of times. Definition 3.1 dictates the functional form of a QAE subroutine, and crucially the relationship RMSE(\hat{a}) $\leq c_{\text{QAE}}q^{-\lambda/2}$ needs to hold for any positive integer q. This is not necessarily the case for "off-the-shelf" implementations of QAE, in which q is effectively only permitted to be a value that aligns with the needs of the QAE algorithm in question. Moreover, in the case of the quadratic advantage-achieving exponentially increasing sequence (EIS) of MLQAE, these numbers of allowed uses spread out exponentially. It is therefore necessary to enhance these algorithms to accept any number of uses – and to use those uses productively.

The constant c_{QAE} is defined to upper-bound the RMSE convergence, and hence it is necessary to consider the worst case for all amplitudes. In the case of PAM this is easy, as we automatically have RMSE(\hat{a}) $\leq 0.5q^{-1/2}$ for any positive integer q – this is just classical MCI, and the bound is saturated when a=0.5. However, here $\lambda=1$ and for the forms of QAE that actually use the quantum mechanical effects to achieve $\lambda=2$ (the full quadratic advantage), some work is needed to adapt the algorithms to work for any number of uses before thereafter evaluating c_{QAE} numerically.

9.1. LCU QAE. Our LCU QAE algorithm, as described in Section 5, is based on the EIS of Ref. [1]. Even when the input number of uses is interpreted as the successful uses, we are still faced with the fact that only certain values of q exactly align with the number needed to complete the EIS for some value of m. As this will not, in general, be the case, it is necessary to curtail the EIS at the final value of m for which the necessary circuit executions can be completed (i.e., as bounded by q), and find some suitable way of deploying the remaining uses.

Letting m^* be the maximum value in the EIS for which it is possible to run all $n_{\rm shots}$, then the idea is to find greatest $m' > m^*$ such that it is possible to run n_{shots} with m' applications of Q. Once the shots for m' have been accounted for, any further remaining uses are allotted to the existing values of m (including m=m') such that the higher values of m are prioritised, but ultimately all uses are taken up (this can always be guaranteed, as adding a single shot to m=0consumes just a single use). This approach is motivated by two principles:

- (1) As the quadratic advantage is obtained by rotating the state to ever higher values of θ , we aim to use up the remaining uses by running the full round of $n_{\rm shots}$ for the highest possible value of m – even if this is not the next value of m in the EIS.
- (2) We may as well always exhaust the assigned uses, and following principle 1, we do this in such a way that the uses are prioritised for the largest values of m.

In LCU QAE, we have two notions of the number of uses of the circuit P, namely successful uses (assuming post-selection on the LCU condition) and expected total uses (including LCU postselection failures). LCU QAE has been designed to fail fast, and so a failed LCU state preparation costs the same regardless of the circuit depth. This means that, as the circuit depths (i.e., the desired accuracy) increases, the contribution of the failed uses to the total uses is ever-diminishing. For this reason, counting only successful uses indeed gives us the asymptotic rate of convergence. However, for any real (finite) problem, we should count the failed uses, and to do so in a manner that is consistent with the overall scaling of RMSE with uses, we choose the minimum upper-bounding line for all of the empirical data.

- 9.2. MLQAE. To enable MLQAE to accept any number of uses, we can use the same strategy as for LCU QAE.
- 9.3. **IQAE.** Our implementation of the IQAE algorithm follows the original presentation given by Grinko et al [2], with some modifications to allow us to directly compare this method with the other forms of QAE we consider.

The original algorithm takes as input a circuit A, a desired error ϵ and a confidence value α and proceeds as follows. Start with the interval $[\theta_{\ell}^0, \theta_u^0] := [0, \pi/2]$, which trivially contains θ . The iterations of the algorithm then produce increasingly smaller α -confidence intervals for θ

$$[\theta_\ell^0, \theta_u^0] \supset [\theta_\ell^1, \theta_u^1] \supset [\theta_\ell^2, \theta_u^2] \supset \dots$$

The algorithm terminates after T iterations, where T is the smallest value such that $\theta_u^T - \theta_\ell^T \leq 2\epsilon$. The algorithms then returns as output $[a_\ell, a_u] = [\sin^2(\theta_\ell^T), \sin^2(\theta_u^T)]$, an α -confidence interval for the amplitude a of A.

The functional form of this algorithm is different from that presented in Definition 3.1. Indeed, instead of taking A and a number of uses q as input it takes A and the pair (α, ϵ) and instead of returning a point estimate it returns an interval, giving it the following form:

$$[a_\ell,a_u] = {\tt IQAE}(A,(\epsilon,\alpha))$$

	Maximum	Median	Minimum
LCU QAE (upper-bound)	13.3	11.4	3.74
LCU QAE (asymptotic)	7.82	6.80	2.15
MLQAE	8.02	6.18	2.13
IQAE	14.4	7.48	1.49
PAM	0.5	0.44	0.14

TABLE 7. c_{QAE} values for the different forms of QAE. By definition, c_{QAE} is taken as the value shown in the column "maximum".

Furthermore, for a given A, α and ϵ , it is not defined from the outset how many uses of A will be required to run the IQAE algorithm. So, we define a function $\mathsf{optAE}(q)$ which chooses a suitably optimal pair (α, ϵ) such that $\mathsf{IQAE}(A, (\epsilon, \alpha))$ uses at most q calls to A. This can be done by using the following upper-bound which can be derived from the original IQAE paper [2]:

$$q_A \leqslant q(\alpha, \epsilon) := \left(\frac{100}{\epsilon} + \frac{32}{(1 - 2\sin(\pi/14))^2}\right) \log\left(\frac{2}{\alpha}\log_2\left(\frac{\pi}{4\epsilon}\right)\right)$$

Noting also that the RMSE of the estimate $\hat{\theta} = (\theta_{\ell}^T + \theta_u^T)/2$ is upper-bounded as

$$RMSE(\hat{\theta}) \leqslant R(\alpha, \epsilon) := (1 - \alpha)\epsilon^2 + \alpha \left(\frac{\pi}{2}\right)^2$$

we implement optAE as the function which takes input q and returns α and ϵ which minimise $R(\alpha, \epsilon)$ subject to $q(\alpha, \epsilon) = q$. Thus, we can define the following version of IQAE which is of the desired form:

$$\mathtt{IQAE}^*(A,q) \coloneqq \mathtt{midpoint}(\mathtt{IQAE}(A,\mathtt{optAE}(q)))$$

where midpoint is a simple function that takes the mean of the two values returned by IQAE (i.e., interpreted as their "midpoint" – as the two values are real numbers).

Empirically, we found that this implementation rarely uses the total number of uses q, indicating that the bound $q(\alpha, \epsilon)$ is quite loose. In order to provide a fair comparison of this algorithm with other forms of QAE we modify IQAE* by first running IQAE(A, optAE(N)) and, then, given the final confidence interval $[\theta_{\ell}^T, \theta_u^T]$ we continue to perform the iterations of the IQAE algorithm until almost all q uses are exhausted. This implementation is the one used to generate the data for example in Table 7.

- 9.4. **Prepare and measure.** As already noted, for PAM we have the analytical result $c_{\text{QAE}} = 0.5$ which automatically holds for any number of uses, but now this is a constant of proportionality for RMSE convergence proportional only to $q^{-1/2}$.
- 9.5. Summary of QAE convergences. We used the numerical data of the simulations described in Section 5 to find c_{QAE} for the various QAE sub-routines included in the QMCI engine. As discussed in the opening of this section, it is necessary to conservatively select the worst case (largest) value of c_{QAE} in each case, and Figs. 38 41 show these for each of the 49 amplitudes for which we simulated QAE. Table 7 summarises the convergences for the three featured forms of QAE, and PAM. By definition, c_{QAE} is taken as the value shown in the column "maximum", however also included are "median" which gives an indication of the typical convergence one would expect for an arbitrarily chosen angle; and "minimum" which shows the best case convergence.

It is also worth briefly comparing to "canonical QAE" – using the quantum Fourier transform [30]. According to Ref. [30, Thm 12] the absolute error can be upper-bounded by a term that is at least $2\pi\frac{\sqrt{a(1-a)}}{q/2}$ – but that this is only achieved with probability $8/\pi^2$. Neglecting the failure probability and assuming that the estimate is uniformly distributed within the margin bounded by the maximum absolute error, we then get $c_{\text{QAE}} \approx \frac{2}{\sqrt{3}}\pi = 5.44$. This figure is comparable to those in Table 7, however the deepest circuits in canonical QAE are about 60 times as deep as those of MLQAE and LCU QAE (for the same value of q). Furthermore, the fact that canonical QAE only outputs the estimate within the confidence bounds with a certain (fixed) probability means some repeats are necessarily going to be required to boost this probability (i.e., to ensure that the overall RMSE – when the non-zero failure probability is also included – is at the required level). Grinko $et\ al$ find that when canonical QAE is run with a suitable number of repeats, it actually performs worse (in terms of RMSE convergence) than QPE-free forms of QAE [2, Fig. 3]. These facts justify our focus on QPE-free forms of QAE, and the need to find a statistically robust form thereof.

10. Resource mode

There is a significant discrepancy between the sizes of circuits which can be executed on presentday hardware (at the time of writing), and those which will be needed to realise a quantum advantage. It is, however, the case that the techniques we describe in the preceding sections do allow the explicit construction of those circuits which will eventually realise a useful quantum advantage - and whilst there is no current prospect of executing or simulating these, there is great value in knowing some broad properties of these circuits. This is because knowing the resources that these circuits require will enable more precise predictions about when certain calculations will see the impact of quantum computing and hence, in turn, will enable forward planning for deployment and timelines for necessary infrastructural developments etc. To facilitate this we have enabled the QMCI engine to run in resource mode, wherein rather than executing the circuits, instead the resource quantification is returned to the user. Resource mode enables the QMCI engine to be used as a prototyping and experimentation tool, wherein the impact on resource requirements of different designer-choices about how to construct various calculations is quantified precisely. For instance, there is often a range of appropriate models for financial time-series, each with myriad possible ways of being encoded in a quantum circuit; additionally, the user has a certain degree of freedom as to how to slice up the time-series and model the derivative instrument payoff. These design choices may well lead to vastly differing resource requirements, and so it is only through research and development using resource mode that the most direct pathway to quantum advantage will be revealed.

Resource mode comes with two sub-modes, namely NISQ resource mode and fault-tolerant resource mode. In each case, resources are counted for all of the component QMCI circuits – although this constitutes something of an over-count, as in practice certain parts of the QMCI could be run classically (indeed this is a further strength of the Fourier series decomposition: the quantum computer need only be used for those parts of the Monte Carlo integral that are truly hard classically):

- circuits corresponding to m = 0 are nothing more than classical Monte Carlo samples with sine / cosine applied;
- indeed, even for low values of $m \neq 0$ it is likely that shallow-circuit (tensor-contraction based) classical simulators could readily be deployed to further reduce the load on the quantum computer;
- harmonics that are awarded only a small number of uses by Fourier QMCI could be wholly estimated using classical MCI instead.

- 10.1. **NISQ resource mode.** To count gates in resource mode, it is necessary to fix a gateset to which the circuit is rebased²⁵. Even though the native gateset varies from machine to machine, an appropriate choice is the universal gateset {TK1, CNOT}, where the parameterised TK1 gate introduced in Eq. (7.37) is universal for single-qubit operations (up to a global phase). After the circuit has been thusly rebased, NISQ resource mode proceeds to construct all of the circuits for the QMCI, but rather than running, instead reports, for each circuit:
 - (1) the number of qubits;
 - (2) the total gate count and depth;
 - (3) the CNOT count and depth,

i.e., the total dimensions of the circuit; and those of the usual bottleneck (the CNOT gate).

10.2. Fault-tolerant resource mode. As many of the eventual applications of QMCI will require error-corrected quantum computers, it is useful to provide an alternative set of resource quantities based on compilation to some fault-tolerant gateset. The standard choice is to compile to the gateset $\{\text{CNOT}, H, S, T\}$ – known as "Clifford+T" and denoted $\{\text{Clifford}, T\}$ – however, this is a discrete gateset, and so it is also necessary to determine the required closeness of approximation. To do this, we first note that in general, the circuits built by the QMCI engine use gates from the set:

$$\mathsf{G}_{\text{full}} = \{ \text{Clifford}, T \} \cup \{ \text{Toffoli}, R_x(\cdot), R_y(\cdot), R_z(\cdot), CR_x(\cdot), CR_y(\cdot), CR_z(\cdot) \}$$
 (10.1)

However, for fault-tolerant (error-corrected) operation we need to (approximately) compile instead to a gate-set:

$$G_{FT} = \{\text{Clifford}, T\} \tag{10.2}$$

This, in turn, means realising the following decomposition to an appropriate approximation:

$$\{\text{Toffoli}, R_x(\cdot), R_y(\cdot), R_z(\cdot), CR_x(\cdot), CR_y(\cdot), CR_z(\cdot)\} \longrightarrow \{\text{Clifford}, T\}$$
(10.3)

In the case of Toffoli, this is straightforward as an exact decomposition exists; and we also note that controlled rotations can exactly decomposed into a circuit with Clifford+T gates and precisely six (uncontrolled) rotation gates [19, Corollary 4.2 & Fig. 4.6]. It is therefore possible to exactly rebase the original circuit to a circuit containing just gates from the set:

$$G_{\text{reduced}} = \{\text{Clifford}, T\} \cup \{R_x(\cdot), R_y(\cdot), R_z(\cdot)\}$$
(10.4)

However, Clifford+T compilation of the (uncontrolled) rotation gates can only be achieved up to a specified approximation. This approximation can be optimised – by which we mean the total T-gate count can be minimised, as this is typically the bottleneck in fault-tolerant circuit execution. This optimisation uses the fact that each $R_x(.)$, $R_y(.)$ and $R_z(.)$ gate can be approximated with a circuit containing $3\log_2(1/\epsilon)$ T gates, where ϵ is an upper-bound on the operator norm distance between the exact and approximate unitary [86].

If a positive operator-valued measurement (POVM) is made for a circuit containing a total of n_R gates from the set $\{R_x(\cdot), R_y(\cdot), R_z(\cdot)\}$, each of which is approximated to at most ϵ operator norm distance, then the maximum absolute discrepancy in probability of measuring any given POVM element is upper-bounded by [19, Eq. (4.62)]

$$\epsilon_{\text{tot}} \leqslant 2n_R \epsilon$$
 (10.5)

 $^{^{25}}$ In the future, we may extend this to allow user-defined gatesets, however for now – as our focus is on a generic quantification of resources, primarily aimed at problem sized that cannot yet be executed – having just a single standard gateset meets the requirements.

To see how we can use this to optimise the choice of ϵ for a certain instance of QMCI, first, we imagine that all of the circuits to be executed are arranged parallel, such that the whole algorithm is a tensor product of the component circuits. The output is then a partial computational basis measurement, to give an output bitstring x, which will then be mapped to $\hat{\kappa}$ – the estimate of the quantity to be estimated, κ . This mapping from x to $\hat{\kappa}$ need not be 1-to-1 (that is, different bitstrings may map to the same $\hat{\kappa}$), however, using the fact that the upper-bound on the discrepancy in measurement outcome probability holds for any POVM, we can choose a POVM such that bitstrings which map to the same value of $\hat{\kappa}$ are grouped in the same POVM element. In this way we label the possible measurement outcomes by $\hat{\kappa}$ without repetition, and hence we can consider the measurement to be directly supported over some discrete values of $\hat{\kappa}$. Thus we have that the final measurement is such that each possible outcome is observed with probability whose magnitude differs at most by ϵ_{tot} , and we can use this to bound the probability of obtaining some $\hat{\kappa}$ under approximate gate synthesis:

$$p(\hat{\kappa}) \leqslant p_{\text{exact}}(\hat{\kappa}) + \epsilon_{tot}$$
 (10.6)

where $p_{\text{exact}}(\hat{\kappa})$ is the probability had all of the QMCI circuits been executed exactly rather than approximately. Therefore, using simple statistics:

$$MSE(\hat{\kappa}) = \int (\hat{\kappa} - \kappa)^2 p(\hat{\kappa}) d\hat{\kappa}$$

$$\leq \int (\hat{\kappa} - \kappa)^2 p_{\text{exact}}(\hat{\kappa}) d\hat{\kappa} + \epsilon_{tot} \int (\hat{\kappa} - \kappa)^2 d\hat{\kappa}$$

$$\leq MSE_{\text{exact}}(\hat{\kappa}) + \epsilon_{tot} \left[\frac{(\hat{\kappa} - \kappa)^3}{3} \right]$$

$$\leq MSE_{\text{exact}}(\hat{\kappa}) + \frac{\epsilon_{tot}}{3} \left(\max_{x \in [x_\ell, x_u]} (g(x)) - \min_{x \in [x_\ell, x_u]} (g(x)) \right)^3$$
(10.7)

where $MSE_{exact}(\hat{\kappa})$ is the MSE had all of the QMCI circuits been executed exactly rather than approximately.

From Eq. (8.4) we also have that the convergence using QMCI is

$$MSE_{exact}(\hat{\kappa}) \leqslant c_g^2 c_{QAE}^2 \left(\max_{x \in [x_\ell, x_u]} (g(x)) - \min_{x \in [x_\ell, x_u]} (g(x)) \right)^2 q^{-2}$$
(10.8)

and thus

$$MSE(\hat{\kappa}) \leqslant c_g^2 c_{QAE}^2 \left(\max_{x \in [x_\ell, x_u]} (g(x)) - \min_{x \in [x_\ell, x_u]} (g(x)) \right)^2 q^{-2} + \frac{\epsilon_{tot}}{3} \left(\max_{x \in [x_\ell, x_u]} (g(x)) - \min_{x \in [x_\ell, x_u]} (g(x)) \right)^3$$

$$(10.9)$$

If we make the slightly conservative simplifying assumption that every time the circuit P is used it is done so to construct the circuit Q, then we get an upper-bound on the number of T gates as a function of the number of uses of P. Specifically, we let $n_R^{(Q)}$ and $n_T^{(Q)}$ be respectively the number of rotation and T gates in the circuit Q (prior to the approximate T synthesis for the rotation gates, but after every other gate has been rebased to Clifford+T) and thus the total amount of T gates is $\frac{q}{2}\left(3n_R^{(Q)}\log_2(1/\epsilon) + n_T^{(Q)}\right)$ where q is the number of uses, and hence q/2 upper-bounds the number of instances of Q. Putting this together, we can obtain an optimisation problem, where

the aim is to satisfy the user specified MSE (MSE_{user}), whilst minimising the T gate count:

$$\begin{aligned} \text{Minimise: } & \frac{q}{2} \left(3n_R^{(Q)} \log_2(1/\epsilon) + n_T^{(Q)} \right) \\ \text{such that: } & \text{MSE}_{\text{user}} = c_g^2 c_{\text{QAE}}^2 \left(\max_{x \in [x_\ell, x_u]} (f(x)) - \min_{x \in [x_\ell, x_u]} (f(x)) \right)^2 q^{-2} \\ & + \frac{q n_R^{(Q)} \epsilon}{3} \left(\max_{x \in [x_\ell, x_u]} (f(x)) - \min_{x \in [x_\ell, x_u]} (f(x)) \right)^3 \end{aligned}$$

which uses $\epsilon_{\rm tot} \leqslant 2n_R \epsilon = 2\frac{q}{2} n_R^{(Q)} \epsilon = q n_R^{(Q)} \epsilon$.

In the QMCI engine, this is solved to give the optimal q and ϵ for the user-defined overall MSE. (If the user specifies the number of uses rather than (R)MSE, the relationship between number of uses and RMSE convergence, for the selected statistical quantity and choice of QAE, as detailed in Section 8, is used to obtain the users implied desired (R)MSE.) Once this optimisation has been performed, resource mode proceeds to quantify the resources assuming the synthesis has been performed to the specified accuracy and reports for each circuit:

- (1) the number of qubits;
- (2) the T count and depth,

where T is now the most relevant single "bottle-neck" gate.

10.3. Tightening the fault-tolerant resource estimate. The bound $\epsilon_{tot} \leq 2n_R \epsilon$ is, in reality, extremely conservative – in the sense that it presumes that all of the errors line up such that the discrepancy between the ideal state and that prepared by the imperfect synthesis is a coherent sum of the individual discrepancies. A more reasonable assumption would be that the discrepancies are uniform iid, in which case the large dimension of the Hilbert space 2^n relative to the number of T gates means that the errors will be quasi-orthogonal with very high-probability [87] and hence a more useful approximation is given by:

$$\epsilon_{tot} \approx 2\sqrt{n_R}\epsilon$$
 (10.10)

Using this would enable the Clifford+T synthesis to be optimised further, and there is good reason to believe that further improvements will be possible in time:

- effective fault-tolerant circuit compilation is not yet a mature field, and it is reasonable to expect that in due course there will be techniques to greatly reduce rotation and T gate counts:
- Clifford+T synthesis performed by simply replacing each rotation gate by a string of Hadamard and T gates is sub-optimal a better approach is to re-write small sub-circuit blocks, and if ancillas are available the T gate counts can be heavily reduced in some cases;
- the bank of rotation gates used in Fourier QMCI is highly structured each rotation is a power of 2 multiplied by the same constant and this structure could be exploited to synthesise the controlled rotations with many fewer T gates;
- the permutational subcircuits that the enhanced *P*-builder adds have not yet been fully optimised for fault-tolerant execution.

As well as these general reasons to suppose that fault-tolerant resource requirements will be greatly reduced, for the specific benchmarks given in Section 11, it is worth observing that the distribution loading circuits are variationally trained and hence have a high proportion of rotation gates. When expressly designed for fault-tolerant execution (the benchmarks are for both NISQ and fault-tolerant execution), such circuits could be alternatively constructed to have more favourable resource requirements.

10.4. **Resource boxes.** In resource mode it is possible to include resource boxes which specify the number of qubits and number and types of gates rather than the explicit circuits. This is included to allow for resource quantification when prototyping new ideas; for instance, it may be that some application requires a certain arithmetic or logical operation that is not currently included in the enhanced P-builder. In this case, the user can approximate the resources needed for the operation, include these as a resource box, and thus estimate the total resources for the calculation of interest. This also further aids the development in terms of ascertaining which functions should be prioritised for future inclusion in the enhanced P-builder.

Part IV. Benchmarks and Discussion

11. Benchmarks

To demonstrate the capability of the quantum Monte Carlo integration engine, and in particular its ability to output resource estimates, we propose a set of benchmarks relevant to finance. Our motivation is to provide a series of calculations that are hard enough to be of interest – they are all path-dependent options where MCI would be likely to be the most effective method classically – but simple enough to be comprehensible and of relatively near-term interest. With this in mind, we give resource estimates for computing the expected payoff for:

- (1) barrier options;
- (2) look-back options;
- (3) autocallables;

where each of these derivative instrument depends on a single low-volatility underlying asset.

11.1. **Modelling the financial time-series.** In order to provide a simple and re-usable set of benchmarks, we propose a slightly simplified problem that is inspired by a financial time-series. In particular, any digital computer (classical or quantum) can only work with a discrete approximation of a continuously varying financial time-series, and for our purposes we take this discrete approximation as the ground truth.

We use the standard textbook model of geometric Brownian motion as the starting point for our discrete financial time-series, however, as we choose to work in return space, it is a discretisation of Brownian motion that is required. Specifically, at each time step an iid random variable that is a 64-point (i.e., six-qubit) discrete approximation of a Gaussian random variable is added to the logarithm of the price. Owing to the location-scale property of the Gaussian distribution it suffices to generate a single quantum circuit which encodes a Gaussian distribution, and to then interpret the support according to the needs of the problem. The specific circuit we use to do so is given in Fig. 42; the resulting distribution approximates a unit Gaussian, with support extending out to $\pm 5\sigma$. From this, we consider two discrete time-series, each with total volatility 10%:

- (1) a time-series with four time-slices thus each iid Gaussian is $\mathcal{N}(0, 1/400)$ (i.e., the total volatility is thus $\sqrt{4 \times 1/400} = 10\%$);
- (2) a time-series with eight time-slices thus each iid Gaussian is $\mathcal{N}(0, 1/800)$ (i.e., the total volatility is thus $\sqrt{8 \times 1/800} = 10\%$);

We choose to work in return space as, even though $c_{\exp} > c_x$ (hence the rate of convergence is slower by a constant factor), constructing Brownian motion rather than geometric Brownian motion is especially advantageous in the low-volatility regime. In this setting, when working in price space the random variable takes a value close to one (when normalised by the asset's starting price) with high probability, and therefore the most efficient use of the qubit registers representing the random variable is a grid of values closely spread around one (so with x_{ℓ} slightly smaller than one, and small Δ). However, multiplication of such lognormal random variables (as is required to construct geometric Brownian motion) is not straightforward, and the necessary register size effectively grows as if the random variables had instead been initialised with $x_{\ell} = 0$, thus leading to vast numbers of qubits and gates. No such problem exists when working in return space, where the Brownian motion is constructed simply by summing zero-mean Gaussian random variables. For these reasons, even though the sample complexity is higher working in return space, we expect the overall gate complexity (and qubit number) to be significantly lower when working therein.

Even though our preference is to work in return space, for completeness we also give circuits to sample the lognormal random variables, as would be required to work in price space, and all of the circuits are displayed in Appendix \mathbb{C}^{26} .

- 11.2. **Details of the selected instruments.** As we are interested in resource estimates, the exact settings are of little consequence as the resources do not vary significantly with these; however, for the sake of definiteness we use the following settings²⁷.
 - The barrier option is a knock-out call option, with (constant) barrier of 1.1 times the initial price; and strike price of 1.05 times the initial price.
 - The look-back option is a call option with strike price of 1.05 times the original price.
 - The autocallable is defined by two components.
 - (1) a series of binary call options $\{(K_i, t_i, b_i)\}_{i=1,...m}$, which pay out b_i times the initial price if the price at t_i exceeds K_i times the initial price. In our examples we include calls on every odd numbered timestep with linearly increasing values for K_i and b_i .
 - (2) A knock-out put option with barrier of 0.9 times the initial price and strike price 0.95 times the initial price.

If any of the call options are satisfied the payoff of the autocallable is the payoff of the earliest option (smallest t_i) that is satisfied. If none of the call options are fulfilled, the payoff is determined by the knock-out put and will be negative in this case. This is done in practice to reduce the overall price of an autocallable option, see for example Ref. [13] from which we have adapted this example.

For each instrument, we quantify the resources required to compute the expected payoff when the payoff is each of the value of the asset, and a fixed binary payoff (when the option is exercised)²⁸. For the former we set the target RMSE as $10^{-2}(e^{0.5}-e^{-0.5})=1.04\times10^{-2}$ and for the latter the target RMSE as 10^{-2} . In each case, these RMSE values correspond to 1% of the range of the support and have been chosen to approximately coincide with the cross-over to quantum advantage in sample complexity. As we use MLQAE to generate the results, Table 6 shows that RMSE of 3.12% of the support and 1.2% of the support is the cross-over for value and binary payoffs respectively, and so we have indeed entered into the regime of quantum advantage in sample complexity. Fig. 33 gives pseudocode for the four time-slice barrier option.

It is also worth mentioning that the specified error of 10^{-2} also motivates the choice of using six qubits for the circuits sampling iid Gaussian random variables. Recalling that the measurement of the state prepared by the six-qubit circuit amounts to sampling a discrete random variable supported on $2^6 = 64$ points, and that the PMF is a discrete approximation of a Gaussian PDF out to $\pm 5\sigma$, simple statistics tells us that after summing four such random variables (as is the final time-slice of the shorter of the two time-series that we consider) the corresponding $\pm 5\sigma$ will be supported by $\sqrt{4} \times 64 = 128$ points. Hence we use the ability to apply a function only to part of the support (i.e., the range of the support corresponding to $\pm 5\sigma$), as detailed in Section 8.7, and so it follows that, with 128 supported points (and hence a resolution of 1/128 = 0.0078 of the support), it is reasonable to calculate the RMSE to 1% = 0.01 of the total support.

11.3. **Benchmark results.** We give resource quantifications for both NISQ and fault-tolerant settings, and for each values are given for both the total algorithm and also the largest single circuit.

²⁶Pytket code for the circuits can also be supplied upon request.

²⁷Note that the barriers have all been set at approximately one standard deviation – and this is a conservative choice, in terms of the resource counts, as the barriers need to be checked at every time slice in this case.

²⁸As autocallables are already somewhat sophisticated instruments, combining different payoffs under different scenarios, it is doubtful that one would further define the payoff to be binary – however as it is possible to construct such calculations we report the binary payoff for completeness.

```
1 # USING STATE PREPARATION MODULE TO LOAD THE DISTRIBUTION P WITH 4 DIMENSIONS
      state_preparation = StatePreparation()
       {\tt state\_preparation\_parameters} \ = \ {\tt StatePreparationParameters} \ (
 3
               distribution=Normal(mean=0, variance=1), # Load the univariate unit Gaussian
 4
 5
               n_dimensions=4,
                                                                                                    # Four identical copies
              n_qubits=6,
                                                                                                    # Same number of qubits for each dimension
 6
              x1=-5,
                                                                                                   # Same xl for each dimension
 7
               delta=10/63,
                                                                                                   # Same delta for each dimension
 8
               ansatz="HWE".
                                                                                                    # (or "TN") Specify the ansatz/loader
 9
               n_layers=6,
                                                                                                    # Same number of ansatz layers for each dimension
10
               )
11
      distribution_circuit = state_preparation.get_circuit(state_preparation_parameters)
       # Get the distribution circuit for the multi-variate normal (to be used in QMCI)
13
14
      # USING ENHANCED-P BUILDER TO ENHANCE THE LOADED STATE
15
      operations = [Sum(4, 1), Sum(5, 2), Sum(6, 3)]
16
       # The Brownian motion is prepared in dimensions 4,5,6,7
17
18
       thresholds =
19
           [Threshold(dimension=4, value=log(1.1), type=BoundType.Upper), # Knock-out condition, 1st time-slice
20
            Threshold(dimension=5, value=log(1.1), type=BoundType.Upper), # Knock-out condition, 2nd time-slice
21
             \label{thm:condition} Threshold (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \quad \# \ \textit{Knock-out condition}, \ \textit{3rd time-slice} \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Threshold} (\texttt{dimension=6}, \ value= \log (\texttt{1.1}), \ type= \texttt{BoundType.Upper}), \\ \text{Thres
22
             Threshold(dimension=7, value=log(1.1), type=BoundType.Upper), # Knock-out condition, 4th time-slice
23
             Threshold(dimension=7, value=log(1.05), type=BoundType.Lower)] # Check strike price, 4th time-slice
24
25
      esop = ESOP(products=[[(0, True), (1, True), (2, True), (3, True), (4, True)]])
26
27
       # If the knock-out conditions haven't occurred AND the price is above the strike price
28
       # USING MONTE-CARLO INTEGRATION MODULE TO RUN QMCI
29
       amplitude_estimator = AmplitudeEstimator(type="MLQAE") # Specify type of amplitude estimator
30
31
       function_applied = Exponential()
                                                                          # In return space -- use exponential
32
       function_applied.flag_shift = 1.05 # Re-base final time-slice support by subtracting strike price
33
34
      monte_carlo_integration = MonteCarloIntegrationBuilder.build(
35
                               distribution_circuit=distribution_circuit,
36
37
                               dimension=7, # Integrate over the last dimension (i.e. the final time slice)
38
                               amplitude_estimator=amplitude_estimator,
                               function_applied=function_applied,
39
                               operations=operations,
40
                               thresholds=thresholds,
41
                               esop = esop,
42
                               condition=5, # Conditional expectation
43
                               resource_only=True, # Run in resource mode
44
                       ) # Build Monte Carlo Integrator
45
46
      target_rmse = 1.04E-2 # Set desired RMSE for estimate
47
48 monte_carlo_integration_result = monte_carlo_integration.estimate_result(target_rmse=target_rmse)
49 resource_estimates = monte_carlo_integration_result.resource_estimates # Get resource_estimates
```

FIGURE 33. Pseudocode for estimating the payoff of a four time-slice knock-out call option. Note that the indicator qubits are implicitly indexed (starting at 0) in order of creation, hence indicator qubits 0-4 are the threshold conditions in L20-24, and indicator qubit 5 is the esop in L26 – and it is upon this that the expectation is conditioned in L43.

Instrument	Payoff	Qubits	Gate count	CNOT count	Gate depth	CNOT depth
Barrier (4 slices)	Value Binary	119 117	$4.95 \times 10^{6} \\ 1.84 \times 10^{6}$	$2.44 \times 10^{6} \\ 8.98 \times 10^{5}$	$2.42 \times 10^{6} \\ 8.63 \times 10^{5}$	1.30×10^6 4.56×10^5
Barrier (8 slices)	Value Binary	307 305	1.30×10^{7} 4.98×10^{6}	$6.44 \times 10^6 \\ 2.46 \times 10^6$	5.15×10^6 1.96×10^6	$\begin{array}{c} 2.71 \times 10^{6} \\ 1.02 \times 10^{6} \end{array}$
Look-back (4 slices)	Value Binary	167 165	$9.26 \times 10^{6} \\ 4.20 \times 10^{6}$	$4.62 \times 10^6 \\ 2.09 \times 10^6$	$4.91 \times 10^{6} \\ 2.24 \times 10^{6}$	$\begin{array}{c} 2.72 \times 10^{6} \\ 1.23 \times 10^{6} \end{array}$
Look-back (8 slices)	Value Binary	439 437	2.45×10^7 1.12×10^7	$\begin{array}{c} 1.22 \times 10^{7} \\ 5.57 \times 10^{6} \end{array}$	$\begin{array}{c} 1.25 \times 10^{7} \\ 5.76 \times 10^{6} \end{array}$	6.90×10^6 3.15×10^6
Autocallable (4 slices)	Value Binary	123 121	1.07×10^7 5.93×10^6	5.27×10^6 2.89×10^6	$5.41 \times 10^{6} \\ 3.00 \times 10^{6}$	$\begin{array}{c} 2.89 \times 10^{6} \\ 1.59 \times 10^{6} \end{array}$
Autocallable (8 slices)	Value Binary	315 313	7.76×10^7 2.87×10^7	3.83×10^7 1.41×10^7	3.22×10^7 1.21×10^7	$1.68 \times 10^{7} \\ 6.33 \times 10^{6}$

(A)

Instrument	Payoff	Qubits	Gate count	CNOT count	Gate depth	CNOT depth
Barrier (4 slices)	Value Binary	119 117	8.37×10^4 3.10×10^4	$4.13 \times 10^4 \\ 1.52 \times 10^4$	4.14×10^4 1.48×10^4	$2.22 \times 10^4 \\ 7.80 \times 10^3$
Barrier (8 slices)	Value Binary	307 305	2.20×10^5 8.41×10^4	$1.09 \times 10^5 \\ 4.16 \times 10^4$	8.87×10^4 3.37×10^4	4.67×10^4 1.76×10^4
Look-back (4 slices)	Value Binary	167 165	9.50×10^4 7.07×10^4	4.75×10^4 3.52×10^4	5.10×10^4 3.80×10^4	2.82×10^4 2.08×10^4
Look-back (8 slices)	Value Binary	439 437	2.49×10^5 1.88×10^5	$\begin{array}{c} 1.25 \times 10^5 \\ 9.40 \times 10^4 \end{array}$	$\begin{array}{c} 1.29 \times 10^5 \\ 9.76 \times 10^4 \end{array}$	7.11×10^4 5.34×10^4
Autocallable (4 slices)	Value Binary	123 121	$\begin{array}{c} 1.14 \times 10^5 \\ 5.60 \times 10^4 \end{array}$	5.64×10^4 2.75×10^4	5.77×10^4 2.83×10^4	3.08×10^4 1.49×10^4
Autocallable (8 slices)	Value Binary	315 313	5.29×10^5 2.63×10^5	$2.62 \times 10^5 \\ 1.30 \times 10^5$	$2.35 \times 10^{5} \\ 1.19 \times 10^{5}$	$1.22 \times 10^5 \\ 6.17 \times 10^4$

(B)

Table 8. NISQ resource quantification for the benchmark problems: (A) total; (B) largest circuit.

			To	tal	Largest circuit		
Instrument	Payoff	Qubits	T count	T depth	T count	T depth	
Barrier (4 slices)	Value Binary	119 117	7.27×10^{7} 1.46×10^{7}	3.03×10^7 1.04×10^6	$1.18 \times 10^{6} \\ 2.72 \times 10^{5}$	4.91×10^5 1.96×10^4	
Barrier (8 slices)	Value Binary	307 305	1.19×10^8 3.05×10^7	3.55×10^{7} 1.55×10^{6}	1.93×10^6 5.68×10^5	5.76×10^5 2.93×10^4	
Look-back (4 slices)	Value Binary	167 165	8.72×10^{7} 2.10×10^{7}	3.52×10^{7} 2.60×10^{6}	8.22×10^5 3.91×10^5	3.32×10^5 4.86×10^4	
Look-back (8 slices)	Value Binary	439 437	1.64×10^8 4.80×10^7	5.01×10^7 5.72×10^6	$\begin{array}{c} 1.63 \times 10^{6} \\ 8.95 \times 10^{5} \end{array}$	5.00×10^5 1.07×10^5	
Autocallable (4 slices)	Value Binary	123 121	1.38×10^8 5.39×10^7	4.79×10^{7} 4.66×10^{6}	$1.53 \times 10^{6} \\ 4.10 \times 10^{5}$	6.40×10^5 3.08×10^4	
Autocallable (8 slices)	Value Binary	315 313	7.54×10^8 2.27×10^8	$1.10 \times 10^8 \\ 1.32 \times 10^7$	$4.71 \times 10^{6} \\ 1.39 \times 10^{6}$	$1.26 \times 10^{6} \\ 7.84 \times 10^{4}$	

Table 9. Fault-tolerant resource quantification for the benchmark problems.

The results are shown in Tables 8 and 9, and a first observation is that the number of qubits needed for these calculations is not excessive. It can be seen that non-trivial QMCI calculations, inspired by financial problems can be performed with total qubit number in the 100-500 range, and whilst these numbers will increase to account for the discrepancy between the actual time-series model and its discrete approximation and to represent a more finely time-sliced discretisation of the financial time-series, this increase will not be exorbitant. Specifically, the qubit number will grow only logarithmically with the reciprocal of the target RMSE and quasi-linearly with the number of time-steps (when working in return space). Furthermore, building the operator Q requires a large multi-controlled-NOT gates, and at present we decompose this in a manner that is extremely ancilla-hungry – approximately half of the total number of qubits are ancillas – and there is room to vastly reduce the ancilla count with only a modest increase to the total gate count.

Thus our attention turns to the number of operations, and the question of whether physical or error-corrected, logical qubits will be required. We can see from Table 8 that in all likelihood it will be the latter for the eight time-slice benchmarks. However, for the four time-slice benchmarks it is foreseeable that NISQ execution may be possible. Consider, for instance, the four time-slice barrier option, for which the largest circuit contains 1.44×10^4 CNOT gates – a simple rule-of-thumb calculation suggests that about $1 - 1/(1.52 \times 10^4) = 99.99\%$ two-qubit gate fidelity would therefore be required.

Note that this estimation, whilst somewhat useful, implies that we should run the circuit to a depth where one error is expected (and so can expect to obtain an error-free output with only a small number of repeats)²⁹, however this does not really capture how noise or QAE works. In particular, as discussed in Section 3.4, we can use noise-aware QAE to run much deeper QMCI

²⁹Note the precise conversion of fidelity to error-rate depends on the noise model.

circuits than one would expect from such a rule-of-thumb. Moreover, we can reasonably expect future circuit optimisation to reduce the CNOT gate count considerably (especially noting that no special optimisation of the CNOT-heavy enhanced P-builder circuits has been undertaken for these benchmarks); and it may be possible to define a slightly further simplified version of the problem, without losing its essential nature. Accordingly, it is entirely reasonable to speculate that a future quantum computer with ~ 100 qubits and two-qubit gate fidelity $\sim 99.99\%$ should be capable of running some simple, but not trivial financial QMCI calculations. However, whilst such a putative future quantum computer may be able to obtain an advantage in sample complexity for a non-trivial financial Monte Carlo integral – which would itself constitute a valuable outcome – it is doubtful that it would make practical sense to price such an option on a quantum computer, as we discuss in more detail in Section 12.

Turning then to the resource counts for fault-tolerant circuit execution, we should first note that T-gate counts are typically high for fault-tolerant circuit execution, largely owing to relative paucity of research on optimised synthesis into fault-tolerant gate-sets meaning that long-known conservative upper-bounds must be relied on, as was discussed (along with other reasons that the fault-tolerant resource quantification is likely to come down significantly) in Section 10.3. For these reasons, the absolute values in Table 9 are of relatively little meaning, apart from in the relative sense – that they give a target to aim for in the reduction of fault-tolerant resources.

12. Discussion

This paper has set out the full technical details of Quantinuum's quantum Monte Carlo integration engine, including our statistically robust QAE algorithm, and general framework for building financial derivative and risk calculations. We have also proposed a set of simple benchmarks, inspired by (slightly) simplified versions of MCI calculations that are ubiquitous in mathematical finance. Whilst these benchmarks clearly indicate that there is some significant further hardware scaling required for quantum advantage, we believe that the QMCI engine will provide utility well ahead of quantum advantage, namely in providing a research and development tool.

- 12.1. The QMCI engine as an R&D tool. The QMCI engine allows users to try out different means of encoding mathematical finance calculations in quantum circuits. Notably, users will be able to test out design choices about data-loading i.e., constructing quantum circuits that sample from financial time-series (potentially for multiple correlated assets) with resource mode quantifying the relative merits of each. Related questions about how finely to slice up the time-series and whether to work in return or price space can also be probed in this way. It should also be noted that the modular design of the QMCI engine is a deliberate choice, and serves to future proof it, as we can easily add in "full-scale" components like quantum arithmetic as hardware matures.
- 12.2. From theoretical quantum advantage to useful quantum advantage. In the above analysis, we have largely focused comparing classical and quantum sample complexity. This is a natural comparison to make in theoretical analysis, and owing to Ref. [27] we know that a quantum sample is at least as easy to prepare as a classical sample so we can assert that a sample complexity advantage automatically translates into a computational complexity advantage. However, simply beating the classical MCI in terms of sample or even computational complexity does not guarantee the automatic adoption of the quantum algorithm, once quantum computers scale to such a size that they can execute the requisite circuits and hence realise the theoretical advantage. Instead there are a whole host of factors that must be taken into consideration.

A first thing to note on this topic is that both classical and quantum MCI (using QPE-free QAE) enjoy the possibility of massive parallelisation – however such a possibility is already a reality in the world of classical computing, but in the case of quantum requires scaling of quantum hardware from a small set of laboratory experiments, to a fully-fledged industry (as well as the more widely-discussed scaling of the number of qubits in each device). We may summarise the factors outside of our (direct) control which will determine when it is beneficial to run QMCI rather than MCI:

- (1) (as above)the classical and quantum parallelisation factors which we assume to subsume the economic cost of using multiple quantum / classical cores as well as the basic question of what is available;
- (2) the clock speeds for the classical and quantum hardware;
- (3) the error-correction overhead (if applicable) for the quantum algorithm.

These all depend on a host of factors that could radically swing the time to quantum advantage one way or the other, but as expounded in Ref. [88] there is reason to be optimistic (or at least not unduly pessimistic) about the timeline to useful quantum advantage in QMCI. Indeed, Chakrabarti et al make a valiant effort to translate the timeline to quantum advantage in QMCI for finance into quantities such as actual clock-rates and the like [13]. Our view is that at this stage it is too early to do so, and it is more meaningful to give rigorous and objective comparisons between QMCI and MCI in terms of somewhat abstract quantities such as gate and sample counts; rather than giving extremely speculative comparisons between "real-world" quantities such as clock-speeds and parallelisation factors.

We do, however, know enough about the source of quantum advantage in QMCI to predict which types of calculations will see the greatest benefits. Notably, as QMCI offers a quadratic improvement in precision with the number of samples relative to classical MCI, it follows we must focus on applications where a sufficiently small error (RMSE) is demanded that QMCI indeed outperforms classical MCI, even when factors such as parallelisation and clock speed are taken into account. In finance, such a setting comes up on (at least) two important occasions. Firstly, when pricing derivatives, whilst estimation of the expected payoff itself (as exemplified in the benchmarks in Section 11) may in many instances not be too onerous to accomplish classically, it is often the case that estimating the (mathematical) derivative of the price with respect to various factors (quantities known colloquially as the *Greeks* in quantitative finance – as they are typically denoted by Greek letters) is something that would benefit from computational enhancement, as quantum computing promises to deliver. This is because, in simple terms, the Greeks can be computed by taking some finite difference approximation of the derivative, and the accuracy that can be obtained in the point estimates of the payoff therefore directly affects the minimum separation possible to compute the finite difference estimate. (Note that there are tricks to partially mitigate some of the computational difficulties in computing derivatives by finite differences, such as using the same random seed for each point estimate – and it has been shown that QMCI can benefit from these too [18]: in time we will supplement the QMCI engine with methods of this type for efficient calculation of the Greeks.) Secondly, precision is always critical when estimating financial risk quantities such as VaR and CVaR. As these calculations concern whole portfolios (and so the financial time-series for many assets therefore need to be encoded into qubit registers) it follows that the quantum resource requirements will be correspondingly higher than for (for example) derivative instruments on a single or small number of underlying. For this reason, examples risk calculations have been used sparingly throughout this paper, however, in the fullness of time, it is perfectly foreseeable that the most dramatic impact of quantum computing on mathematical finance will be in portfolio risk quantification.

Another reason to be optimistic about the outlook for QMCI in finance is that, whilst it has been shown that the computational complexity of sampling is never worse quantumly than classically [27], the converse does not hold. In contrived instances, should the random process in question be sampling an IQP circuit then, up to widely held complexity conjectures, the computational complexity of classical sampling is exponentially larger than its quantum counterpart [89]. For real world data, such as financial time-series, it is not necessarily the case that there will be such a spectacular asymptotic advantage in quantum over classical sampling, however it is reasonable to suppose that the additional functionality that quantum operations offer when navigating through Hilbert space to prepare samples may offer significant reductions in circuit depth in practice. That is, the construction in Ref. [27] uses a reversible circuit with a single bank of Hadamards followed by gates from the set $\{X, \text{ CNOT}, \text{ Toffoli}\}$ to construct P, however it is highly plausible that more "natively quantum" sampling circuits, exploiting a quantum-universal gateset, may prepare the desired samples in fewer operations, hence representing an extremely welcome complementary (to the quadratic speed-up in QAE) advantage.

With these observations in mind, we can draw up a check-list for applications that are likely to enjoy a useful quantum advantage by deploying QMCI:

- (1) MCI is the best technique classically;
- (2) precision is important: specifically, more precision that the current state-of-the-art would be valuable, but is hard to obtain by classical computational means;
- (3) there is an additional, complementary advantage in loading the random process as a quantum state.

12.3. Further applications of the QMCI engine. Whilst we have focused on mathematical finance in this paper, it is of course the case that the potential applications of QMCI are more widespread than this. Myriad problems across business, supply chain / logistics, energy, operations and data-intensive science have computational workflows that rely heavily on MCI, and in principle each of these can be enhanced by QMCI. Broadly speaking, such applications can be broken down into (i) estimation and forecasting, to which the QMCI engine can be applied in its present form, with only minor modifications; and (ii) numerical integration – where the randomness is merely a device for computing high dimensional integrals. In the latter case, the integrand must be decomposed into the product of a first term that itself integrates to value one (and hence may be thought of as a probability distribution) and can be encoded in a quantum circuit; and a second term that depends on only one variable, and can be expressed as a periodic piecewise function (This decomposition essentially amounts to importance sampling in the classical setting.). A prominent example of computational-intensive numerical integration that has garnered some attention as a potential application of QMCI are the high-dimensional integrals in high-energy physics [90].

Part V. End matter and appendices

ACKNOWLEDGMENTS

The authors thank Sam Duffield, Konstantinos Meichanetzidis, and Matthias Rosenkranz for kindly reviewing an earlier version of this article. We also thank Edwin Agnew and Jose Gefaell for their work that helped form the background to this paper, and for all of the fruitful discussions we had along the way.

References

- [1] Y. Suzuki, S. Uno, R. Raymond, T. Tanaka, T. Onodera, and N. Yamamoto, "Amplitude estimation without phase estimation," *Quantum Information Processing*, vol. 19, no. 2, Jan 2020. [Online]. Available: http://dx.doi.org/10.1007/s11128-019-2565-2
- [2] D. Grinko, J. Gacon, C. Zoufal, and S. Woerner, "Iterative quantum amplitude estimation," mar 2021. [Online]. Available: https://doi.org/10.1038/s41534-021-00379-1
- [3] S. Aaronson and P. Rall, "Quantum approximate counting, simplified," Symposium on Simplicity in Algorithms, p. 24–32, Jan 2020. [Online]. Available: http://dx.doi.org/10.1137/1.9781611976014.5
- [4] K. Nakaji, "Faster amplitude estimation," Quantum Information and Computation, vol. 20, no. 13&14, pp. 1109–1123, nov 2020. [Online]. Available: https://doi.org/10.26421/qic20.13-14-2
- [5] S. Herbert, "Quantum monte carlo integration: The full advantage in minimal circuit depth," p. 823, sep 2022. [Online]. Available: https://doi.org/10.22331/q-2022-09-29-823
- [6] P. Rebentrost, B. Gupt, and T. R. Bromley, "Quantum computational finance: Monte Carlo pricing of financial derivatives," *Physical Review A*, vol. 98, no. 2, Aug 2018. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.98.022321
- [7] N. Stamatopoulos, D. J. Egger, Y. Sun, C. Zoufal, R. Iten, N. Shen, and S. Woerner, "Option pricing using quantum computers," *Quantum*, vol. 4, p. 291, Jul 2020. [Online]. Available: http://dx.doi.org/10.22331/q-2020-07-06-291
- [8] S. Woerner and D. J. Egger, "Quantum risk analysis," npj Quantum Information, vol. 5, no. 1, Feb 2019. [Online]. Available: http://dx.doi.org/10.1038/s41534-019-0130-6
- [9] A. Bouland, W. van Dam, H. Joorati, I. Kerenidis, and A. Prakash, "Prospects and challenges of quantum finance," 2020. [Online]. Available: https://doi.org/10.48550/arxiv.2011.06492
- [10] R. Orús, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," Reviews in Physics, vol. 4, p. 100028, 2019. [Online]. Available: https://doi.org/10.1016/j.revip.2019.100028
- [11] D. J. Egger, R. G. Gutierrez, J. Mestre, and S. Woerner, "Credit risk analysis using quantum computers," *IEEE Transactions on Computers*, vol. 70, no. 12, pp. 2136–2145, dec 2021. [Online]. Available: https://doi.org/10.1109/TC.2020.3038063
- [12] K. Kaneko, K. Miyamoto, N. Takeda, and K. Yoshino, "Quantum pricing with a smile: Implementation of local volatility model on quantum computer," 2020. [Online]. Available: https://doi.org/10.48550/arxiv.2007.01467
- [13] S. Chakrabarti, R. Krishnakumar, G. Mazzola, N. Stamatopoulos, S. Woerner, and W. J. Zeng, "A threshold for quantum advantage in derivative pricing," *Quantum*, vol. 5, p. 463, Jun 2021. [Online]. Available: https://doi.org/10.22331\/q-2021-06-01-463
- [14] P. Rebentrost and S. Lloyd, "Quantum computational finance: quantum algorithm for portfolio optimization," 2018. [Online]. Available: https://doi.org/10.48550/arxiv.1811.03975
- [15] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, and E. Yndurain, "Quantum computing for finance: State-of-the-art and future prospects," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–24, 2020. [Online]. Available: https://doi.org/10.1109/TQE.2020.3030314
- [16] D. An, N. Linden, J.-P. Liu, A. Montanaro, C. Shao, and J. Wang, "Quantum-accelerated multilevel monte carlo methods for stochastic differential equations in mathematical finance," *Quantum*, vol. 5, p. 481, jun 2021. [Online]. Available: https://doi.org/10.22331/q-2021-06-24-481
- [17] D. Herman, C. Googin, X. Liu, A. Galda, I. Safro, Y. Sun, M. Pistoia, and Y. Alexeev, "A survey of quantum computing for finance," 2022. [Online]. Available: https://doi.org/10.48550/arxiv.2201.02773
- [18] N. Stamatopoulos, G. Mazzola, S. Woerner, and W. J. Zeng, "Towards quantum advantage in financial market risk using quantum gradient algorithms," *Quantum*, vol. 6, p. 770, jul 2022. [Online]. Available: https://doi.org/10.22331/q-2022-07-20-770
- [19] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, 2010.
- [20] P. Wilmott, Paul Wilmott Introduces Quantitative Finance, 2nd ed. USA: Wiley-Interscience, 2007.
- [21] R. E. Caflisch, "Monte Carlo and quasi-Monte Carlo methods," Acta Numerica, vol. 7, p. 1–49, 1998. [Online]. Available: https://www.cambridge.org/core/journals/acta-numerica/article/abs/monte-carlo-and-quasimonte-carlo-methods/FE7C779B350CFEA45DB2A4CCB2DA9B5C
- [22] R. Leluc, F. Portier, A. Zhuman, and J. Segers, "Speeding up Monte Carlo integration: Control neighbors for optimal convergence," Université catholique de Louvain, Institute of Statistics, Biostatistics and Actuarial Sciences (ISBA), LIDAM Discussion Papers ISBA 2023019, 2023. [Online]. Available: https://EconPapers.repec.org/RePEc:aiz:louvad:2023019
- [23] N. Chopin and M. Gerber, "Higher-order stochastic integration through cubic stratification," arXiv:2210.01554, 2022. [Online]. Available: https://arxiv.org/abs/2210.01554

- [24] J. Datta and N. G. Polson, "Quantile importance sampling," 2023. [Online]. Available: https://arxiv.org/abs/2305.03158
- [25] L. Grover and T. Rudolph, "Creating superpositions that correspond to efficiently integrable probability distributions," 2002. [Online]. Available: https://arxiv.org/abs/quant-ph/0208112
- [26] S. Herbert, "No quantum speedup with grover-rudolph state preparation for quantum monte carlo integration," Phys. Rev. E, vol. 103, p. 063302, Jun 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE. 103.063302
- [27] ——, "Every classical sampling circuit is a quantum sampling circuit," 2021. [Online]. Available: https://arxiv.org/abs/2109.04842
- [28] S. Herbert, R. Guichard, and D. Ng, "Noise-aware quantum amplitude estimation," 2021. [Online]. Available: https://arxiv.org/abs/2109.04840
- [29] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive-sums-of-products," 2001. [Online]. Available: https://pdxscholar.library.pdx.edu/ece_fac/195
- [30] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," 06 2000.
- [31] J. Wachter, "Rare events and financial markets," 2020. [Online]. Available: https://www.econstor.eu/bitstream/10419/234000/1/2020number1-2.pdf
- [32] A. Childs and N. Wiebe, "Hamiltonian simulation using linear combinations of unitary operations," Nov 2012. [Online]. Available: https://doi.org/10.26421/qic12.11-12
- [33] L. K. Grover, "A fast quantum mechanical algorithm for database search," in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219. [Online]. Available: https://doi.org/10.1145/237814.237866
- [34] P. Del Moral, "Nonlinear filtering: Interacting particle resolution," Comptes Rendus de l'Académie des Sciences Series I Mathematics, vol. 325, no. 6, pp. 653–658, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0764444297847787
- [35] T. J. DiCiccio and B. Efron, "Bootstrap confidence intervals," Statistical Science, vol. 11, no. 3, pp. 189 228, 1996. [Online]. Available: https://doi.org/10.1214/ss/1032280214
- [36] R. Barlow, "Asymmetric systematic errors," 2003. [Online]. Available: https://arxiv.org/abs/physics/0306138
- [37] E. Ryu, "Effects of skewness and kurtosis on normal-theory based maximum likelihood test statistic in multilevel structural equation modeling," *Behavior research methods*, vol. 43, pp. 1066–74, 06 2011. [Online]. Available: https://doi.org/10.3758%2Fs13428-011-0115-7
- [38] M. Amy, A. N. Glaudell, and N. J. Ross, "Number-theoretic characterizations of some restricted clifford+t circuits," *Quantum*, vol. 4, p. 252, apr 2020. [Online]. Available: https://doi.org/10.22331/q-2020-04-06-252
- [39] X. Qiang, T. Loke, A. Montanaro, K. Aungskunsiri, X. Zhou, J. L. O'Brien, J. B. Wang, and J. C. Matthews, "Efficient quantum walk on a quantum processor," *Nature communications*, vol. 7, no. 1, p. 11511, 2016. [Online]. Available: https://doi.org/10.1038%2Fncomms11511
- [40] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, "Exponential algorithmic speedup by a quantum walk," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, 2003, pp. 59–68. [Online]. Available: https://doi.org/10.1145%2F780542.780552
- [41] D. W. Berry and A. M. Childs, "Black-box hamiltonian simulation and unitary implementation," arXiv:0910.4157, 2009. [Online]. Available: https://arxiv.org/abs/0910.4157
- [42] W. T. Tutte and W. T. Tutte, Graph theory. Cambridge university press, 2001, vol. 21.
- [43] S. E. Venegas-Andraca, "Quantum walks: a comprehensive review," *Quantum Information Processing*, vol. 11, no. 5, pp. 1015–1106, 2012. [Online]. Available: https://doi.org/10.1007%2Fs11128-012-0432-5
- [44] M. Christandl, N. Datta, T. C. Dorlas, A. Ekert, A. Kay, and A. J. Landahl, "Perfect transfer of arbitrary states in quantum spin networks," *Physical Review A*, vol. 71, no. 3, p. 032312, 2005. [Online]. Available: https://doi.org/10.1103/PhysRevA.71.032312
- [45] C. Godsil, "State transfer on graphs," Discrete Mathematics, vol. 312, no. 1, pp. 129–147, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0012365X11002974
- [46] Y. Ge, B. Greenberg, O. Perez, and C. Tamon, "Perfect state transfer, graph products and equitable partitions," *International Journal of Quantum Information*, vol. 9, no. 03, pp. 823–842, 2011. [Online]. Available: https://doi.org/10.1142%2Fs0219749911007472
- [47] R. Bachman, E. Fredette, J. Fuller, M. Landry, M. Opperman, C. Tamon, and A. Tollefson, "Perfect state transfer on quotient graphs," arXiv:1108.0339, 2011. [Online]. Available: https://arxiv.org/abs/1108.0339
- [48] H. Risken, Fokker-planck equation. Springer, 1996. [Online]. Available: https://doi.org/10.1007% 2F978-3-642-61544-3_4
- $[49]\,$ S. M. Ross, $Stochastic\ processes.\,\,$ John Wiley & Sons, 1995.
- [50] S. Wiesner, "Simulations of Many-Body Quantum Systems by a Quantum Computer," 1996. [Online]. Available: https://arxiv.org/abs/quant-ph/9603028

- [51] C. Zalka, "Simulating quantum systems on a quantum computer," Proc. Roy. Soc. Lond. A, vol. 454, pp. 313–322, 1998. [Online]. Available: http://doi.org/10.1098/rspa.1998.0162
- [52] S. K. Leyton and T. J. Osborne, "A quantum algorithm to solve nonlinear differential equations," 2008.
 [Online]. Available: https://arxiv.org/abs/0812.4423
- [53] Y. Cao, A. Papageorgiou, I. Petras, J. Traub, and S. Kais, "Quantum algorithm and circuit design solving the poisson equation," *New Journal of Physics*, vol. 15, no. 1, p. 013021, 2013. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/15/1/013021
- [54] D. W. Berry, "High-order quantum algorithm for solving linear differential equations," *Journal of Physics A: Mathematical and Theoretical*, vol. 47, no. 10, p. 105301, feb 2014. [Online]. Available: https://dx.doi.org/10.1088/1751-8113/47/10/105301
- [55] A. Montanaro and S. Pallister, "Quantum algorithms and the finite element method," *Phys. Rev. A*, vol. 93, p. 032324, Mar 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.93.032324
- [56] D. W. Berry, A. M. Childs, A. Ostrander, and G. Wang, "Quantum Algorithm for Linear Differential Equations with Exponentially Improved Dependence on Precision," Communications in Mathematical Physics, vol. 356, no. 3, pp. 1057–1081, 2017. [Online]. Available: https://doi.org/10.1007/s00220-017-3002-y
- [57] P. C. S. Costa, S. Jordan, and A. Ostrander, "Quantum algorithm for simulating the wave equation," *Phys. Rev. A*, vol. 99, p. 012323, Jan 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.99.012323
- [58] M. Lubasch, J. Joo, P. Moinier, M. Kiffner, and D. Jaksch, "Variational quantum algorithms for nonlinear problems," *Phys. Rev. A*, vol. 101, p. 010301, Jan 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.101.010301
- [59] A. M. Childs and J.-P. Liu, "Quantum Spectral Methods for Differential Equations," Communications in Mathematical Physics, vol. 375, no. 2, pp. 1427–1457, 2020. [Online]. Available: https://doi.org/10.1007/s00220-020-03699-z
- [60] S. Lloyd, G. D. Palma, C. Gokler, B. Kiani, Z.-W. Liu, M. Marvian, F. Tennie, and T. Palmer, "Quantum algorithm for nonlinear differential equations," 2020. [Online]. Available: https://arxiv.org/abs/2011.06571
- [61] A. M. Childs, J.-P. Liu, and A. Ostrander, "High-precision quantum algorithms for partial differential equations," Quantum, vol. 5, p. 574, Nov. 2021. [Online]. Available: https://doi.org/10.22331/q-2021-11-10-574
- [62] J.-P. Liu, H. Ø. Kolden, H. K. Krovi, N. F. Loureiro, K. Trivisa, and A. M. Childs, "Efficient quantum algorithm for dissipative nonlinear differential equations," *Proceedings of the National Academy of Sciences*, vol. 118, no. 35, p. e2026805118, 2021. [Online]. Available: https://doi.org/10.1073%2Fpnas.2026805118
- [63] A. J. Pool, A. D. Somoza, M. Lubasch, and B. Horstmann, "Solving Partial Differential Equations using a Quantum Computer," in 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), 2022, pp. 864–866. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9951265/
- [64] S. Jin, N. Liu, and Y. Yu, "Quantum simulation of partial differential equations via Schrodingerisation," 2022.
 [Online]. Available: https://arxiv.org/abs/2212.13969
- [65] —, "Quantum simulation of partial differential equations via Schrodingerisation: technical details," 2022. [Online]. Available: https://arxiv.org/abs/2212.14703
- [66] S. Lloyd, "Universal Quantum Simulators," Science, vol. 273, no. 5278, pp. 1073–1078, 1996. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.273.5278.1073
- [67] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum Algorithm for Linear Systems of Equations," Phys. Rev. Lett., vol. 103, p. 150502, Oct 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.103. 150502
- [68] A. Gnanasekaran, A. Surana, and T. Sahai, "Efficient Quantum Algorithms for Nonlinear Stochastic Dynamical Systems," 2023. [Online]. Available: https://arxiv.org/abs/2303.02463
- [69] M. Benedetti, M. Fiorentini, and M. Lubasch, "Hardware-efficient variational quantum algorithms for time evolution," *Phys. Rev. Res.*, vol. 3, p. 033083, Jul 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.3.033083
- [70] C. Mc Keever and M. Lubasch, "Classically optimized Hamiltonian simulation," *Phys. Rev. Res.*, vol. 5, p. 023146, Jun 2023. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.5.023146
- [71] Y. Kikuchi, C. M. Keever, L. Coopmans, M. Lubasch, and M. Benedetti, "Realization of quantum signal processing on a noisy quantum computer," 2023. [Online]. Available: https://arxiv.org/abs/2303.05533
- [72] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043001, 2019. [Online]. Available: https://doi.org/10.1088%2F2058-9565%2Fab4eb5
- [73] F. Verstraete, V. Murg, and J. I. Cirac, "Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems," *Advances in Physics*, vol. 57, no. 2, pp. 143–224, 03 2008. [Online]. Available: https://doi.org/10.1080/14789940801912366

- [74] R. Orús, "A practical introduction to tensor networks: Matrix product states and projected entangled pair states," *Annals of Physics*, vol. 349, pp. 117–158, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003491614001596
- [75] M. C. Bañuls, "Tensor Network Algorithms: A Route Map," Annual Review of Condensed Matter Physics, vol. 14, no. 1, pp. 173–191, 2023. [Online]. Available: https://doi.org/10.1146/annurev-conmatphys-040721-022705
- [76] F. Verstraete, J. J. García-Ripoll, and J. I. Cirac, "Matrix Product Density Operators: Simulation of Finite-Temperature and Dissipative Systems," *Phys. Rev. Lett.*, vol. 93, p. 207204, Nov 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.93.207204
- [77] M. Zwolak and G. Vidal, "Mixed-State Dynamics in One-Dimensional Quantum Lattice Systems: A Time-Dependent Superoperator Renormalization Algorithm," Phys. Rev. Lett., vol. 93, p. 207205, Nov 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.93.207205
- [78] J. B. Keller, "Closest Unitary, Orthogonal and Hermitian Operators to a Given Operator," *Mathematics Magazine*, vol. 48, no. 4, pp. 192–197, 1975. [Online]. Available: http://www.jstor.org/stable/2690338
- [79] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *nature*, vol. 549, no. 7671, pp. 242–246, 2017. [Online]. Available: https://doi.org/10.1038%2Fnature23879
- [80] S. McArdle, A. Gilyén, and M. Berta, "Quantum state preparation without coherent arithmetic," arXiv:2210.14892, 2022. [Online]. Available: https://arxiv.org/abs/2210.14892
- [81] A. G. Rattew, Y. Sun, P. Minssen, and M. Pistoia, "The efficient preparation of normal distributions in quantum registers," *Quantum*, vol. 5, p. 609, 2021. [Online]. Available: https://doi.org/10.22331/q-2021-12-23-609
- [82] D. Herman, C. Googin, X. Liu, A. Galda, I. Safro, Y. Sun, M. Pistoia, and Y. Alexeev, "A survey of quantum computing for finance," arXiv:2201.02773, 2022. [Online]. Available: https://arxiv.org/abs/2201.02773
- [83] A. Holmes and A. Y. Matsuura, "Efficient quantum circuits for accurate state preparation of smooth, differentiable functions," in 2020 IEEE International Conference on Quantum Computing and Engineering (QCE). IEEE, 2020, pp. 169–179. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9259933/
- [84] P. Gundlapalli and J. Lee, "Deterministic and entanglement-efficient preparation of amplitude-encoded quantum registers," *Physical Review Applied*, vol. 18, no. 2, p. 024013, 2022. [Online]. Available: https://doi.org/10.1103/PhysRevApplied.18.024013
- [85] C. Zoufal, A. Lucchi, and S. Woerner, "Quantum generative adversarial networks for learning and loading random distributions," npj Quantum Information, vol. 5, no. 1, p. 103, 2019. [Online]. Available: https://doi.org/10.1038%2Fs41534-019-0223-2
- [86] N. J. Ross and P. Selinger, "Optimal ancilla-free clifford+t approximation of z-rotations," Quantum Info. Comput., vol. 16, no. 11–12, p. 901–953, sep 2016.
- [87] S.-A. Wegner, "Lecture notes on high-dimensional data," 2021. [Online]. Available: https://arxiv.org/abs/2101. 05841
- [88] S. Herbert, "Quantum computing for data-centric engineering and science," *Data-Centric Engineering*, vol. 3, p. e36, 2022. [Online]. Available: https://www.cambridge.org/core/journals/data-centric-engineering/article/quantum-computing-for-datacentric-engineering-and-science/DBBE294A860DB71EA0C7B26D2DB4660B
- [89] M. J. Bremner, R. Jozsa, and D. J. Shepherd, "Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 467, no. 2126, pp. 459–472, aug 2010. [Online]. Available: https://doi.org/10.1098/rspa.2010.0301
- [90] G. Agliardi, M. Grossi, M. Pellen, and E. Prati, "Quantum integration of elementary particle processes," *Physics Letters B*, vol. 832, p. 137228, sep 2022. [Online]. Available: https://doi.org/10.1016%2Fj.physletb.2022.137228

APPENDIX A. QAE STATISTICAL ROBUSTNESS BENCHMARKS

This section includes the (absolute) bias (Fig. 34), RMSE (Fig. 35), (absolute) skewness (Fig. 36) and kurtosis (Fig. 37) of each form of QAE for each amplitude (over a finely-spaced grid covering the entire range).

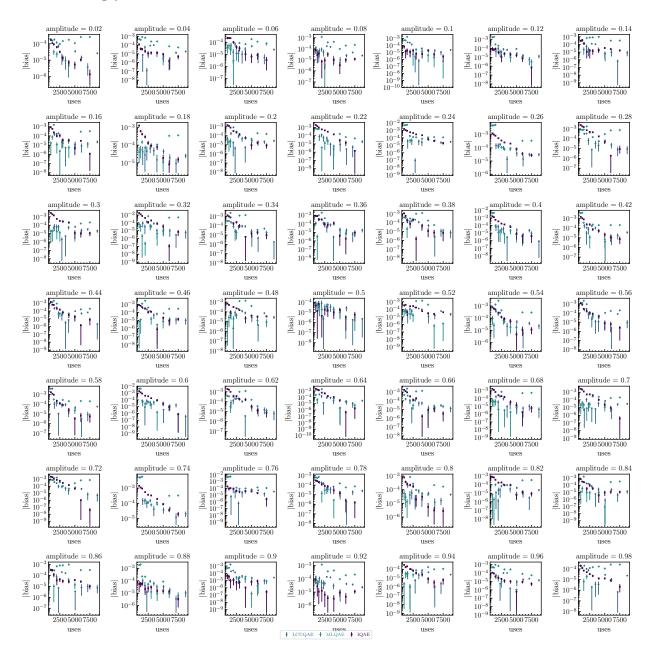


FIGURE 34. The bias, as found by numerical simulations for the various forms of QAE, for each of the 49 amplitudes. It can be seen that the bias is generally small for all amplitudes and all forms of QAE.

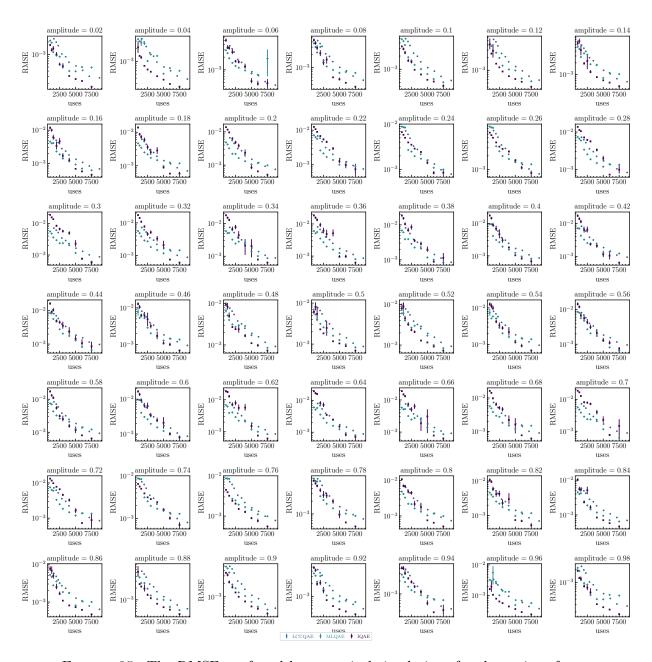


FIGURE 35. The RMSE, as found by numerical simulations for the various forms of QAE, for each of the 49 amplitudes. This is addressed extensively elsewhere, and included here merely for completeness.

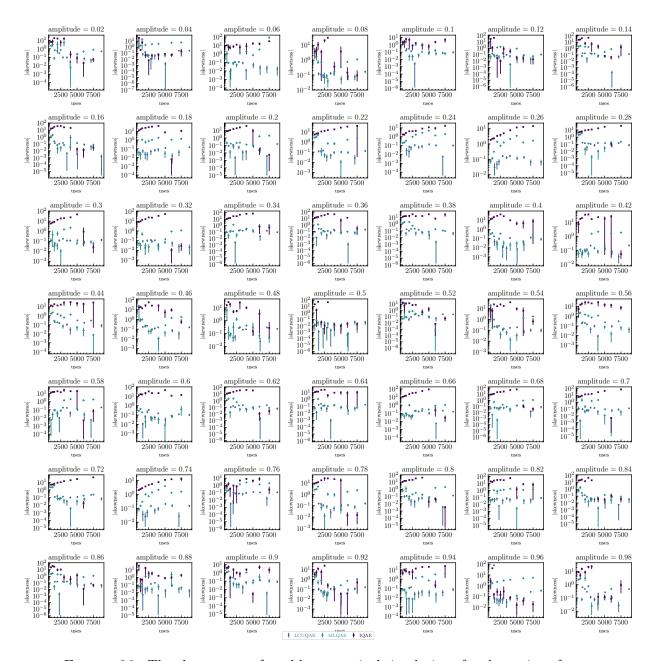


FIGURE 36. The skewness, as found by numerical simulations for the various forms of QAE, for each of the 49 amplitudes. It can be seen that LCU QAE generally has good skewness, whereas there are certain amplitudes for which MLQAE and IQAE have poor skewness.

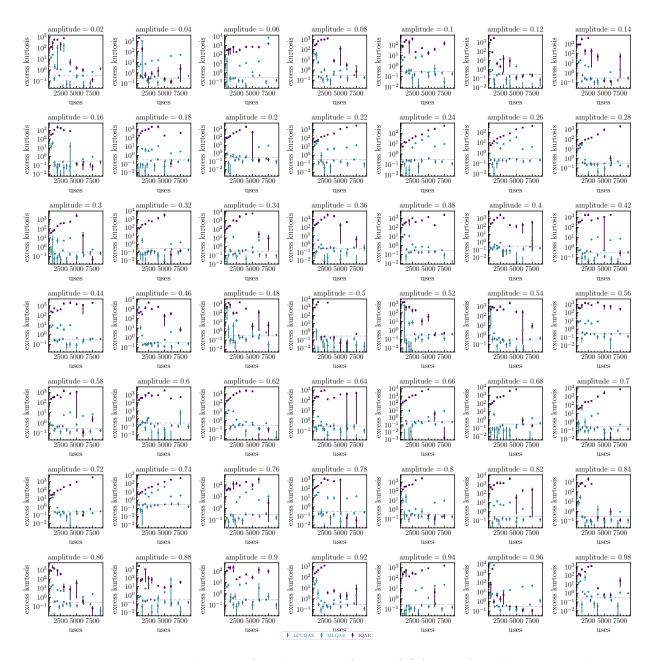


FIGURE 37. Excess kurtosis for the various forms of QAE, as found by numerical simulations for each of the 49 amplitudes. It can be seen that, aside from the odd outlier, LCU QAE has very low excess kurtosis, whereas neither of the other forms of QAE have good performance for all amplitudes.

APPENDIX B. RMSE CONVERGENCE PLOTS

This Appendix includes values from numerical simulation for RMSE at various values of number of uses, for each of the 49 amplitudes simulated. Lines of best fit are used to find numerical values of $c_{\rm QAE}$.



FIGURE 38. RMSE convergence for LCU QAE versus successful uses. In this case, the line of best fit gives the asymptotic value of $c_{\rm QAE}$, which is equal to 7.83 – as is the worst case which occurs at amplitude = 0.56.

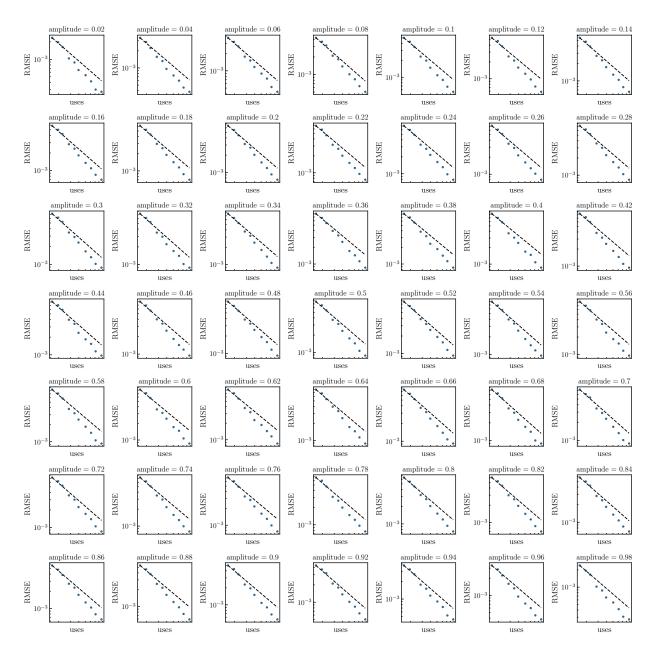


FIGURE 39. RMSE convergence for LCU QAE versus expected total uses. In this case, the upper-bounding line gives a conservative value of $c_{\rm QAE}$ that is appropriate for finite numbers of uses, and is equal to 13.3 – as is the worst case which occurs at amplitude = 0.58.



FIGURE 40. RMSE convergence for MLQAE versus total uses. In this case, the line of best fit gives a value of $c_{\rm QAE}$ which is equal to 8.02 – as is the worst case which occurs at amplitude = 0.26.

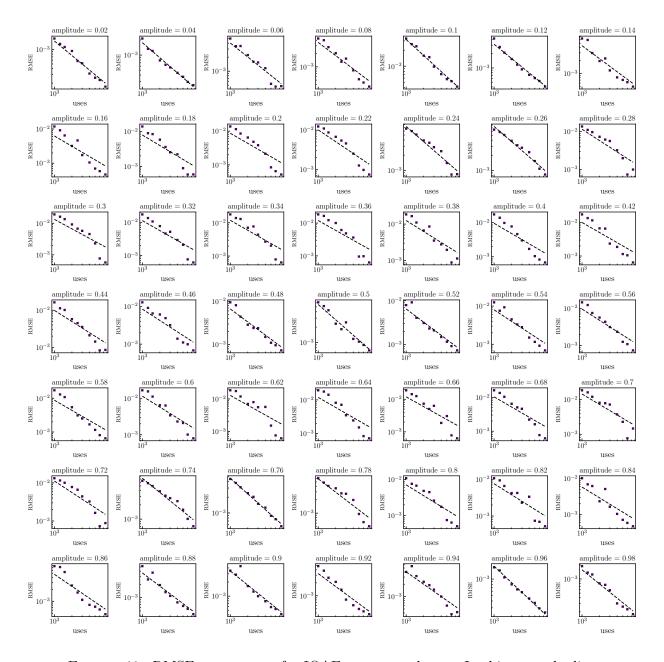


FIGURE 41. RMSE convergence for IQAE versus total uses. In this case, the line of best fit gives a value of c_{QAE} which is equal to 14.4 – as is the worst case which occurs at amplitude = 0.7.

APPENDIX C. CIRCUITS FOR BENCHMARKS

Below we give the circuits used in the benchmarks in Section 11.

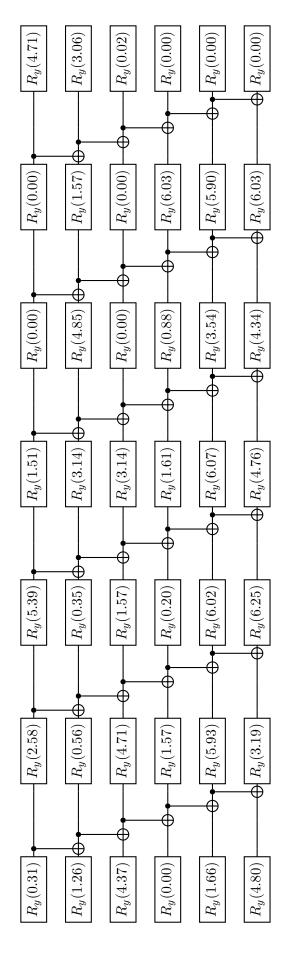


FIGURE 42. The 6-qubit circuit that prepares the 64-point PMF of a discrete approximation of a random variable and are specified up to global phases; $x_{\ell} = -5$ and $\Delta = 10/63$, such that the support out to $\pm 5\sigma$ is covered. Here the distributed as $\mathcal{N}(0,1)$, where all rotations are given in radians (based on the standard definition of a rotation operator) angles are "absolute" values – not given as multiples of π .

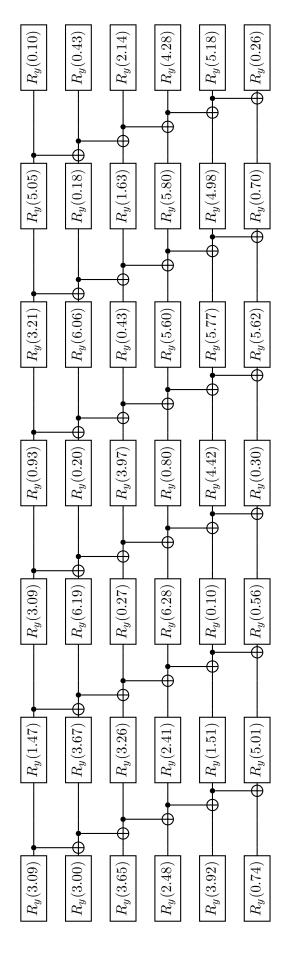


FIGURE 43. The 6-qubit circuit that prepares the 64-point PMF of a discrete approximation of a random variable distributed as $\mathcal{LN}(0,1/800)$, where all rotations are given in radians (based on the standard definition of a rotation operator) and are specified up to global phases; $x_{\ell} = 0.83$ and $\Delta = 0.01$, such that the support out to $\pm 5\sigma$ is covered. Here the angles are "absolute" values – not given as multiples of π .

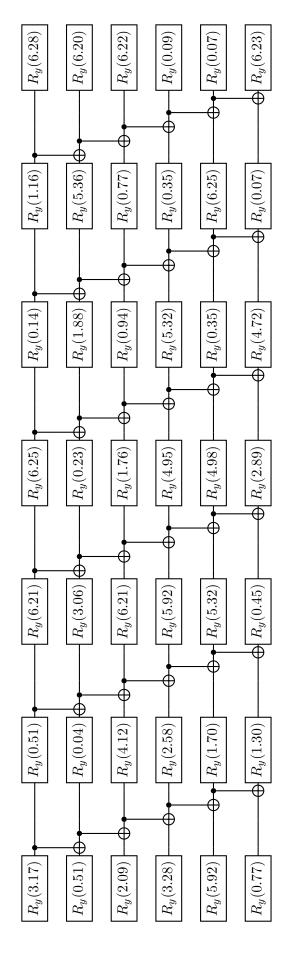


FIGURE 44. The 6-qubit circuit that prepares the 64-point PMF of a discrete approximation of a random variable distributed as $\mathcal{LN}(0,1/400)$, where all rotations are given in radians (based on the standard definition of a rotation operator) and are specified up to global phases; $x_{\ell} = 0.77$ and $\Delta = 0.01$, such that the support out to $\pm 5\sigma$ is covered. Here the angles are "absolute" values – not given as multiples of π .