Centro de Investigación en Tecnologías de la Información y Comunicaciones
Universidade da Coruña (UDC)

Design of the Benchmark Cases of the Suite

Lista de autores

Oluwatosin Esther Odubanjo
Diego Andrade Canosa
Juan Touriño Domínguez
María José Martín Santamaría
Basilio Fraguela Rodríguez

01/09/2023

Despregamento dunha infraestructura baseada en tecnoloxías cuánticas da información que permita impulsar a I+D+i en Galicia.

Apoiar a transición cara a una economía dixital.

Operación financiada pola Unión Europea, a través do FONDO EUROPEO DE DESENVOLVEMENTO REXIONAL (FEDER) como parte da resposta da Unión á pandemia da COVID-19

DATA	AUTHOR	CHANGES	VERSION
15 / 03 / 2023	Diego Andrade Canosa	Initial version and structure	0.1
20 / 03 / 2023	Diego Andrade Canosa	First complete initial version	1.0
21 / 03 / 2023	María José Martín Santamaría	Review of the document	1.1
22 / 03 / 2023	Basilio Fraguela Rodríguez	Review of the document	1.2
23 / 03 / 2023	Juan Touriño Domínguez	Review of the document	1.3

Table of contents

List of figures

List of tables

List of acronyms

QC	*Quantum Computing*
TNBS	*The NEASQC Benchmark Suite*
PL	*Probability Loading*
HHL	*Harrow-Hassidim-Lloyd*
PCA	*Principal Component Analysis*
PDF	*Probability Distribution Function*
AE	*Amplitude Estimation*
QPE	*Quantum Phase Estimation*
IQPE	*Iterative Quantum Phase Estimation*
QFT	*Quantum Fourier Transform*

1 Introduction

The Quantum Computing (QC) Benchmark Suite, defined within the framework of this contract, comprises a set of benchmark cases defined in this document. These cases are extracted from several domains where Quantum Computing (QC) has been probed to be useful.

The QC benchmark suite does not address the topic of "quantum supremacy", i.e., finding an algorithm that can be solved on a quantum computer but is difficult or almost impossible to solve in a classical computer in a reasonable time.

Each case has four components: the selection criteria, the kernel definition, the test case, and the benchmark execution procedure. We define now separately each of these parts.

- **Selection Criteria**: it is the justification of why the case is representative of a task that is commonly performed in QC and suitable to be part of the suite. The cases that compose the suite meet the following characteristics:
  - They are based on a quantum algorithm or kernel that is common to several use cases and representative of the needs of other algorithms of the same family.
  - They scale with the number of qubits up to a reasonable number. Certainly, many of the benchmarks are limited by the classical capacity for pre-processing and post-processing information.
  - They are defined at high level, i.e., they must be agnostic of the quantum computer architecture, programming language, etc.
- **Kernel**: it is a core quantum subroutine or step which may be implemented using different procedures or algorithmic approaches. Because of this, a Kernel is described using a high-level mathematical or procedural definition. Examples are the *Quantum Fourier Transformation*, the loading of an initial quantum state in a quantum circuit, etc.
- **Benchmark**: it is each one of the use cases in which the TNBS is divided. This document contains a description of one of them. Each benchmark is composed of a description of the Kernel and its corresponding Benchmark Test Case.
- **Benchmark Test Case**: it is a particular problem that involves the execution of the Kernel. The output of this Test Case must be verifiable analytically or through a classical simulation. The Test Case is used for evaluating the performance of a Quantum platform that executes the Kernel. For example, the loading of a specific statistical distribution into a quantum circuit (**Benchmark Test Case**) is used to evaluate the performance of a platform for loading an initial quantum state in a quantum circuit (**Kernel**).
- **Benchmark execution procedure**: it is the detailed definition of the procedure to execute each benchmark, collect the evaluation metrics and verified the accuracy of the output.

This document defines three benchmark cases: Probability Loading (Section 2), Quantum Amplitude Estimation (Section 3), and Quantum Phase Estimation (Section 4).

2 Benchmark for Probability Loading Algorithms

This section describes the T1: Probability Loading (PL) benchmark.

In Section 2.1, we justify its selection, and in Section 2.2, we describe its form as indicated by the suite. Each kernel is associated with a Benchmark Test Case, which we detail in Section 2.3.

2.1 Kernel selection justification

The **PL kernel** is common to many different quantum algorithms like the **HHL** [8], quantum **PCA** [12], quantum amplitude estimation algorithms [3] etc. This initialization step is, usually, a very demanding part of any quantum algorithm because its number of operations typically scales as $\sim 2^n$, being n the number of qubits to be initialized. In addition, this kernel meets the three main requirements from the QC benchmark methodology:

1. The kernel can be described mathematically or procedurally. Using this description, a standalone circuit can be generated (see section 2.2).
2. The kernel can be defined for different number of qubits.
3. The output can be verified with a classical computation (in the proposed **Benchmark Test Case**, see section 2.3, the result is known *a priori*)

For all these reasons, the **PL kernel** is a good candidate for the **TNBS**.

2.2 Kernel Description

The **PL kernel** can be defined, mathematically as follows:

Let $\mathbf{V}$ be a normalised vector of complex values:

$$\mathbf{V} = \{v_0, v_1, \cdot, v_{2^n-1}\}, v_i \in \mathbb{C} \tag{1}$$

such that

$$\sum_{i=0}^{2^n} |v_i|^2 = 1 \tag{2}$$

The main task of the **PL kernel** is the creation of an operator $\mathbf{U}$, from the normalised vector $\mathbf{V}$, which satisfies equation (3):

$$\mathbf{U}|0\rangle_n = \sum_{i=0}^{2^n-1} v_i|i\rangle_n \tag{3}$$

This procedure can be used for the loading of a probability density function (**PDF**). In this case, equation (1) can be reformulated as (4)

$$\mathbf{P} = \{p_0, p_1, \cdot, p_{2^n-1}\}, p_i \in [0, 1] \tag{4}$$

Equation (2) must be transformed into equation (5)

$$\sum_{i=0}^{2^n} |p_i|^2 = 1 \tag{5}$$

And (3) can be written as (6):

$$\mathbf{U}|0\rangle_n = \sum_{i=0}^{2^n-1} \sqrt{p_i}|i\rangle_n \tag{6}$$

The **Benchmark Test Case** for the **PL kernel** developed in this document is based on this particular case.

Note: The **PL kernel** definition is agnostic about the implementation of the loading operator **U**. The kernel only provides conditions about the mandatory input vector **P** and about the behaviour of the operator **U**. Different algorithms and procedure approaches can be used for constructing such an operator. This operator implementation agnosticism will be kept in the design of the correspondent **Benchmark Test Case** so it can be used as a methodology for evaluating different algorithms or procedures for data loading into quantum circuits.

2.3 Description of the benchmark test case

This section presents the complete description of the **Benchmark Test Case** for the **PL kernel**. Section 2.3.1 describes the problem addressed by the test case. Section 2.3.2 provides a high-level description of the case. Finally, Section 2.3.3 provides the execution workflow.

2.3.1 Description of the problem

The loading of a fixed normal probability distribution function, **PDF**, $N_{\mu,\sigma}(x)$, into a quantum circuit is the **Benchmark Test Case** associated to the **PL kernel**. An operator **U** for loading this normal **PDF** must be built using a *probability loading* algorithm, and the probabilities of the different possible final states must be measured and compared with the original normal **PDF**.

Finally, the verification of the output can be done by comparing the obtained measurements with the original **PDF**, using several metrics.

2.3.2 Benchmark test case description

This section introduces a detailed step-by-step workflow of the **Benchmark Test Case**. Given a number of qubits, n, and using a specific input *probability loading* algorithm, the test case must follow the following steps:

1. Take a random uniform distribution with a particular mean, $\tilde{\mu}$ and standard deviation, $\tilde{\sigma}$, selected within the following ranges:
   - $\tilde{\mu} \in [-2, 2]$
   - $\tilde{\sigma} \in [0.1, 2]$
2. So the normal **PDF** is: $N_{\tilde{\mu},\tilde{\sigma}}(x)$
3. Create an array of 2^n values: $\mathbf{x} = \{x_0, x_1, x_2, \cdots, x_{2^n-1}\}$ where:

- x_0 such that
$$\int_{-\infty}^{x_0} N_{\tilde{\mu},\tilde{\sigma}}(x)dx = 0.05$$
- x_{2^n-1} such that
$$\int_{-\infty}^{x_{2^n-1}} N_{\tilde{\mu},\tilde{\sigma}}(x)dx = 0.95$$
- $x_{i+1} = x_i + \Delta x$
- $\Delta x = \frac{x_{2^n-1}-x_0}{2^n}$

4. Create a 2^n values array, $\mathbf{P}$ from $\mathbf{x}$ by:
$$\mathbf{P}(\mathbf{x}) = \{P(x_0), P(x_1), \cdots, P(x_{2^n-1})\} = \{N_{\tilde{\mu},\tilde{\sigma}}(x_0), N_{\tilde{\mu},\tilde{\sigma}}(x_1), \cdots, N_{\tilde{\mu},\tilde{\sigma}}(x_{2^n-1})\}$$

5. Normalize the $\mathbf{P}$ array:
$$\mathbf{P_{norm}}(\mathbf{x}) = \{P_{norm}(x_0), P_{norm}(x_1), \cdots, P_{norm}(x_{2^n-1})\}$$
where
$$P_{norm}(x_i) = \frac{P(x_i)}{\sum_{j=0}^{2^n-1} P(x_j)}$$

6. Compute the number of shots n_{shots} as:
$$n_{shots} = \min\left(10^6, \frac{100}{\min\left(\mathbf{P_{norm}}(\mathbf{x})\right)}\right)$$

7. Use the $\mathbf{P_{norm}}(\mathbf{x})$ array as input of the particular *probability loading* algorithm for creating the $\mathbf{U}$ operator such that :
$$\mathbf{U}|0\rangle_n = \sum_{i=0}^{2^n-1} \sqrt{P_{norm}(x_i)}|i\rangle_n \tag{7}$$

8. Execute the quantum program $\mathbf{U}|0\rangle_n$ and measure all the n qubits a number of times n_{shots}. Store the number of times each state $|i\rangle_n$ is obtained, m_i, and compute the probability of obtaining it as
$$Q_i = \frac{m_i}{n_{shots}} \forall i = \{0, 1, \cdots, 2^n - 1\}$$

9. With the measured array $\mathbf{Q} = \{Q_i\}\forall i = \{0, 1, \cdots, 2^n - 1\}$ and the initial normalised array $\mathbf{P_{norm}}$ compute following metrics:
   - The Kolmogorov-Smirnov (**KS**) between $\mathbf{Q}$ and $\mathbf{P_{norm}}$. This is the maximum of the absolute difference between the cumulative distribution functions of $\mathbf{P_{norm}}$ and $\mathbf{Q}$:
$$KS = \max\left(\left|\sum_{j=0}^{i} P_{norm}(x_j) - \sum_{j=0}^{i} Q_j\right|, \ \forall i = 0, 1, \cdots, 2^n - 1\right)$$
   - The Kullback-Leibler divergence is defined as:
$$\mathbf{KL}(\mathbf{Q}/\mathbf{P_{norm}}) = \sum_{j=0}^{2^n-1} P_{norm}(x_j) \ln \frac{P_{norm}(x_j)}{\max(\epsilon, Q_k)}$$
   where $\epsilon = \min(P_{norm}(x_j)) * 10^{-5}$ which guarantees the logarithm exists when $Q_k = 0$

10. Execute a χ^2 test using $n_{shots}\mathbf{Q}$ and $n_{shots}\mathbf{P_{norm}}$ and get its p-value (using as null hypothesis that both sets are equal). If the p-value is lower than 0.05 then the obtained result should be considered invalid. Este χ^2 test no aparece antes, no?

Additionally, the time from steps 1 to 10 is measured as the **elapsed time**. If possible, the time of the quantum part, step 8, should be measured separately as the **quantum time**.

2.3.3 Complete benchmark procedure

To execute a complete **Benchmark Test Case** of the **PL kernel** the next procedure must be followed:

- We must select in advance the set of the number of qubits to be tested (for example from n=4 to n=8).
- For each number of qubits the following steps must be performed:
  1. Execute a warm-up step consisting in:
     (a) Execute 10 iterations of the **Benchmark Test Case**, section 2.3.2, and the computation of the mean and the standard deviation of the **elapsed time**, $\tilde{T}$ and σ_T metrics, respectively.
     (b) Compute the number of repetitions, M, using equation (8):

$$M = \left(\frac{\sigma_T Z_{1-\frac{\alpha}{2}}}{r\tilde{T}}\right)^2 \tag{8}$$

     where r is the desired relative error for the **elapsed time** (fixed to $r = 0.1$) and $Z_{1-\frac{\alpha}{2}}$ is the percentile for $\alpha = 0.95$
  2. Execute the complete **Benchmark Test Case**, section 2.3.2, M times. M must be greater than 5.
  3. Compute the mean and the standard deviation for the **elapsed time**, **quantum time**, if possible, and for the mentioned metrics in steps 9 and 10 of section 2.3.2: χ^2, **KS** and **KL**.
- If the verification χ^2 test fails (the p-value is lower than 0.05), the process must be stopped.

The method used to calculate the number of repetitions M in the previous procedure guarantees that the **elapsed time** will have a relative error lower than 10% with a confidence level of 95%.

3 Benchmark for Amplitude Estimation Algorithms

This section describes the T2:Amplitude Estimation benchmark.

Section 3.1 provides a justification for kernel selection according to the **TNBS** benchmarking methodology meanwhile section 3.2 presents a complete description of this **AE kernel**. The kernel is associated to a **Benchmark Test Case** which is described in Section 3.3.

3.1 Kernel selection justification

The **AE kernel** is a core step in quantum computation for various applications like finance [16, 25, 6], chemistry [11, 1], machine learning [23, 24] and, even, can be used for generic tasks such as numeric integration [15]. For executing **AE kernel**, different algorithm approaches, **AE** algorithms from now, were proposed recently [3, 18, 5, 14, 26, 21, 13]. So the **AE kernel** can be considered as an interesting candidate for **TNBS kernel**. Additionally, it satisfies the three main requirements from the QC benchmark methodology:

1. A mathematical definition of the kernel can be given with enough accuracy to allow the construction of a standalone circuit (see sections 3.2 and 3.3.2).
2. The kernel can be defined using a smaller or larger number of qubits.
3. The output can be verified with a classical computation (in the proposed **Benchmark Test Case**, see section 3.3, the result is known *a priori*)

3.2 Kernel Description

The **AE kernel**, also know as the **Amplitude Estimation** problem, can be defined in the following way:

Let an unitary operator $\mathbf{A}$ that acts upon an initial n-qubits state $|0\rangle_n = |0\rangle^{\otimes n}$ as shown in equation (9):

$$|\Psi\rangle = \mathbf{A}|0\rangle_n = \sum_{i=0}^{2^n-1} a_i|i\rangle_n \tag{9}$$

Now we are interested in the sub-state composed by some basis states $J = \{j_0, j_1, \cdots, j_l\}$, so we can write down (10):

$$|\Psi\rangle = \mathbf{A}|0\rangle_n = \sum_{j\in J} a_j|j\rangle_n + \sum_{i\notin J} a_i|i\rangle_n \tag{10}$$

If we define the sub-states $|\Psi_0\rangle$ and $|\Psi_1\rangle$ using (11):

$$|\Psi_0\rangle = \frac{1}{\sqrt{a}}\sum_{j\in J} a_j|j\rangle_n \;\; and \;\; |\Psi_1\rangle = \frac{1}{\sqrt{1-a}}\sum_{i=0,i\notin J}^{2^n-1} a_i|i\rangle_n \tag{11}$$

The final $|\Psi\rangle$ can be expressed as (12):

$$|\Psi\rangle = \mathbf{A}|0\rangle_n = \sqrt{a}|\Psi_0\rangle + \sqrt{1-a}|\Psi_1\rangle \tag{12}$$

The **AE kernel** consists in getting an estimation of the amplitude of $|\Psi_0\rangle$: a.

Following subsections present different approaches for solving the **AE kernel**.

3.2.1 Monte Carlo Solution

One naive procedure for solving **AE kernel**, *Monte Carlo* solution from now, is measuring all the qubits N times and getting the probability of obtaining the desired state $|\Psi_0\rangle$. In this case the estimator of a, $\tilde{a}$, will be given by equation (13):

$$\tilde{a} = P_{|\Psi_0\rangle} = \frac{Number\ of\ times\ |\Psi_0\rangle\ was\ measured}{N} \tag{13}$$

The error ϵ_a of this $\tilde{a}$ estimator can be obtained using the *Chernoff-Hoeffding* [9] bound (14):

$$P[\tilde{a} \in |a_j - \epsilon_a, a_j + \epsilon_a|] \geq 2e^{-2N\epsilon_a^2} \tag{14}$$

So if we want $P[\tilde{a} \in |a_j - \epsilon_a, a_j + \epsilon_a|] \geq \alpha$ ($\alpha \in [0, 1]$) then the error is given by (15):

$$\epsilon_a^2 \leq \frac{1}{2N} Ln[\frac{2}{\alpha}] \tag{15}$$

So the error for the estimator $\tilde{a}$ has the following behaviour with the number of measurements N:

$$\epsilon_a \sim \frac{1}{\sqrt{N}} \tag{16}$$

Usually, for the **AE kernel**, instead of the number of measurements, the number of calls to the oracle (this is the operator $\mathbf{A}$), N_{oracle}, is used. In the *Monte Carlo* solution: $N = N_{oracle}$, so equation (16) can be rewritten as equation (17)

$$\epsilon_a \sim \frac{1}{\sqrt{N_{oracle}}} \tag{17}$$

3.2.2 Canonical AE solution with Quantum Phase Estimation

In equation (12) the following substitution: $\sqrt{a} = \sin(\theta)$ can be performed and equation (18) can be obtained:

$$|\Psi\rangle = \mathbf{A}|0\rangle_n = \sqrt{a}|\Psi_0\rangle + \sqrt{1-a}|\Psi_1\rangle = \sin(\theta)\,|\Psi_0\rangle + \cos(\theta)\,|\Psi_1\rangle \tag{18}$$

Now a Grover-like operator [3] based on $\mathbf{A}$, can be built following equation (19):

$$\mathbf{G}(\mathbf{A}) = \mathbf{A}\left(\hat{I} - 2|0\rangle\langle 0|\right)\mathbf{A}^\dagger\left(\hat{I} - 2|\Psi_0\rangle\langle\Psi_0|\right) \tag{19}$$

This Grover-like operator acts as shown in equation (20):

$$\mathbf{G}^k(\mathbf{A})|\Psi\rangle = \mathbf{G}^k(\mathbf{A})\mathbf{A}|0\rangle_n = \sin((2k+1)\theta)|\Psi_0\rangle + \cos((2k+1)\theta)|\Psi_1\rangle \tag{20}$$

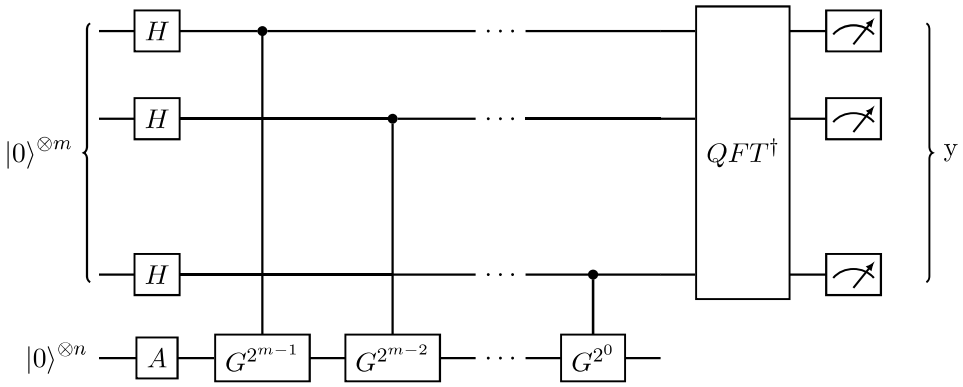

Figure 1: Canonical *Amplitude Estimation* using *Quantum Phase Estimation*.

being k the number of times that operator **G** is applied.

The *canonical Quantum Amplitude Estimation* solution for **AE kernel**, uses the *Quantum Phase Estimation* algorithm, **QPE**, [3] over the operator **G**(**A**) for computing $\tilde{a}$. This algorithm allocates m auxiliary qubits and applies, over $|\Psi\rangle$, geometrically increasing controlled, by the different auxiliary qubits, powers of **G** as shown in the Figure 1

Finally, the complex conjugate of the *Quantum Fourier Transformation* ($QFT^\dagger$ in Figure 1) is applied over the auxiliary qubits, that will be measured generating an integer $y \in \{0, 1, ...M-1\}$, where $M = 2^m$. This integer can be mapped to an angle using:

$$\tilde{\theta} = \frac{y\pi}{2^m} \tag{21}$$

In this case the estimation will be $\tilde{a} = \sin^2(\tilde{\theta})$. With a probability of at least $\frac{8}{\pi^2} \sim 81\%$ the error of the estimator will be given by (22) [3]:

$$\epsilon = |\tilde{a} - a| \leq \frac{2\pi\sqrt{a(1-a)}}{M} + \frac{\pi^2}{M^2} \tag{22}$$

So in this case the error for the estimator $\tilde{a}$ scales with:

$$\epsilon_a \sim \frac{1}{M} \tag{23}$$

The number of auxiliary qbits, m, is related to the number of oracle calls by equation (24):

$$M = 2^m = \frac{N_{oracle} + 1}{2} \tag{24}$$

By plugging (24) into (23), the error for the *canonical Quantum Amplitude Estimation* algorithm can be obtained as a function of the number of oracle calls (25):

$$\epsilon_a \sim \frac{2}{N_{oracle} + 1} \sim \frac{1}{N_{oracle}} \tag{25}$$

This approach yields a quadratic speed up over the *Monte Carlo* method (17).

3.2.3 Amplitude Estimation without Phase Estimation

Canonical Quantum Amplitude Estimation is computationally expensive and presents some caveats to be implemented in current quantum computers. However, there are several algorithms that can solve the **AE kernel**, without the use of **QPE**, where the error of the $\tilde{a}$ estimation, ϵ_a, scales between *Monte Carlo* and *Canonical Quantum Amplitude Estimation* one, this is:

$$\frac{1}{N_{oracle}} < \epsilon_a < \frac{1}{\sqrt{N_{oracle}}} \tag{26}$$

The main idea of these algorithms is to take advantage of the fact:

$$\mathbf{G}^k|\Psi\rangle = \mathbf{G}^k\mathbf{A}|0\rangle_n = \sin\left((2k+1)\theta\right)|\Psi_0\rangle + \cos\left((2k+1)\theta\right)|\Psi_1\rangle \tag{27}$$

And in the use of very smart strategies for selecting k in order to maximize the probability of measuring the $|\Psi_0\rangle$:

$$P[|\Psi_0\rangle] = \sin^2\left((2k+1)\theta\right) \tag{28}$$

3.3 Description of the benchmark test case

This section presents the complete description of the **Benchmark Test Case** for the **AE kernel**. The main idea is the computation of the integral of a particular function, in a well-defined interval, using some particular implementation, usually an **AE** algorithm, of the **AE kernel**.

Section 3.3.1 presents the base problem, integral computation, of the **Benchmark Test Case** in a formal way. Section 3.3.2 describes, exhaustively, how the **Benchmark Test Case** should be implemented from a formal perspective. Finally, Section 3.3.3 provides the workflow for complete execution of **Benchmark Test Case**.

3.3.1 Description of the problem

The computation of the integral of a function, $f(x)$, in a closed interval $[a, b] \subset \mathbb{R}$, is the proposed **Benchmark Test Case** for **AE kernel**.

An operator **A** must be constructed in such a way that the desired integral: **F**,

$$\mathbf{F} = \int_a^b f(x)dx \tag{29}$$

must be encoded into the amplitude of a very well final defined state. This is, the operator **A** must act as showed in equation (30)

$$|\Psi\rangle = \mathbf{A}|0\rangle_n = \sqrt{a}|\Psi_0\rangle + \sqrt{1-a}|\Psi_1\rangle \tag{30}$$

where $\sqrt{a} = \mathbf{F}$

This **A** operator must be given as input to an **AE** algorithm, that must return the estimation of the $\tilde{\mathbf{F}}$. To evaluate the performance of the operator the estimator should be compared to the actual integral value **F**

The proposed function for the **Benchmark Test Case** is $f(x) = \sin x$, whose integral can be calculated easily as (31):

$$\mathbf{F} = \int_a^b \sin(x)dx = -\cos x|_a^b = \cos(a) - \cos(b) \tag{31}$$

and the two following integration intervals will be used:

- $[0, \frac{3\pi}{8}]$: $\mathbf{F}^0 = \int_0^{\frac{3\pi}{8}} \sin(x)dx = 0.6173165676349102$. This computation will be mandatory.
- $[\pi, \frac{5\pi}{4}]$: $\mathbf{F}^1 = \int_\pi^{\frac{5\pi}{4}} \sin(x)dx = -0.2928932188134523$. This computation will be optional.

In summary, for the **Benchmark Test Case**, an operator $\mathbf{A}^0$ must be built and a particular **AE** algorithm must compute and report the integral $\mathbf{F}^0$. Additionally, a second $\mathbf{A}^1$ operator can be built and the corresponding integral $\mathbf{F}^1$ can be reported.

3.3.2 Benchmark test case description

This section presents a complete mathematical description of the **Benchmark Test Case** for the **AE kernel**.

The **Benchmark Test Case** proposed requires a set of steps that are explained in detail in Sections 3.3.2.1-3.3.2.4, namely: the discretization of the domain and the function, the normalization of the array, and the encoding of the function as a quantum circuit. Section 3.3.2.7 describes the metrics used to verify the output of the circuit, and Section 3.3.2.8 describes the general workflow of the algorithm including the steps described before.

3.3.2.1 Domain Discretization The first step is the discretization of each domain in 2^n intervals, with $n \in \mathbb{N}$ as shown in (32):

$$\{[x_0, x_1], [x_1, x_2], ..., [x_{2^n-1}, x_{2^n}]\} \tag{32}$$

Where

- $x_{i+1} > x_i$
- $a = x_0$
- $b = x_{2^n}$

3.3.2.2 Function discretization For each domain following array with the discretization of the desired function, $f(x) = \sin(x)$, must be computed:

$$f_{x_i} = \frac{f(x_{i+1}) + f(x_i)}{2}$$

The desired integral, for each interval, can be approximated as Riemann sum (33):

$$S_{[a,b]} = \sum_{i=0}^{2^n-1} f_{x_i} \cdot (x_{i+1} - x_i) \tag{33}$$

Using $x_{i+1} - x_i = \frac{b-a}{2^n}$ then we can write down (33) as:

$$S_{[a,b]} = \sum_{i=0}^{2^n-1} f_{x_i} \frac{b-a}{2^n} = \frac{b-a}{2^n} \sum_{i=0}^{2^n-1} f_{x_i} \tag{34}$$

When $(x_{i+1} - x_i) \to 0$ (i.e., $n \to \infty$), $S_{[a,b]} \to \mathbf{F} = \int_a^b \sin(x)dx$

3.3.2.3 Array Normalisation A normalization step, over the f_{x_i} array, must be performed before creating the operator $\mathbf{A}$ that will encode the function into a quantum circuit. This should be done using (35):

$$f\_norm_{x_i} = \frac{f_{x_i}}{\max(|f_{x_i}|)} \tag{35}$$

Now the computed integral will be

$$S_{[a,b]} = \frac{b-a}{2^n} \sum_{i=0}^{2^n-1} f_{x_i} = \frac{b-a}{2^n} \sum_{i=0}^{2^n-1} \max(|f_{x_i}|) f\_norm_{x_i} = \frac{\max(|f_{x_i}|)(b-a)}{2^n} \sum_{i=0}^{2^n-1} f\_norm_{x_i} \tag{36}$$

3.3.2.4 Encoding function into a quantum circuit The next step is to codify $f\_norm_{x_i}$ array in a quantum circuit. The following procedure must be used:

1. Initialize a quantum register with at least $n+1$ qubits[1], where n must be equal to the n used to define the 2^n discretization intervals (see section 3.3.2.1):

$$|0\rangle \otimes |0\rangle_n \tag{37}$$

2. Apply the uniform distribution over the first n qubits as shown in (38):

$$(\mathbb{I} \otimes H^{\otimes n})(|0\rangle \otimes |0\rangle_n) = |0\rangle \otimes H^{\otimes n}|0\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |0\rangle \otimes |i\rangle_n \tag{38}$$

3. Create an operator $\mathbf{U}_f$ for encoding the $f\_norm_{x_i}$. This operator must act as shown in (39):

$$\mathbf{U}_f (|0\rangle \otimes |i\rangle_n) = (f\_norm_{x_i}|0\rangle + \beta_i|1\rangle) \otimes |i\rangle_n \tag{39}$$

4. Apply the $\mathbf{U}_f$ operator over the $n+1$ qubits:

$$\mathbf{U}_f \left(\mathbb{I} \otimes H^{\otimes n}\right) |0\rangle \otimes |0\rangle_n \tag{40}$$

5. Applying equation (38) and (39) into (40) equation (41) is obtained:

$$\begin{aligned} \mathbf{U}_f \left(\mathbb{I} \otimes H^{\otimes n}\right) |0\rangle \otimes |0\rangle_n = \mathbf{U}_f \left(\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |0\rangle \otimes |i\rangle_n \right) = \\ = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \mathbf{U}_f (|0\rangle \otimes |i\rangle_n) = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} (f\_norm_{x_i}|0\rangle + \beta_i|1\rangle) \otimes |i\rangle_n \end{aligned} \tag{41}$$

[1] Additional auxiliary qubits may be used

6. In equation (41) the amplitude β_i is not important.
7. Finally the uniform distribution is applied over the first n qubits again as shown in (42):

$$|\Psi\rangle = (\mathbb{I} \otimes H^{\otimes n})\, \mathbf{U}_f\, (\mathbb{I} \otimes H^{\otimes n})\, |0\rangle \otimes |0\rangle_n \tag{42}$$

8. So applying (41) into (42) the equation (43) can be obtained:

$$|\Psi\rangle = (\mathbb{I} \otimes H^{\otimes n})\, \mathbf{U}_f\, (\mathbb{I} \otimes H^{\otimes n})\, |0\rangle \otimes |0\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \Big(f\_norm_{x_i}|0\rangle + \beta_i|1\rangle \Big) \otimes H^{\otimes n}|i\rangle_n \tag{43}$$

9. Taking into account only the $|0\rangle \otimes |i\rangle_n$ terms, equation (43) can be expressed as (44):

$$|\Psi\rangle = (\mathbb{I} \otimes H^{\otimes n})\, \mathbf{U}_f\, (\mathbb{I} \otimes H^{\otimes n})\, |0\rangle \otimes |0\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} f\_norm_{x_i}|0\rangle \otimes H^{\otimes n}|i\rangle_n + \cdots \tag{44}$$

10. It is know that:

$$H^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n} \sum_{k=0}^{2^n} (-1)^{jk} |j\rangle_{nn}\langle k| \tag{45}$$

11. So $H^{\otimes n}|i\rangle_n$ can be expressed using equation (46):

$$H^{\otimes n}|i\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n} \sum_{k=0}^{2^n} (-1)^{jk} |j\rangle_{nn}\langle k|i\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n} (-1)^{ji} |j\rangle_n = \frac{1}{\sqrt{2^n}}|0\rangle_n + \frac{1}{\sqrt{2^n}} \sum_{j=1}^{2^n} (-1)^{ji} |j\rangle_n \tag{46}$$

12. Finally applying (46) into (44) and taking only into account the $|0\rangle \otimes |0\rangle_n$ term, equation (47) can be obtained:

$$|\Psi\rangle = (\mathbb{I} \otimes H^{\otimes n})\, \mathbf{U}_f\, (\mathbb{I} \otimes H^{\otimes n})\, |0\rangle \otimes |0\rangle_n = \frac{1}{2^n} \sum_{i=0}^{2^n-1} f\_norm_{x_i}|0\rangle \otimes |0\rangle_n + \cdots \tag{47}$$

Using the previous steps, two different operators $\mathbf{A}^I(f_{x_i})$ must be created following equation (48):

$$\mathbf{A}^I(f_{x_i}) = (\mathbb{I} \otimes H^{\otimes n})\, \mathbf{U}_f^I\, (\mathbb{I} \otimes H^{\otimes n}) \tag{48}$$

where the superscript I can take 0 or 1 depending on the domain integration interval

Using (47) the behaviour of such operators will be:

$$|\Psi\rangle = \mathbf{A}^I(f_{x_i})|0\rangle \otimes |0\rangle_n = \frac{1}{2^n} \sum_{i=0}^{2^n-1} f\_norm_{x_i}|0\rangle \otimes |0\rangle_n + \cdots \tag{49}$$

Equation (49) can be compared with the equation (18):

$$|\Psi\rangle = \mathbf{A}|0\rangle_n = \sqrt{a}|\Psi_0\rangle + \sqrt{1-a}|\Psi_1\rangle$$

where

$$|\Psi_0\rangle = |0\rangle \otimes |0\rangle_n$$

and

$$\sqrt{a} = \frac{1}{2^n}\sum_{i=0}^{2^n-1} f\_norm_{x_i}$$

Now the Riemann sum approximation of the desired integral can be computed by measuring the probability of obtaining the state $|\Psi_0\rangle = |0\rangle \otimes |0\rangle_n$ as shown in (50)

$$\mathbf{P}[|\Psi_0\rangle] = |\ \langle\Psi_0\ |\Psi\rangle\ |^2 = \left|\ \langle\Psi_0\ |\frac{1}{2^n}\sum_{i=0}^{2^n-1} f\_norm_{x_i}|\Psi_0\rangle\ \right|^2 = \left|\frac{1}{2^n}\sum_{i=0}^{2^n-1} f\_norm_{x_i}\right|^2 = \tilde{a} \tag{50}$$

The $\sim$ in $\tilde{a}$ indicates that the amplitude was obtained using a quantum measurement.

Now, plugging (50) into the Riemann sum (36) the desired integral can be computed as (51)

$$\tilde{S}_{[a,b]} = \frac{\max(f_{x_i})\,(b-a)}{2^n}\left(2^n\sqrt{\mathbf{P}[|\Psi_0\rangle]}\right) \tag{51}$$

In (51) the $\sim$ in $\tilde{S}_{[a,b]}$ indicates that the integral was obtained using a measurement meanwhile the $S_{[a,b]}$ is for pure Riemann sum calculation as shown in (34).

The 2^n terms can be removed from the equation but they will be kept for the moment.

3.3.2.5 Operator $\mathbf{U}_f$ This subsection describes the steps for building the $\mathbf{U}_f$ operator, equation (39):

- The following array must be computed: $\phi_{x_i} = \arccos(f\_norm_{x_i})$, using the values of array $f\_norm_{x_i}$.
- For a given state $|i\rangle_n \otimes |0\rangle$, it must be implemented a rotation around the *y-axis* over the last qubit, $|0\rangle$, controlled by the state $|i\rangle_n$ of $2 * \phi_{x_i}$. So the following operation must be built:

$$|0\rangle \otimes |i\rangle_n \rightarrow \mathbf{R}_y(2 * \phi_{x_i})|0\rangle \otimes |i\rangle_n = (\cos(\phi_{x_i})|0\rangle + \sin(\phi_{x_i})|1\rangle) \otimes |i\rangle_n \tag{52}$$

- Now undoing the ϕ_{x_i} and doing $\beta_i = \sin(\phi_{x_i})$ the desired operator $\mathbf{U}_f$ can be obtained by:

$$(f\_norm_{x_i}|0\rangle + \beta_i|1\rangle) \otimes |i\rangle_n = \mathbf{U}_f\,(|0\rangle \otimes |i\rangle_n) \tag{53}$$

So the operator $\mathbf{U}_f$ can be constructed following equations (52) and (53) that can be summarized into (54):

$$\mathbf{U}_f \left(|0\rangle \otimes |i\rangle_n\right) = \left(\mathbf{R}_y(2 * \phi_{x_i})|0\rangle\right) \otimes |i\rangle_n \tag{54}$$

The $\mathbf{U}_f$ is a controlled rotation by state. The recommended way for implementing it, is using **quantum multiplexors** [17]. A direct implementation of this operator can be used but, in general, deeper circuits with redundant operations are obtained with respect to the **quantum multiplexors** implementation.

3.3.2.6 Amplitude Estimation Algorithm As shown in section 3.3.2.4 the two $\mathbf{A}^I$ operators allow to encode each correspondent integral in the amplitude of the state $|0\rangle \otimes |0\rangle_n$.

For a given **AE** algorithm:

- Operator $\mathbf{A}^0$ must be provided as input of the **AE** algorithm and the obtained $\tilde{S}^0_{[a,b]}$ integral must be reported as output.
- Additionally, operator $\mathbf{A}^1$ can be provided as input of the **AE** algorithm and the obtained $\tilde{S}^1_{[a,b]}$ integral can be reported as output.

Note: in general most **AE** algorithms use the Grover-like operator of $\mathbf{A}$, equation (19), for solving the **AE kernel**. The **AE kernel** and the correspondent **Benchmark Test Case** presented in this document is agnostic to Grover operators. The only mandatory input is the operator $\mathbf{A}$.

3.3.2.7 Getting the metrics The quality of the **AE** estimation of the integrals obtained in the previous steps, $\tilde{S}^I_{[a,b]}$ (where $I = \{0, 1\}$ stands for each of the intervals where the integral can be computed) must be evaluated using the following metrics:

- *Sum absolute error*: between the AE estimator and the Riemann sum computed using (34): $\epsilon = |\tilde{S}^I_{[a,b]} - S_{[a^I,b^I]}|$
- *Oracle calls*: total number of calls of the operator $\mathbf{A}^I$

3.3.2.8 Summary AE benchmark kernel A step-by-step workflow of the **Benchmark Test Case** for the **AE kernel**, with references to the before explained components, is presented here.

For a desired number of qubits and for one of the integration intervals, described in Section 3.3.1, following steps must be executed:

1. Create the domain discretization as explained in Section 3.3.2.1
2. Create the array with the correspondent *sine* function discretization as explained in Section 3.3.2.2
3. Compute normalization of the array as explained in Section 3.3.2.3
4. Create the $\mathbf{A}^I$ oracle operator for encoding the array as explained in Section 3.3.2.4
5. Using the $\mathbf{A}^I$ oracle operator as input of the AE algorithm for computing the estimation of $\tilde{a}$, see equation (50).
6. Post-process the result of the AE algorithm for getting the estimation of the integral as explained in Section 3.3.2.7 an in equation (51)

7. Compute the desired metrics as explained in Section 3.3.2.7

Additionally, the following times should be computed:

- Complete benchmark step time: this is the time from step 1 to step 7. This will be the **elapsed time**.
- The execution time of the *Amplitude Estimation* algorithm. This is the time of step 5. This will be the **run time**
- If it is possible the time of the pure quantum part of the algorithm should be registered. This will be the **quantum time**

3.3.3 Complete benchmark procedure

To execute the **Benchmark Test Case** for the **AE kernel** following steps should be done:

- A range of a number of qubits should be selected (for example from n=4 to n=8).
- For each selected number of qubits and each 2 described integral intervals following steps should be executed:
  1. Execute a pre-benchmark step consisting in:
     (a) Executing 10 times the complete benchmark step (section 3.3.2.8) for a number of qubits n, and the interval integration.
     (b) For *Sum absolute error*, *Oracle calls* metrics and for the *run time* compute the mean, μ_m, and the standard deviation, σ_m, where m is each of these quantities.
     (c) For each of quantity m computes the number of repetitions M_m using (55):

$$M_m = \left(\frac{\sigma_m Z_{1-\frac{\alpha}{2}}}{r\mu_m}\right)^2 \tag{55}$$

     where r is the desired relative error for all the quantities m (it should be fixed to $r = 0.1$) and $Z_{1-\frac{\alpha}{2}}$ is the percentile for a confidence level of $\alpha = 0.95$
     (d) Compute the maximum value of the different M_m obtained values: $M = \max M_m$
  2. Execute the complete benchmark test case (section 3.3.3) M times. M must be greater than 5.
  3. Compute the mean and the standard deviation for each of the metrics presented in section 3.3.2.7 (*Sum absolute error*, *Oracle calls*) and for all the measured times explained in section 3.3.2.8 (*elapsed time*, *run time* and *quantum time*)

If the before workflow is followed it can be said that, for each qubit and interval executed, the provided mean for the different obtained metrics (*Sum absolute error*, *Oracle calls* and *elapsed time*) will have a relative error lower than 10% with a confidence level of 95%.

4 Benchmark for Phase Estimation Algorithms

This section describes the T3: Phase Estimation benchmark.

Section 4.2 justifies why this is a good benchmark case and explains its suitability for the suite. Section 4.2 provides a formal mathematical description of the kernels, while Section 4.3 offers a detailed description of the test case, including a formal definition of the execution procedure.

4.1 Kernel selection justification

The Quantum Phase Estimation (QPE) kernel[10] is utilized to estimate the eigenvalues (or phases) of a unitary operator. The QPE implements a measurement for any Hermitian operator and serves as a key component in numerous quantum algorithms, such as Shor's algorithm, the quantum algorithm for solving linear systems of equations, or the measurement of the energy of a Hamiltonian in Variational Quantum Eigensolver (VQE) algorithms[4].

Moreover, it fulfills the three main requirements of the QC benchmark methodology:

1. A mathematical definition of the kernel can be provided with sufficient accuracy to enable the construction of a standalone circuit (refer to sections 4.2 and 4.3).
2. The kernel can be defined for a configurable number of qubits.
3. The output can be verified through a classical computation (see section 4.3.2).

4.2 Kernel Description

Let $\mathcal{U}$ be an m-qubit unitary operator. The eigenvalues of this operator are phases that can be represented as $e^{2i\pi\lambda_j}$ for $j = 0, 1, 2, \cdots 2^m - 1$. For a particular *eigenstate* $|\psi_j\rangle$, then:

$$\mathcal{U}\,|\psi_j\rangle = e^{2i\pi\lambda_j}\,|\psi_j\rangle$$

where $0 \leq \lambda_j < 1$. The objective of the **QPE** kernel is to obtain an estimation, up to a finite level, of the different eigenvalues λ_j. Consider a unitary operator U that operates on m qubits and has an eigenvector $|\psi\rangle$ with an associated eigenvalue $e^{2\pi i\theta}$, where $0 \leq \theta < 1$. Our goal is to estimate this eigenvalue to a finite level of precision by estimating the phase θ. The eigenvalue $e^{2\pi i\theta}$ can be expressed in this form because U is a unitary operator over a complex vector space, and therefore, its eigenvalues must be complex numbers with an absolute value of 1.

The quantum phase estimation kernel operates on two registers, a m qubits for the input state $|\psi\rangle$ and a t qubits for the counting register used for the phase estimation. Thus, the total number of qubits will be $m + t$. The initial state is set to $|\psi\rangle|\mathbf{0}\rangle$, where $|\mathbf{0}\rangle$ is the t-qubit state $|0\rangle^{\otimes t}$. The quantum phase estimation kernel applies a sequence of controlled-U^{2^j} operations, where the j-th control qubit is the $t - j$-th counting qubit and U is the unitary operator being estimated. This operation maps the input state to

$\frac{1}{\sqrt{2^t}}(|\psi\rangle + e^{2\pi i 2^{t-1}\theta}U|\psi\rangle + e^{2\pi i 2^{t-2}\theta}U^2|\psi\rangle + \cdots + e^{2\pi i\theta}U^{2^{t-1}}|\psi\rangle)|\mathbf{0}\rangle$

where θ is the unknown phase to be estimated.

The canonical **QPE** kernel approach, whose circuit implementation is shown in Figure 2, operates on two different registers: a n qubits one, initialized to the eigenstate $|\psi_j\rangle$, and a m qubits one to estimate the phase. Thus, the total number of qubits will be $n + m$.

The initial state is set to (dashed line 1 in Figure 2):

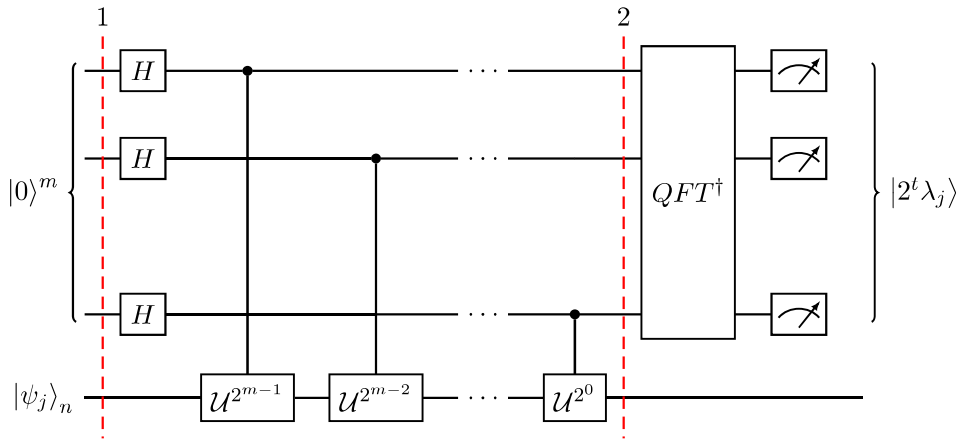

Figure 2: Canonical **QPE** circuit.

$$|\psi_j\rangle \otimes |\mathbf{0}\rangle$$

where $|\mathbf{0}\rangle = |0\rangle^{\otimes m}$.

As shown in Figure 2, a sequence of controlled-$\mathcal{U}^{2j}$ operations, where the j-th control qubit is the $m-j$-th counting qubit, is applied over the state $|\psi_j\rangle$ (i.e. over the n qubits register)

This operation maps the initial state to the state (dashed line 2 in Figure 2):

$$\frac{1}{\sqrt{2^m}}\left(|0\rangle + e^{2\pi i 2^{m-1}\lambda_j}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2\pi i 2^{-1}\lambda_j}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i 2^{0}\lambda_j}|1\rangle\right) \otimes |\psi_j\rangle$$

where λ_j is the unknown eigenvalue correspondent to the eigenstate $|\psi_j\rangle$. After applying the complex conjugate of the **Quantum Fourier Transformation** (operator $QFT^\dagger$ in Figure 2) on the m qubits register, they are measured and the resulting binary string is converted to a decimal fraction. The output of the canonical **QPE kernel** is an estimator $\tilde{\lambda_j}$ of the eigenvalue λ_j to m bits of precision.

4.2.1 Methods to perform the QPE kernel

It is not in the spirit of this benchmark suite to link a kernel to a given method or implementation. However, to gain perspective on the topic, we discuss some of the potential implementation variations proposed for this algorithm[19, 10].

There are two main methodological approaches to the QPE: (1) invoking an inverse Quantum Fourier Transform ($\text{QFT}^\dagger$) to extract the phase information or (2) performing a basic measurement operation followed by classical post-processing instead.

The iterative quantum phase estimation algorithm (IQPE)[7] is a family of implementations of the algorithm designed to address some of the limitations of NISQ-era quantum computers. For this, the quantum platform interacts with a classical computer to iteratively estimate the phase. The role of the classical computer is to provide feedback to the quantum computer based on the intermediate measurements, which yields better accuracy. This approach implements the QPE with a single counting bit.

The Variational Eigensolver family of methods [20] uses a quantum variational approach to compute the eigenvalues of the ground state of the energy of a Hamiltonian. This approach trades complexity for time, as it reduces the complexity of the circuit with respect to traditional QPE methods, but it requires several steps until the parameters (the rotation angle in this case) of the variational circuit are properly adjusted.

This adjustment has been recently improved using Bayesian inference techniques[22]. These techniques allow extracting the maximum amount of information from previous measurements, improving the adjustment of the rotation angles for the next evaluation of the algorithm.

Other approaches do not require time evolution [2], like those based on the use of unitaries that encode the spectrum of the Hamiltonian but are easier to implement than e^{-iHt}.

In those algorithms that rely on the $\text{QFT}^\dagger$ for transforming the result into the computational base, the method to implement the $\text{QFT}^\dagger$ is also an important source of variability, as many methods have been proposed to implement this task [4].

4.3 Description of the benchmark test case

This section introduces the **Benchmark Test Case** for the **QPE kernel**. Subsection 4.3.1 describes the base operator used. Subsection 4.3.2 provides the high-level description that any implementation must follow. Subsection 4.3.3 provides the workflow that a complete **Benchmark Test Case** execution must observe.

4.3.1 Description of the problem

The computation of the eigenvalues of a n qubits unitary operator $R_z(\theta)^n = \otimes_{i=1}^{n} R_z(\theta)$, given a fixed $\theta = \frac{\pi}{2}$, is the proposed **Benchmark Test Case** for the **QPE kernel**. Figure 3 shows the circuit implementation of this operator. The $R_z(\theta)$ operator is a Z-axis rotation gate:

$$R_z(\theta) = (e^{-i\frac{\theta}{2}} |0\rangle \langle 0| + e^{i\frac{\theta}{2}} |1\rangle \langle 1|) \tag{56}$$

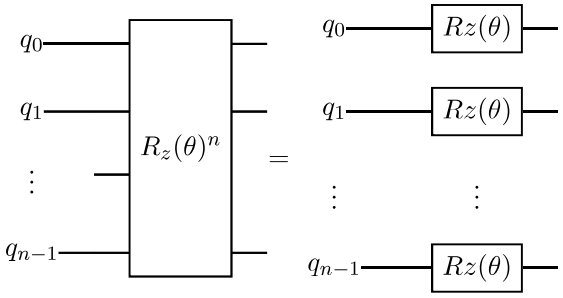

Figure 3: Circuit implementation of the $R_z(\theta)^n$ operator

A fixed QPE implementation must be used to compute the probabilities of the different eigenvalues of the $R_z(\theta)^n$ operator, and these probabilities must be compared with the theoretical probability distribution using various metrics.

4.3.2 Benchmark test case description

This section provides a step-by-step workflow of the **Benchmark Test Case** for the **QPE kernel**. The benchmark requires fixing a discretization parameter m for each number of qubits n test. The procedure follows 8 steps:

1. **Generation of the reference probability distribution of the eigenvalues.** This must be done for each number of qubits (n) and using its associated discretization of 2^m bins of the probability distribution. Thus, the following probabilities must be computed $P^{th}_{\lambda,m}(\frac{k}{2^m})$ where $k = 1, 2, \cdots 2^m - 1$ following the next steps:

   (a) Compute each state of the 2^n possible states as a binary string.

   (b) For each of the computed states $|i\rangle$ $i = 0, 1, 2, \cdots 2^n - 1$ perform the following computations:

   - Count the number of zeros in the bit string representation of the state $|i\rangle$: n^i_0
   - Count the number of ones in the bit string representation of the state $|i\rangle$: n^i_1
   - Calculate the difference between the number of zeros and the number of ones: $d^i = n^i_0 - n^i_1$
   - Compute the associated angle, θ^i to the difference, d^i as: $\theta^i = -\frac{2d^i}{\theta}$ where $\theta = \frac{\pi}{2}$
   - Compute the correspondent angle but only between $[0, 2\pi)$: $\theta^i_{[0,2\pi)} = \theta^i \bmod 2\pi$
   - Finally, compute the correspondent $|i\rangle$ eigenvalue as: $\lambda^i = \frac{\theta^i_{[0,2\pi)}}{2\pi}$

   (c) A eigenvalue λ^i was computed for each possible state $|i\rangle$ $i = 0, 1, 2, \cdots 2^n - 1$. Some λ^i eigenvalues will appear several times.

   (d) Draw a histogram for the complete list of eigenvalues λ^i according to the following guidelines:

   - The number of histogram bins will be 2^m.
   - The range of the histogram will be $[0, 1]$
   - The frequency of eigenvalues in each bin k should be computed: f_{λ_k}
   - Each bin must be labelled as $\frac{k}{2^m}$ where k is the number of the bin ($k = 0, 1, 2, \cdots 2^m - 1$)

   (e) This histogram must be used to build a theoretical discrete probability distribution of the eigenvalues: $P^{th}_{\lambda,m}(\frac{k}{2^m}) = f_{\lambda_k}$. Figure 4 shows an example of the theoretical probability distribution of the eigenvalues, for a $n = 7$ qubits $R_z(\theta)^n$ operator.

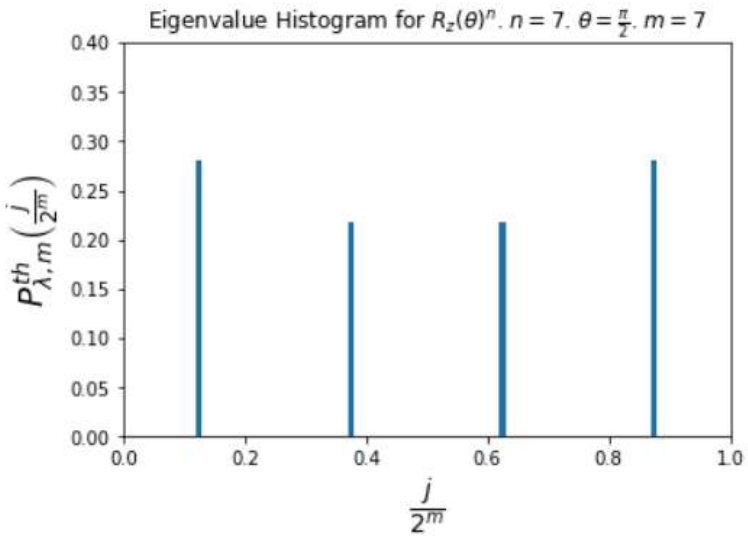

Figure 4: Example of histogram showing the theoretical probability distribution of the eigenvalues for a $n = 7$ qubits operator $R_z(\theta)^n$ with $\theta = \frac{\pi}{2}$. The histogram of the eigenvalues was discretized in 2^m bins with $m = 7$

2. Compute the number of eigenvalues that will be measured using the **QPE** routine $n_{eigenvalues}$ as:

$$n_{eigenvalues} = \min\left(10^6, \frac{100}{\min\left(P^{th}_{\lambda,m}\right)}\right)$$

3. Create the operator $R_z(\theta)^n$ with $\theta = \frac{\pi}{2}$.

4. Create an initial state for the QPE routine that is an equiprobable combination of all the 2^n possible states:

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{n-1} |i\rangle$$

5. Provide the $R_z(\theta)^n$ operator and the initial state $|\psi_0\rangle$ to the **QPE** routine, execute it and measure the eigenvalue. This should be done $n_{eigenvalues}$ times. So a complete list of $n_{eigenvalues}$ **QPE** eigenvalues is generated: λj^{QPE} where $j = 1, 2, \cdots n_{eigenvalues}$

6. The list of eigenvalues generated by the **QPE** (λj^{QPE}) must be used to draw a second histogram using the same procedure as in step **1.d**. This generates the measured probability distribution of the values generated by the **QPE**: $P^{QPE}_{\lambda,m}(\frac{k}{2^m})$ with $k = 0, 1, \cdots 2^m - 1$.

7. The two discrete probability distributions, $P^{th}_{\lambda,m}$ and $P^{QPE}_{\lambda,m}$, must be compared using the following metrics:

   - The Kolmogorov-Smirnov (**KS**) between $P^{th}_{\lambda,m}$ and $P^{QPE}_{\lambda,m}$. This is the maximum of the absolute difference between the cumulative distribution functions of $P^{th}_{\lambda,m}$ and $P^{QPE}_{\lambda,m}$:

$$KS = \max\left(\left|\sum_{k=0}^{i} P^{th}_{\lambda,m}(\frac{k}{2^m}) - \sum_{k=0}^{i} P^{QPE}_{\lambda,m}(\frac{k}{2^m})\right|, \ \forall k = 0, 1, \cdots, 2^m - 1\right)$$

- The Kullback-Leibler divergence, defined as:

$$\mathbf{KL}(P^{QPE}_{\lambda,m}/P^{th}_{\lambda,m}) = \sum_{k=0}^{2^m-1} P^{th}_{\lambda,m}(\frac{k}{2^m}) \ln \frac{P^{th}_{\lambda,m}(\frac{k}{2^m})}{\max(\epsilon, P^{QPE}_{\lambda,m}(\frac{k}{2^m}))}$$

  where $\epsilon = 10^{-5}$ which guarantees the logarithm exists when $P^{QPE}_{\lambda,m}(\frac{k}{2^m}) = 0$

8. Execute a χ^2 test using $n_{eigenvalues}P^{th}_{\lambda,m}$ and $n_{eigenvalues}P^{QPE}_{\lambda,m}$ and get its p-value (using as null hypothesis that both sets are equal). If the p-value is lower than 0.05 then the obtained result should be considered invalid.

Additionally, the time from steps 2 to 8 should be measured and labeled as the **elapsed time**. If possible, the time of the pure quantum part, step 5, should be measured separately as the **quantum time**.

4.3.3 Complete benchmark procedure

The procedure of execution of the **Benchmark Test Case** of the **QPE kernel** is as follows:

1. We must fix in advance the different numbers of qubits to be tested (for example from n=4 to n=8).
2. For each number of qubits n, we have to fix the number of bins m of the discretization (in general it is recommended that $m \geq n$). Then, the following steps must be performed:
   (a) Execute a warm-up step consisting in:
      i. Execute 10 iterations of the **Benchmark Test Case**, subsection 4.3.2, and compute the mean and the standard deviation of the **elapsed time**, $\tilde{T}$ and σ_T, respectively.
      ii. Compute the number of repetitions, M, using equation (57):

$$M = (\frac{\sigma_T Z_{1-\frac{\alpha}{2}}}{r\tilde{T}})^2 \tag{57}$$

      where r is the desired relative error for the **elapsed time** (fixed to $r = 0.1$) and $Z_{1-\frac{\alpha}{2}}$ is the percentile for $\alpha = 0.95$

   (b) Execute the complete **Benchmark Test Case** M times for n qubits following section 4.3.2. M being at least 5.
   (c) Compute the mean and the standard deviation for the **elapsed time**, **quantum time**, if possible, and for the mentioned metrics in step 6 and 7 of section 4.3.2: χ^2, **KS** and **KL**.
3. If the verification χ^2 test fails (the p-value is lower than 0.05), the process must be stopped.

The method used to calculate the number of repetitions M in the previous procedure guarantees that the **elapsed time** will have a relative error lower than 10% with a confidence level of 95%.

Bibliography

[1] Alán Aspuru-Guzik, Anthony D. Dutoi, Peter J. Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. Science, 309(5741):1704–1707, 2005. doi: 10.1126/science.1113479.

[2] Dominic W Berry, Mária Kieferová, Artur Scherer, Yuval R Sanders, Guang Hao Low, Nathan Wiebe, Craig Gidney, and Ryan Babbush. Improved techniques for preparing eigenstates of fermionic hamiltonians. npj Quantum Information, 4(1):22, 2018.

[3] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. AMS Contemporary Mathematics Series, 305, 2000. doi: 10.1090/conm/305/05215.

[4] Yudong Cao, Jonathan Romero, Jonathan P. Olson, Matthias Degroote, Peter D. Johnson, Mária Kieferová, Ian D. Kivlichan, Tim Menke, Borja Peropadre, Nicolas P. D. Sawaya, Sukin Sim, Libor Veis, and Alán Aspuru-Guzik. Quantum chemistry in the age of quantum computing. Chemical Reviews, 119(19):10856–10915, 2019. doi: 10.1021/acs.chemrev.8b00803. URL https://doi.org/10.1021/acs.chemrev.8b00803. PMID: 31469277.

[5] Dmitry Grinko, Julien Gacon, Christa Zoufal, and Stefan Woerner. Iterative quantum amplitude estimation. npj Quantum Information, 7(1), 2021. doi: 10.1038/s41534-021-00379-1.

[6] Andrés Gómez, Alvaro Leitao Rodriguez, Alberto Manzano, Maria Nogueiras, Gustavo Ordóñez, and Carlos Vázquez. A survey on quantum computational finance for derivatives pricing and var. Archives of Computational Methods in Engineering, 2022. doi: 10.1007/s11831-022-09732-9.

[7] Dipanjali Halder, V. Srinivasa Prasannaa, Valay Agarawal, and Rahul Maitra. Iterative quantum phase estimation with variationally prepared reference state. International Journal of Quantum Chemistry, 123(3):e27021, 2023. doi: https://doi.org/10.1002/qua.27021. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/qua.27021.

[8] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. Physical Review Letters, 103(15), oct 2009. doi: 10.1103/physrevlett.103.150502.

[9] Hoeffding's inequality. Hoeffding's inequality Wikipedia, the free encyclopedia, 2004. URL https://en.wikipedia.org/wiki/Hoeffding%27s_inequality.

[10] Duo Jiang, Xiaonan Liu, Huichao Song, and Haoshan Xie. An survey: Quantum phase estimation algorithms. In 2021 IEEE 5th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC), volume 5, pages 884–888, 2021. doi: 10.1109/ITNEC52019.2021.9587010.

[11] Emanuel Knill, Gerardo Ortiz, and Rolando D. Somma. Optimal quantum measurements of expectation values of observables. Physical Review A, 75:012328, 2007. doi: 10.1103/PhysRevA.75.012328.

[12] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. Nature Physics, 10(9):631–633, jul 2014. doi: 10.1038/nphys3029.

[13] Xi Lu and Hongwei Lin. Random-depth quantum amplitude estimation, 2023.

[14] Alberto Manzano, Daniele Musso, and Álvaro Leitao. Real quantum amplitude estimation. EPJ Quantum Technology, 10, 2023. doi: 10.1140/epjqt/s40507-023-00159-0.

[15] Ashley Montanaro. Quantum speedup of monte carlo methods. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 471(2181):20150301, 2015.

[16] Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. Quantum computational finance: Monte Carlo pricing of financial derivatives. Physical Review A, 98(2), 2018. ISSN 2469-9934.

[17] V.V. Shende, S.S. Bullock, and I.L. Markov. Synthesis of quantum-logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(6):1000–1010, 2006. doi: 10.1109/tcad.2005.855930.

[18] Yohichi Suzuki, Shumpei Uno, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto. Amplitude estimation without phase estimation. Quantum Information Processing, 19(2), 2020. doi: 10.1007/s11128-019-2565-2.

[19] Krysta M. Svore, Matthew B. Hastings, and Michael Freedman. Faster phase estimation. Quantum Info. Comput., 14(3–4):306–328, mar 2014. ISSN 1533-7146.

[20] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. The variational quantum eigensolver: A review of methods and best practices. Physics Reports, 986:1–128, 2022. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2022.08.003. URL https://www.sciencedirect.com/science/article/pii/S0370157322003118. The Variational Quantum Eigensolver: a review of methods and best practices.

[21] Shumpei Uno, Yohichi Suzuki, Keigo Hisanaga, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto. Modified grover operator for quantum amplitude estimation. New Journal of Physics, 23(8):083031, 2021. doi: 10.1088/1367-2630/ac19da.

[22] Nathan Wiebe and Chris Granade. Efficient bayesian phase estimation. Physical review letters, 117(1):010503, 2016.

[23] Nathan Wiebe, Ashish Kapoor, and Krysta M. Svore. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. Quantum Info. Comput., 15(3–4):316–356, 2015. ISSN 1533-7146.

[24] Nathan Wiebe, Ashish Kapoor, and Krysta M. Svore. Quantum deep learning. Quantum Info. Comput., 16(7–8):541–587, 2016. ISSN 1533-7146.

[25] Stefan Woerner and Daniel J. Egger. Quantum risk analysis. npj Quantum Information, 5(1), 2019. doi: 10.1038/s41534-019-0130-6.

[26] Yunpeng Zhao, Haiyan Wang, Kuai Xu, Yue Wang, Ji Zhu, and Feng Wang. Adaptive algorithm for quantum amplitude estimation, 2022.