Leveraging quantum computing for heat conduction analysis: A case study in thermal engineering

Case Studies in Thermal Engineering 79 (2026) 107813

Contents lists available at ScienceDirect

Case Studies in Thermal Engineering

journal homepage: www.elsevier.com/locate/csite

Leveraging quantum computing for heat conduction analysis: A case study in thermal engineering

Pietro Asinari [a,b,*], Matteo Maria Piredda [a], Giulio Barletta [a], Paolo De Angelis [a], Nada Alghamdi [a], Giovanni Trezza [a,c], Marina Provenzano [a], Matteo Fasano [a], Eliodoro Chiavazzo [a,b]

[a] Dipartimento Energia, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino (TO), Italy
[b] Istituto Nazionale di Ricerca Metrologica, Strada delle Cacce 91, 10135, Torino (TO), Italy
[c] Université Grenoble Alpes, 1130 Rue de la Piscine, St Martin D'Heres, 38402, France

HIGHLIGHTS

- Explores quantum computing for simulating heat transfer in engineering.
- Uses heat conduction as a case study linking quantum computing and thermal science.
- Compares quantum circuits, packages, and optimization algorithms.
- Tests ideal, shot-based, and real executions on the IQM Lagrange quantum computer.
- Evaluates quantum hardware performance versus classical computational methods.

GRAPHICAL ABSTRACT

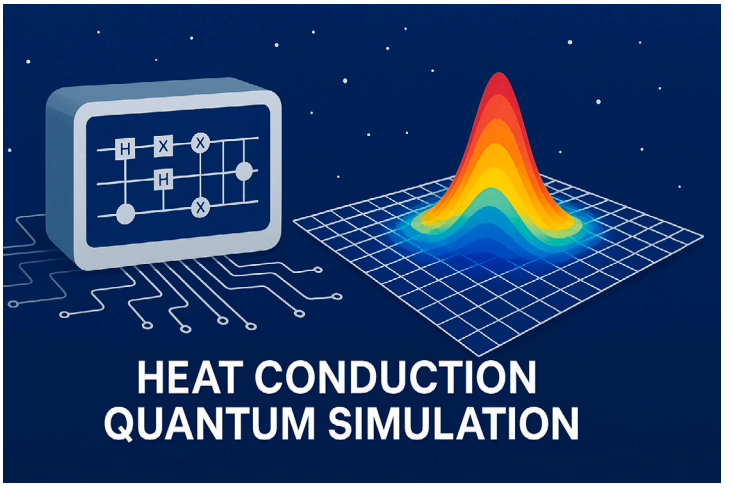


Image generated with the assistance of ChatGPT, an AI language model developed by OpenAI, under the CC BY 4.0 license.

ARTICLE INFO

Dataset link: https://github.com/SMaLL-PoliTo/QuantumThermal

Keywords:
Quantum computing

ABSTRACT

The rapid development of quantum computing is creating new opportunities in thermal sciences for the exploration and simulation of heat and mass transfer phenomena, turbulent flows, as well as plasma and multiphase flows in porous media. In particular, high-fidelity simulations of turbulent flows could benefit from quantum computing through the acceleration

* Corresponding author at: Dipartimento Energia, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino (TO), Italy.
E-mail address: pietro.asinari@polito.it (P. Asinari).

https://doi.org/10.1016/j.csite.2026.107813
Received 2 November 2025; Received in revised form 23 January 2026; Accepted 7 February 2026
Available online 10 February 2026

Thermal engineering
Heat transfer
Variational quantum eigensolver
Real noisy quantum hardware
Quantum–classical comparison

of linear algebra solvers. However, the practical performance of current quantum computers remains limited. Major challenges include noise and fault tolerance—or, at least, effective error mitigation—as well as the development of improved quantum and hybrid quantum–classical algorithms, the tuning of algorithms for specific problem classes, and the management of both cloud-based platforms and physical quantum hardware. This work presents a case study on heat conduction as a paradigmatic test case to promote the exploration of quantum computing for thermal science. Innovative algorithmic strategies are discussed, supported by numerical and experimental case studies, and the performance of real quantum hardware (IQM Lagrange at Torino) is compared with classical computational methods. In particular, preliminary investigations on this quantum hardware yielded a mean nodal outcome error of up to 14% for the problem under consideration. In the current Noisy Intermediate-Scale Quantum (NISQ) era, these numerical results obtained on real quantum hardware are preliminary but provide valuable insights for guiding further hardware development, an effort that also requires advanced metrology. The paper highlights opportunities, current challenges, and future directions, with a particular emphasis on engineering applications.

1. Introduction

Quantum computing has generated enormous expectations both within the scientific community and among the general public since its early realizations. This explains the substantial investments in this field by companies developing real quantum hardware. While progress is being made, the development of noise-resilient quantum computers capable of addressing a wide range of problems is expected to take another 10–15 years at least [1]. So far, researchers have obtained only theoretical proofs that quantum computers might provide significant advantages over current classical computers in: (i) simulating quantum physics and chemistry, (ii) breaking the current public-key cryptography used to secure sensitive communications, and (iii) solving problems in finance and logistics optimization [2]. Beyond chemistry, cryptography, and finance, another application of quantum computing generating considerable attention is (iv) machine learning [3]. All of these applications share a common feature: they rely on combinatorial problems that are difficult to tackle with classical hardware but could benefit from quantum hardware, as all possible options can be explored in parallel.

However, computational challenges in engineering applications at large extend well beyond combinatorial problems and could, in principle, benefit from quantum computing, once fault tolerance or, at least, error mitigation is achieved. In some engineering contexts, quantum algorithms for linear algebra and inference may eventually provide computational acceleration when integrated into hybrid classical–quantum workflows. In particular, the numerical solution of partial differential equations, such as those governing turbulent flows [4], heat and mass transfer [5], plasma in fusion reactors [6], multiphase flows in porous media for reservoir engineering [7], represents a major computational benchmark for quantum computing due to the need to repeatedly solve large, sparse linear systems. Materials science for engineering applications, including the design of catalysts and energy-storage materials, structural mechanics and inverse problems represent other meaningful examples, because they often involve high-dimensional and/or ill-posed computational challenges.

Among these engineering applications, we focus here on Computational Fluid Dynamics (CFD) for thermal engineering problems. High-fidelity simulation of turbulent flows (DNS, LES, and high-resolution RANS) is dominated by a small number of extremely costly numerical tasks: (i) the solution of very large sparse linear systems arising from discretized Navier–Stokes equations, pressure–velocity coupling, and implicit time stepping; (ii) the repeated evaluation of these solvers across many time steps and parameter variations; and (iii) the need for fine spatial and temporal resolution due to the multiscale nature of turbulence. The computational cost scales super-linearly with Reynolds number, making many practically relevant regimes inaccessible even on leadership-class supercomputers. Quantum computing could, in principle, contribute to turbulent flow simulation through acceleration of linear algebra subroutines, which dominate wall-clock time in CFD solvers.

In order to understand this point, let us consider a very simple argument. In most of the engineering applications, the computational domain is discretized by a spatial mesh with N nodes. State-of-the-art CFD simulations can utilize up to several hundreds of billion mesh cells on advanced supercomputers. For example, a study documented the use of a grid with 780 billion cells ($N_{\text{classic}} \sim 7.8 \times 10^{11}$) on Tianhe-2 supercomputer, leveraging over 1.376 million heterogeneous cores [8]. This state-of-the-art number of mesh nodes on classical computers could be significantly increased by an ideal quantum computer. A quantum computer with n qubits (see Appendix A for the fundamental concepts) has

$$N = 2^n, \tag{1}$$

computational basis states, analogous to those of a classical system. These basis states are given by all possible tensor products of single-qubit basis states. However—and this is the crucial difference—a quantum system can exist in a superposition (see Appendix A.2) of all these basis states, with each state weighted by a complex probability amplitude, which may also exhibit correlations. In multi-qubit systems, superposition can therefore give rise to correlations between qubits. These correlations become genuinely quantum when they cannot be expressed as products of independent single-qubit probability amplitudes. In this case, the quantum state is said to be entangled, and *entanglement* represents the first truly quantum phenomenon highlighted here (see Appendix A.3). The complex amplitudes associated with the quantum state can be manipulated by applying a sequence of elementary quantum

gates, in close analogy with logical gates in classical digital computing (see Appendix A.4). At this stage, a second distinctly quantum phenomenon emerges: different computational paths contribute amplitudes that can interfere with one another. Such *interference* may be constructive (positive) or destructive (negative), allowing desired outcomes to be amplified while suppressing incorrect ones (see Appendix A.5). The resulting amplitudes ultimately determine the probabilities of obtaining each computational basis state upon measurement. As a consequence, a quantum computer can encode and manipulate a number of real probabilities that is at least proportional to the number of computational basis states given by Eq. (1). This exponential scaling in the number of quantum states is what makes quantum systems particularly compelling compared to classical ones, because few qubits would be enough to ensure $N = 2^n \gg N_{\text{classic}}$. For example, $n = 50$ (ideal) qubits would be enough to realize $N \sim 1.1 \times 10^{15}$, which could eventually overcome the capability of Tianhe-2 supercomputer.

In spite of enormous expectations, the actual performance of current quantum computers remains limited by many unresolved challenges. First, we are currently in the Noisy Intermediate-Scale Quantum (NISQ) era of quantum computing, in which a clear quantum advantage has yet to be demonstrated in most practical applications [2]. Noise is a broad term encompassing all sources of error in a quantum processor, including decoherence (arising from system-environment interactions), control errors (imperfect pulses), crosstalk between qubits, measurement errors, and classical electronic noise, among others. While noise can be partially mitigated, a decisive breakthrough is expected only when NISQ devices are superseded by fault-tolerant, application-scale quantum (FASQ) computers, which exploit quantum error-correcting codes to enable a wide range of useful applications [9]. This transition, which is still in its infancy, remains highly uncertain, as the path from NISQ to FASQ is widely regarded as arduous, costly, and prolonged [9]. Secondly, though no less importantly, challenges related to quantum algorithms remain significant. Achieving effective synergy between classical computational resources and quantum hardware requires carefully designed hybrid quantum–classical algorithms, whose performance can be hindered by issues such as convergence difficulties in variational quantum algorithms [10]. Moreover, purely quantum algorithms must be robust against residual noise, even in fault-tolerant computers. Determining which hybrid quantum–classical or fully quantum algorithms are best suited to specific problem classes remains largely an open and insufficiently explored area of research. Finally, quantum run management continues to pose practical challenges, owing to the constraints imposed by cloud-based quantum services and the limited availability and maturity of current quantum hardware.

Given the above challenges, adapting established engineering software to the quantum computing paradigm represents a substantial effort, one that can be regarded as justified only if clear and unambiguous evidence of a genuine quantum advantage over classical approaches is demonstrated. Until such evidence is available, the investigation of quantum computing should be viewed primarily as an intellectually stimulating and scientifically valuable endeavor. At present, however, it is premature to assert that these methods will lead to practical or impactful outcomes, either for specific applications or for engineering practice more broadly. In this context, this work investigates the use of a real quantum computer for the numerical solution of the heat conduction equation, which serves as a simple yet paradigmatic test case for more advanced thermal engineering problems, with an emphasis on a practical, engineering-oriented approach.

This paper is organized as follows. In Section 2, the fundamentals of the heat conduction equation and its numerical solution are presented. Section 3 discusses the main steps of the algorithm called Variational Quantum Eigensolver (VQE). In Section 4, the quantum results are presented and analyzed—specifically, those obtained from ideal (statevector) simulations, ideal shot-based simulations, and runs on noisy real quantum hardware. Finally, Section 5 draws the main conclusions and includes a discussion of the challenges and future perspectives. In Appendix A, the fundamental concepts of quantum computing are reported, including basics of binary coding, qubit and superposition, multi-qubit systems and entanglement, gate-model quantum computing, quantum interference and quantum Hamiltonian. In Appendix B, the fundamental proof underlying the VQE methodology is presented. In Appendix C, the Pauli decomposition is presented because of its relevance to the VQE methodology. In Appendix D, the graphical representation of a single-qubit state in Hilbert space and on the Bloch sphere are presented and compared.

2. Heat conduction

2.1. One-dimensional heat conduction equation

Let us consider the one-dimensional heat conduction equation, as a simple yet paradigmatic test case, namely

$$\frac{\partial T}{\partial t} = D\,\frac{\partial^2 T}{\partial z^2}, \tag{2}$$

with the function $T = T(z,t)$ being the local temperature, z the space coordinate, t the time, and the positive coefficient D the thermal diffusivity of the medium. Let us consider a constant diffusivity, a given initial profile $T(z,0)$ and the periodic spatial boundary condition. This problem can be solved analytically using the Fourier transform and it is usually trivial for most of the current classical numerical techniques.

Let us solve the previous equation by the classical finite-difference (FD) method, which consists in solving differential equations by approximating derivatives with finite differences. Both the spatial domain and time domain are discretized by a regular mesh: the unknown function T is evaluated at the generic lth mesh node and at the τ-th time step, where both l and τ belong to $\mathbb{N}^+$, namely $T_l^\tau = T(z_l, t_\tau)$ where $z_l = l\,\Delta z$ with $0 \le l \le N-1$ and $t_\tau = \tau\,\Delta t$ with $0 \le \tau \le N_t$ ($\tau = 0$ identifies the given initial profile). The quantities Δz and Δt are the spatial and temporal partitions of the grid, while N is the number of space mesh nodes and N_t is the number of time steps. Let us use a fully implicit FD scheme that yields the stability of the solution for arbitrary diffusivity of the equation and the grid size:

$$\frac{T_l^{\tau+1} - T_l^\tau}{\Delta t} = D\,\frac{T_{l-1}^{\tau+1} - 2\,T_l^{\tau+1} + T_{l+1}^{\tau+1}}{\Delta z^2}. \tag{3}$$

The previous formula can be reformulated as

$$-r\,T_{l-1}^{\tau+1}+(1+2\,r)\,T_{l}^{\tau+1}-r\,T_{l+1}^{\tau+1}=T_{l}^{\tau}, \tag{4}$$

where $r=D\,\Delta t/\Delta z^2$ is the (dimensionless) numerical Fourier number. Let us define a new operator $\hat{C}$ as

$$\hat{C}:=\begin{bmatrix}(1+2\,r) & -r & 0 & 0 & \dots & -r\\ -r & (1+2\,r) & -r & 0 & \dots & 0\\ 0 & -r & (1+2\,r) & -r & \dots & 0\\ \dots & \dots & \dots & \dots & \dots & \dots\\ 0 & \dots & -r & (1+2\,r) & -r & 0\\ 0 & \dots & 0 & -r & (1+2\,r) & -r\\ -r & \dots & 0 & 0 & -r & (1+2\,r)\end{bmatrix}, \tag{5}$$

which can be used to formulate a linear system of equations which is consistent with Eq. (3). In particular, using the operator $\hat{C}$ defined by Eq. (5), Eq. (3) becomes in compact form

$$\hat{C}\,\vec{T}^{+}=\vec{T}, \tag{6}$$

where $\vec{T}^{+}$ stands for $\vec{T}^{\tau+1}=\left(T_0^{\tau+1},T_1^{\tau+1},T_2^{\tau+1},\dots,T_{N-1}^{\tau+1}\right)^T$ and $\vec{T}$ stands for $\vec{T}^{\tau}=\left(T_0^{\tau},T_1^{\tau},\ T_2^{\tau},\dots,T_{N-1}^{\tau}\right)^T$. Clearly the inverse of operator $\hat{C}$ can be used as a time-progress operator for the temperature profile subject to heat conduction, namely

$$\vec{T}^{+}=\vec{T}(t+\Delta t)=\hat{C}^{-1}\,\vec{T}, \tag{7}$$

which can be also generalized by $\tau\in\mathbb{N}^{+}$ in the following formula

$$\vec{T}(t+\tau\,\Delta t)=(\hat{C}^{-1})^{\tau}\,\vec{T}^{t=0}. \tag{8}$$

In the following sections, for the sake of simplicity and without loss of generality, we will focus on $\tau=1$ and hence on Eq. (7) only.

2.2. Given initial temperature profile as test case

The initial temperature profile $\vec{T}^{t=0}$ can excite different eigenmodes of the heat conduction equation. Consequently, the choice of the initial temperature profile—and thus its complexity—is an integral part of defining the test case. For the sake of simplicity, we consider an initial temperature profile $\vec{T}^{t=0}$ defined by a single harmonic, namely

$$T_{l}^{t=0}=T\,(l\,\Delta z,0)=1+\frac{1}{2}\sin\left[\frac{2\pi}{N}\,(l+1)\right]. \tag{9}$$

The previous profile can be considered an input of the test case, while the output is given by applying the time-progress operator $\hat{C}^{-1}$ given by Eq. (7) to the input profile $\vec{T}^{t=0}$. In other words, a single time step is performed to update the initial temperature profile $\vec{T}^{t=0}$ of the target problem. The expected results can be anticipated by referring to Fig. 6, which can be also helpful for visualizing the problem setup.

In the following sections, this simple test case is reformulated so as to be solvable using quantum computing methods.

3. Variational Quantum Eigensolver (VQE)

Quantum computing is deeply rooted in quantum physics and intimately connected with quantum information. Both considerations make clear that quantum computing must first overcome an educational barrier before becoming a widespread approach within the engineering community at large. Moreover, the rapid pace of progress in this field and its inherently cross-disciplinary nature make it difficult for newcomers to obtain a broad overview of the most important techniques and results, as foundational quantum concepts span physics, mathematics, computer science, and engineering [11].

Here we focus on solving a linear system of equations as a paradigmatic task for many engineering problems. Quantum algorithms for solving linear systems of equations, commonly referred to as Quantum Linear System Algorithms (QLSAs), constitute a key building block for quantum-enhanced engineering simulations. The canonical QLSA is the Harrow–Hassidim–Lloyd (HHL) [12] algorithm and its subsequent improvements, which provide an exponential speedup under sparsity and conditioning assumptions, but require deep, fault-tolerant quantum circuits. In this context, fault-tolerant refers to quantum algorithms that assume the availability of a fully error-corrected quantum computer, capable of running very long and precise quantum circuits without the computation being destroyed by noise. For near-term quantum hardware, variational quantum algorithms (VQAs) [10] have emerged as practical alternatives. Variational quantum algorithms are hybrid in nature, meaning that they combine quantum and classical computational resources: a parameterized quantum circuit is executed on the quantum hardware to prepare and measure quantum states, while a classical optimizer iteratively updates the circuit parameters based on the measurement outcomes. The term variational refers to the optimization of these parameters in order to minimize (or maximize) a cost function, typically defined as the expectation value of an observable. VQAs are particularly well suited for near-term, noisy intermediate-scale quantum (NISQ) hardware because they rely on relatively shallow quantum circuits, tolerate a certain level of noise, and offload computationally demanding optimization tasks to classical processors. Among these, the Variational Quantum Linear Solver (VQLS) [13] is specifically designed to approximate the solution of linear systems by minimizing a problem-tailored cost function in a hybrid quantum–classical optimization loop. While

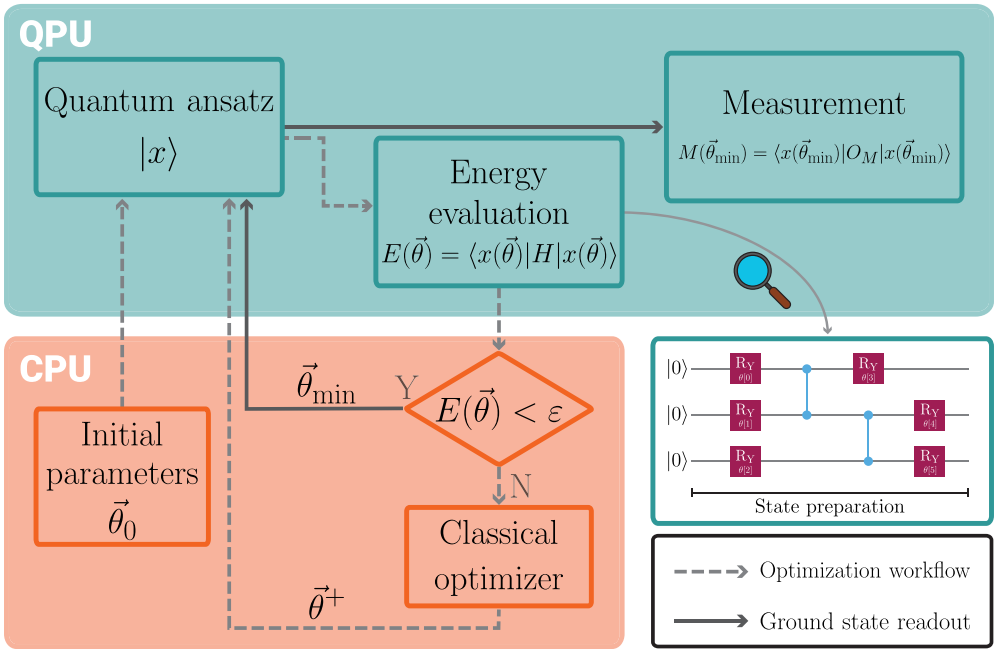


Fig. 1. Schematic representation of the key features of the VQE algorithm. The QPU denotes the quantum processing unit, while the CPU denotes the classical processing unit. The dashed lines represent the optimization loop, in which the CPU updates the parameters $\vec{\theta}$ and the QPU estimates the loss function $E(\vec{\theta}) \geq 0$, as defined in Eq. (19). When the estimated energy $E(\vec{\theta})$ falls below the tolerance ε, the optimization terminates. At this point, the measurement observable $\hat{O}_M$, defined in Eq. (28), can be evaluated through the readout workflow, indicated by the solid lines.

the Variational Quantum Eigensolver (VQE) [14,15] is also a prominent VQA, it is primarily aimed at eigenvalue problems and only indirectly applicable to linear systems through problem reformulation. However, in this work we focus on VQE, as it provides a more versatile package applicable to a broader class of problems, including eigenvalue and ground-state calculations that are central to materials modeling, while still allowing linear systems to be addressed through suitable reformulations, which are discussed in the following sections.

Before proceeding, it is important to note that VQAs, and the VQE in particular, suffer from intrinsic limitations. For instance, the VQE requires the evaluation of a cost (loss) function that is typically expressed as a weighted sum of elementary (Pauli) operators. The number of terms appearing in this decomposition generally grows exponentially with the number of qubits, which can lead to a substantial measurement overhead [16]. More sophisticated variational methods have been already proposed, based on evaluating the loss function by an adaptation of a fundamental quantum circuit, the so-called Hadamard test [17]. An even more effective implementation consists in combining the Hadamard test with the quantum Fourier transform [16] and adopting the so-called ansatz tree [16,18]. Despite these improvements, variational algorithms may suffer from convergence issues, i.e. the so-called barren plateaus [19], in which the parameter optimization landscape of the tentative solution becomes exponentially flat and featureless as the number of qubits increases. To mitigate its negative impact on finding the solution, for example, one could adopt local cost functions and alternating-layered quantum circuits [20]. However leveraging these concepts remains challenging for effectively scaling quantum simulations.

Setting aside these more advanced considerations, the basic VQE workflow is illustrated in Fig. 1 and can be summarized in five main steps:

1. Identification of the quantum state (Normalization);
2. Design of the quantum system (Observable);
3. Selection of the quantum parameterized trial solution (Quantum ansatz);
4. Minimization of the loss function (Optimization);
5. Extraction of useful results (De-normalization).

3.1. Normalization

The first step is to identify the quantum state where to store the relevant information by normalizing Eq. (6). Let us indicate the quantum states by the Dirac notation $|\cdot\rangle$, which stands for the standard notation for normalized vectors in quantum mechanics (see Appendix A.2) [21]. Because a discrete quantum state $|\psi\rangle$ is normalized, namely $\langle\psi|\psi\rangle = 1$, let us divide Eq. (6) by a factor

such that it becomes possible to identify a quantum state which depends on the temperature profile. In particular, let us choose this factor as follows:

$$\frac{1}{(\vec{T}^{+}\cdot\vec{T}^{+})(\vec{T}\cdot\vec{T})}\,\hat{C}\,\vec{T}^{+}=\frac{1}{(\vec{T}^{+}\cdot\vec{T}^{+})(\vec{T}\cdot\vec{T})}\,\vec{T}, \tag{10}$$

where $\vec{T}\cdot\vec{T}$ denotes the inner product (i.e., $\vec{T}^{\dagger}\cdot\vec{T}$, and since $\vec{T}$ is real-valued, $\vec{T}^{\dagger}$ reduces to the transpose of $\vec{T}$); similarly for $\vec{T}^{+}\cdot\vec{T}^{+}$. Let us define the quantum state $|b\rangle$ for mapping the initial temperature profile, namely

$$|b\rangle:=\frac{1}{\sqrt{\vec{T}\cdot\vec{T}}}\,\vec{T}, \tag{11}$$

where, by construction, $\langle b|b\rangle=1$. Before proceeding, let us be sure to appreciate the true meaning of the previous relation. Essentially, it implies that each node z_l of the computational mesh is associated with a quantum computational basis state $|j\rangle$ (where $|j\rangle\in\{0,1\}^{\otimes n}$ involves the binary representation of integer l and $\otimes$ denotes the tensor product, see Appendix A.3). Similarly let us proceed with the quantum state $|x\rangle$ for mapping the updated temperature profile at the new time step, namely

$$|x\rangle:=\frac{1}{\sqrt{\vec{T}^{+}\cdot\vec{T}^{+}}}\,\vec{T}^{+}, \tag{12}$$

where the same normalization holds. Introducing the definitions given by Eqs. (11) and (12) into Eq. (10) yields

$$\sqrt{\frac{\vec{T}^{+}\cdot\vec{T}^{+}}{\vec{T}\cdot\vec{T}}}\,\hat{C}\,|x\rangle=|b\rangle. \tag{13}$$

It is also possible to define a normalization factor f given by

$$f=\sqrt{\frac{\vec{T}^{+}\cdot\vec{T}^{+}}{\vec{T}\cdot\vec{T}}}, \tag{14}$$

and to derive the linear system of target equations as

$$\hat{A}|x\rangle=|b\rangle, \tag{15}$$

where $\hat{A}=f\,\hat{C}$. The problem is that the normalization factor is not known at the beginning of the numerical procedure because it depends on the solution $\vec{T}^{+}$, which derives from solving the linear system of equations. This means that $\hat{A}$ can be used to discuss the theoretical setup, but the practical numerical procedure must involve $\hat{C}$, because the latter depends only on the adopted FD formula.

As a closing remark of this step, it is worth highlighting that, if a quantum advantage is to be realized, the state $|b\rangle$ must be generated directly on the quantum computer. Indeed, the size of the computational basis of the quantum state, $N\gg N_{\text{classic}}$, would exceed the capabilities of any classical computer for state preparation or storage.

3.2. Observable

The second step is to derive a quantum system, which provides information relevant to solve the target problem. Let us follow the strategy proposed in Ref. [22] to construct a Hamiltonian (refer to Appendix A.6), which admits the quantum state $|x\rangle$ as the ground state. Although heat conduction is intrinsically a dissipative process and thus not Hamiltonian in the strict sense, we introduce a "Hamiltonian" here to refer to the matrix arising from the finite-difference discretization, which structurally resembles a Hamiltonian operator.[1] This formalism facilitates the following analysis. Applying the methodology described in Ref. [22] to Eq. (15) yields the following Hamiltonian

$$\hat{H}'=\hat{A}^{\dagger}\,(I-|b\rangle\langle b|)\,\hat{A}, \tag{16}$$

where $|\cdot\rangle\langle\cdot|$ denotes the outer product and $\hat{A}^{\dagger}$ is the conjugate transpose of $\hat{A}$ in general, while, in this case, $\hat{A}^{\dagger}=\hat{A}^{T}$, because $\hat{A}$ is real. In this case, $\hat{A}$ is also symmetric and therefore $\hat{A}^{T}=\hat{A}$. As pointed out before, let us formulate the quantum algorithm in terms of the practical operator $\hat{C}$ which does not involve the normalization factor. Since $\hat{A}^{T}=\hat{A}=f\,\hat{C}$, the previous Hamiltonian can be computed as $\hat{H}'=f^{2}\,\hat{H}$ where

$$\hat{H}:=\hat{C}^{T}\,(I-|b\rangle\langle b|)\,\hat{C}. \tag{17}$$

We remind that the Hamiltonian is just a special case of quantum observable (where energy is the actual observed quantity). It is possible to prove that the quantum state $|x\rangle$, in the linear system given by Eq. (15), also minimizes the observable $\hat{H}$ given by Eq. (17). See the proof reported in Appendix B. Hence the linear algebra task given by Eq. (15) is converted to the task of finding the ground state of the Hamiltonian $\hat{H}$ [22]. As already pointed out, in future quantum computing, also $\hat{H}$ must be generated directly on the quantum computer, because the size of the computational basis of the quantum state should exceed the capabilities of any classical computer.

[1] In an ideal quantum computer, where there is no interaction with the environment, state evolution is unitary and therefore reversible. The only source of irreversibility in such a system is measurement. The key idea, therefore, is to construct a reversible quantum evolution that, upon measurement, collapses onto the target solution of the dissipative dynamics. Hence the measurement process in quantum mechanics is often used as a way to model irreversible phenomena.

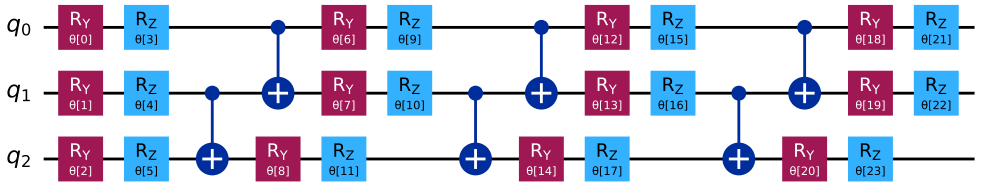


Fig. 2. Ansatz A1 (R_Y and R_Z, CX, linear entanglement, 3 and a half layers). For clarity, horizontal lines represent quantum wires which correspond to qubits in the circuit, red squares are the R_Y gates (see Eq. (31)), blue squares are the R_Z gates (see Eq. (32)), blue dots represent control points in controlled gates, $\oplus$ symbol is used for a controlled-X (CX or $CNOT$) gate. The latter gate explicitly guarantees the desired entanglement. More details about gate-model quantum computing are reported in Appendix A.4.

3.3. Quantum ansatz circuit

The third step is the quantum circuit ansatz, that is, a parameterized trial solution designed to approximate the ground state $|x\rangle$. In classical computational engineering, selecting an appropriate ansatz is analogous to designing a suitable mesh for discretizing the computational domain of partial differential equations. The ansatz selection involves defining its circuit topology, the degree of entanglement, and the number of tunable parameters. In general, increasing the circuit complexity enhances the ability to represent a wide range of solution profiles (expressivity), but it also leads to higher computational cost and reduced performance. The key point is that, in order to preserve a potential quantum advantage, there are too many elements in the vector $|x\rangle$ to work on them directly by a classical computer. Let us recall that VQE is a hybrid algorithm, where the optimization is supposed to be done by a classical computer [21]. Hence let us introduce a vector of parameters $\vec{\theta}$, which are fewer such that they can be handled by a classical computer. The quantum circuit ansatz enforces a parameterized state $|x(\vec{\theta})\rangle$ which makes possible to map these parameters on a generic quantum state. The parameterized state is the output of a unitary transformation (quantum gate), namely

$$|x(\vec{\theta})\rangle = U(\vec{\theta})\,|0\rangle^{\otimes n}, \tag{18}$$

where $|0\rangle^{\otimes n}$ stands for $|0\rangle \otimes |0\rangle \otimes \ldots |0\rangle = |0\rangle|0\rangle \ldots |0\rangle = |00\ldots0\rangle$ (the computational basis is always separable) and $\vec{\theta}$ is a vector of tunable parameters. As already pointed out, $\otimes$ denotes the tensor product. See Appendix A.3 for clarifying the meaning of the tensor product notation. The parameters $\vec{\theta}$ are typically generic "rotations" of qubits which are optimized during the minimization step of the VQE algorithm (see next Section 3.4). In order to preserve a potential quantum advantage, the number of parameters N_θ to optimize over in the ansatz circuit must be much less than the size of the computational basis of the quantum states N, namely $N_\theta \ll N$, because the former is handled by a classical optimizer/minimizer, while the latter exploits the full capability of the quantum computer. In other words, the number of parameters must grow as a polynomial in the number n of qubits, namely $N_\theta \sim n$, while the size of the full vector $|x\rangle$ is exponential in the number of qubits [23]. It is not important which polynomial describes the growth of the number of parameters, because any polynomial cannot compete with the growth of the exponential function 2^n for large n. In the following sections, for example, we will see $8\,n$ parameters in the ansatz depicted in Fig. 2 and $4\,n$ parameters in the simplified ansatz depicted in Fig. 4(b).

3.4. Optimization

The fourth step is the actual minimization of the loss function. A loss function quantifies the difference ("loss") between a quantum state predicted by the ansatz for a given input and the ground state. Taking into account Eq. (B.6) and the ansatz given by Eq. (18), the loss function can be defined as

$$L(\vec{\theta}) \equiv E(\vec{\theta}) := \langle x(\vec{\theta})|\hat{H}|x(\vec{\theta})\rangle \geq 0. \tag{19}$$

The optimal set of parameters can be formally defined by the argument of the minimization problem with regard to the loss function, namely

$$\vec{\theta}_{\min} := \arg\min_{\vec{\theta}} L(\vec{\theta}) = \arg\min_{\vec{\theta}} \langle x(\vec{\theta})|\hat{H}|x(\vec{\theta})\rangle. \tag{20}$$

Let us define $|x_{\min}\rangle$ as

$$|x_{\min}\rangle \equiv |x(\vec{\theta}_{\min})\rangle. \tag{21}$$

Because of numerical errors, $|x_{\min}\rangle$ is different from the theoretical ground state, i.e. $|x_{\min}\rangle \neq |x\rangle$, but it is usually considered a reasonable approximation, as far as the tunable parameters of the optimization procedure are properly selected.

3.5. De-normalization and measurement

The final step is to de-normalize the numerical quantum approximation $|x_{\min}\rangle$ to return to the original quantity of interest, i.e. the temperature. Let us start with the initial temperature profile. Let us define the auxiliary quantity

$$\eta = \sqrt{\vec{T} \cdot \vec{T}}, \tag{22}$$

which can be used to express Eq. (11) as $\vec{T} = \eta |b\rangle$. Let us compute this auxiliary quantity η by the spatial average of the initial temperature profile, namely

$$\eta = \frac{\sum_l T_l}{\sum_j \langle j|b\rangle}. \tag{23}$$

where $\sum_j \langle j|b\rangle$ is the sum of all real amplitudes in $|b\rangle$. Please remember that $|j\rangle \in \{0,1\}^{\otimes n}$ involves the binary representation of integer l. Similarly, recalling Eq. (12), let us define

$$\eta^+ = \sqrt{\vec{T}^+ \cdot \vec{T}^+}. \tag{24}$$

which can now be computed by using the quantum numerical approximation $|x_{\min}\rangle$. Taking advantage of the energy conservation, which implies

$$\sum_l T_l^+ = \sum_l T_l, \tag{25}$$

the quantity η^+ can be computed by the following formula

$$\eta^+ = \frac{\sum_l T_l}{\sum_j \langle j|x_{\min}\rangle}. \tag{26}$$

It is worth to note that, in case of accurate minimization, all terms in the summation at the denominator in Eq. (26) are positive because they correspond to the nodal values of the normalized new temperatures. Clearly $f = \eta^+/\eta$. The quantity η^+ is essential to de-normalize the quantum solution and to come back to the updated temperature profile, namely

$$\vec{T}^+ = \eta^+ |x_{\min}\rangle, \tag{27}$$

which is exactly what we are searching for.

As a closing remark of this step, it is worth emphasizing that, if a quantum advantage is to be realized, the state $|x_{\min}\rangle$ cannot be fully extracted from the quantum computer, as we do in this work for the sake of clarity. Indeed, estimating all amplitudes of $|x_{\min}\rangle$ would require an impractically large number of measurements. Moreover, the size of the computational basis of the quantum state $|x_{\min}\rangle$ would exceed the storage capabilities of any classical computer. Hence, the idea is to perform a limited number of measurements of the following form:

$$M := \langle x_{\min}|\hat{O}_M|x_{\min}\rangle, \tag{28}$$

where $\hat{O}_M$ is the measurement observable. A possible choice for $\hat{O}_M$ is an observable that extracts a single nodal temperature value (with this procedure repeated for all relevant temperatures). Alternatively, one may design an observable that estimates the relevant thermal flux for the application under consideration. The measurement observable $\hat{O}_M$ is used in the readout workflow, indicated by the solid lines, in Fig. 1.

In this work, however, for the sake of clarity and given the small dimensionality of the problem, we compute all components of the statevector, namely the complete temperature profile.

3.6. Practical details of implementation

In this Section, we complete the previous Section by providing additional details about the implementation of the algorithm on a quantum computer.

3.6.1. Decomposition in Pauli matrices

In the presented algorithm, the loss function to be minimized $L(\vec{\theta})$ is defined by the expectation value of the observable $\hat{H}$ defined by Eq. (17). The evaluation of the cost function relies on expressing the target Hamiltonian in a form compatible with the capabilities of quantum hardware. In practice, quantum processors can directly measure only simple observables, typically local projective measurements in a fixed basis. For this reason, any physical Hamiltonian acting on qubits is decomposed into a linear combination of experimentally measurable observables, namely tensor products of Pauli operators (see Appendix C). Hence, to measure the observable $\hat{H}$ given by Eq. (17) on a quantum computer, one must typically represent it as a sum of tensor products of Pauli matrices, that is

$$\hat{H} = \sum_{p=0}^{N_P-1} \gamma_p \hat{P}_p, \tag{29}$$

where N_P is the number of terms in the Pauli decomposition of the Hamiltonian, $\hat{P}_p$ belongs to the set $\{I, X, Y, Z\}^{\otimes n}$ and the Pauli elementary matrices [21] are

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{30}$$

The coefficients γ_p are obtained via the Hilbert–Schmidt inner product [21]. In our case, because $\hat{H}$ is real and symmetric, then $\hat{H} = \hat{H}^\dagger$ is Hermitian and consequently $\gamma_p \in \mathbb{R}$. More details can be found in Appendix C. The key point is that this Pauli decomposition provides an exact representation of the Hamiltonian and allows its expectation value to be reconstructed from a set of elementary measurements, at the expense of an increased measurement overhead when many non-commuting terms are present.

3.6.2. Quantum ansatz circuits

In the presented algorithm, the loss function to be minimized $L(\vec{\theta})$ is defined with regard to a parameterized trial solution $|x(\vec{\theta})\rangle$, which is called the ansatz. The ansatz given by Eq. (18) is a parameterized trial solution $|x(\vec{\theta})\rangle$ which should be able to approximate the ground state $|x\rangle$. The parameterized solution is the output of a unitary transformation (quantum gate) $U(\vec{\theta})$, which depends on a vector $\vec{\theta}$ of N_θ tunable parameters. Naively, one would like to have a procedure for correlating the parameters in $\vec{\theta}$ with the real amplitudes in $|x(\vec{\theta})\rangle$ by means of some analytical formulas. This approach is usually called real data loading or better encoding, and some algorithms have been proposed in literature [24]. For optimization problems—and for VQE in particular—real data loading/encoding does not make sense (because it must be $N_\theta \ll N$) and it will be omitted here in favor of an approach with less tunable parameters.

The unitary transformation $U(\vec{\theta})$ is defined by heuristic quantum circuits (also called patterns), which experimentally proved to be effective in preparing trial states for variational quantum algorithms or in realizing classification for machine learning. More details about gate-model quantum computing, quantum circuit diagrams and most important gates are reported in Appendix A.4. There are typically four ingredients that determine the unitary transformation: (i) single-qubit operations (either a combined rotation gate $R_Y R_Z$ or a simple rotation gate R_Y); (ii) entanglement blocks (controlled gate CX or CZ); (iii) entanglement type (full, linear, or ring-type); (iv) number of repeated layers (either complete layers or layers without final rotations). By combining these ingredients, different ansatz circuits (A1, A2, A3, A3p and A4) can be constructed, which are described in the following. The general concepts underlying these designs are summarized first:

- A simple rotation gate R_Y is sufficient to represent real-valued amplitudes, as in our case. However, the combined rotation gate $R_Y R_Z$ is often included to provide additional flexibility during the optimization process.
- The controlled-Z (CZ) gate is symmetric, meaning that the control and target qubits can be exchanged without affecting the operation [21]. In contrast, the controlled-X (CX), or $CNOT$, gate is asymmetric in the computational basis.
- Full entanglement exploits the maximum degrees of freedom available in a quantum circuit, but it quickly becomes difficult to manage as the system size increases. Linear entanglement, on the other hand, couples adjacent qubits in a sequential manner. Ring-type entanglement extends the linear configuration by adding an additional coupling between the last and the first qubits.
- A single layer consists of one rotation step followed by one entanglement step. Therefore, when multiple layers are used, they typically end with an entanglement step and do not include final rotations. If a final set of rotations is added to fine-tune the amplitudes, this effectively corresponds to using half of an additional layer.

Ansatz A1 (R_Y and R_Z, CX, linear entanglement, 3 and a half layers). This ansatz is called "*EfficientSU2*" circuit in Qiskit [23] and it is plotted in Fig. 2 for a system with 3 qubits. More details about gate-model quantum computing are reported in Appendix A.4. In gate-model quantum computing, it is crucial to manage properly the computer memory, which requires that the endianness must be specified—that is, how bits or qubits are stored in memory. In this paper, the little-endian convention is used [21], meaning that the least significant bit (LSB, i.e. the bit representing the smallest place value 2^0) is stored at the lowest memory address. Consequently, in circuit diagrams, as the one reported in Fig. 2, the topmost qubit represents the LSB and the bottom qubit represents the most significant bit (MSB). The single-qubit operations consist of the sequential application $R_Y R_Z$ (in this case, there is no tensor product implied because both apply to the same qubit) of a R_Y gate and a R_Z gate, defined as

$$R_Y(\theta_Y) = \exp\left(-i\,\frac{\theta_Y}{2}\,Y\right) = \begin{pmatrix} \cos(\theta_Y/2) & -\sin(\theta_Y/2) \\ \sin(\theta_Y/2) & \cos(\theta_Y/2) \end{pmatrix}, \tag{31}$$

and

$$R_Z(\theta_Z) = \exp\left(-i\,\frac{\theta_Z}{2}\,Z\right) = \begin{pmatrix} \exp(-i\,\theta_Z/2) & 0 \\ 0 & \exp(i\,\theta_Z/2) \end{pmatrix}. \tag{32}$$

The previous definitions can be thought of as derived from the same generic formula

$$\exp\left(-i\,\theta_\sigma\,\sigma\right) = \cos\theta_\sigma\, I - i\,\sin\theta_\sigma\,\sigma, \tag{33}$$

where θ_σ is a parameter and σ is a matrix which can be the Pauli matrix Y or Z, taking into account that $\sigma^2 = I$. The previous generic formula derives from the property of Pauli matrices (after multiplication by $i = \sqrt{-1}$ to make them anti-Hermitian) to generate transformations in the sense of Lie algebras [21]. This formula is analogous for Pauli matrices to the Euler's formula of complex analysis. The entanglement is realized by controlled NOT gates (also called controlled-X gates or CX) [21] in a linear arrangement. The linear arrangement means that qubits are entangled in a "linear chain" pattern, rather than all-to-all entanglement

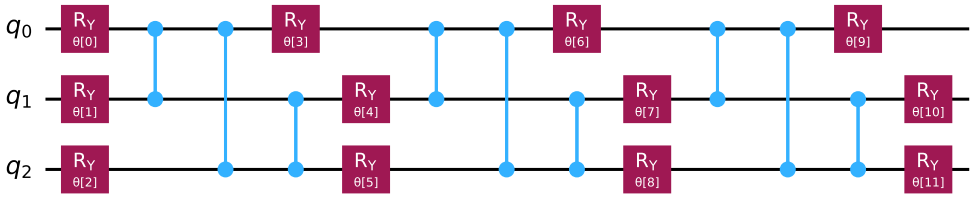
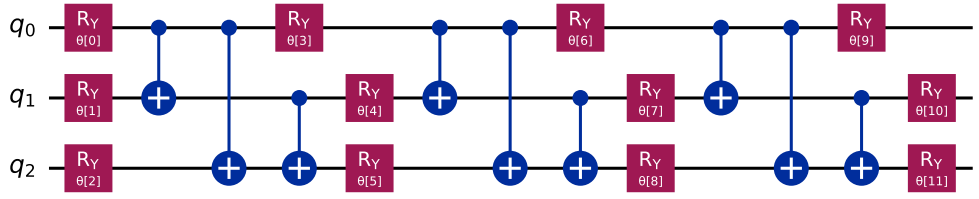
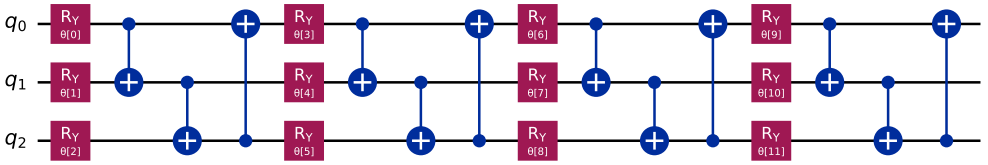


Fig. 3. Ansatz A2 (R_Y, CZ, full entanglement, 3 and a half layers). For clarity, horizontal lines represent quantum wires which correspond to qubits in the circuit, red squares are the R_Y gates (see Eq. (31)), connected couple of blue dots represents a (symmetric) controlled-Z (or CZ) gate. The latter gate explicitly guarantees the desired entanglement.



(a) Ansatz A3



(b) Ansatz A3p

Fig. 4. Ansatz A3 and A3p (R_Y, CX, full entanglement, 3 and a half layers for A3 and 4 layers for A3p). For clarity, horizontal lines represent quantum wires which correspond to qubits in the circuit, red squares are the R_Y gates (see Eq. (31)), blue dots represent control points in controlled gates, $\oplus$ symbol is used for a controlled-X (CX or $CNOT$) gate. The difference between A3 in (a) and A3p in (b) is due to the actual topology of the full entanglement.

which is more complex to realize on real quantum hardware. The combination of single-qubit operations and linear entanglement forms a fundamental layer of the circuit. This layer is repeated three and a half times, meaning that the final repetition includes only the single-qubit operations. This ansatz implies, for a system with $n = 3$ qubits, $N_\theta = 24$ because there are four sequences of single-qubit operations with six parameters each (two gates R_Y and R_Z for each qubit) or, equivalently, eight parameters per qubit. For the sake of comparison, for $n = 4$ the number of parameters in this ansatz becomes $N_\theta = 8\,n = 32$. It is essential in VQE that the number of ansatz parameters to optimize over is linear as in this case (or polynomial at worst) in the number of qubits, in order to ensure a potential quantum advantage (because it must be $N_\theta \sim k_\theta\, n \ll 2^n = N$, where k_θ is a proper constant).

Ansatz A2 (R_Y, CZ, full entanglement, 3 and a half layers). In this ansatz, which is plotted in Fig. 3, the single-qubit operations consist solely of the sequential application of the R_Y gate, reducing the number of parameters to four per qubit. Moreover the entanglement is realized by controlled-Z (or CZ) gates, which are symmetric with respect to the exchange of its qubits and this is why they are represented by a connected couple of blue dots (which is symmetric) in Fig. 3. Controlled-Z gates are sometimes considered easier to be realized by real quantum hardware (because there is no preference in the direction of information). Finally, this ansatz realizes full entanglement (sometimes called full connectivity or all-to-all entanglement) because it is a circuit in which every qubit can become entangled with every other qubit.

Ansatz A3 and A3p (R_Y, CX, full entanglement, 3 and a half layers for A3 and 4 layers for A3p). These ansatz circuits, which are plotted in Fig. 4, are sort-of hybrid circuits of the previous ones: they have simplified single-qubit operations (as A2) but with controlled-X (or CX) gates for the entanglement (as A1). There are two variants, namely A3 reported in Fig. 4(a) and A3p in Fig. 4(b), which slightly differ because of the actual topology of the full entanglement.

Ansatz A4 (R_Y, CZ, linear entanglement, 1 and a half layer). This ansatz, illustrated in Fig. 5, is designed for implementation on real quantum hardware. The single-qubit operations consist solely of the sequential application of two R_Y gates, resulting in two parameters per qubit. Entanglement is implemented using controlled-Z (CZ) gates, which are generally easier to realize on actual quantum devices. A linear qubit arrangement is adopted to restrict entanglement to neighboring qubits only.

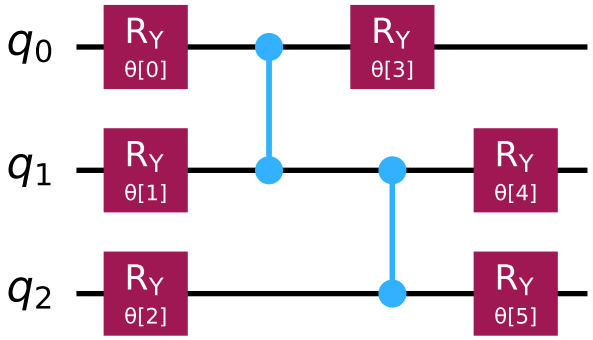


Fig. 5. Ansatz A4 (R_Y, CZ, linear entanglement, 1 and a half layer). For clarity, horizontal lines represent quantum wires which correspond to qubits in the circuit, red squares are the R_Y gates (see Eq. (31)), connected couple of blue dots represents a (symmetric) controlled-Z (or CZ) gate. The latter gate explicitly guarantees the desired entanglement.

3.6.3. Quantum simulations and executions

It is important to highlight that, in quantum computation, simulations and executions can be broadly categorized into three types, depending on how measurement and noise are treated.

- Ideal simulation (statevector simulation) refers to a noise-free, deterministic calculation of the quantum state's evolution. In this case, the full statevector $|\psi\rangle$ is computed exactly, and measurement probabilities are derived analytically as $\langle j|\psi\rangle^2$, corresponding to an effectively infinite number of measurement samples.
- Ideal shot-based simulation introduces statistical sampling into this otherwise ideal setting. Here, the simulator draws a finite number N_S of measurement outcomes ("shots") from the theoretical probability distribution, mimicking the discrete, probabilistic nature of real quantum measurements while still neglecting any physical noise.
- Noisy simulation (real hardware run) incorporates the effects of noise and imperfections, by executing the circuit on an actual quantum processor. In this regime, results reflect both statistical fluctuations from finite sampling and physical noise from gate errors, decoherence, and measurement inaccuracies.

Together, these three regimes establish a hierarchy of realism—ranging from exact theoretical predictions to fully experimental implementations—useful for algorithm development, benchmarking, and hardware validation.

3.6.4. Software packages

In this work, we aimed to keep the discussion of solving the heat conduction equation through quantum computing as general as possible. Nevertheless, since the ultimate objective is to provide case studies suitable for performance analyses and benchmarking, certain practical aspects of the implementation necessarily depend on the specific features of the software packages employed. To ensure an unbiased and neutral assessment, the case studies were implemented using three software packages for quantum computing: Qiskit (Quantum Information Software Kit) by IBM [23], Qrisp by Fraunhofer [25], Quantum Matcha Tea (Quantum MAny Target quantum Circuit Hpc App) by University of Padua [26]. Qiskit, developed by IBM [23], is an open-source package for designing, simulating, and executing quantum circuits on both local simulators and IBM Quantum hardware, providing a comprehensive Python-based ecosystem for algorithm development and hardware integration. Qrisp, developed by the Fraunhofer Institute [25], offers a high-level, modular environment for quantum programming that emphasizes flexibility and interoperability with classical computing resources. Quantum Matcha Tea, developed at the University of Padua [26], is a research-oriented package focused on the high-performance simulation of quantum circuits and hybrid quantum–classical workflows on distributed architectures. Together, these tools illustrate the diversity of current approaches to quantum software development, ranging from industrial platforms to research-driven, high-performance computing applications.

3.6.5. Minimization of the loss function

VQE is a hybrid algorithm that combines (i) classical operations for the converging iterations and (ii) loss function evaluations by quantum operations, to find the ground state of the target quantum system, which is designed in our case to update the one-dimensional temperature profile consistently with the heat conduction equation. For the converging iterations by classical operations, there are several classical optimization algorithms which can be also employed to minimize the cost function associated with variational quantum circuits. In this work, we consider four representative methods: *COBYLA*, *COBYQA*, *BFGS*, and *SPSA*. *COBYLA* (Constrained Optimization BY Linear Approximations) and *COBYQA* (Constrained Optimization BY Quadratic Approximations) are derivative-free algorithms that iteratively construct local linear or quadratic models of the objective function, respectively, and are particularly suited for noisy or hardware-based evaluations. *BFGS* (Broyden–Fletcher–Goldfarb–Shanno) is a quasi-Newton gradient-based method that estimates the Hessian to accelerate convergence on smooth cost landscapes. In contrast, *SPSA* (Simultaneous Perturbation Stochastic Approximation) efficiently estimates gradients through random perturbations of all parameters

simultaneously, making it highly effective for high-dimensional and noisy quantum optimization tasks. These algorithms provide a representative balance between robustness to measurement noise and convergence efficiency in variational quantum algorithms. These classical methods are available in SciPy, which is an open-source Python library that provides a large collection of numerical algorithms and scientific computing tools [27].

In spite of these state-of-the-art tools, a VQE algorithm may converge to a local rather than a global minimum. This behavior arises from the intrinsic non-convexity of the variational loss function $L(\vec{\theta})$ given by Eq. (19), which typically exhibits multiple local minima, flat regions (i.e. barren plateaus), and oscillatory features due to the structure of the parameterized quantum circuit. Moreover, the expressivity of the chosen ansatz and parameter redundancies can further restrict exploration of the parameter space, preventing convergence to the true global optimum. For this reason, it is a good practice to initialize the VQE parameters $\vec{\theta}$ by random values. However, when statistical sampling is involved, finite-shot fluctuations may additionally distort the estimated gradients, amplifying the risk of premature convergence. Consequently, local-minimum trapping is a fundamental characteristic of variational quantum optimization, even under ideal simulation conditions. In this case, the good news is that we know—by construction—that the minimum of $L(\vec{\theta})$ is equal to zero. Hence, it is possible to introduce a sanity check within the minimization loop: in cases of local-minimum trapping, defined as situations where the final loss function L_f remains larger than a given threshold after the selected number of iterations N_{IT} is reached, the minimization procedure is simply randomly initialized again and repeated. Including such quality control in the minimization loop realizes an enhanced minimizer, which is quite effective.

3.6.6. Error norms

Regardless of whether we are referring to simulations or actual executions, it is essential to characterize case studies using an appropriate error norm. Let us recall the numerical outcome of the quantum computation, given by Eq. (27), and let us extract the corresponding nodal outcome as

$$T_j^+ = \eta^+ \langle j | x_{\min} \rangle. \tag{34}$$

This nodal result can be compared with the reference nodal value $\overline{T}_j^+$ (ground truth) in order to define the relative nodal error and consequently the mean of the relative nodal errors as

$$\mathbb{M}_e = \frac{1}{N} \sum_j \frac{\left\| T_j^+ - \overline{T}_j^+ \right\|}{\overline{T}_j^+}, \tag{35}$$

which clearly depends on the number of qubits of the system (by means of the number of mesh nodes N).

In shot-based and noisy simulations, it is necessary to account for the fact that N_R repetitions are required to obtain statistically reliable results. Each repetition, denoted by r, yields multiple values for the nodal outcomes, which collectively form the matrix T_{rj}^+. To quantify the accuracy of the numerical solution, the mean nodal relative error $\mathbb{M}_E$ may be defined as

$$\mathbb{M}_E = \frac{1}{N_R} \sum_{r=0}^{N_R-1} \frac{1}{N} \sum_j \frac{\left\| T_{rj}^+ - \overline{T}_j^+ \right\|}{\overline{T}_j^+}. \tag{36}$$

This metric is rather stringent, as it does not fully exploit the information contained in the ensemble of simulations. A more representative indicator of accuracy is the error of the mean nodal values $\mathbb{E}_M$, defined as

$$\mathbb{E}_M = \frac{1}{N} \sum_j \frac{\left\| (1/N_R) \sum_{r=0}^{N_R-1} T_{rj}^+ - \overline{T}_j^+ \right\|}{\overline{T}_j^+}. \tag{37}$$

By virtue of Jensen's inequality—since the absolute value is a convex function—the error of the mean nodal values $\mathbb{E}_M$ is necessarily smaller than or equal to the mean nodal relative error $\mathbb{M}_E$, that is,

$$\mathbb{E}_M \leq \mathbb{M}_E. \tag{38}$$

These relative errors $\mathbb{M}_e$, $\mathbb{M}_E$ and $\mathbb{E}_M$ are used to characterize the proposed case studies and are reported in Table 1.

4. Quantum results

4.1. Case studies and numerical results

Solving heat conduction equation, despite its apparent simplicity, poses a significant challenge given the current state of quantum computing and is valuable for assessing the performance of real quantum hardware and algorithms. Moreover, as quantum hardware is evolving rapidly, it remains unclear which hardware and approaches will ultimately be most suitable for solving linear systems of equations. For these reasons, we believe it is important to focus on case studies that are flexible enough to be implemented across different hardware and software packages. In this Section, we present multiple implementations of the proposed case studies across a wide variety of software packages, reflecting the efforts of different research communities worldwide, including Qiskit in the USA,

Table 1
Case studies for general overview. Notation: (3q) means that the results refer to 3 qubits; (4q) means 4 qubits; (5q) means 5 qubits. If not specified otherwise, (3q) is intended. Relative errors $\mathbb{M}_e$, $\mathbb{M}_E$ and $\mathbb{E}_M$ are defined in Eqs. ((35), (36), (37)), respectively. When not explicitly stated, $\mathbb{M}_e$ is intended. Enhanced minimizer means that there is a quality check for avoiding local-minimum trapping ($L_f \leq 0.5$). The notation $(*)$ denotes that the maximum number of iterations is reached (10^6).

Case study	Ansatz	Parameters per qubit	Execution (ideal/shot-based/noisy)	Package	Minimizer (simple/ enhanced)	Number of repetitions N_R	Relative errors $\mathbb{M}_e$, $\mathbb{M}_E$, $\mathbb{E}_M$
#1	A1	8	ideal	Qiskit	*COBYLA* (10^{-3} tol.)	1	(3q) 3.22×10^{-3}
					COBYLA (10^{-6} tol.)		(3q) 1.95×10^{-6} (4q) 7.68×10^{-3} $(*)$
#2	A2	4	ideal	Quantum Matcha TEA	*BFGS* (with default tolerance)	1	(3q) 1.47×10^{-7} (4q) 5.80×10^{-6} (5q) 2.35×10^{-5}
#3.1	A3	4	shot-based	Qiskit Aer	*SPSA* ($L_f \leq 0.5$, 1024 shots)	30	$\mathbb{M}_E$: 5.48×10^{-2} $\mathbb{E}_M$: 2.13×10^{-2}
					SPSA ($L_f \leq 0.5$, 4096 shots)		$\mathbb{M}_E$: 6.87×10^{-2} $\mathbb{E}_M$: 1.20×10^{-2}
#3.2	A3p	4	shot-based	Qrisp	*COBYQA* (10^{-3} tol.)	30	$\mathbb{M}_E$: 3.40×10^{-2} $\mathbb{E}_M$: 1.04×10^{-2}
					COBYQA (5×10^{-4} tol.)		$\mathbb{M}_E$: 2.58×10^{-2} $\mathbb{E}_M$: 4.58×10^{-3}
#4 (real)	A4	2	noisy (IQM Lagrange)	Qiskit	*SPSA* (1024 shots)	1	(3q) 1.38×10^{-1} meaning 14%

Qrisp in Germany, and Quantum Matcha Tea in Italy. The datasets and the custom simulation code used in this study are publicly available at GitHub[2] (see also the Section about data availability).

In this work, we propose a set of case studies which are described in Tables 1–3. The central objective is to present a representative set of case studies that facilitate the investigation of different ansatz circuits, simulation and execution approaches, as well as the unique characteristics of distinct software packages. Starting from each software package, we selected—based on the available documentation—the most commonly used ansatz circuits for general-purpose variational problems, together with the most suitable and effective optimizers for that package. In particular, Table 1 reports the ansatz circuit and the corresponding number of tunable parameters (see Section 3.6.2), the execution type (see Section 3.6.3), the software package (see Section 3.6.4), the adopted minimizer (see Section 3.6.5) and the error norms (see Section 3.6.6). Table 2 reports ideal simulations for investigating the effect of adopted ansatz. Table 3 reports investigations about the effect of the number of shots on the results of shot-based simulations.

The numerical results of ideal simulations are visualized in Fig. 6 for case study #1 and in Fig. 7 for case study #2. The numerical results of shot-based simulations are visualized in Fig. 8 for case study #3.1 and in Fig. 9 for case study #3.2. Finally, numerical results obtained by real noisy quantum hardware are visualized in Fig. 10.

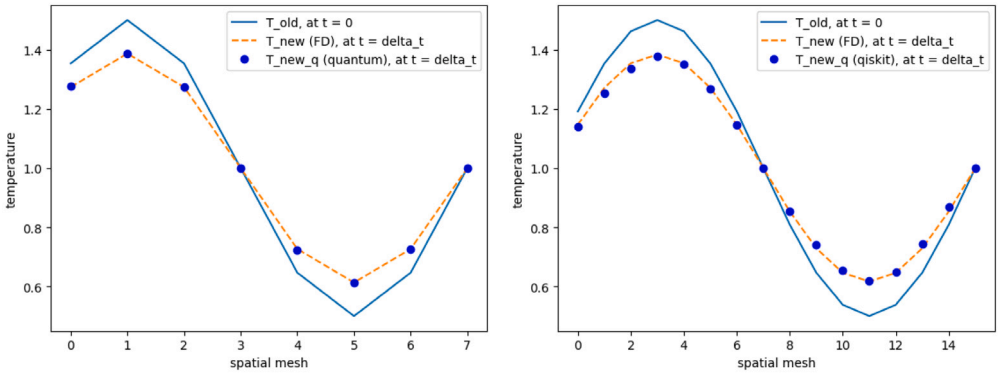
4.2. Discussion

4.2.1. Ideal (statevector) simulations

Let us start with ideal simulations. Ideal simulation (statevector simulation) refers to a noise-free, deterministic calculation of the quantum state's evolution. In this case, the full statevector $|\psi\rangle$ is computed exactly, and measurement probabilities are derived analytically as $\langle j|\psi\rangle^2$, corresponding to an effectively infinite number of measurement samples. Hence, an ideal (statevector) simulation is actually a classical simulation, without statistical uncertainty or decoherence. A real quantum computer can prepare and evolve a quantum state, but it cannot access the full statevector, because quantum measurement is probabilistic by nature. To estimate the probability of a state, or any observable, one must perform many repetitions (shots). On a real quantum computer, ideal (noise-free, deterministic) simulations are impossible because they would require an infinite number of measurements to reconstruct the full quantum state with infinite precision. However these ideal simulations are still useful to check the consistency of the algorithms in solving a problem of interest and they will be documented here for this goal.

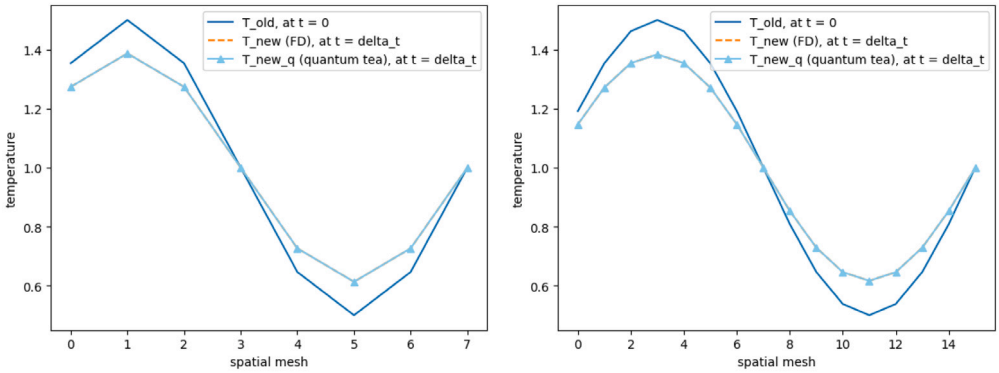
We first consider the numerical results obtained from statevector simulations of the VQE methodology applied to the test case presented in Section 2. This configuration corresponds to case study #1 in Table 1, for which additional details are provided and which is illustrated in Fig. 6. This case study is implemented using the ideal Qiskit package (see Section 3.6.4 for details) and optimized with the *COBYLA* algorithm (see Section 3.6.5), using convergence tolerances of 10^{-3} and 10^{-6} for the three- and four-qubit cases, respectively. In Fig. 6(a), the results for one time-step update of the temperature profile according to the heat conduction equation are reported in case of $n = 3$ qubits ($N = 8$). This minimization required 3523 evaluations of the loss function, which is

[2] https://github.com/SMaLL-PoliTo/QuantumThermal



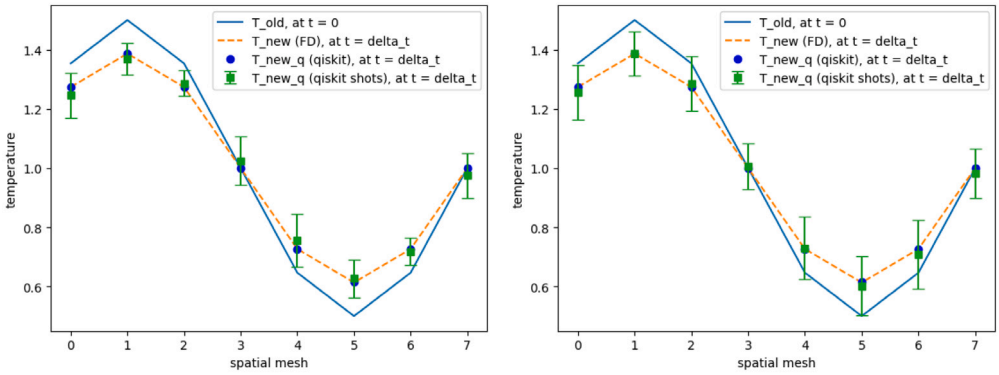
(a) $n = 3$ qubits (b) $n = 4$ qubits

Fig. 6. Case study #1 (see Table 1 for details): One time-step update of the temperature profile according to heat conduction equation by quantum computing with $n = 3$ qubits in panel (a) and with $n = 4$ qubits in panel (b). The blue line is the initial temperature profile (sinusoidal profile with mean equal to 1 given by Eq. (9)), the orange dashed line is the new temperature profile at time Δt, computed by finite-difference (FD) method. The markers are the mesh temperatures computed by the ideal quantum simulator of the Qiskit package.



(a) $n = 3$ qubits (b) $n = 4$ qubits

Fig. 7. Case study #2 (see Table 1 for details): One time-step update of the temperature profile according to heat conduction equation by quantum computing with $n = 3$ qubits in panel (a) and with $n = 4$ qubits in panel (b). The blue line is the initial temperature profile (with mean equal to 1), the orange dashed line is the new temperature profile at time Δt, computed by finite-difference method. The markers are the mesh temperatures computed by the ideal quantum simulator of the Quantum Matcha TEA package.



(a) 1024 shots (b) 4096 shots

Fig. 8. Case study #3.1 (see Table 1 for details): One time-step update of the temperature profile according to heat conduction equation. Results by 1024 shots in panel (a) and results by 4096 shots in panel (b). The markers are the mesh temperatures computed by the shot-based quantum simulator Qiskit Aer. The error bars corresponding to standard deviations are also reported.

Table 2
Ideal simulations for investigating the effect of adopted ansatz. The simulations are done by Qiskit software package. Notation: (3q) means that the results refer to 3 qubits; (4q) means 4 qubits. Relative error $\mathbb{M}_e$ is defined in Eq. (35). The minimizer tolerance is 10^{-6}. The reported CPU times are indicative, as they exhibit slight variations between different runs. The notation $(*)$ denotes that the maximum number of iterations is reached (10^6).

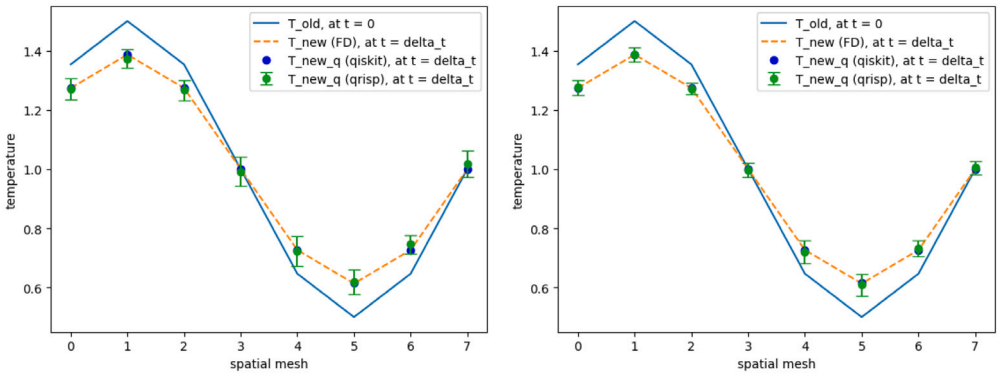
Case study	Ansatz	Minimizer	CPU time	Relative error, $\mathbb{M}_e$
#1.1	A1	*COBYLA*	21.1 s	(3q) 1.95×10^{-6}
		COBYLA	4 h (*)	(4q) 7.68×10^{-3} (*)
		BFGS	6.4 s	(3q) 4.00×10^{-7}
		BFGS	11.5 min	(4q) 2.20×10^{-6}
#2.1	A2	*COBYLA*	8.6 s	(3q) 1.44×10^{-6}
		BFGS	1.2 s	(3q) 9.21×10^{-8}
		BFGS	1.4 min	(4q) 2.01×10^{-5}
#3.1.1	A3	*COBYLA*	8.6 s	(3q) 2.82×10^{-6}
		BFGS	1.3 s	(3q) 1.67×10^{-7}
		BFGS	1.5 min	(4q) 1.37×10^{-6}
#3.2.1	A3p	*COBYLA*	5.1 s	(3q) 6.81×10^{-7}
		BFGS	1.5 s	(3q) 5.30×10^{-7}
		BFGS	1.8 min	(4q) 2.78×10^{-2}
#4.1	A4	*COBYLA*	25.8 s	(3q) 4.13×10^{-2}
		BFGS	1.7 s	(3q) 4.13×10^{-2}

Table 3
Shot-based simulations for investigating the effect of the number of repetitions. The simulations are done by Qiskit Aer software package. The number of qubits is equal to 3. The enhanced minimizer is based on $SPSA$ method. The loss function L_f is the final value after the selected number of iterations ($N_{IT} = 1000$) is reached: if L_f is larger than the reported threshold, then the minimization is re-initialized and repeated. The mean relative error $\mathbb{M}_E$ is defined in Eq. (36), while the relative error of the mean outcomes $\mathbb{E}_M$ is defined in Eq. (37). The reported CPU times are indicative, as they exhibit slight variations between different runs.

Case study	Ansatz	Number of shots	Number of repetitions	CPU time	Relative errors
#3.1.2	A3	1024	30	1.1 h	$\mathbb{M}_E : 5.48 \times 10^{-2}$
		($L_f \leq 0.5$)			$\mathbb{E}_M : 2.13 \times 10^{-2}$
		1024	30	3.5 h	$\mathbb{M}_E : 4.44 \times 10^{-2}$
		($L_f \leq 0.001$)			$\mathbb{E}_M : 6.63 \times 10^{-3}$
		1024	60	6.0 h	$\mathbb{M}_E : 4.22 \times 10^{-2}$
		($L_f \leq 0.001$)			$\mathbb{E}_M : 1.00 \times 10^{-2}$
		1024	120	11.4 h	$\mathbb{M}_E : 5.05 \times 10^{-2}$
		($L_f \leq 0.001$)			$\mathbb{E}_M : 3.79 \times 10^{-3}$
#3.1.2	A3	4096	30	2.4 h	$\mathbb{M}_E : 6.87 \times 10^{-2}$
		($L_f \leq 0.5$)			$\mathbb{E}_M : 1.20 \times 10^{-2}$
		4096	30	8.3 h	$\mathbb{M}_E : 2.82 \times 10^{-2}$
		($L_f \leq 0.001$)			$\mathbb{E}_M : 5.73 \times 10^{-3}$
		4096	60	12.7 h	$\mathbb{M}_E : 3.02 \times 10^{-2}$
		($L_f \leq 0.001$)			$\mathbb{E}_M : 4.63 \times 10^{-3}$

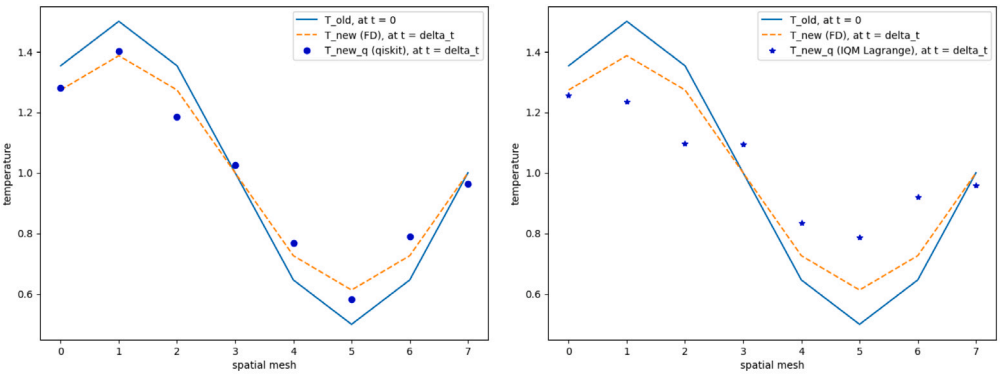
still a significant number for most existing quantum computers. Similarly, in Fig. 6(b), the results for the same problem in case of $n = 4$ qubits ($N = 16$) are reported. It is important to note that, in order to ensure a simple visual comparison between the two subplots in Fig. 6, the Fourier number is adjusted to maintain the same physical time step, regardless of the mesh resolution. Since $r = D\,\Delta t/\Delta z^2 \propto 1/\Delta z^2$, it follows that $r_{4q}/r_{3q} = (N_{4q}/N_{3q})^2 = (2^4/2^3)^2 = 4$. In this second case, even though the results are acceptable because $\mathbb{M}_e = 7.68 \times 10^{-3}$, we performed 10^6 iterations of the loss function, hitting the upper maximum limit, which we set in advance. It is clear that an increased number of tunable parameters, passing from $N_\theta = 24$ to 32, makes more complex the search for the minimum of the Hamiltonian.

Talking about deterministic (hence classical) simulations of quantum circuits, it is important to highlight that the adopted minimizer matters. For example, in the case study #2 reported in Fig. 7 (see Table 1 for details), the Quantum Matcha TEA package (see Section 3.6.4 for details) uses the gradient-based *BFGS* minimizer (see Section 3.6.5). In Figs. 7(a) and 7(b), the results for one time-step update of the temperature profile according to the heat conduction equation are reported for the three- and four-qubit cases, respectively. The relative errors $\mathbb{M}_e$ obtained by *BFGS* (case #2) are noticeably smaller than those obtained by *COBYLA* (case #1), despite a significant reduction in computational time (up to one order of magnitude), at least on the hardware considered.



(a) Precision: 10^{-3} (b) Precision: 5×10^{-4}

Fig. 9. Case study #3.2 (see Table 1 for details): One time-step update of the temperature profile according to heat conduction equation. Results with precision equal to 10^{-3} in panel (a) and results with precision equal to 5×10^{-4} in panel (b). The markers are the mesh temperatures computed by the shot-based quantum simulator of Qrisp. The error bars corresponding to standard deviations are also reported.



(a) Ideal simulation (b) Real quantum hardware run

Fig. 10. Case study #4 on IQM Lagrange at Torino by Qiskit implementation (see Table 1 for details): One time-step update of the temperature profile according to heat conduction equation. Ideal simulation in panel (a) and (noisy) real quantum hardware run in panel (b). Only 1024 shots per measurement are used.

However, several significant challenges remain. Table 2 highlights the non-trivial interplay between the choice of ansatz and the adopted minimizer, and its impact on the numerical results. For the problem considered here, simplified ansatz circuits—namely A2, A3, and A3p—consistently yield better performance. In addition, the *BFGS* minimizer generally outperforms *COBYLA* in terms of both computational efficiency and accuracy, at least in ideal simulation settings where gradient-based optimization methods can be expected to provide an advantage. A further challenge becomes apparent when examining the numerical results for case #2 obtained with the *BFGS* minimizer in Table 1. As the system size increases from three to five qubits, the relative error $\mathbb{M}_e$ grows by roughly one order of magnitude per additional qubit. This trend suggests that the underlying optimization landscape becomes increasingly difficult to navigate as the number of qubits increases, underscoring the scalability limitations of the VQE approach.

4.2.2. Ideal shot-based simulations (with statistical sampling)

Ideal shot-based simulation introduces statistical sampling into the previous otherwise ideal setting. Here, the simulator draws a finite number N_S of measurement outcomes ("shots") from the theoretical probability distribution, mimicking the discrete, probabilistic nature of real quantum measurements while still neglecting any physical noise due to decoherence and environmental interactions (see Section 3.6.3). As already discussed in Section 3.6.1, quantum processors can directly measure only simple observables, typically local projective measurements in a fixed basis. For this reason, any physical Hamiltonian acting on qubits is decomposed into a linear combination of N_P experimentally measurable observables (see Appendix C). For example, for $n = 3$ the Pauli decomposition requires $N_P = 34$. This implies that, in principle, estimating the target observable requires a total number of shots equal to $N_S \times N_P$.

Before proceeding with the numerical results of this Section, it is important to realize an intrinsic feature of VQE which may be exacerbated by the statistical sampling. Even in an ideal, noise-free, shot-based simulation, a VQE algorithm may converge to

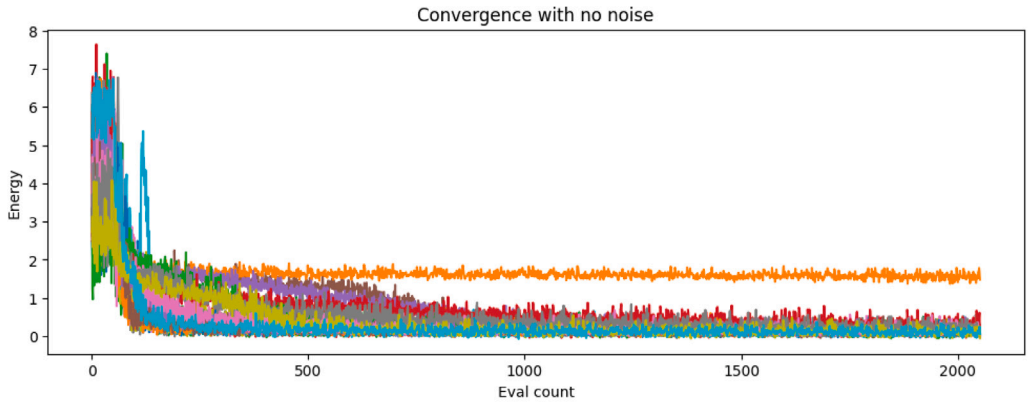


Fig. 11. Shot-based VQE simulation with 30 repetitions. Among the 30 repetitions of the minimization procedure, one shows a bad convergence because the final loss function L_f is significantly larger than zero, which is the target value by design. The number of evaluations exceeds twice the number of iterations ($N_{IT} = 1000$) because *SPSA* estimates the gradient using two energy evaluations, with occasional additional evaluations required.

a local rather than a global minimum. This problem has been already discussed in Section 3.6.5, where the concept of enhanced minimizer is introduced. In the enhanced minimizer, when the final loss function L_f remains larger than a given threshold after the selected number of iterations N_{IT} is reached, the minimization procedure is simply randomly re-initialized and repeated. In order to clarify this concept, let us consider an example: The convergence history of 30 repetitions of a shot-based VQE simulation reported in Fig. 11. In this example, after 1000 iterations, among the 30 repetitions of the minimization procedure, one shows a bad convergence because the final loss function L_f is significantly larger than zero, which is the target value by design. In this case, adopting an enhanced minimizer with a quality check, which ensures that $L_f \leq 0.5$, is enough to be effective, as proved by results reported in Fig. 12. Finally, shot-based simulations require N_R repetitions to obtain statistically reliable results, as described in Section 3.6.6. Summing up, shot-based simulations by an enhanced minimizer require a large number of measurements given by $N_S \times N_P \times N_{IT} \times N_R$. For example, the results reported in Table 1 and Fig. 8, which show good statistical behavior, require theoretically $1024 \times 34 \times 1000 \times 30 \sim 1 \times 10^9$ measurements.

We investigated in Table 3 the non-trivial interplay among the number of shots, the number of repetitions and the acceptance threshold of the enhanced minimizer, and their impact on the numerical results. In particular, lowering the acceptance threshold of the enhanced minimizer (from 0.5 to 0.001, as shown in Table 3) requires repeating a significant fraction of the minimization procedures, but it leads to an improvement in the statistical quality of the results. The latter phenomenon seems more pronounced for larger numbers of shots. On the other hand, increasing the number of shots broadens the statistical distribution of individual outcomes—and consequently their associated errors—while improving the convergence of the mean value toward the ideal result.

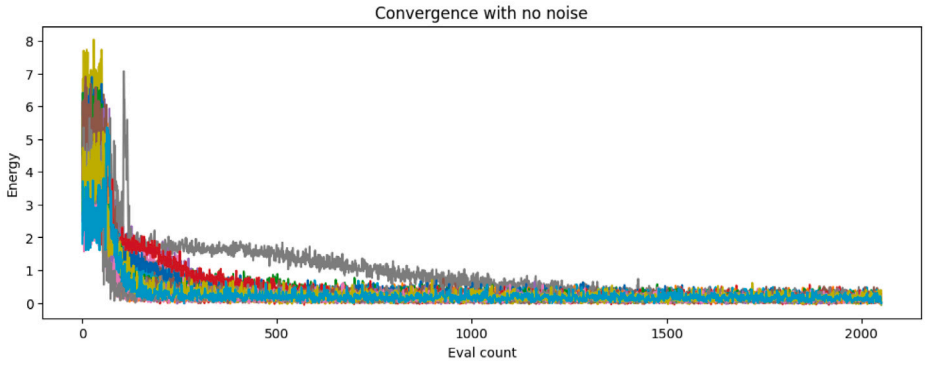
In ideal shot-based simulations, determining the optimal number of shots required to achieve the desired accuracy in the results can sometimes be challenging. Different software packages adopt different strategies to address this issue (see Section 3.6.4 for details). In particular, the Qrisp package allows the user to specify the desired precision directly in the VQE procedure. By default, the target precision is set to 0.01, which refers to the accuracy with which the Hamiltonian is evaluated [25]. The number of shots performed by the simulator per iteration scales quadratically with the inverse of the precision. The results of the case studies executed using Qrisp are reported in Table 1 and Fig. 9. It is interesting to note that case study #3.2, with a precision of 10^{-3}, produces an error of the mean nodal values $\mathbb{E}_M = 1.04 \times 10^{-2}$, which is similar to what is achieved by Qiskit Aer using 4096 shots in case study #3.1. This suggests that, at least for the present case study, 4096 shots with Qiskit Aer correspond approximately to a precision of 10^{-3} in Qrisp. On the classical hardware considered, the execution times are also comparable (~ 2.4 hours, see Table 3), further supporting this equivalence. Moreover, in case study #3.2, the precision equal to 5×10^{-4} improves the accuracy proportionally, as the error of the mean nodal values decreases to $\mathbb{E}_M = 4.58 \times 10^{-3}$.

4.2.3. Results on real quantum hardware

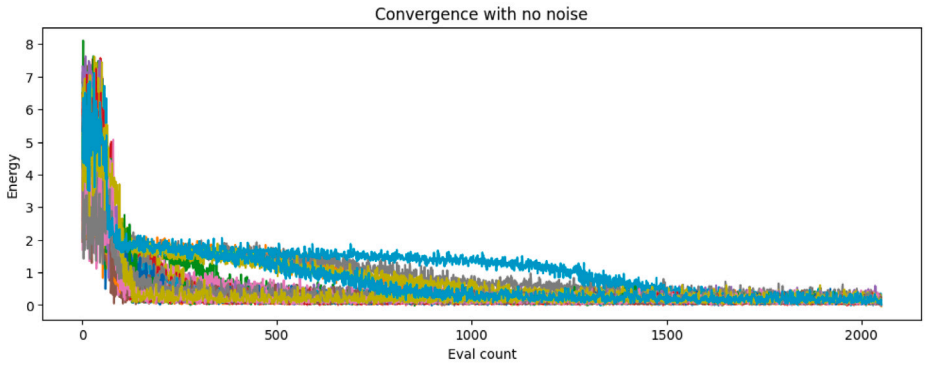
Finally we discuss the practical issues in the current stage of development of real quantum computers, which are affected by noise and imperfections. In this case, results reflect both statistical fluctuations from finite sampling and physical noise from gate errors, decoherence, and measurement inaccuracies.

In particular, we focus here on the IQM Spark system named “Lagrange”, which is a 5-qubit superconducting quantum processor developed by IQM Quantum Computers and recently installed at the Politecnico di Torino data center.[3] This installation represents the first operational on-premises quantum computer of its kind in Italy. Lagrange integrates a full-stack architecture encompassing

[3] The first IQM quantum computer in Italy was turned on in Turin (22/05/2025).



(a) $N_S = 1024$ shots, $N_{IT} = 1000$, $N_R = 30$.



(b) $N_S = 4096$ shots, $N_{IT} = 1000$, $N_R = 30$.

Fig. 12. Convergence of case study #3.1 (see Table 1 for details). Both simulations for (a) 1024 shots and (b) 4096 shots show good convergence because, in case of local-minimum trapping, the (enhanced) minimization procedure is randomly re-initialized and repeated, in order to ensure $L_f \leq 0.5$. The number of evaluations exceeds twice the number of iterations because *SPSA* estimates the gradient using two energy evaluations, with occasional additional evaluations required.

cryogenic control, microwave electronics, and quantum programming interfaces within a single modular platform. The device operates on superconducting transmon qubits, which are coupled and controlled via tunable microwave pulses enabling high-fidelity single- and two-qubit gate operations [28]. The quantum processor is maintained at a base temperature of approximately 20 mK using a dilution refrigerator, effectively suppressing thermal excitations and decoherence effects. To ensure optimal signal integrity and qubit coherence, the entire setup is housed within a vibration-isolated and electromagnetically shielded environment occupying roughly $4\,\mathrm{m}^2$ of laboratory space.

In order to use IQM Lagrange in Torino, it is important to understand the peculiarities of the IQM Spark hardware [28]. In general, the circuit depth—that is, the number of layers required to complete a quantum computation—plays a crucial role and should be minimized. However, one of the key aspects of quantum computing is entanglement, which—ideally—enables an exponential growth in the number of computational states, but is also very challenging to implement correctly on actual quantum hardware. In this regard, the following recommendations apply.

- First, it is generally advantageous to use the native CZ gates instead of CX gates (although the performance difference is not dramatic). In physical hardware, the native entangling interaction between qubits determines which of these two gates is more directly implementable. On most superconducting systems (e.g., IQM, IBM, Rigetti, etc.), the hardware implements CZ natively, whereas CX is a derived and composite operation.
- Second, it is important to limit the number of entangling blocks to one or two, even though this may pose challenges in accurately describing the ground state for the present application.
- Third, one must take into account the qubit topology—specifically, which qubits are connected in the star-like arrangement of IQM Lagrange [28]. Typically, “fully” entangling blocks would require many operations after transpilation, so a “linear” block is a more efficient choice.

Taking these considerations into account, it becomes clear why case study #4 in Table 1 employs ansatz A4, shown in Fig. 5: it uses only CZ gates in a linear arrangement, with just two entangling blocks and two layers of gates (i.e., 6 tunable parameters for 3 qubits). For running this case study on real quantum hardware, we adopted Qiskit as the software package. We set the number of

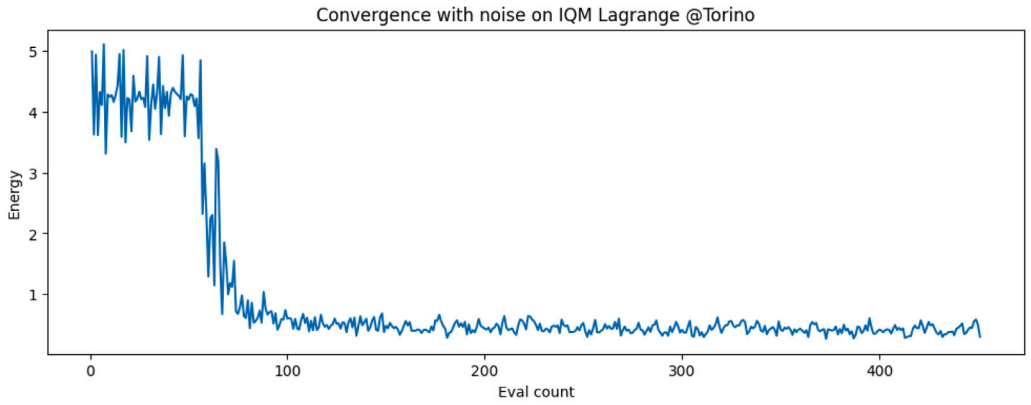


Fig. 13. Convergence of case study #4 on IQM Lagrange at Torino by Qiskit implementation (see Table 1 for details, *SPSA* minimizer, $N_S = 1024$, $N'_P = 28 < 34$, $N_{IT} = 200$, $N_R = 1$). The execution required 203 jobs and 45.3 min total QPU time: 5 min of QPU time for calibration and 12 s of QPU time per iteration. The number of evaluations exceeds twice the number of iterations because *SPSA* estimates the gradient using two energy evaluations, with occasional additional evaluations required.

measurement shots $N_S = 1024$ for reducing the computational demand. For $n = 3$ the Pauli decomposition requires $N_P = 34$ terms to measure the observable (see Appendix C). In practice, quantum hardware employs Pauli-term grouping, whereby multiple Pauli operators sharing the same measurement basis are measured using a single circuit. As a result, the effective number of measurement settings satisfies $N'_P \leq N_P$. In this case study, we obtained $N'_P = 28 < 34$. Moreover, we selected *SPSA* as minimizer with $N_{IT} = 200$ iterations. Finally we run only one calculation ($N_R = 1$), again for minimizing the QPU time. Summing up, the results reported in Table 1 for case study #4 required $1024 \times 28 \times 200 \times 1 \sim 6 \times 10^6$ measurements (excluding preliminary operations for calibration). After transpilation, the VQE minimization required the execution of $N_J = 203$ jobs and a total of 45.3 min of QPU time. The number of jobs N_J is slightly larger than the number of iterations N_{IT} because additional preliminary jobs are required to calibrate the learning rate and perturbation parameters in *SPSA*. These preliminary jobs took 5 min of QPU time, while the iterations were executed (on average) at 12 s per iteration. This case study could have been also executed on existing cloud-based quantum platforms, although the associated costs depend on the selected provider and they may be currently significant. Using these operational parameters, case study #4, executed on IQM Lagrange in Torino, demonstrated the convergence shown in Fig. 13 and the results reported in Fig. 10.

Looking at the actual results reported in Fig. 10, it is clear that there remains substantial room for improvement both (i) on the hardware side, by relaxing the physical constraints on circuit implementation, and (ii) on the error-mitigation side, by reducing gate errors, decoherence, and measurement inaccuracies. In fact, the hardware constraints imposed during circuit simplification already limit the ability of the ansatz circuit to approximate the ground state of the present problem, as evident from the results of the ideal simulator in Fig. 10(a). Moreover, the noise inherent to real quantum hardware further degrades the accuracy of the numerical solution which is reported in Fig. 10(b), leading to a mean nodal error of $\mathbb{M}_e = 1.38 \times 10^{-1}$, corresponding to roughly 14%. Clearly, these results obtained on actual quantum hardware must be regarded as preliminary, since only a few months have passed since its installation and many tunable parameters remain available to improve accuracy. This will be the focus of future extensive experimentation, but it already represents a meaningful step in defining a roadmap toward the broader application of quantum computing in engineering.

5. Conclusions

In this paper, we propose a case study to investigate the potential of quantum computing for simulating heat transfer phenomena relevant to engineering applications. We use the solution of the heat conduction equation as a paradigmatic case study to explore the adoption of quantum computing in thermal science. Despite its simplicity—which may even appear trivial to the thermal science community—this case study remains surprisingly challenging for the current state of real quantum computers. We believe that its apparent simplicity is, in fact, an advantage, as it allows sufficient flexibility for implementation across different hardware and software packages. In particular, we investigated various quantum ansatz circuits (5), software packages (Qiskit, Quantum Matcha TEA, Qiskit Aer, Qrisp), and optimization algorithms (*COBYLA*, *COBYQA*, *BFGS*, *SPSA*). We compared results from ideal simulations, shot-based simulations, and noisy executions on real quantum hardware (the IQM Lagrange quantum computer recently installed in Torino). This comprehensive effort enabled us to evaluate the performance of state-of-the-art quantum hardware against classical computational methods.

The main conclusions are as follows.

- Among the investigated software packages for ideal simulations, libraries based on tensor-network methods (e.g., Quantum Matcha TEA) offer a tangible advantage for efficient implementation on classical high-performance computing (HPC) systems.

This advantage stems primarily from their ability to efficiently compress entanglement in classical representation, exploiting the underlying structure of quantum correlations to significantly reduce the computational cost of the simulation, at least in variational problems.

- Regarding ideal shot-based simulations with statistical sampling, we examined two families of packages: those based on the explicit specification of the number of shots and those based on selecting the desired precision (i.e., the associated uncertainty). It is possible to heuristically establish an equivalence between these two approaches; therefore, the preferred choice depends on the type of investigation. In any case, it is crucial to implement methodologies to prevent local-minimum trapping, which is an intrinsic risk arising from the non-convexity of the variational loss function in quantum problems.
- Finally, for noisy executions on real quantum hardware, we reported preliminary results on the recently installed IQM Lagrange quantum computer in Torino, with a mean nodal outcome error of up to 14%. Although this performance is currently unsatisfactory from an application perspective, it should be regarded as a first step along a longer path toward a deeper understanding and eventual improvement of this emerging computational technology.

It is worth concluding with some reflections on the challenges and future perspectives. Identifying the most critical challenges for such a rapidly evolving technology is extremely difficult. Ideal quantum simulations performed on classical high-performance computing (HPC) systems provide a valuable testing ground, but they may lead the community to underestimate the inherent difficulties associated with statistical sampling on real quantum computers—even on future devices with significantly reduced noise levels.

At present, physical noise arising from gate errors, decoherence, and measurement inaccuracies prevents most genuinely engineering-oriented applications. Of course, some applications are less sensitive to these issues. Nevertheless, eliminating the fundamental sources of noise—particularly the unwanted coupling with the surrounding environment—appears extremely challenging. The situation may eventually be mitigated through advanced error-mitigation strategies that go beyond the simple averaging of physical qubits to emulate a logical qubit.

Finally, a word of realism. Porting existing engineering software to the quantum computing paradigm is an enormous undertaking, which can be considered realistic only if convincing evidence of a genuine quantum advantage over classical systems can be unambiguously demonstrated. In the meantime, while exploring the potential of quantum computing may be intellectually stimulating and scientifically worthwhile, it remains far too early to claim that such approaches will yield practical or useful results, whether in specific applications or in engineering as a whole.

CRediT authorship contribution statement

Pietro Asinari: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Matteo Maria Piredda:** Writing – review & editing, Writing – original draft, Resources, Data curation. **Giulio Barletta:** Writing – review & editing, Writing – original draft, Resources, Data curation. **Paolo De Angelis:** Writing – review & editing, Writing – original draft, Resources, Data curation. **Nada Alghamdi:** Writing – review & editing, Writing – original draft. **Giovanni Trezza:** Writing – review & editing, Writing – original draft. **Marina Provenzano:** Writing – review & editing, Writing – original draft. **Matteo Fasano:** Writing – review & editing, Writing – original draft, Supervision. **Eliodoro Chiavazzo:** Writing – review & editing, Writing – original draft, Supervision.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work the authors used ChatGPT, an AI language model developed by OpenAI under the CC BY 4.0 license, in order to linguistically refine portions of the text. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Declaration of competing interest

The authors declared that they have no conflict of interest and the paper presents their own work which does not infringe any third-party rights, especially authorship of any part of the article is an original contribution, not published before and not being under consideration for publication elsewhere.

Acknowledgment

The authors gratefully acknowledge: Paolo Viviani, Emanuele Dri and Olivier Terzo of *Fondazione LINKS*; Francesco Benfenati and Cosma Belli of IQM Quantum Computers; Davide Calonico of *Istituto Nazionale di Ricerca Metrologica (INRiM)*; Simone Montangero of University of Padua.

The authors acknowledge access to the IQM quantum computer Lagrange, hosted at Politecnico di Torino (Turin, Italy), jointly acquired by *Politecnico di Torino*, the *Fondazione LINKS*, and *Istituto Nazionale di Ricerca Metrologica (INRiM)*, which was used to perform the quantum experiments reported in this work.

Appendix A. Fundamental quantum computing concepts

In this Appendix, we aim to introduce the fundamental concepts of quantum computing.

A.1. Basics of binary coding

Before familiarizing with the data structure of a quantum computer, let us recall some basics of binary coding, which is useful for both classical and quantum computing systems. To represent a number in binary, every digit has to be either 0 or 1 (as opposed to decimal numbers where a digit can take any value from 0 to 9). In both cases, the mathematical notation used to write binary numbers or quantum states usually implies that the least significant bit (LSB) is written at the rightmost position. In other words, the rightmost bit typically represents the LSB. For example, in a 4-bit system, 0001 in binary corresponds to the decimal value 1, because the "1" is the LSB, representing 2^0 instead of 2^3 (if it were the most significant bit, MSB). A decimal integer, for instance the previously mentioned index l with $0 \leq l \leq N-1$, which identifies the mesh node, can be equivalently expressed by the corresponding binary number j, namely $j_{(2)} = l_{(10)}$, where the binary number can be expressed by an equivalent bit string $j_{(2)} \equiv \beta_{n-1}\beta_{n-2} \dots \beta_1\beta_0$. The latter equivalence is expressed by the following formula:

$$l \equiv l_{(10)} = \sum_{b=(n-1)}^{0} \beta_b \, 2^b = \sum_{b=0}^{n-1} \beta_b \, 2^b, \tag{A.1}$$

where β_0 is the LSB. In the last expression, $\beta_q \in \{0, 1\}$ is the generic digit (bit) of the string. In the following, we will use equivalently the decimal integer l or the corresponding binary string $j \equiv j_{(2)} = \beta_{n-1}\beta_{n-2} \dots \beta_1\beta_0$.

A.2. Qubit and superposition

The building block of a quantum computer is a qubit. A qubit is a two-level quantum-mechanical system, with many states which are linear combinations (often called superpositions) of the fundamental basis states, corresponding to the states 0 and 1 for a classical bit. Let us indicate the quantum states by the Dirac notation $|\cdot\rangle$, which stands for the standard notation for normalized vectors in quantum mechanics. Consequently, the computational basis states of each qubit, or simply the computational basis, are $|0\rangle$ and $|1\rangle$. The state vector for the generic qth qubit in a system can be expressed as

$$|\psi_q\rangle = \delta_q^{|0\rangle}|0\rangle + \delta_q^{|1\rangle}|1\rangle, \tag{A.2}$$

where $\delta_q^{|0\rangle} \in \mathbb{C}$ and $\delta_q^{|1\rangle} \in \mathbb{C}$ are complex numbers that represent the weight of $|0\rangle$ and $|1\rangle$ states of the superposition, and are called complex probability amplitudes [21]. In principle, these two complex numbers may suggest that there are four degrees of freedom in each qubit, but there are also two physical constraints to be considered. First of all, if these complex numbers are presented in polar form, it is possible to realize that their global phases can be disregarded, because only the relative phase matters with regard to the expectation value of any observable [21]. Secondly, the corresponding probabilities are normalized such that $|\delta_q^{|0\rangle}|^2 + |\delta_q^{|1\rangle}|^2 = 1$ [21]. Taking into account these two constraints, the state of each qubit can be described by two angles and the state vector can be visualized by means of the so-called Bloch sphere (more details in Appendix D) [21].

A.3. Multi-qubit systems and entanglement

A quantum computer involves more than one qubit and therefore we need to familiarize also with the multi-qubit systems. The computational basis states of a system with n qubits are all $N = 2^n$ possible tensor products[4] of single-qubit basis states. The generic computational basis state is indicated by $|j\rangle$ and it can be $|00 \dots 0\rangle$, $|00 \dots 1\rangle$, …, $|11 \dots 1\rangle$. Formally $|j\rangle$ is defined as

$$|j\rangle = |\beta_{n-1}\beta_{n-2} \dots \beta_0\rangle = |\beta_{n-1}\rangle \otimes |\beta_{n-2}\rangle \otimes \dots |\beta_0\rangle \in \{|0\rangle, |1\rangle\}^{\otimes n}, \tag{A.3}$$

where $\otimes$ denotes the tensor product and the last expression is a compact notation for indicating the set of all computational basis states, which has $N = 2^n$ elements. Clearly, a computational basis state is always separable by definition [21], meaning that it can be written as a tensor product of individual single-qubit states. An n qubit system has $N = 2^n$ computational basis states, analogously to a classical system, but, unlike classical bits that exist in one state at a time, qubits can also exist in superpositions of all these states, according to some probabilities, which can also be correlated. This means that, in general, superposition in multi-qubit systems can give rise to correlations between qubits, which become genuinely quantum when the resulting state is not separable. When such correlations cannot be expressed as products of independent single-qubit probability amplitudes, the quantum state is said to be entangled. Entanglement is therefore a uniquely quantum phenomenon in which the joint state of the system cannot be factorized into a tensor product of individual qubit states, leading to correlations that have no classical counterpart.

Entanglement allows a quantum computer to represent and process states described by an exponentially large number of complex amplitudes associated with the 2^n computational basis states; however, these non-classical correlations are fragile and can be rapidly degraded by decoherence arising from interactions with the environment [21]. In order to exploit the full capability of a quantum computer with n qubits, let us consider a system quantum state $|\psi\rangle$ defined as

$$|\psi\rangle = |\psi_0 \, \psi_1 \, \dots \, \psi_{n-1}\rangle = \sum_{|j\rangle \in \{|0\rangle, |1\rangle\}^{\otimes n}} \alpha_j |j\rangle, \tag{A.4}$$

[4] The tensor product $\otimes$ of two vectors $|\psi_A\rangle$ and $|\psi_B\rangle$ creates a new vector $|\psi_A\rangle \otimes |\psi_B\rangle$ in a space with dimension $d_A \times d_B$, which is larger than the individual spaces having dimensions d_A and d_B, respectively. On the other hand, the dyadic product (sometimes indicated by the same symbol in fluid-dynamics), produces a matrix (operator), not a vector, and it is indicated here by $|\psi_A\rangle\langle\psi_B|$.

where $\alpha_j \in \mathbb{C}$ is a complex coefficient called amplitude. Please note that, when composing physical systems, the sequential labeling of their components (e.g., $|\psi_0\rangle, |\psi_1\rangle, \ldots, |\psi_{n-1}\rangle$) may differ from the mathematical notation used to represent the bit strings, i.e., $\beta_{n-1} \ldots \beta_1 \beta_0$. The discrete vector quantum state $|\psi\rangle$ is normalized, namely $\langle\psi|\psi\rangle = 1$ where $\langle\cdot|\cdot\rangle$ denotes the inner product. Because the computational basis is orthonormal, then $\alpha_j = \langle j|\psi\rangle$. The probability p_j of finding the system state in the computational basis state $|j\rangle$ is given by $p_j = |\alpha_j|^2 = \alpha_j^* \alpha_j$, where $\sum_j p_j = \sum_j |\alpha_j|^2 = 1$. The key point here is that, in general, $|\psi\rangle = |\psi_0\, \psi_1\, \ldots\, \psi_{n-1}\rangle \neq |\psi_0\rangle|\psi_1\rangle \ldots |\psi_{n-1}\rangle = |\psi_0\rangle \otimes |\psi_1\rangle \otimes \ldots |\psi_{n-1}\rangle$, namely the state vector of the composite system is usually not separable. When the system state $|\psi\rangle$ is not separable, one can say that the system is in a state correlated by entanglement. The number of entangled states grows exponentially as a function of the number of qubits.

The latter point is crucial in quantum computing. It is possible to understand this key feature by means of real data loading (or better encoding) in quantum computers. Some algorithms have been proposed in literature to efficiently load real-world data into a quantum system, e.g. a divide-and-conquer algorithm for quantum state preparation [24]. Data loading (or encoding) is a good way to understand how a quantum computer works and hence it will be discussed briefly here. The main idea of the divide-and-conquer paradigm is to decompose a problem into subproblems of the same type and then recursively combine their solutions to obtain the solution of the original problem [24]. Let us analyze a generic step of this recursive procedure. Suppose a quantum system consists of $n-1$ qubits, realizing N_{n-1} quantum states. Now, let us add one more qubit, designed to provide specified probabilities of being in the state $|1\rangle$ for each of the N_{n-1} quantum states. This design introduces an additional N_{n-1} degrees of freedom, corresponding to the N_{n-1} probabilities. Consequently, the new quantum system with n qubits yields $N_n = 2N_{n-1}$ quantum states. Starting with a single qubit and applying this relation recursively, it is straightforward to show that $N \equiv N_n = 2^n$. This expression is, however, simplified because (i) it assumes real amplitudes and (ii) it neglects the loss of one degree of freedom due to the normalization condition $\langle\psi|\psi\rangle = 1$. Nevertheless, the simplified formula $N = 2^n$ is useful for capturing the essence.

A.4. Gate-model quantum computing

Gate-model quantum computing is the dominant paradigm for implementing general-purpose quantum algorithms. In this package, information is encoded in qubits, and computations are carried out through the application of a sequence of elementary quantum gates, analogous to logical gates in classical digital computing. These gates correspond to unitary operations acting on one or more qubits and are composed into quantum circuits that implement the desired algorithm. After the circuit execution, the quantum state is measured, yielding classical outcomes from which physical observables or solution properties are inferred. The gate-model approach is universal, meaning that any quantum algorithm can, in principle, be expressed as a quantum circuit built from a finite set of elementary gates. However, current gate-model quantum processors are affected by noise, limited coherence times, and finite gate fidelities, which strongly constrain circuit depth and motivate the use of hybrid and variational algorithms tailored to near-term hardware. Other quantum computing paradigms, such as quantum annealing and analog quantum computing, follow different computational principles and are therefore suited to distinct classes of problems.

Quantum algorithms in the gate-model package are commonly represented using quantum circuit diagrams. In these diagrams, each horizontal line corresponds to a qubit, which is initialized at the beginning of the computation and persists throughout the circuit. Quantum gates are depicted as symbols placed along these lines, indicating operations applied to individual qubits or jointly to multiple qubits. The circuit is read from left to right, reflecting the temporal order in which the operations are applied, with earlier gates acting first and later gates acting on the resulting quantum state. Measurement operations are typically shown at the end of the circuit, marking the transition from quantum information to classical outcomes.

In the main text, some gates are mentioned, which belong to two main categories. It is possible to define a quantum gate by a Pauli elementary matrix, e.g. X and Z, which are defined by Eqs. (30). These gates are elementary because they are not parameterized. On the other hand, it is also possible to define parameterized gates, e.g. R_Y and the R_Z, which are defined in terms of a rotation parameter by Eqs. (31) and (32), respectively. Finally, given the single-qubit Pauli gates X and Z, their controlled counterparts introduce conditional logic between two qubits. A controlled-X (CX) gate, also known as the $CNOT$ gate, acts on a pair of qubits: a control qubit and a target qubit. The X operation is applied to the target qubit if and only if the control qubit is in the state $|1\rangle$; otherwise, the target qubit is left unchanged. Similarly, a controlled-Z (CZ) gate applies a Z operation to the target qubit conditional on the control qubit being in the state $|1\rangle$. Both CX and CZ gates are two-qubit operations and play a fundamental role in generating entanglement and implementing conditional logic in quantum circuits. More details about quantum circuit diagrams and quantum gates can be found in Ref. [21].

A.5. Quantum interference

A defining feature of quantum computing is the role of interference arising from the complex-valued probability amplitudes associated with quantum states. Unlike classical or probabilistic computing, where probabilities are real and non-negative and therefore combine only additively, quantum amplitudes can be complex and may interfere either constructively or destructively. During a quantum computation, different computational paths contribute amplitudes that can reinforce one another (positive interference) or cancel out (negative interference), thereby amplifying desired outcomes while suppressing incorrect ones. This controlled use of interference underpins the potential advantage of many quantum algorithms and represents a fundamental departure from classical probabilistic approaches, in which only positive interference is possible and cancellation effects cannot occur.

A.6. Quantum Hamiltonian

In physics, a Hamiltonian is an operator that represents the total energy of a system and fully determines its dynamical and spectral properties. In quantum computing, the Hamiltonian plays a central role by encoding the problem of interest into an operator whose ground state or eigen-structure corresponds to the desired solution. In the context of the Variational Quantum Eigensolver (VQE), a problem-specific quantum Hamiltonian is constructed for a discrete system of qubits such that its lowest-energy state encodes the solution to a target problem, for example a linear system of equations. By designing the Hamiltonian so that its ground state minimizes an appropriate cost function, VQE enables the solution to be obtained through variational optimization without explicitly simulating the system dynamics, making this formulation particularly suitable for near-term quantum hardware.

Appendix B. VQE fundamental proof

In this Appendix, we want to prove that the quantum state $|x\rangle$, in the linear system given by Eq. (15), also minimizes the observable $\hat{H}$ given by Eq. (17). In other words, $|x\rangle$ is the ground state of the operator $\hat{H}$, as expected by design.

Let us decompose the operator as $\hat{H} = \hat{C}^T \hat{J} \hat{C}$, where $\hat{J} = I - |b\rangle\langle b|$ is a projector operator. It is possible to prove that $\hat{J}$ is a projector operator because $\hat{J}^2 = I - 2|b\rangle\langle b| + |b\rangle\langle b| = \hat{J}$. It projects onto the orthogonal complement of $|b\rangle$. That is, it removes the component of a vector along $|b\rangle$, leaving only the part orthogonal to $|b\rangle$. First of all, let us analyze the eigenvalues of $\hat{J}$, by calling λ_m the m-th eigenvalue and $|\phi_m\rangle$ the corresponding eigenstate, namely $\hat{J}|\phi_m\rangle = \lambda_m|\phi_m\rangle$. The m-th eigenvalue can be computed as

$$\lambda_m = \langle\phi_m|\hat{J}|\phi_m\rangle = 1 - \langle\phi_m|b\rangle^2. \tag{B.1}$$

Recalling the Cauchy–Schwarz inequality, namely

$$\langle\phi_m|b\rangle \leq \sqrt{\langle\phi_m|\phi_m\rangle}\sqrt{\langle b|b\rangle} = 1, \tag{B.2}$$

it is possible to find out that

$$\lambda_m = 1 - \langle\phi_m|b\rangle^2 \geq 0. \tag{B.3}$$

This means that all eigenvalues of $\hat{J}$ are larger than or equal to zero, i.e. the core operator $\hat{J}$ is positive semi-definite. Using the same orthonormal basis, it is possible to express $\hat{J}$ by spectral decomposition

$$\hat{J} = \sum_{m=0}^{N-1} \lambda_m |\phi_m\rangle\langle\phi_m|. \tag{B.4}$$

Now, coming back to the main observable $\hat{H}$, let us consider the expectation value of the observable $\hat{H}$ with regard to the generic state vector $|\psi\rangle$, namely

$$\langle\psi|\hat{H}|\psi\rangle = \langle\psi|\hat{C}^T \left(\sum_{m=0}^{N-1} \lambda_m |\phi_m\rangle\langle\phi_m| \right) \hat{C}|\psi\rangle = \sum_{m=0}^{N-1} \lambda_m \langle\phi_m|\hat{C}|\psi\rangle^2, \tag{B.5}$$

where we used the fact that $\hat{C}$ is real and $\hat{C}^T = \hat{C}$. Using the result given by Eq. (B.3) yields

$$\langle\psi|\hat{H}|\psi\rangle \geq 0, \tag{B.6}$$

which proves that the main observable $\hat{H}$ is also positive semi-definite, i.e. all eigenvalues of $\hat{H}$ are positive or equal to zero. Clearly $|x\rangle$ is the eigenvector of $\hat{H}$ corresponding to the zero eigenvalue, namely

$$\hat{H}|x\rangle = \hat{H}\,\hat{A}^{-1}|b\rangle = f^{-1}\,\hat{C}^T\,(I - |b\rangle\langle b|)\,|b\rangle = 0, \tag{B.7}$$

which proves that $|x\rangle$ is the ground state of the operator $\hat{H}$, as expected by design.

Appendix C. Pauli decomposition

In this Appendix, we want to understand better how Pauli decomposition works. First, we need to understand the tensor product between matrices in the present context. As an example, let us consider a composite system made of two qubits. In this case, the generic pth element of the decomposition looks like

$$\hat{P}_p = \sigma^{p0} \otimes \sigma^{p1} = \begin{bmatrix} \sigma^{p0}_{11}\,\sigma^{p1} & \sigma^{p0}_{12}\,\sigma^{p1} \\ \sigma^{p0}_{21}\,\sigma^{p1} & \sigma^{p0}_{22}\,\sigma^{p1} \end{bmatrix} =$$

$$= \begin{bmatrix} \sigma^{p0}_{11} \begin{pmatrix} \sigma^{p1}_{11} & \sigma^{p1}_{12} \\ \sigma^{p1}_{21} & \sigma^{p1}_{22} \end{pmatrix} & \sigma^{p0}_{12} \begin{pmatrix} \sigma^{p1}_{11} & \sigma^{p1}_{12} \\ \sigma^{p1}_{21} & \sigma^{p1}_{22} \end{pmatrix} \\ \sigma^{p0}_{21} \begin{pmatrix} \sigma^{p1}_{11} & \sigma^{p1}_{12} \\ \sigma^{p1}_{21} & \sigma^{p1}_{22} \end{pmatrix} & \sigma^{p0}_{22} \begin{pmatrix} \sigma^{p1}_{11} & \sigma^{p1}_{12} \\ \sigma^{p1}_{21} & \sigma^{p1}_{22} \end{pmatrix} \end{bmatrix} =$$

$$= \begin{bmatrix} \sigma_{11}^{p0}\sigma_{11}^{p1} & \sigma_{11}^{p0}\sigma_{12}^{p1} & \sigma_{12}^{p0}\sigma_{11}^{p1} & \sigma_{12}^{p0}\sigma_{12}^{p1} \\ \sigma_{11}^{p0}\sigma_{21}^{p1} & \sigma_{11}^{p0}\sigma_{22}^{p1} & \sigma_{12}^{p0}\sigma_{21}^{p1} & \sigma_{12}^{p0}\sigma_{22}^{p1} \\ \sigma_{21}^{p0}\sigma_{11}^{p1} & \sigma_{21}^{p0}\sigma_{12}^{p1} & \sigma_{22}^{p0}\sigma_{11}^{p1} & \sigma_{22}^{p0}\sigma_{12}^{p1} \\ \sigma_{21}^{p0}\sigma_{21}^{p1} & \sigma_{21}^{p0}\sigma_{22}^{p1} & \sigma_{22}^{p0}\sigma_{21}^{p1} & \sigma_{22}^{p0}\sigma_{22}^{p1} \end{bmatrix}, \tag{C.1}$$

where σ^{p0} and σ^{p1} are matrices which can be I, X, Y or Z and they refer to the first and the second qubit, respectively.

After becoming more familiar with this product, let us come back to the case with n qubits. In this case, each term of the summation in Eq. (29) looks like

$$\hat{P}_p = \sigma^{p0} \otimes \sigma^{p1} \otimes \ldots \sigma^{p(n-1)}, \tag{C.2}$$

where the detailed expressions are omitted for the sake of simplicity, but it is clear that each term involves a tensor product of n Pauli matrices. The difficulty is that the number of terms, denoted by N_P, typically grows very rapidly with the number of qubits n. To illustrate this point, we consider the following example. In case of a system with 3 qubits, the decomposition in Pauli matrices given by Eq. (29) is the following

$$\begin{aligned} \hat{H} \;=\; & \gamma_0 III + \gamma_1 IIX + \gamma_2 IXI + \gamma_3 IXX + \gamma_4 IXZ + \gamma_5 IYY + \\ & \gamma_6 IZI + \gamma_7 IZX + \gamma_8 IZZ + \gamma_9 XII + \gamma_{10} XIX + \gamma_{11} XXI + \\ & \gamma_{12} XXX + \gamma_{13} XXZ + \gamma_{14} XYY + \gamma_{15} XZI + \gamma_{16} XZX + \gamma_{17} XZZ + \\ & \gamma_{18} YIY + \gamma_{19} YXY + \gamma_{20} YYI + \gamma_{21} YYX + \gamma_{22} YYZ + \gamma_{23} YZY + \\ & \gamma_{24} ZII + \gamma_{25} ZIX + \gamma_{26} ZIZ + \gamma_{27} ZXI + \gamma_{28} ZXX + \gamma_{29} ZXZ + \\ & \gamma_{30} ZYY + \gamma_{31} ZZI + \gamma_{32} ZZX + \gamma_{33} ZZZ, \end{aligned} \tag{C.3}$$

where, for example, III means $I \otimes I \otimes I$ and similarly for all remaining terms. The coefficients γ_p are obtained via the Hilbert–Schmidt inner product [21]. The Hilbert–Schmidt inner product defines a natural inner product on the space of linear operators acting on a finite-dimensional Hilbert space and is given by the trace of the product of one operator with the adjoint of the other [21]. In our example, for $n = 3$ the Pauli decomposition requires $N_P = 34$. For the sake of comparison, for $n = 4$ the Pauli decomposition requires $N_P = 120 \sim \exp(k_p\, n)$, where k_p is a proper constant. Therefore, the present application appears to be a case where the number of Pauli products in the Hamiltonian decomposition grows exponentially with the number of qubits, as suggested in [16]. However, the reference does not explicitly analyze this scaling, and as noted, leveraging the distribution of Pauli string weights can potentially reduce the complexity. While a brute-force or naive approach would indeed be impractical, the literature suggests alternative methods—such as truncation, grouping of Pauli strings, and other techniques—that might be applicable in this context. More details about the Pauli decomposition can be found in Ref. [29].

Appendix D. Hilbert space versus Bloch sphere

In this Appendix, we aim to intuitively explain the construction of the Bloch sphere representation of a single qubit, and its relationship to the Hilbert space representation. As previously discussed, a single qubit state vector can be expressed using Eq. (A.2):

$$|\psi_q\rangle = \delta_q^{|0\rangle}|0\rangle + \delta_q^{|1\rangle}|1\rangle,$$

where $\delta_q^{|0\rangle}, \delta_q^{|1\rangle} \in \mathbb{C}$ are complex numbers. These coefficients define a state in a two-dimensional Hilbert space over the complex numbers, which can be thought of as having four real parameters; thus, even in this simple case, it is difficult to visualize the state. Fortunately, we can overcome this limitation by using the polar (Euler) representation of complex numbers and applying the normalization condition, $|\delta_q^{|0\rangle}|^2 + |\delta_q^{|1\rangle}|^2 = 1$. We first express the complex amplitudes as $\delta_q^{|0\rangle} = Ae^{i\alpha}$ and $\delta_q^{|1\rangle} = Be^{i\beta}$. Substituting into Eq. (A.2), we obtain:

$$|\psi_q\rangle = Ae^{i\alpha}|0\rangle + Be^{i\beta}|1\rangle. \tag{D.1}$$

The normalization condition constrains the amplitudes to lie on the unit circle in the real two-dimensional space: $A^2 + B^2 = 1$. Therefore, we can parameterize the amplitudes using a single *polar* angle θ:

$$|\psi_q\rangle = \cos\theta\, e^{i\alpha}|0\rangle + \sin\theta\, e^{i\beta}|1\rangle.$$

Since an overall (global) phase factor has no physical effect, we can factor out $e^{i\alpha}$ and ignore it, defining a relative phase $\zeta = \beta - \alpha$. We then obtain:

$$|\psi_q\rangle = e^{i\alpha}\left(\cos\theta|0\rangle + \sin\theta\, e^{i\zeta}|1\rangle\right),$$

where the term $e^{i\alpha}$ can be safely omitted, since it represents a phase shift, and does not affect measurement outcomes [21], yielding the simplified and physically equivalent expression:

$$|\psi_q\rangle = \cos\theta|0\rangle + \sin\theta\, e^{i\zeta}|1\rangle. \tag{D.2}$$

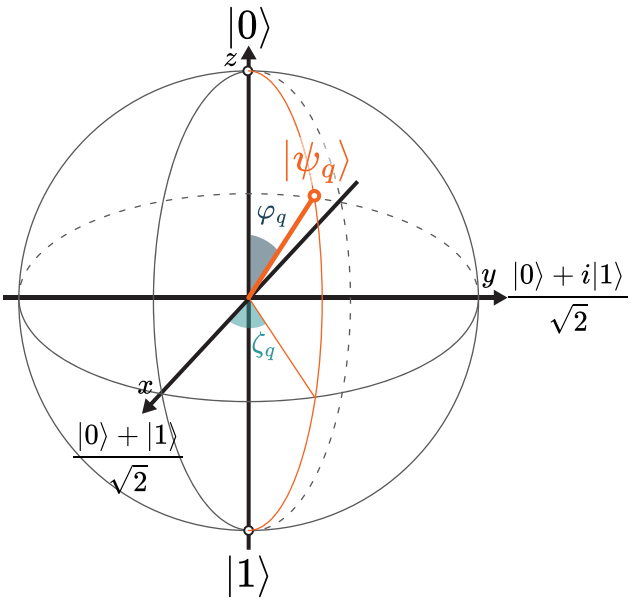


Fig. D.14. Bloch sphere representation of the qubit state $|\psi_q\rangle$.

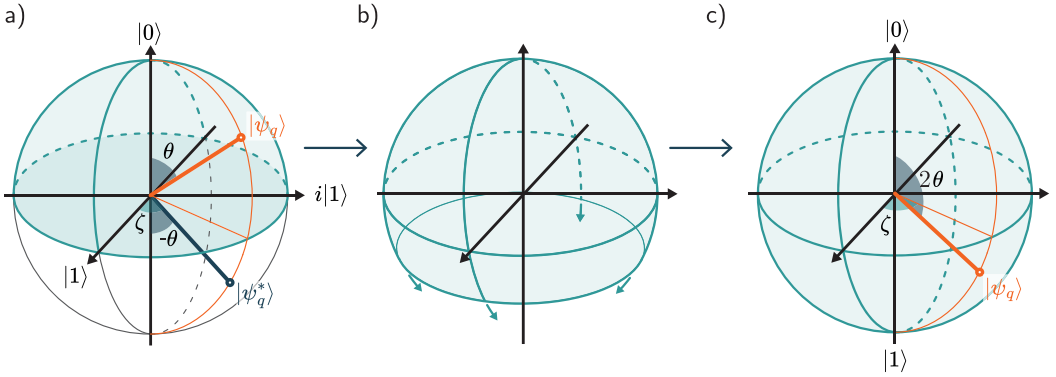


Fig. D.15. Intuitive construction of the Bloch sphere. (a) The qubit vector state $|\psi_q\rangle$ (and the reflected state $|\psi_q^*\rangle$) represented in Hilbert space. (b) The conceptual stretching of the upper hemisphere onto a sphere (and the equator of the upper hemisphere in a point). (c) The final representation of $|\psi_q\rangle$ as a vector state on the Bloch sphere.

On the other hand, in many textbooks, the state of each qubit is described conventionally by two angles φ_q and ζ_q, and the state vector can be expressed as:

$$|\psi_q\rangle = \cos\left(\varphi_q/2\right)|0\rangle + e^{i\,\zeta_q}\sin\left(\varphi_q/2\right)|1\rangle, \tag{D.3}$$

which can be visualized by means of the so-called Bloch sphere (see Fig. D.14) [21].

Comparing Eq. (D.2) with Eq. (D.3) yields $\theta = \varphi_q/2$ and $\zeta = \zeta_q$, which will be clearer at the end of this Appendix and is the main point of this derivation. Now, using Euler's formula $e^{i\zeta} = \cos\zeta + i\sin\zeta$, we can rewrite the qubit state:

$$|\psi_q\rangle = \cos\theta|0\rangle + \sin\theta\cos\zeta|1\rangle + i\sin\theta\sin\zeta|1\rangle.$$

Recalling the spherical coordinate vector $\vec{r} = (\sin\theta\cos\zeta, \sin\theta\sin\zeta, \cos\theta)^T$, we recognize that it is possible to visualize the state $|\psi_q\rangle$ in a tridimensional Hilbert space with the bases $|1\rangle = (1,0,0)^T$, $i|1\rangle = (0,i,0)^T$ (the generic axis of the Hilbert space may be complex, but the inner product must be still non-commutative[5]), $|0\rangle = (0,0,1)^T$, as shown in Fig. D.15(a). Using the full sphere in Hilbert

[5] In a two-dimensional complex space, the inner product is non-commutative because it is conjugate symmetric, meaning $\langle\psi|\phi\rangle = \langle\phi|\psi\rangle^*$ (complex conjugate), so swapping the vectors changes the result unless the inner product is real. For example, if $|a\rangle = |1\rangle$ and $|b\rangle = i|1\rangle$, then $\langle a|b\rangle = i \neq \langle b|a\rangle = -i$, therefore $\langle a|b\rangle = \langle b|a\rangle^*$.

space has the problem that multiple states (on the equator) can result in the same measurement outcomes. For example, consider the state $|\psi_q\rangle$ and the reflected state through the equatorial plane $|\psi_q^*\rangle = \cos(-\theta)|0\rangle + \sin(-\theta)\, e^{i\varphi}|1\rangle$, depicted in Fig. D.15(a), which yield the same measurement probabilities:

$$p_0 = \langle\psi_q|P_0|\psi_q\rangle = \langle\psi_q^*|P_0|\psi_q^*\rangle = \cos^2\theta,$$
$$p_1 = \langle\psi_q|P_1|\psi_q\rangle = \langle\psi_q^*|P_1|\psi_q^*\rangle = \sin^2\theta,$$

where the projectors operators of the measurement are defined as $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$. Thus, we see that $|\psi_q\rangle$ and $|\psi_q^*\rangle$ are physically indistinguishable by measurement. As a result, only the *upper hemisphere* of the sphere in the Hilbert space is needed to uniquely describe a qubit state, which corresponds to restricting the angle θ to the interval $\theta \in [0, \pi/2]$.

This means that all the states that lie along the equator ($\theta = \pi/2$ in the Hilbert space) represent the same measurement outcome (i.e., for $\theta = \pi/2$ we have $|\psi_q'\rangle = \cos\zeta|1\rangle + \sin\zeta\ i|1\rangle$, so the measurement probability is $p_1 = \langle\psi_q'|P_1|\psi_q'\rangle = \cos^2\zeta + \sin^2\zeta = 1$). Therefore, we can conceptually imagine "pulling" this equatorial circumference downward until it collapses into a single point. This transformation simplifies the visualization and results in the familiar Bloch sphere representation, as illustrated in Fig. D.15(b) and Fig. D.15(c). In this representation, the polar angle θ from the Hilbert space mapping is effectively *doubled* on the Bloch sphere. To maintain consistency between the two representations, we can introduce the Bloch polar angle $\varphi_q = 2\theta$, which is equivalent to $\theta = \varphi_q/2$ with $\varphi_q \in [0, \pi]$, as already previously discussed.

It is important to note that this is an illustrative non-rigorous explanation of the Bloch sphere construction. Mathematically, the Bloch sphere corresponds to the complex projective line of the two-dimensional Hilbert space, constructed using a stereographic projection of the qubit state onto a plane and topologically represented as the *Riemann sphere*.

Data availability

The datasets and the custom simulation code used in this study are publicly available at GitHub: https://github.com/SMaLL-PoliTo/QuantumThermal.

References

[1] J.-W. Pan, Quantum technologies need big investments to deliver on their big promises, Nature 638 (8052) (2025) http://dx.doi.org/10.1038/d41586-025-00564-8, 862–862, bandiera_abtest: a Cg_type: World View Nature Publishing Group Subject_term: Quantum physics, Technology, Policy, URL https://www.nature.com/articles/d41586-025-00564-8.

[2] M. Brooks, Quantum computers: what are they good for? Nature 617 (7962) (2023) S1–S3, http://dx.doi.org/10.1038/d41586-023-01692-9, bandiera_abtest: a Cg_type: Spotlight Nature Publishing Group Subject_term: Computer science, Applied physics, Quantum information, URL https://www.nature.com/articles/d41586-023-01692-9.

[3] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, P.J. Coles, Challenges and opportunities in quantum machine learning, Nat. Comput. Sci. 2 (9) (2022) 567–576, http://dx.doi.org/10.1038/s43588-022-00311-3, publisher: Nature Publishing Group, URL https://www.nature.com/articles/s43588-022-00311-3.

[4] X. Li, X. Yin, N. Wiebe, J. Chun, G.K. Schenter, M.S. Cheung, J. Mülmenstädt, Potential quantum advantage for simulation of fluid dynamics, Phys. Rev. Res. 7 (1) (2025) http://dx.doi.org/10.1103/PhysRevResearch.7.013036, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85214879505&doi=10.1103%2fPhysRevResearch.7.013036&partnerID=40&md5=2601c4625e6caba9b4bf973654db62d3.

[5] S.A. Razzak, O.H. Alkhalaf, S.M. Rahman, J. Zhu, Quantum machine learning – a novel approach for hydrodynamics analysis and modeling of liquid–solid circulating fluidized bed risers, Chem. Eng. Sci. 282 (2023) http://dx.doi.org/10.1016/j.ces.2023.119310, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85172185197&doi=10.1016%2fj.ces.2023.119310&partnerID=40&md5=432e31dc741ded2ea3910efcd681a9e6.

[6] Y. Yang, The role of quantum computing in advancing plasma physics simulations for fusion energy and high-energy, Front. Phys. 13 (2025) http://dx.doi.org/10.3389/fphy.2025.1551209, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-105000476582&doi=10.3389%2ffphy.2025.1551209&partnerID=40&md5=c1fcc1db1cdb53cbcf74b6a662a29f70.

[7] X. Rao, Performance study of variational quantum linear solver with an improved ansatz for reservoir flow equations, Phys. Fluids 36 (4) (2024) http://dx.doi.org/10.1063/5.0201739, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85189674654&doi=10.1063%2f5.0201739&partnerID=40&md5=acb2d1a791e51a73bda852dc7e7b5148.

[8] Y.-X. Wang, L.-L. Zhang, W. Liu, X.-H. Cheng, Y. Zhuang, A.T. Chronopoulos, Performance optimizations for scalable CFD applications on hybrid CPU+MIC heterogeneous computing system with millions of cores, Comput. & Fluids 173 (2018) 226–236, http://dx.doi.org/10.1016/j.compfluid.2018.03.005, URL https://www.sciencedirect.com/science/article/pii/S0045793018301038.

[9] J. Eisert, J. Preskill, Mind the gaps: The fraught road to quantum advantage, 2025, http://dx.doi.org/10.48550/arXiv.2510.19928, arXiv:2510.19928 [quant-ph], URL http://arxiv.org/abs/2510.19928.

[10] M. Cerezo, A. Arrasmith, R. Babbush, S.C. Benjamin, S. Endo, K. Fujii, J.R. McClean, K. Mitarai, X. Yuan, L. Cincio, P.J. Coles, Variational quantum algorithms, Nat. Rev. Phys. 3 (9) (2021) 625–644, http://dx.doi.org/10.1038/s42254-021-00348-9, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85112707065&doi=10.1038%2fs42254-021-00348-9&partnerID=40&md5=98f7383a69f2c4d9efcddac7430a28db.

[11] N. Sharma, V. Saxena, A systematic review of strategic approaches and applications in quantum computing, Opt. Quantum Electron. 57 (438) (2025) http://dx.doi.org/10.1007/s11082-025-08364-0, URL https://link.springer.com/article/10.1007/s11082-025-08364-0.

[12] A.W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. 103 (15) (2009) 150502, http://dx.doi.org/10.1103/PhysRevLett.103.150502, publisher: American Physical Society, URL https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[13] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subaşı, L. Cincio, P.J. Coles, Variational quantum linear solver, Quantum 7 (2023) http://dx.doi.org/10.22331/q-2023-11-22-1188, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85179605147&doi=10.22331%2fq-2023-11-22-1188&partnerID=40&md5=b745d06efc7674720c441ea4b64cb156.

[14] A. Peruzzo, P. Shadbolt J. McClean, M.-H. Yung, X.-Q. Zhou, P.J. Love, A. Aspuru-Guzik, J.L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. 5 (2014) http://dx.doi.org/10.1038/ncomms5213, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84904819361&doi=10.1038%2fncomms5213&partnerID=40&md5=946c62d9e79b20702cf35f5e0b89ea33.

[15] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G.H. Booth, J. Tennyson, The variational quantum eigensolver: A review of methods and best practices, Phys. Rep. 986 (2022) 1–128, http://dx.doi.org/10.1016/j.physrep.2022.08.003, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85138335876&doi=10.1016%2fj.physrep.2022.08.003&partnerID=40&md5=32b7124bd3ae8bcb62ce01f1fc2208d2.

[16] N.M. Guseynov, A.A. Zhukov, W.V. Pogosov, A.V. Lebedev, Depth analysis of variational quantum algorithms for the heat equation, Phys. Rev. A 107 (5) (2023) 052422, http://dx.doi.org/10.1103/PhysRevA.107.052422, publisher: American Physical Society, URL https://link.aps.org/doi/10.1103/PhysRevA.107.052422.

[17] J. Ingelmann, S.S. Bharadwaj, P. Pfeffer, K.R. Sreenivasan, J. Schumacher, Two quantum algorithms for solving the one-dimensional advection–diffusion equation, Comput. & Fluids 281 (2024) 106369, http://dx.doi.org/10.1016/j.compfluid.2024.106369, URL https://www.sciencedirect.com/science/article/pii/S0045793024002019.

[18] H.-Y. Huang, K. Bharti, P. Rebentrost, Near-term quantum algorithms for linear systems of equations with regression loss functions, New J. Phys. 23 (11) (2021) 113021, http://dx.doi.org/10.1088/1367-2630/ac325f, publisher: IOP Publishing.

[19] M. Larocca, S. Thanasilp, S. Wang, K. Sharma, J. Biamonte, P.J. Coles, L. Cincio, Z. Holmes J.R. McClean, M. Cerezo, Barren plateaus in variational quantum computing, Nat. Rev. Phys. 7 (4) (2025) 174–189, http://dx.doi.org/10.1038/s42254-025-00813-9, publisher: Nature Publishing Group, URL https://www.nature.com/articles/s42254-025-00813-9.

[20] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, P.J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits, Nat. Commun. 12 (1) (2021) http://dx.doi.org/10.1038/s41467-021-21728-w, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85102866117&doi=10.1038%2fs41467-021-21728-w&partnerID=40&md5=8476182c4c55af77fd8c4605005b7a60.

[21] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press, USA, 2010.

[22] X. Xu, J. Sun, S. Endo, Y. Li, S.C. Benjamin, X. Yuan, Variational algorithms for linear algebra, Sci. Bull. 66 (21) (2021) 2181–2188, http://dx.doi.org/10.1016/j.scib.2021.06.023, URL https://www.sciencedirect.com/science/article/pii/S2095927321004631.

[23] A. Javadi-Abhari, M. Treinish, K. Krsulich, C.J. Wood, J. Lishman, J. Gacon, S. Martiel, P.D. Nation, L.S. Bishop, A.W. Cross, B.R. Johnson, J.M. Gambetta, Quantum computing with Qiskit, 2024, http://dx.doi.org/10.48550/arXiv.2405.08810, arXiv:2405.08810 [quant-ph], URL http://arxiv.org/abs/2405.08810.

[24] I.F. Araujo, D.K. Park, F. Petruccione, A.J. da Silva, A divide-and-conquer algorithm for quantum state preparation, Sci. Rep. 11 (1) (2021) 6329, http://dx.doi.org/10.1038/s41598-021-85474-1, publisher: Nature Publishing Group, URL https://www.nature.com/articles/s41598-021-85474-1.

[25] R. Seidel, S. Bock, N.V. Tcholtchev, M. Hauswirth, Qrisp: a framework for compilable high-level programming of gate-based quantum computers, in: International Workshop on Programming Languages for Quantum Computing 2022, 2022, p. 7, http://dx.doi.org/10.24406/publica-1631, URL https://publica.fraunhofer.de/handle/publica/445649.

[26] M. Ballarin, D. Jaschke, S. Montangero, Quantum TEA : qmatchatea, language: eng, 2024, http://dx.doi.org/10.5281/zenodo.11619266, URL https://zenodo.org/records/11619266.

[27] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, I. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, Scipy 1.0: fundamental algorithms for scientific computing in Python, Nat. Methods 17 (3) (2020) 261–272, http://dx.doi.org/10.1038/s41592-019-0686-2, publisher: Nature Publishing Group, URL https://www.nature.com/articles/s41592-019-0686-2.

[28] J. Rönkkö, O. Ahonen, V. Bergholm, A. Calzona, A. Geresdi, H. Heimonen, J. Heinsoo, V. Milchakov, S. Pogorzalek, M. Sarsby, M. Savytskyi, S. Seegerer, F. Šimkovic, P.V. Sriluckshmy, P.T. Vesanen, M. Nakahara, On-premises superconducting quantum computer for education and research, EPJ Quantum Technol. 11 (1) (2024) 1–37, http://dx.doi.org/10.1140/epjqt/s40507-024-00243-z, publisher: SpringerOpen, URL https://epjquantumtechnology.springeropen.com/articles/10.1140/epjqt/s40507-024-00243-z.

[29] L. Hantzko, L. Binkowski, S. Gupta, Tensorized Pauli decomposition algorithm, Phys. Scr. 99 (8) (2024) 085128, http://dx.doi.org/10.1088/1402-4896/ad6499, publisher: IOP Publishing.