

Paving the Way to Hybrid Quantum-Classical Scientific Workflows

Sandeep Suresh Cranganore^{a,b}, Vincenzo De Maio^b, Ivona Brandic^b, Ewa Deelman^c

^aForschungszentrum Jülich GmbH, Peter Grünberg Institute, Quantum Control (PGI-8), Wilhelm Johnen Straße, Juelich, 52425, Germany

^bTU Wien, HPC group, Favoritenstrasse 9-11, Vienna, 1040, Austria

^cUniversity of Southern California, Information Sciences Institute, 4676 Admiralty Way Suite 1001, Marina del Rey, 90292, California, USA

Abstract

The increasing growth of data volume, and the consequent explosion in demand for computational power, are affecting scientific computing, as shown by the rise of extreme data scientific workflows. As the need for computing power increases, quantum computing has been proposed as a way to deliver it. It may provide significant theoretical speedups for many scientific applications (i.e., molecular dynamics, quantum chemistry, combinatorial optimization, and machine learning). Therefore, integrating quantum computers into the computing continuum constitutes a promising way to speed up scientific computation. However, the scientific computing community still lacks the necessary tools and expertise to fully harness the power of quantum computers in the execution of complex applications such as scientific workflows. In this work, we describe the main characteristics of quantum computing and its main benefits for scientific applications, then we formalize hybrid quantum-classic workflows, explore how to identify quantum components and map them onto resources. We demonstrate concepts on a real use case and define a software architecture for a hybrid workflow management system.

1. Introduction

Scientific computing is a branch of computer science spanning different disciplines (i.e., biology, chemistry, engineering), whose goal is the development of standardized and accurate simulations of different phenomena. Scientific computation is typically modeled by scientific workflows, whose execution is managed by Workflow Management Systems (WMSs) (e.g., Pegasus [1], ASKALON [2], Airflow [3]). The importance of scientific workflows has been proven by the Nobel-Prize-winning research on gravitational waves, which employed LIGO data analysis workflows managed by Pegasus WMS¹ [4], and their broad applicability to critical fields [5] such as drug design [6], material sciences [7], and simulations of the spread of Covid-19 [8]. The increasing complexity of scientific workflows calls for increasing computing power, which is provided by HPC clusters [9].

Current HPC systems are faced with the end of Moore's law [10]. This means that we cannot increase the computing power at the same rate as before. Consequently, HPC researchers are considering alternative forms of computing to satisfy the increasing demands of scientific applications, enabling the transition to Post-Moore scientific computing [11].

In the landscape of Post-Moore computing, quantum computing promises substantial performance improvement [12, 13]. Quantum computing can increase application performance, due to the proven theoretical speedup for different scientific problems [14, 15] and its native modeling of many scientific phenomena [16]. However, despite the theoretical speedup, the cur-

rent state-of-the-art hardware is bound by the following shortcomings (1) limited availability of hybrid resources, (2) susceptibility of qubits to noise and errors in the *Noisy Intermediate-Scale Quantum* (NISQ) devices [17, 18, 19], (3) limited technical capabilities and engineering shortcomings at the hardware level: several requirements, such as highly-controllable qubits (high-fidelity state preparation and qubit register initialization), large counts of quantum gates (for e.g. deeper quantum circuits have higher CNOT (controlled-NOT) counts, which contribute to larger error rates as compared to single-qubit gates) operating within the coherence limits of the qubits (for e.g. shorter gate times and efficiency), and considerably large circuit depths (starting from qubit initialization to the final measurement). In order to run quantum algorithms (cryptosystems-based algorithms, such as Shor's integer factorization algorithm for breaking the RSA-2048 scheme [20]) require large-scale logical qubit devices or alternatively physical qubits and quantum gates ranging between thousands to millions [21], and (4) challenges in suppressing errors, for e.g. protecting entanglement between logical qubits [22] (fault-tolerant quantum computation) arising due to the lack of fault-tolerant logical algorithms/quantum error correction (QEC) schemes at the experimental level.

A broad category of workflows tasks can be propelled by utilizing quantum processors. These range from (a) accelerators that are interoperable with classical architectures, resulting in hybrid quantum-classical systems [23] or neuromorphic architectures [24], (b) stochastic and probabilistic sampling methods such as Monte Carlo (MC) estimation, (c) Programmable array of qubits, for synthetic simulation of other quantum systems, also known as *quantum simulators*, e.g., quantum phases of matter and critical dynamics of many-body systems [25, 26].

¹Nobel Prize-winning discovery on gravitational waves came about with contributions from USC scientists

In the field of gate-based NISQ, the most influential paradigm subclass of hybrid models are the *Variational Quantum Algorithms* (VQAs) [27], where classic and quantum hardware are tightly coupled and cooperate in the achievement of a specific task. VQAs are turning out to be one of the much anticipated workhorses in the hybrid computation arena. These include fluid dynamics, quantum chemistry simulations, for e.g., accurate calculations of electronic structure using Hartree-Fock methods [28]. Molecular dynamics (MD) is another highly suitable usecase, for e.g., simulating weakly bound, coarse-grained intermolecular interactions and groundstate determination [29].

In this work, we investigate the problem of executing scientific workflows on hybrid quantum-classical ecosystems. First, we identify and formalize the main actors involved in the process. Based on our model, we design a hybrid quantum-classic workflow starting from a classic molecular dynamics workflow designed for Pegasus WMS. Then, we provide an idea of how to allow the execution of scientific workflows on hybrid quantum-classic systems and identify challenges and possible solutions. Finally, we provide an outlook on the field and identify possible trends for future research in the area.

We focus on scientific applications, which provide major opportunities for quantum modeling and quantum speedup [23]. Also, we consider Pegasus [1], a well-known WMS, as the reference architecture for WMSs.

The paper is organized as follows: first, we analyze related work in Section 2, then we provide the theoretical foundations of our work in Section 3 (Appendixes provide additional background in quantum computing). In Section 4, we provide the definition of hybrid quantum-classic workflows and how to transform a classic workflow into a hybrid quantum-classic workflows. In Section 5, we describe our molecular dynamics simulation use case as a running example of the transformation from classic to hybrid classic-quantum workflows. We then describe our vision of hybrid workflow execution in Section 6, while in Section 7 we identify the challenges that must be tackled to enable it. Finally, we conclude our paper in Section 8.

2. Related Work

Quantum computing has been first theorized by Feynman [30]. Advantages and ideas for quantum supremacy over superconducting qubits are described in [31], while [32] describes the circuit-based model of computation. Similarly, [33] focuses on ion-traps, while [34] describes D-Wave quantum annealers.

In [35], a first study on how to transform classic workflows into hybrid classic/quantum workflows is performed. In particular, it focuses on machine learning applications [36] and also describes methods for the identification of quantum candidates and splitting scientific applications between classic and quantum hardware. Applications of quantum computing to scientific applications can be found in many domains, ranging from drug design [37], molecular dynamics [38], financial modelling [39], manufacturing industry [40], linear optimization [41], and healthcare [42]. In the above examples, scientific workflows are not considered.

VQAs, one of the typical applications of hybrid classic-quantum systems, are described in [43]. In [44], different applications of VQAs are described. Still in the context of variational quantum algorithms, [45] focuses on photonic quantum platforms, while [46, 47] focus on the variational quantum eigensolver, that is a common task in scientific computations. Applications of VQAs can be found in different scientific applications, such as molecular dynamics [38], accelerating machine learning workloads [15] and combinatorial optimization [43]. However, few works address the integration of VQAs in scientific workflows.

Efforts in standardizing hybrid applications are performed also from a software engineering perspective. In [48], a survey about the state of the art in quantum software engineering is performed. Works like [49] focus on the software development cycle for quantum applications. Other approaches [50, 51] focus on automatic synthesis of quantum programs.

Classical workflow management systems such as Pegasus, ASKALON, and DagsHub² are described respectively in [52, 2]. Execution of workflows in hyper-heterogeneous architectures is described in [53]. Also, [54] proposes a characterization of workflow management systems for data-intensive applications. Support for quantum workflow is provided in tools such as Orquestra [55] and Covalent [56]. In this work, we provide guidelines on how to adapt existing classical scientific workflows into hybrid quantum-classical scientific workflows and how to extend existing classical WMS to integrate both quantum and classical hardware.

We extend the outlined works by focusing on the integration of quantum machines in the execution of HPC applications. We generalize the concept of hybrid quantum-classical workflows, defining different execution models for hybrid quantum-classical applications and validating our findings on a real-world molecular dynamics simulation workflow. Based on our findings, we identify open challenges and possible solutions for the integration of quantum devices in HPC applications.

3. Background

3.1. Scientific Workflows

Scientific Applications in different domains (i.e., finance, biology, chemistry, engineering) can be decomposed in elementary *tasks* (i.e., aggregate data from different sources, average a set of samples, apply a method to a specific dataset). Tasks can be combined into *workflows*, represented as directed acyclic graphs (DAGs) [52, 2, 57] where nodes represent the tasks and edges model data and control dependencies between tasks.

Definition 3.1 (Scientific Workflows). A workflow W can be formally defined as a DAG $W = (T, E)$, such that T is a set of tasks and E the set of edges, with $E \subset T \times T$.

Workflows modelling scientific applications are called *scientific workflows*. Workflows and tasks can be stored in public

²<https://dagshub.com/>

repositories (i.e., Pegasus workflow gallery³), allowing re-use of validated code, *repeatability* of simulation, (possibility to easily repeat the setup and execution of a simulation), which increases confidence in simulation’s results, and reproducibility of computation (possibility to reproduce and verify results of computation), creating opportunity for new insights and reducing measurements errors. Also, workflows are fundamental for the development of *standardized*, *robust*, and *accurate* simulations of different phenomena.

3.2. Workflow Management Systems

Execution of scientific workflows on HPC infrastructures requires different software layers, to enable (1) scheduling of workflow tasks onto different computing resources, (2) management of data, including intermediate data products (either streaming data, or scientific datasets), (3) interoperability between different heterogeneous resources (e.g., Cloud/Edge nodes, academic clusters), and (4) fault tolerance (e.g., checkpointing of execution, re-execution of tasks).

3.3. Quantum Computing

Recent years have seen a major boom in the areas of quantum information processing and quantum technologies. The huge surge in academic interest and industrial investment in quantum happened more or less after the seminal publication by Google Inc. on *Quantum supremacy using a programmable superconducting processor* [31]. Along with neuromorphic computing architectures, quantum computation, and quantum simulation have emerged as some of the most promising paradigms in alternative computing architectures. Multiple quantum platforms based on superconducting qubits like the IBMQ universal quantum computer [58], programmable atomic arrays [26], trapped-ion quantum computers [33], *D-Wave* 2000Q and 5000Q quantum annealers [34] exist today with the promise of accelerating a wide range of problems that would typically be impossible to solve or simulate on a classical (hardware and software) computers. Some of these include Shor’s groundbreaking integer factorization algorithm [20] offering superpolynomial speedup, Harrow-Hassidim-Lloyd (HHL) algorithm for solving a large system of sparse matrices with exponential speedup [41], and accelerated linear algebra computations like matrix multiplication [59]. Other highly relevant domains include combinatorial optimization (NP-hard problems), finance, machine learning [60, 61, 62], battery design, new novel molecule and drug discovery, quantum materials, grid power management to name a few.

Although the potential to accelerate time to solution using quantum machines is huge, quantum computation is still in its very beginning, suffering from many hardware and software imperfections. Currently, the technology for initial state preparation of quantum registers, precise qubit control, high-fidelity quantum gate preparation, and measurement of qubits involves

a high level of uncertainty. These can be traced mainly to ultra-precise engineering bottlenecks and environmental errors induced by *decoherence*. For example, the measured fidelity of 2 million samples on the 53-qubit *Sycamore* chip fabricated by Google Inc. [31, 63], described in terms of the linear cross-entropy benchmark (XEB), is only at a level of 0.2 % [31, 13]. This does not keep up with the performance of classical simulators, which can be exponentially complex, but provide higher fidelity compared to quantum devices.

3.4. Hybrid Quantum-Classical Systems

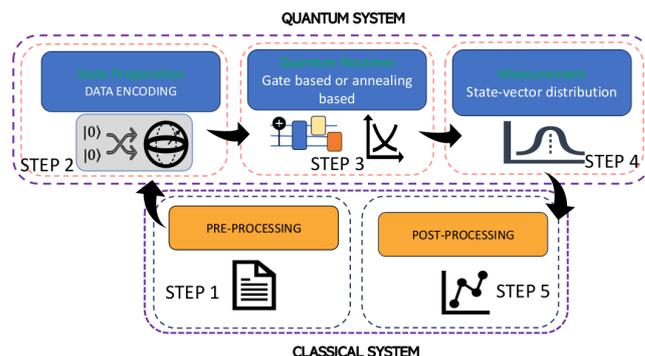


Figure 1: Schematics of hybrid quantum-classical systems.

Hybrid quantum-classic systems define a new class of computing paradigms combining the computing capacities of near-term noisy quantum processors and classical co-processors working in conjunction to solve large-scale scientific problems. The main advantage of using this approach is that it allows the exploitation of the many strengths of classic processors for multiple tasks (e.g., convex optimization, error correction, data pre/post-processing) while at the same time utilizing the capabilities of quantum machines for other specific tasks [23]. The motivation behind hybrid ecosystems is to orchestrate quantum algorithms with classical routines which are more suited and efficient on classical processors (for e.g. classical optimizers for back-propagation computation, data entry, graphics, data pre/post-processing etc.). This allows for the distribution of larger workloads to classical devices, thereby mitigating the burden on error-prone quantum hardware, which is leveraged for specialized and targetted tasks (for instance, quantum phase estimation (QPE), cryptographic schemes, operator quantum expectation value computation, optimization, etc.). This strategic integration yields a substantial reduction in the utilization time of the quantum resources, enhancing the efficacy of task executions. Figure 1 describes the hybrid system pipeline: In step 1, data is pre-processed on the classic system for further encoding onto the quantum registers; in Step 2, the quantum state is prepared based on preprocessed input (typically done using data encoding schemes), Step 3, manipulates a quantum circuit; in step 4, the quantum state is measured and post-processed in step 5.

Data Encoding For Quantum Devices: Classical data in its raw form cannot be processed on quantum devices. Desired initial quantum *state preparation* necessitates converting

³https://pegasus.isi.edu/workflow_gallery/

and encoding the input classical vector (tensor) data in a suitable representation as *quantum data* for embedding, storing the quantum information in the QPUs, and performing quantum operations via quantum algorithms. Porting classical data sets onto quantum devices can be achieved efficiently using multiple DATA ENCODING schemes [64, 65]. In general, choosing a particular data encoding method depends on the use case (model/algorithm dependent). Some of the well-known encoding frameworks are (i) Basis Encoding, (ii) qRAM Encoding, (iii) Angle Encoding, (iv) Amplitude Encoding, etc.

Parametrized Quantum Circuits: Parametrized quantum circuits (PQCs) or variational circuits are basically quantum algorithms that vary certain variables (real/complex valued vectors) or parameters, often denoted as θ . PQCs like any other quantum circuit consists of, (1) Initialized qubit register (cf. Appendix B, section(Appendix B.3) with appropriate quantum state preparation, (2) a quantum circuit with a cascade of quantum logic gates (cf. Appendix C and Appendix D) $U(\theta)$, parameterized by a set of parameters θ (3) classical optimizers augmenting the PQCs and, (4) measurements and resets.

Variational Quantum Algorithms (VQAs) are one of the most important paradigms in hybrid quantum-classical systems and are prime candidates for quantum advantage. VQAs can be defined as hybrid quantum-classical algorithms, wherein a parametrized quantum circuit is iteratively optimized via classical optimization algorithms. The schematic diagram Figure 1 shows that the black box performs VQA executions. The black box can be decomposed into two major blocks, namely the quantum block and a classical block, interconnected by an underlying adaptive feedback-loop mechanism.

- **Quantum System:** A *Noisy-Intermediate-Scale-Quantum* (NISQ) device that at the low level prepares highly entangled parameterized quantum states and performs quantum-subroutines using PQCs (variational circuits). For e.g. this block executes a *forwardpass* by computing the quantum expectation value of certain physical observables /operators (matrices) and the measured quantum state yields the corresponding parameter values which are stored in the memory at every intermediate step.
- **Classic System:** A *classical* optimizer which receives the quantum outputted parameters and executes iterative optimization, (for e.g. gradient descent) by optimizing the cost (loss) function landscape. Calling gradients functions for parameter θ_i update is done using *back-propagation* (accumulated gradients). The data flow in this stage corresponds to passing the updated parameters loop back into quantum circuit for further quantum function calls (gate operations) and initiate subsequent control flow steps.

Other Hybrid Algorithm Frameworks Apart from VQAs, there are other classes of hybrid algorithms, typically analog-based, that leverage existing quantum architectures. These include hybrid forms of quantum annealing, such as hybrid solvers in D-WAVE machines for solving arbitrary structure QUBO

problems (quadratic models, such as Binary Quadratic Models (BQMs), Constrained Quadratic Models (CQMs) or Unconstrained Quadratic Models, Discrete Quadratic Models (DQMs)) etc⁴. Frameworks such as *iterated* adiabatic reverse annealing which are quantum annealing-based techniques embedded in a classical loop have also been useful in tackling multiple scientific and industrial use cases [66]. Some more ongoing developments in the hybrid algorithms sectors are *Quantum Neuromorphic Computing*, wherein brain-inspired classical neural network architectures are conjoined with quantum hardware to offer computational advantage [24].

4. From Classic to Hybrid Workflows

In this section, we provide the main definitions of what is needed to enable our vision of hybrid workflow execution, focusing on hybrid workflows and then on hybrid WMSs.

4.1. Hybrid Workflows

We extend the definition of classic workflows by adding a set of *quantum* tasks Q to Definition 3.1. Quantum tasks $q \in Q$ that can be executed only by quantum machines. Also, quantum tasks in our workflow definition are *functionally equivalent* [67] to some tasks in T . Because of the undecidability of the program equivalence problem [67], we assume that the user defines a mapping function $f : T \mapsto Q$ that maps a classic task into its quantum equivalent. Multiple quantum tasks can be available for different classic tasks: for example, for the classic task of computing a matrix eigenvalue, either HHL [68] or VQE [44] algorithm can be used, depending on available quantum hardware. As a consequence, f is surjective, but not injective. We define *quantum candidates* tasks as tasks for which there is a quantum task, namely,

$$\forall t \in T : f(t) \neq \emptyset, t \text{ is a quantum candidate.} \quad (1)$$

We define the set T' as the set of quantum candidates. We assume that $T' \subseteq T$. If a task t' is a quantum candidate, we add a *decision* node between t' predecessor and connect it to t' and all other tasks $q \in f(t')$. Different execution paths will be executed according to *conditions* or specified by the decision node. An example of a condition could be, for example, to execute the quantum task if quantum hardware is available. The set of decision nodes is defined as D . Figure 2 provides an example of how to transform a classic workflow into a hybrid workflow.

We distinguish three types of quantum tasks, that are described in Figure 3 together with their control and data flow: *circuit execution*, *task execution* (Figure 3b, and *hybrid execution*. A circuit execution (Figure 3a) represents a single execution of a quantum circuit, where a single sampling of the result of execution is performed. Examples of this computation can be simple algorithms, such as the simulation of a coin toss. The result of a task execution, instead, is the most frequent result

⁴https://docs.dwavesys.com/docs/latest/doc_leap_hybrid.html

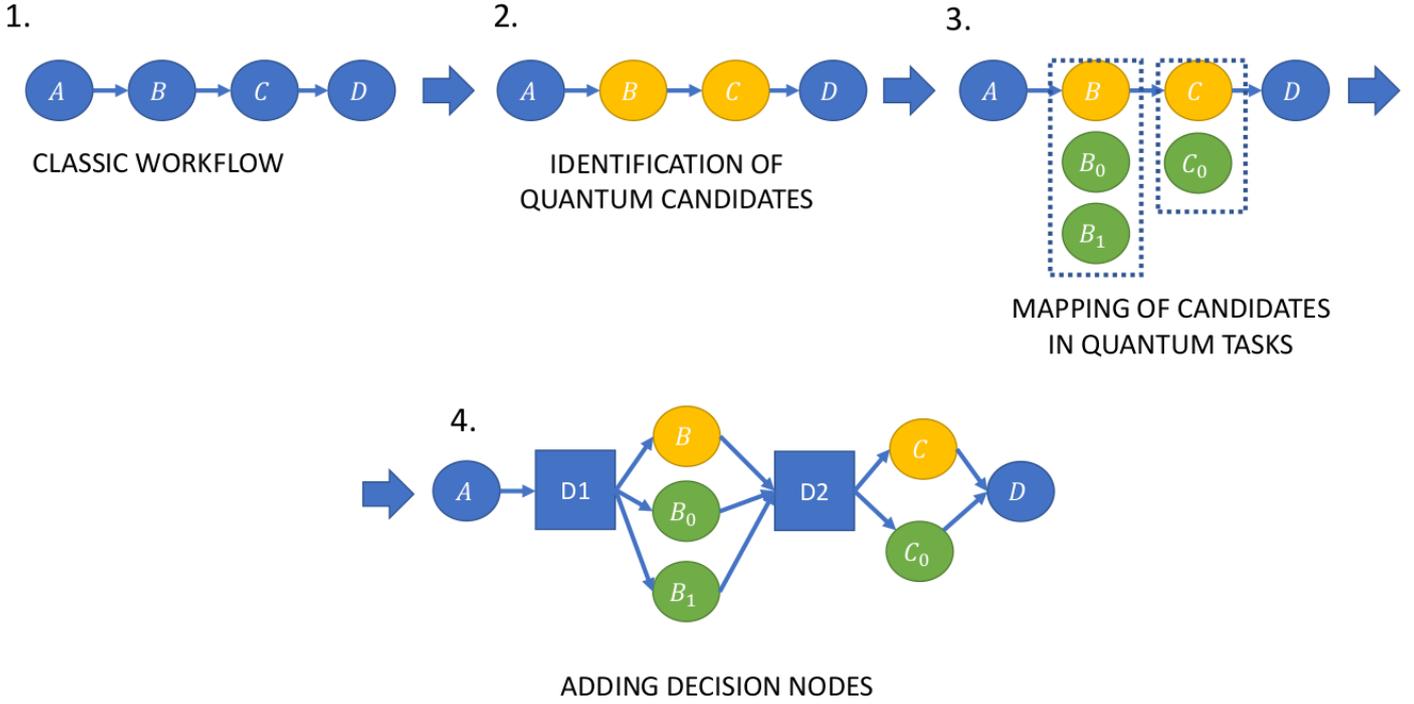


Figure 2: Example of Transformation into Hybrid Workflow

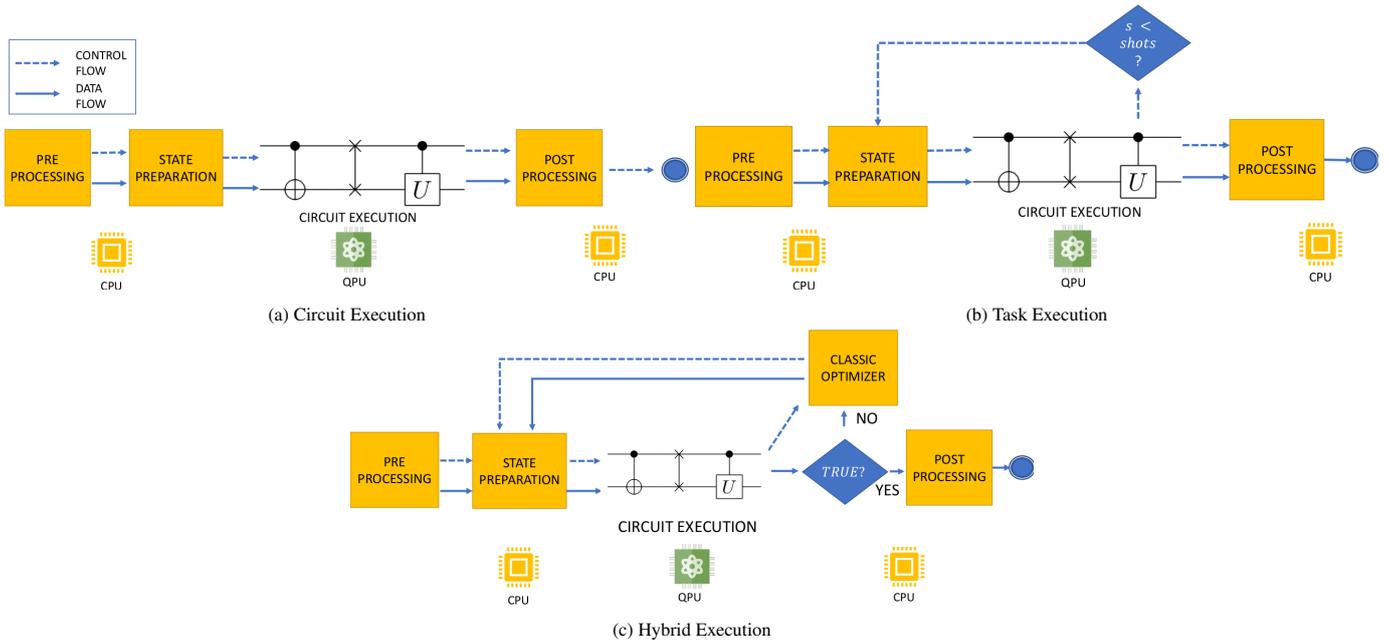


Figure 3: Types of Quantum Tasks.

among s samplings of the execution of a quantum circuit and represents the execution of typical quantum algorithms such as Grover and Deutsch-Jozsa. This model can also be applied to the quantum subroutines of hybrid algorithms, such as Shor's algorithm [20]. Finally, a hybrid execution (Figure 3c) represents computations involving interaction between classic and quantum hardware, similar to what happens with Variational Quantum Algorithms [44], where a quantum state modeling the solution to a specific problem is modeled as a parametrized

quantum circuit with a vector of parameters $\vec{\theta}$, whose optimal values are found by optimizers running on classic hardware. Finally, we define hybrid workflows as follows:

Definition 4.1 (Hybrid Workflows). We define hybrid workflows as $W = (T, Q, E, D, f)$, where:

- T is the set of classic tasks, where $T' \subset T$ is the set of quantum candidates;
- Q is the set of quantum tasks;

- D is the set of decision nodes, where $|D| = |T'|$;
- $E \subset (T \cup Q \cup D) \times (T \cup Q \cup D)$ is the set of edges;
- f is the function mapping tasks in T' in one or more quantum tasks in Q .

5. A Molecular Dynamics Use Case

MD is one of the most popular scientific applications executed on modern HPC systems. MD simulations reproduce the accurate dynamics (time evolution of molecular systems) at a given pressure and temperature by iteratively computing inter-atomic forces and atom dynamics over short time steps. The trajectories generated by these simulations enable a better understanding of conformations and molecular mechanisms. In particular, a trajectory is a series of frames, i.e. sets of atomic coordinates stored at fixed infinitesimal time steps [69].

The quantum adaptation of MD applications builds over a similar pipeline as put forth in the recent works of [70, 69]; dealing with *in-situ* and *in-transit* analytics of MD simulations on state-of-the-art supercomputers. Our hybrid workflows can be efficiently integrated with the *in-situ* and *in-transit* analysis of the data and meta-data generated by MD simulations.

Here, we provide a high-level description of the hybrid workflows pertaining to the aforementioned use case. The motivation to focus on this particular use case is as follows: (a) MD simulations/analyses are a very active field in the distributed deep learning and HPC scientific computing communities with widespread applications to industry. (b) It consists of a purely quantum-based algorithm stack (see Section 5.2) followed by an additional variational hybrid algorithm stacks (VQEs) on the quantum algorithm returned output (see Section 5.2). (c) It is elusive to find applications such as in [70, 69], that offer one-to-one mapping of the entire classical problem (two major compute-intensive tasks) on a hybrid ecosystem and capture the essence of the hybrid workflow pipeline.

Our initial observation was that specific tasks of the *in-situ* MD simulations, for e.g., collective variables (CVs) generation, can be leveraged using state-of-the-art quantum algorithms and quantum subroutines instead of merely parallelizing workloads using GPU accelerator facilities. Just in the way GPUs, TPUs, and FPGAs serve as indispensable tools for accelerated linear algebra for variable dimension tensor calculations (matrix-matrix, matrix-vector, vector-vector multiply), the same logic also carries over to devices based on quantum architectures. It is clear from the current status of quantum computers that a complete MD application cannot be executed on the rudimentary NISQ quantum hardware. Thus, we need to identify the best-suited parts (subspaces) of the workflows that can be accelerated multifold using quantum processing units (QPUs). Moreover, the utilization time of quantum devices should be kept low-key, in order to prevent noise-induced quantum errors and imperfections of quantum hardware.

Figure 4a visualizes the target MD application with quantum candidates highlighted. After reading input from the user,

the application reads a trajectory file, which provides information about the molecule structure, and identifies the atom segments that have to be considered in our calculations. For each pair of atom segments, the application performs parallel computations to calculate B_{IJ} matrices. The CV that we will use in this work is the *Largest Eigenvalue of the Bipartite Matrix (LEBM)*. The bipartite matrices are used as an input for the LEBM calculation. At the end of this process, results for different B_{IJ} are collected and analyzed.

Starting from classical MD workflow (Figure 4a), by applying the procedure described in Figure 2 we obtain the resulting Hybrid Quantum-Classical workflow described in Figure 4b. These quantum-classical (hybrid) counterparts of the purely classical MD workflows were first conducted and benchmarked on a 5-qubit IBM Q devices [38]. In the next sections, we describe each step in detail.

5.1. Identification of Quantum candidates

Strategies for pinning down suitable quantum candidates in a use case is to a large extent problem/model/system dependent. In a hybrid workflow environment, this basically boils down to choosing from a limited set of available quantum and hybrid quantum-classical algorithms that could potentially outperform known classical algorithms.

In the MD simulation landscape, compute-intensive data analysis of time-evolving molecular systems can be systematically solved by designing a suite of collective variables (CVs) [70]. Collective variables (CVs) are a set of statistical metrics that capture relevant molecular motions enabling efficient monitoring of rare events in huge molecular structures and chains [71]. Technically, CVs can also be defined as a function of the atomic coordinates in one frame that helps to reconstruct the free-energy surface for enhanced sampling. Since trajectories are reduced to time series of a small number of such CVs, simulated molecular processes are much more amenable to interpretation and further analysis. Ideally, a CV can be as simple as the distance between two atoms or can involve complex mathematical operations on a large number of atoms. For example, in the domain of Metadynamics [72], scientists use well-chosen collective variables (CVs) to capture important molecular motions in the region of interest.

The works in [70, 69] demonstrated that collective variables could be extracted using Euclidean distance matrices and bipartite distance matrices.

The Euclidean distance matrix D and the bipartite distance matrix B_{IJ} have three fundamental properties: they are symmetric, diagonal elements are zeros, and off-diagonal elements are strictly positive [70]. After calculating matrices D and B , we focus on calculating the molecular system's CVs.

Potential quantum candidates for MD-based experiments were chosen in accordance to:

(i) Replacing classical routines that are compute-intensive with equivalent quantum algorithms that guarantee a theoretical speedup. A task $t \in T$ is defined as **compute-intensive** if

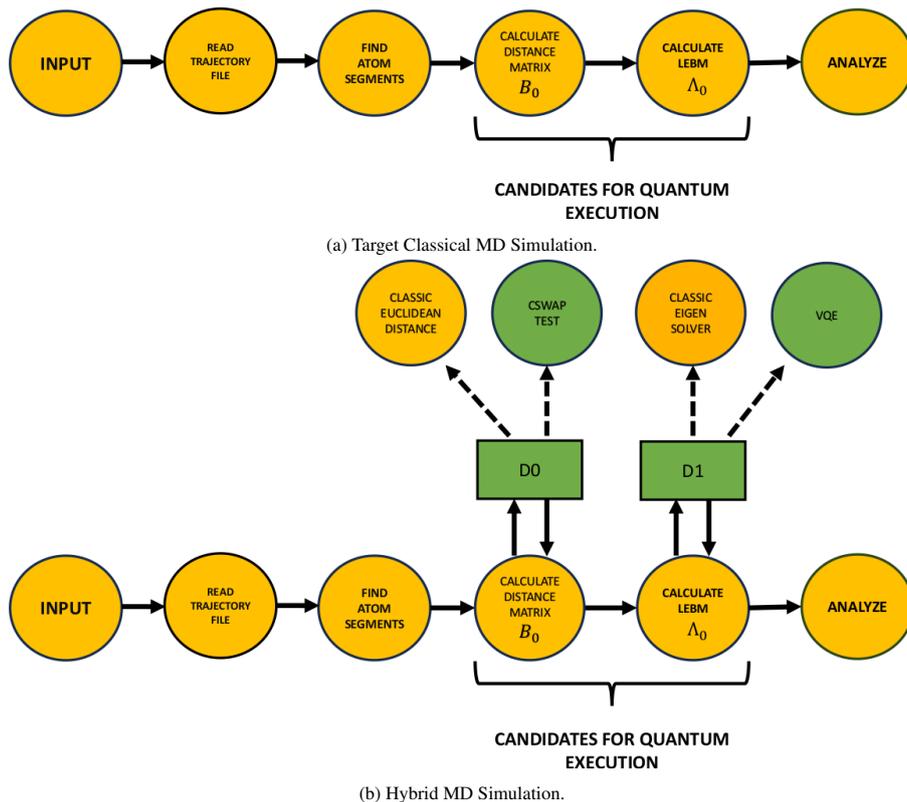


Figure 4: Hybrid Quantum-Classical MD Simulation.

at least 70% of its execution time is spent in performing floating point operations (i.e., no data staging, I/O, and communication).

(ii) Capturing the electronic properties and molecular time-evolution of the system which in turn can be used for efficient computation of collective variables (CVs),

(iii) Most importantly, whether there exists such a quantum algorithm and routine, that is available to the user for successful implementation of the use case. If not, then engineering quantum algorithms/quantum circuits with a considerably shallow circuit depth, (with a modest qubit register size and quantum gates crammed into the circuits) such that the problem can be efficiently simulated on NISQ hardware.

The two classical tasks that exactly satisfy all the aforementioned requirements in our use case are:

(a) Distance (bipartite) matrix generation between different C_α atoms in the molecular system (cf. Target Task 1 5.2) for details), which is calculated using SWAP test;

(b) Calculating target CV corresponding to largest eigenvalues of the bipartite matrices (LEBM) and distance matrices using hybrid quantum algorithms (cf. Target Task 2 5.2) for details). We use Variational Quantum Eigensolvers (VQEs) [73] for eigenvalue estimation.

5.2. Mapping from Classic to Hybrid Workflow

In Figure 4b we show the hybrid workflow resulting from the transformation of the classic MD workflow depicted in Figure 4a, following the transformation procedure described in Fig-

ure 2. Since we have two quantum candidates, we add two decision nodes, $D0$ and $D1$. Decision nodes are responsible for selecting target implementation, depending on whether quantum hardware is available, and performing data encoding from classic to quantum domain. We analyze the implementation of each algorithm in the next sections, together with the control and data flow of target hybrid MD simulation.

Target Task 1: Quantum algorithm for distance matrix generation & computing structural change evolution

It was shown in [74] that quantum computers possess the power to manipulate large numbers of high-dimensional vector/tensor datasets. Typically, vector operations involving vector (tensor) dot products, norms, overlaps, etc., feature in supervised and unsupervised machine-learning tasks. For our MD analytics use case, CVs described by distance/bipartite matrices constitute some of the most compute-intensive routines. Classical algorithms running on classical devices typically require polynomial time in the number of vectors and the dimension of the space to solve these tasks. For an N -dimensional vector space, their time complexity grows linear in N , i.e., $O(N)$. Quantum computers, on the contrary, can remarkably achieve this feat in time $O(\log N)$, owing to their intrinsic capabilities to efficiently manipulate high-dimensional vectors embedded in large tensor product spaces [74].

The reason for this exponential speed-up can be argued as follows: classical data (typically vectors or tensors), expressed in terms of N -dimensional complex-valued vectors can be en-

coded through amplitude encoding onto merely $\log_2(N)$ qubits, thus requiring only logarithmic in the size of the classical dataset. This data stored in a *quantum random access memory* (qRAM) whose mapping takes $O(\log_2 N)$ steps [75]. In this converted quantum form, data post-processing can be done using multiple quantum algorithms like the *quantum Fourier transforms* [76], matrix inversion methods [41], etc., with time-complexity $O(\text{poly}(\log N))$. Thus, distance estimation and inner product operations between post-processed vectors belonging to N -dimensional vector spaces take merely time $O(\log N)$ [74]. Moreover, as per [77], sampling post-processed vectors and distance or inner-product determination between these post-processed vectors is an exponentially hard task.

Inspired by the aforementioned (theoretical) *exponential* quantum speedup, we make use of quantum algorithms to generate the proxies for modeling molecular structure time-evolution, i.e., Euclidean distance matrices D and bipartite distance matrices B_{IJ} . These can be efficiently generated via a quantum subroutine called the *SWAP* (also called *C-SWAP*) test.

The high-level workflow description of the **Target Task 1** with quantum integrated architectures is depicted in Figure (5). We can see that its execution resembles the control and data flow of a job execution, as shown in Figure 3b. Typically, it requires manual re-direction of tasks onto quantum devices for generating distance matrices. The proposed workflow schematic is a quantum adaptation (counterpart) of the *in situ* or *in transit* integrated classical workflows used for molecular dynamics (MD) analytics, originally introduced in [78].

Target Task 2: Largest Eigenvalues of Bipartite Matrix
Hybrid Quantum-Classical algorithm for Eigenvalue Determination: It was first posited in the works of Johnston *et al.* [79, 78] that the measurement of largest eigenvalues of the bipartite matrices B_{IJ} or the Euclidean distance matrices D_I can serve as collective variables (CVs) and suffices for monitoring structural changes in the conformation of I relative to J . Later, this idea was extrapolated in [69, 70], wherein the largest eigenvalues of each one of these matrices were computed without retaining other frames in memory. Nevertheless, with ever-increasing molecule sizes, numerical linear algebra calculations such as eigenvalue and eigenspectrum (typically using numerical diagonalization techniques) or singular value decomposition (SVD) computations may fall short due to the saturating computational power of classical (HPC) systems.

Hybrid quantum-classical systems offer a possibility to alleviate the aforementioned bottleneck by semi-utilizing quantum devices as accelerators orchestrated by classical optimization protocols for parameter updates. The *variational quantum eigensolver* (VQE) [45, 47, 43] form an important subset of the variational quantum algorithms (VQAs), that computes the *eigenvalues* of typically large Hermitian matrices \hat{A} (see Appendix F) using the *Rayleigh-Ritz* variational approach [80]. This heuristic algorithm was developed with a strong focus on solving the ground state of many-body interacting quantum systems (strongly correlated) using iterative numerical optimization. Multiple highly complex systems appearing in quantum chemistry remain intractable even for the capabilities of cur-

rent leading-edge high-performance computing (HPC) systems. Augmenting NISQ devices or the near-future quantum devices to operational support given by supercomputers increases the hopes for a faster convergence to solutions for such large-scale chemistry target applications based on quantum simulations. [44, 81, 82]

The LEBM computation of the target matrices B_{IJ} or D falls as a fitting use-case for the variational quantum eigensolver engine. The machinery is described in detail in Appendix F. Initially one begins with an ansatz wavefunction (trial state) outputted from a parameterized quantum circuit. The core principle operating under the hood is iteratively updating the wavefunction parameter whilst minimizing the *cost function* Eq.(2) $C(\theta)$ by employing a classical optimizer. This cost function is typically chosen to be the quantum expectation value of a given hermitian matrix with respect to the parametrized wavefunction $|\Psi(\theta)\rangle$. The input to the VQE engine is either the C-SWAP quantum-subroutine generated bipartite matrix or generated bipartite matrices generated using classical machines. (which is Pauli encoded, since it is a Hermitian in nature cf. Appendix E), the corresponding function to minimize reads,

$$\theta^* = \arg \min_{\theta} C(\theta) = \langle \psi(\theta) | \hat{B}_{IJ} | \psi(\theta) \rangle, \quad (2)$$

The goal is to approach the sets of LEBM as close to the actual values, i.e., the values calculated on a classic machine. However, since VQE performs iterative optimization of a cost function C (Equation 2), it can approach the exact value only after multiple executions or iterative loops [44]. To this end, we define the error function that we use to quantify the VQE benchmarks.

Our quantum-enhanced MD simulations were conducted on a 5 qubit IBMQ hardware. The maximum permissible matrix dimension possible with our requested quantum resources was a restrictive 16×16 dimensional distance matrix blocks E_{IJ} that feature in the bipartite matrix. The corresponding *VQE scientific workflow* for molecular-dynamics (MD) target applications (cf. Appendix F for detailed mathematical description) is depicted in Figure 6, which resembles hybrid execution on Figure 3c.

Cost Function Optimization. For the \hat{B}_{IJ} , we define its classic LEBM $\Lambda_c(B_{IJ})$ and its quantum counterpart calculated using VQE using a specific initially chosen hyperparameter setting Π as $\Lambda_{vqe}(\hat{B}_{IJ}, \Pi)$. We construct a figure of merit to compare VQE results obtained from different architectures, quantified by a Mean Square Error (MSE). For a set of matrices, $\hat{B} = \{B_{IJ}^0, B_{IJ}^1, \dots, B_{IJ}^n\}$, we define

$$Err(\hat{B}, \Pi) = \sum_{i \in [0, |\hat{B}|]} \frac{(\Lambda_c(B_{IJ}^i) - \Lambda_{vqe}(\hat{B}_{IJ}^i, \Pi))^2}{|\hat{B}|} \quad (3)$$

as the MSE between the classic and quantum eigenvalues, calculated using hyperparameters Π .

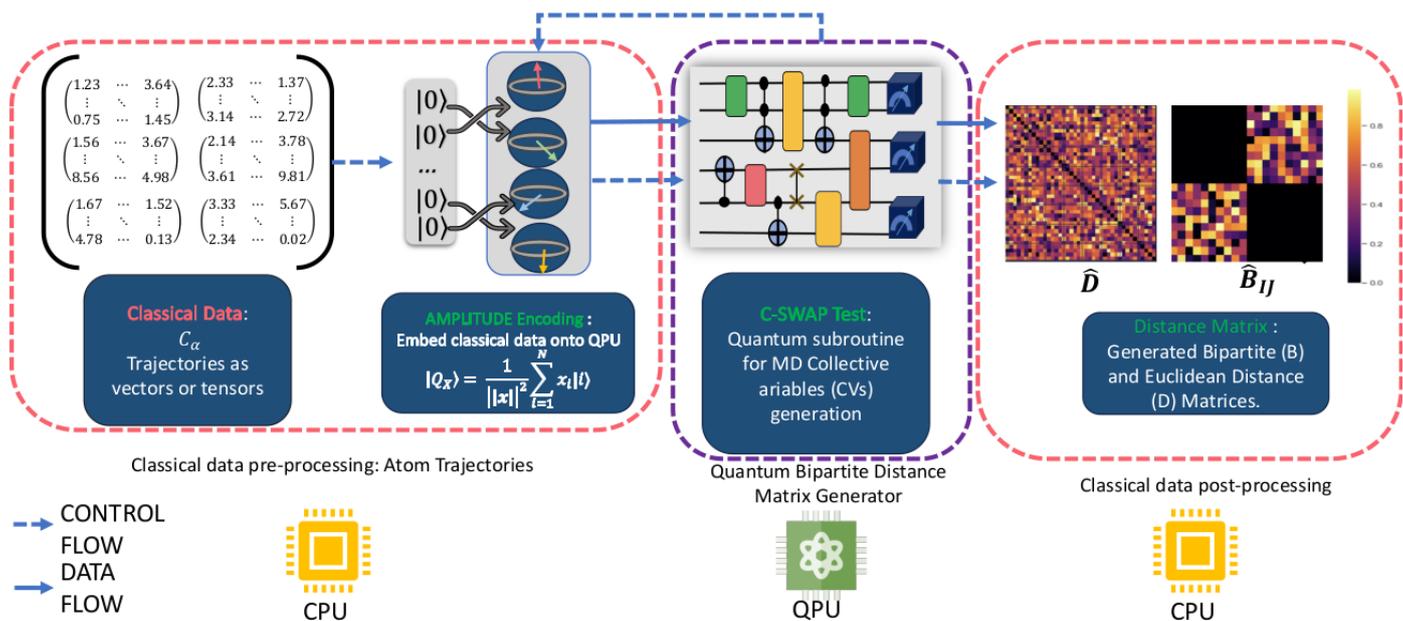


Figure 5: Quantum integrated MD workflows for distance matrix computations.

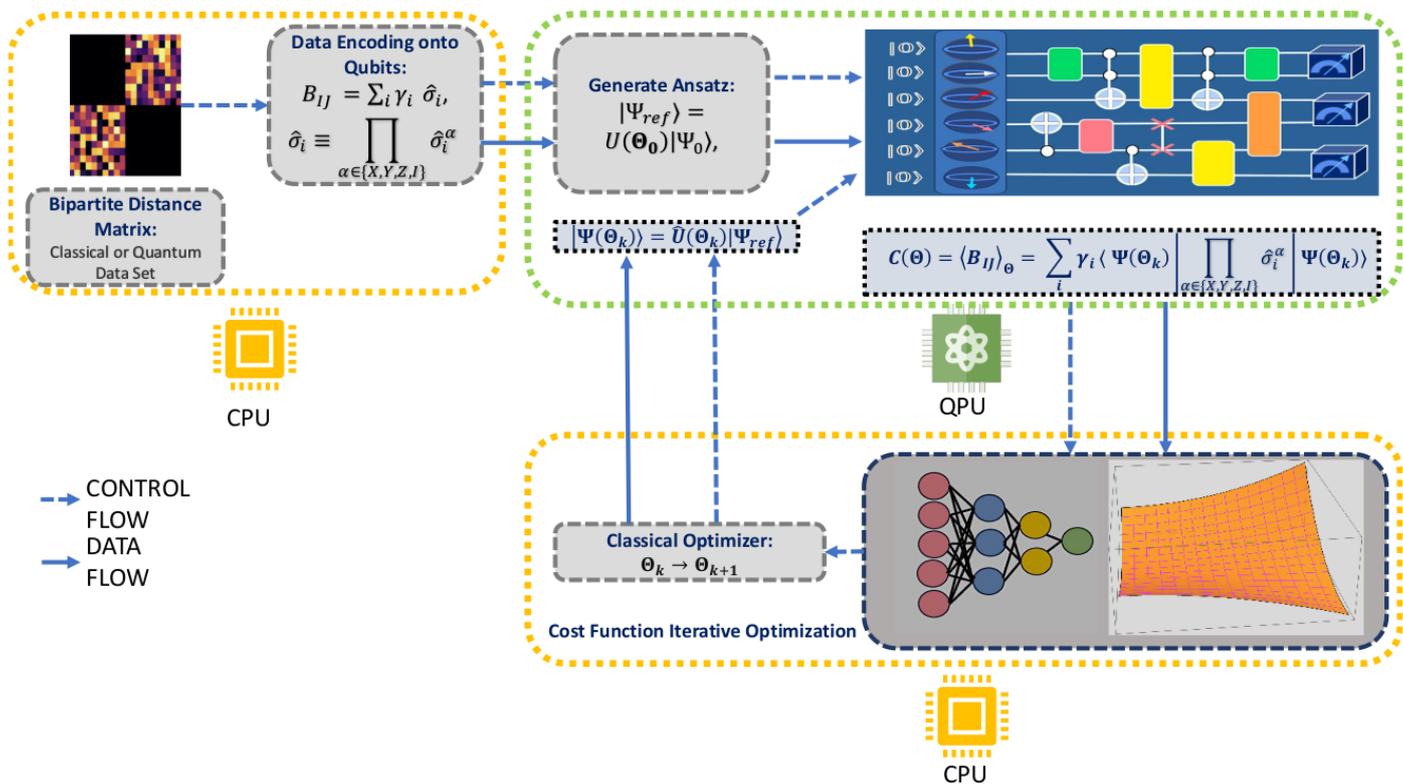


Figure 6: Quantum workflows for variational quantum eigensolvers

Data-Driven Hyperparameters Selection. It is important to note that the VQE executions can be largely improved by opting for different hyperparameter Π tuning schemes that are available on the IBM Qiskit API. In practice, VQE execution results are affected by tuning hyperparameters and converge to the optimal value after the user pins down a suitable configuration after taking into account possible error-mitigation strategies. There-

fore, a grid-search-based algorithm forms an important part of the so-called data-driven methods which will be used to identify the most suitable hyperparameters setting. In [38], we describe a method to select a suitable set of hyperparameters for VQE.

5.3. Generalizing Hybrid WMSs

The rapid progress of quantum facilities has led to a surge in challenges across various application domains. Consequently, generalizing the hybrid workflow frameworks to accommodate diverse applications is necessary. A comprehensive overview is depicted in Fig. (2), which could encapsulate a broad spectrum of techniques and be embedded at various stages in a generalized hybrid ecosystem. These encompass (a) Hamiltonian simulation and analog/single ancilla Linear Combination of Unitaries (LCU) [83] methodology for ground state preparation, (b) Graph-based combinatorial optimization employing quantum-approximate-optimization algorithms (QAOA) [84], (c) Quantum Fourier Transformations (QFT) and Quantum Phase Estimation techniques (QPE) as alternatives to Variational Quantum Eigensolvers (VQEs), (d) Classical machine learning (ML) assisting quantum algorithms [85], (e) Quantum machine learning (QML) workloads such as quantum support vector machines (QSVMs) [86], Quantum Kernel-based ML (QKE) methods [15, 87], and (f) Simulations of quantum Hamiltonians for determining electronic state energies [88, 37]. The following sections build over this generalized scheme.

5.3.1. Hybrid Data/Control Flow

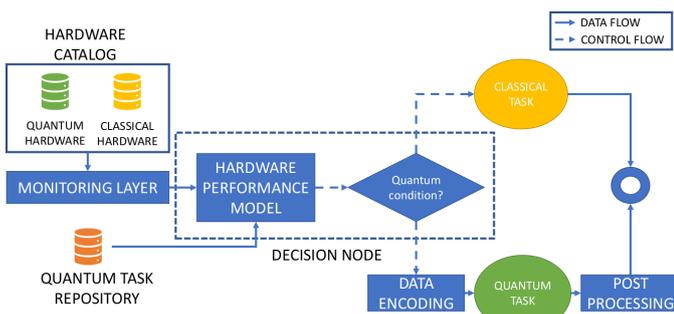


Figure 7: Decision Node.

In a Hybrid Workflow Engine (HWE), characterizing the control and data flows becomes highly relevant. Since the control flows (defining execution paths) and data flows (information movement through computations) are highly interlinked, it is necessary to demarcate and differentiate these two flows systematically. In such intertwined workload sharing environments arising in *variational quantum algorithms* between quantum-classical components, WMSs become highly non-trivial. Here we relate/map the conducted quantum MD computational experiments to the formally defined hybrid workflows from Section 4.1.

The data and the control flow are handled by the decision nodes, whose flow is described in Figure 7. The goal of decision nodes is to manage the control and data flow of the hybrid workflow, based on pre-defined conditions for execution on quantum hardware. Decision nodes collect data about the available quantum and classical hardware through the monitoring layer. Also, available quantum tasks are fetched by the quantum task repository, following indications provided by annotations. Afterward, performance is evaluated for each task on the available

hardware, by applying a specific hardware performance model. Performance models can be either defined as a single metric, i.e., quantum volume [89] or total quantum factor [90], or exploit machine learning-based approaches such as [91]. Based on the results of the hardware performance models, a specific condition is evaluated to make a decision. Based on the condition, the classic or the quantum task is executed. If the quantum task is executed, classic data has to be encoded into the quantum domain by applying different data encoding methods. After execution on quantum hardware, post-processing is applied to mitigate the effects of quantum noise. Finally, the output of the task, either classic or quantum, can be used by the subsequent workflow task.

Similar approaches to the selection of quantum tasks have been proposed. For example, [92] proposes NISQAnalyzer, a method for selecting the most suitable quantum machine among a set of available superconducting quantum hardware and Qiskit implementations, while [93] focuses on the execution of quantum services on Amazon BraKet. In [91], these ideas are further explored considering also ion-traps architecture and different programming frameworks. However, different hardware, for the same algorithm might require different implementations and different programming models. An example is photonics hardware, where Measurement-Based Quantum Computing [94] provides better performance [95]. Therefore, we propose an approach where a joint selection of algorithm implementation and target quantum hardware is performed.

In the next sections, we analyze control and data flow in detail.

Control-Flow Modeling for Hybrid Systems. Control-flow modeling deals with deconstructing the order of operations in a computer program and dictates the sequence in which computational tasks need to be executed [96]. In the case of hybrid quantum-classical applications, execution paths can be bifurcated into the classical control flow pipeline (pertaining to classical flows and smaller sub-flows) and quantum control flow pipeline (related to quantum flows and sub-flows) respectively.

Classical side of quantum computation: Typically, the classical control flow dependencies involve:

1. Initializing set of variational parameters of the quantum circuit.
2. Spawning multiple quantum circuits and processes simultaneously.
3. Delegating (directing) user-defined quantum tasks Q onto the quantum processors and controlling the execution order of one or several quantum candidates.
4. Executing classical programs in concert with quantum routines (e.g., for iterative optimization algorithms).
5. Measurement of quantum states and classical parameter feedback. Lastly, generating the final output after multiple iterations (checking convergence to solution) and data post-processing.

Data-Flow Modeling for Hybrid Systems. Data-flow analysis on the other hand deals with collecting information about the

possible set of values calculated at various stages in a computer program and describes the information passage within a workflow [96]. Data generation, feature extraction, data transformation, data storage, and exchange of input and output data within the workflows are some examples.

Classical Data flow modeling:

1. Preprocessing input classical data, implementing feature extraction methods on classical datasets, and developing suitable forms of data encoding schemes for QPU embedding.
2. Postprocessing results after quantum measurements, mitigating readout errors and extracting useful information.

Quantum Data flow modeling:

1. Quantum data flows concerns with quantum parallelism arising due to multi-qubit states $|\Psi\rangle \in (\mathbb{C}^2)^{\otimes d}$ within a quantum register and the flow of entanglement prepared via controlled gate sequence applications. (for e.g., multi-QPU interactions and managing multi-QPU communication). This may be necessary to communicate a computational state between QPUs or to prepare both registers in a mutually entangled state. These operations require the presence of a quantum network, which uses quantum physical systems to communicate quantum states between registers) [97].
2. Exchanging data within the several spawned quantum circuits or subparts of the different circuits.

6. Hybrid Workflow Management Systems

6.1. Software Components

In this section, we describe the main software components that are necessary to enable the execution of tasks into hybrid quantum-classical systems. The main components are also depicted in Figure 8.

Hardware Catalog contains information about available hardware in underlying hybrid computing systems, including not only which hardware is available, but also different characteristics (i.e., CPU power, storage capacity, network throughput). This component is common in many other WMSs, such as Pegasus [52]. Hardware-specific implementations of the hardware catalog are available in different quantum hardware with Cloud frontends, i.e., IBM Quantum, Amazon BraKet, Google Quantum AI, and Azure Quantum. The main difference between these services is that, while IBM and Google have their own quantum devices, Amazon provides an interface to access quantum devices from other vendors, e.g., Rigetti, Quantinuum, and Pasqal. Azure Quantum, instead, proposes a hybrid solution, exposing the same interface not only for Microsoft devices but also for devices of other vendors. The first step towards the integration of quantum hardware in the hardware catalog would first of all require the design of APIs to interconnect these services with WMS hardware catalog. Existing approaches described in [92, 93, 91] are currently not integrated into WMS,

targeting only specific frameworks (e.g., Qiskit and IBM Quantum [92] or Amazon BraKet [93]) or specific hardware, e.g., superconducting and ion-traps [91]. Also, information about the characteristics of available hardware, also known as quantum hardware descriptors, should be exposed. Typical descriptors are a number of available qubits, qubits topology, and error rate. Moreover, since quantum computers at the time could not be used concurrently, information about the queueing status should be exposed. In some cases, such as IBM, also other performance metrics that are specific to quantum, i.e., Circuit Layer Operations per Second (CLOPS) or Quantum Volume [89], can be exposed. However, such metrics might not be directly applicable to other types of hardware, i.e., from other vendors or relying on different technologies, such as, for example, ion-traps, photonics, or neutral atoms.

Quantum Task Repository includes the implementation of different quantum tasks, in different software variants depending on the available quantum machines. Tasks can be implemented in graphical languages, such as ZX-Calculus [98], different high-level programming frameworks (e.g., Qiskit, PennyLane, Q#), or in low-level languages such as OpenQASM (quantum assembler)⁵. Examples of quantum task repositories are provided by the IBM Quantum Lab, BraKet git repository⁶, Cirq Quantumlib github⁷, Microsoft Quantum github⁸. However, each one of these repositories targets specific frameworks and devices. A generic quantum task repository should include implementations for different devices available in the hardware catalog. Also, each task should be labeled to describe its goal (i.e., unstructured search, optimization), as well as its input and output. Compilation of selected tasks will then be performed in the subsequent transpilation layer.

Classic-Quantum Mapper is responsible for identifying a mapping of classic tasks into equivalent quantum tasks. This can be performed by means of *code classification* [99, 100], using labels defined in the Quantum Task Repository, or by applying circuit synthesis methods, such as [51, 50] if no corresponding quantum task is found. Currently, the mapping of quantum candidates into quantum tasks is performed manually by the application developer. One could think of exploiting code classification approaches [99]

Transpilation Layer is responsible for transpile, i.e., adapting the high-level definition of quantum circuits that define the quantum tasks in the quantum tasks repository, to the target quantum architecture. This step is necessary because the definition of the circuit might not fit the topology of the underlying architecture. To address this issue, a sequence of operations is applied to the circuit definition to reorganize qubits and quantum gates. Transpilers are available in different frameworks, such as Qiskit, PennyLane, and Q#. However, they are mostly designed for circuit-based quantum computing. Work available for measurement-based quantum computing [101], that allows to fully exploit photonics quantum devices is at the moment still

⁵<https://github.com/openqasm/openqasm>

⁶<https://github.com/amazon-braket>

⁷<https://github.com/quantumlib/Cirq>

⁸<https://github.com/microsoft/Quantum>

at the experimental phase. In the future, we should be able to have universal transpilers, capable of optimizing for different computational models depending on available quantum hardware.

Hybrid Monitoring Layer is designed to collect data about the execution of both quantum and classic hardware. On classic hardware, there are many methods to collect either low-level metrics about the systems (e.g., CPU, bandwidth) or aggregated metrics (e.g. reliability), ranging from distributed [102] and centralized [103] monitoring approaches. To enable monitoring of hybrid quantum-classical systems, the existing monitoring layer provided by existing WMS should be integrated with API provided by existing quantum framework (i.e., Amazon BraKet, IBM Quantum, Xanadu) to expose to the workflow developer the hardware descriptors collected by quantum computers, described in the hardware catalog. The implementation of the monitoring service for each provider is related to the underlying framework: for example, in IBM Quantum, it is provided by the runtime service, that allows keeping track of the state of a submitted quantum job by querying the state of a Job object. Similar approaches are available on PennyLane and Q#.

Hybrid Intercommunication Layer constitutes the API that allows communication between quantum and classic hardware. This layer allows data encoding, offloading of data and computation to quantum hardware, retrieving measurements, and performing error correction. Currently, such features are framework-specific and therefore depend on the available API between each framework and the hardware vendors.

6.2. Hybrid Workflows Execution

Figure 8 summarizes how we envision the execution of hybrid workflows. First, the user submits a scientific workflow using the WMS interface. Workflow tasks are then selected by the WMS scheduler, based on its scheduling policy. Based on the workflows annotations, the scheduler knows if the current t task is a **classic** or a **quantum** task. If t is classic, the scheduler will select target machine based on (1) task requirements, (2) availability of hardware resources (based on input hardware catalog), and (3) task-related cost function, which takes in input task t and machine m and outputs a score, which defines whether it is convenient to execute t on machine m and it is used by the scheduler for its decisions. Execution is then performed on the identified machine m , and the results of task t are forwarded to the user or to the following tasks that require them as input.

If t is a quantum task, the first thing to do is to verify whether a quantum **target**, i.e., an equivalent quantum algorithm for the classic quantum candidate, is available in WMS quantum codebase. If no equivalent quantum target is found, the workflows continue its execution by scheduling the classic task on the available classic hardware; otherwise, according to the logic implemented in the decision node, the WMS can decide either to (1) select a quantum algorithm among the available quantum targets, or (2) execute the classic implementation of the task. In the first case, execution depends on the quantum task type (as defined in Figure 3b), in the latter, execution proceeds as in classical workflows.

6.3. Assumptions and Limitations

Description of quantum task repository, as well as transpilation as mapper, are focused on the quantum circuit model, that is typical of superconducting machines used by the most common quantum hardware platforms (e.g., IBM Quantum, Amazon BraKet) and also by ion-trapped machines (e.g., AQT). However, different technologies are available for quantum hardware, such as photonics quantum machines (e.g., Xanadu). While each circuit can be also executed on photonics hardware, these machines proved to have the best performance with measurement-based model [94], therefore circuit synthesis and mapping might not guarantee the best performance.

Another issue is related to the code classification-based approach since we assume that there is a label for each classic task that allows us to map it to its quantum counterpart. As a consequence, this approach is inherently dependent on the source code corpus that we use to train the model.

Concerning hardware descriptors, we mostly focus on definitions provided by IBM Qiskit, such as quantum volume and CLOPS, whose definition is based on superconducting IBM machines. While these metrics could be extracted also for other types of hardware employing the circuit-based model, they might not be applicable to annealers or photonics.

7. Challenges

Based on our vision of execution of hybrid workflows, we identify the challenges that must be addressed to realize our vision of hybrid classic/quantum workflows.

7.1. Quantum Hardware Descriptors

As already mentioned in Section 6, hardware descriptors should be part of the hardware catalog, to characterize different available quantum hardware and to perform decisions on whether to allocate. The first challenge is to identify descriptors that can be used to describe quantum hardware: while on classic CPUs we have well-known descriptors (e.g., CPU frequency, RAM frequency, and memory, available storage), quantum hardware cannot be characterized using the same descriptors, due to the inherently different computational model. Also, there are different hardware technologies available for quantum computers (e.g., superconducting, neutral-atoms, ion-traps, photonics, and annealers), which makes it difficult to identify the descriptors most impacting the performance of quantum computing.

To address this challenge, first of all, in-depth empirical studies are required to identify the most impacting parameters. Empirical studies require also the definition of purposely designed benchmarks, similar to [31, 34, 104]. Based on data collected from the benchmarking, we can identify a subset of descriptors that is of interest for a specific quantum hardware. Descriptors can be also composed of aggregated hardware parameters, such as quantum volume [89], that is used by IBM Qiskit.

Moreover, applications need to be able to target different QPUs automatically depending on the problem type, involving

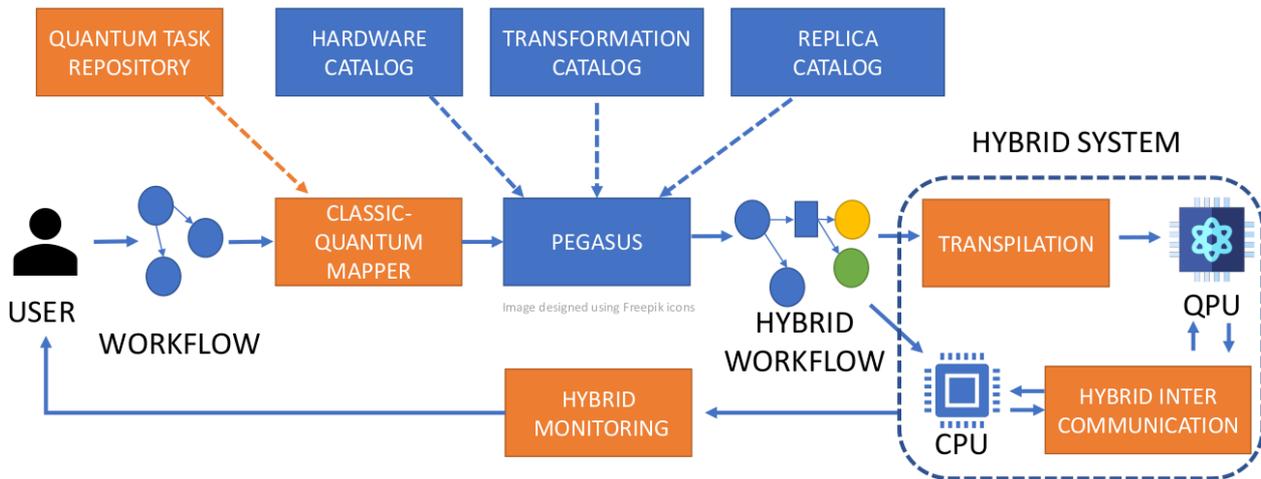


Figure 8: Quantum Workflow Management System.

minimal code modifications. Hence, a *QPU agnostic* development allows greater freedom and efficiency in hybrid ecosystems. Recently, a framework named *MQT Predictor* [91] that automates quantum device selection and according to compilation of quantum algorithms for suitable hardware, execution has been proposed. This toolkit⁹ offers a) a prediction method, based on Supervised Machine Learning, which without performing any compilation predicts the most suitable hardware specific to the application in consideration. b) A reinforcement learning (RL) based method producing device-specific quantum circuit compilers. The compilation passes from several compiler tools are combined by learning (trained) optimized sequences of those passes (following a mix-and-match compiler pass) with respect to a customizable fidelity.

7.2. Performance Models

Performance models should be able to predict different performance metrics (i.e., running time, error rate) taking as input (1) a quantum task, and (2) a target quantum hardware where the input task should be executed. Such performance models are also required by WMS schedulers to decide where to allocate input tasks. Performance models should be based on values of identified hardware descriptors to facilitate the integration of different architectures. The main challenges in designing performance models lie in the difference between quantum architectures, which makes it difficult to design a generalized model that can be applied to different hardware. Also, since quantum error varies with time, depending on different environmental factors [19], performance models should be automatically updated when a change in hardware descriptors is detected.

Existing performance models have been designed for different quantum hardware [105], targeting specific applications, without considering applications' performance with relation to values of typical hardware descriptors.

7.3. Optimization of Hybrid Workflows

Optimization of hybrid workflows embraces different phases of workflow execution, depending on the type of quantum tasks: for circuits, optimization focuses on adaptation of circuits to the underlying qubit topology, or to perform gate-level optimization on the circuit (i.e., removing redundant parts). Also, on the classic side, data encoding has to be optimized for the underlying classic architecture. Concerning variational quantum algorithms, as shown by [44, 38], optimization requires setting different hyperparameters, both on classic (i.e., optimizer parameters, cost function) and on the quantum side (i.e., circuit structure). Optimizations should be applied at the time of workflow execution.

Since data encoding methods require different algebraic operations on input data [106], optimization could include typical operations used in HPC to improve mathematical computations, or even the use of specific hardware (i.e., GPUs, FPGAs) that are known to perform well for these operations. For optimization of circuits according to the underlying hardware, deep-learning-based approaches [107] or structural optimization approaches [108, 109] can be considered. Finally, for variational quantum algorithms, hyperparameter optimization approaches can be applied, as discussed by [38].

7.4. Error Mitigation

Current NISQ architectures are subject to noise, due to environmental factors and technological limitations [19]. Considering the effect of noise on the output of quantum algorithms, the typical approach in hybrid systems is to perform error mitigation on classic hardware. Different approaches are available, either ML-based [110, 111] or based on error correction codes [112]. In both cases, designing a model for error correction requires deep knowledge of the target hardware, and collecting different error metrics. Once error metrics have been identified, since error is constantly varying over time, data about such metrics have to be constantly collected in order to timely update the error model. Finally, the model should be updated timely, as soon as error goes above a given threshold.

⁹<https://github.com/cda-tum/mqt-predictor>

To address these challenges, first of all, different methods to constantly monitor quantum hardware-induced errors and readout error mitigation strategies [113] must be implemented. Moreover readout Error mitigation strategies for quantum workflows ha Then, techniques based on reinforcement learning, such as described in [114, 115]. These methods can be also improved by applying FPGAs [112] or Edge AI methods, such as [116]. Updates of the model could be triggered by staleness control methods, similar to [117].

7.5. Integration within WMSs

The final challenge is to integrate quantum machines into the HPC continuum. At the moment, different frameworks are available to manage quantum hardware, (i.e., Qiskit, BraKet, PennyLane). In order to fully exploit available quantum machines, a WMS should be capable of communicating with different frameworks, integrating all functionalities that are required for the execution of hybrid workflows (i.e., data encoding, transpilation, error mitigation). Each of these functionalities should be integrated with the HPC infrastructure without affecting performance of workflow execution.

8. Conclusion and Outlook

In this work, we describe the main components that would be needed for the execution of scientific workflows in hybrid ecosystems. Our investigation starts from the study of an MD use case, where we describe a pipeline that starts with the identification of the quantum candidates and the equivalent quantum tasks. Based on this analysis, we describe a possible architecture for a hybrid WMS, identifying software components that would facilitate the integration of classic and quantum architectures. Finally, we identify the challenges that need to be addressed for the execution of hybrid workflows.

In the future, we plan to further investigate the integration of quantum architectures in HPC, considering different types of architectures (i.e., photonic, annealers) and different programming models (i.e., measurement-based instead of gate-based). Also, we plan to extend this study on different scientific use cases, including the development of performant software stacks.

Acknowledgements

This work is partially funded through the projects "Runtime Control in Multi Clouds" (Rucon), Austrian Science Fund (FWF): Y904-N31 START-Programm 2015; Standalone Project "Transprecise Edge Computing" (Triton), Austrian Science Fund (FWF): P 36870-N; Trustworthy and Sustainable Code Offloading (Themis), Austrian Science Fund (FWF): PAT1668223; and by the Flagship Project "High-Performance Integrated Quantum Computing" (HPQC) # 897481 Austrian Research Promotion Agency (FFG) funded by the European Union – NextGenerationEU. Ewa Deelman's work was funded by the U.S. National Science Foundation under award # 2331153.

References

- [1] E. Deelman, K. Vahi, M. Rynge, R. Mayani, R. F. da Silva, G. Papadimitriou, M. Livny, The evolution of the pegasus workflow management software, *Computing in Science & Engineering* 21 (2019) 22–36. URL: <https://doi.org/10.1109/MCSE.2019.2919690>. doi:10.1109/MCSE.2019.2919690.
- [2] M. Wiczołek, R. Prodan, T. Fahringer, Scheduling of scientific workflows in the askalon grid environment, *Acm Sigmod Record* 34 (2005).
- [3] S. Haines, Workflow orchestration with apache airflow, in: *Modern Data Engineering with Apache Spark: A Hands-On Guide for Building Mission-Critical Streaming Applications*, Springer, 2022, pp. 255–295.
- [4] B. P. Abbott, et al. (LIGO Scientific and Virgo Collaboration), Gw170104: Observation of a 50-solar-mass binary black hole coalescence at redshift 0.2, *Phys. Rev. Lett.* 118 (2017) 221101. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.118.221101>. doi:10.1103/PhysRevLett.118.221101.
- [5] I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields, et al., *Workflows for e-Science: scientific workflows for grids*, volume 1, Springer, 2007.
- [6] Q. Vanhaelen, P. Mamoshina, A. M. Aliper, A. Artemov, K. Lezhnina, I. Ozerov, I. Labat, A. Zhavoronkov, Design of efficient computational workflows for in silico drug repurposing, *Drug Discovery Today* 22 (2017) 210–222.
- [7] H. S. Stein, J. M. Gregoire, Progress and prospects for accelerating materials science with automated and autonomous workflows, *Chemical science* 10 (2019) 9640–9649.
- [8] J. Ozik, J. M. Wozniak, N. Collier, C. M. Macal, M. Binois, A population data-driven workflow for covid-19 modeling and learning, *The International Journal of High Performance Computing Applications* 35 (2021) 483–499.
- [9] B. P. Abbott, R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, V. B. Adya, et al., Gw170104: Observation of a 50-solar-mass binary black hole coalescence at redshift 0.2, *Physical Review Letters* 118 (2017). URL: <http://dx.doi.org/10.1103/PhysRevLett.118.221101>. doi:10.1103/physrevlett.118.221101.
- [10] J. Shalf, The future of computing beyond moore's law, *Philosophical Transactions of the Royal Society A* 378 (2020) 20190061.
- [11] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina, et al., The opportunities and challenges of exascale computing, Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee (2010) 1–77.
- [12] J. A. Ang, D. J. Mountain, New horizons for high-performance computing, *Computer* 55 (2022) 156–162.
- [13] Y. A. Liu, X. L. Liu, F. N. Li, H. Fu, Y. Yang, J. Song, P. Zhao, Z. Wang, D. Peng, H. Chen, C. Guo, H. Huang, W. Wu, D. Chen, Closing the "quantum supremacy" gap: Achieving real-time simulation of a random quantum circuit using a new sunway supercomputer, in: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '21*, Association for Computing Machinery, New York, NY, USA, 2021. URL: <https://doi.org/10.1145/3458817.3487399>. doi:10.1145/3458817.3487399.
- [14] P. Givi, A. J. Daley, D. Mavriplis, M. Malik, Quantum speedup for aerospace and engineering, *AIAA Journal* 58 (2020) 3715–3727.
- [15] Y. Liu, S. Arunachalam, K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, *Nature Physics* 17 (2021) 1013–1017.
- [16] W.-L. Chang, J.-C. Chen, W.-Y. Chung, C.-Y. Hsiao, R. Wong, A. V. Vasilakos, Quantum speedup and mathematical solutions of implementing bio-molecular solutions for the independent set problem on ibm quantum computers, *IEEE Transactions on NanoBioscience* 20 (2021) 354–376.
- [17] S. Aaronson, Read the fine print, *Nature Physics* 11 (2015) 291–293.
- [18] B. Cheng, X.-H. Deng, X. Gu, Y. He, G. Hu, P. Huang, J. Li, B.-C. Lin, D. Lu, Y. Lu, C. Qiu, H. Wang, T. Xin, S. Yu, M.-H. Yung, J. Zeng, S. Zhang, Y. Zhong, X. Peng, F. Nori, D. Yu, Noisy intermediate-scale quantum computers, *Frontiers of Physics* 18 (2023) 21308.
- [19] J. Preskill, Quantum Computing in the NISQ era and beyond, *Quantum* 2 (2018) 79. URL: <https://doi.org/10.22331/q-2018-08-06-79>. doi:10.22331/q-2018-08-06-79.

- [20] P. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134. doi:10.1109/SFCS.1994.365700.
- [21] C. Gidney, M. Ekerå, How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, *Quantum* 5 (2021) 433. URL: <https://doi.org/10.22331/q-2021-04-15-433>. doi:10.22331/q-2021-04-15-433.
- [22] W. Cai, X. Mu, W. Wang, J. Zhou, Y. Ma, X. Pan, Z. Hua, X. Liu, G. Xue, H. Yu, H. Wang, Y. Song, C.-L. Zou, L. Sun, Protecting entanglement between logical qubits via quantum error correction, *Nature Physics* (2024).
- [23] S. A. Stein, R. L'Abbate, W. Mu, Y. Liu, B. Baheri, Y. Mao, G. Qiang, A. Li, B. Fang, A hybrid system for learning classical data in quantum states, in: *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, IEEE, 2021, pp. 1–7.
- [24] D. Marković, J. Grollier, Quantum neuromorphic computing, *Applied Physics Letters* 117 (2020) 150501. URL: <https://doi.org/10.1063/5.0020014>. doi:10.1063/5.0020014.
- [25] A. Keesling, A. Omran, H. Levine, H. Bernien, H. Pichler, S. Choi, R. Samajdar, S. Schwartz, P. Silvi, S. Sachdev, P. Zoller, M. Endres, M. Greiner, V. Vuletić, M. D. Lukin, Quantum Kibble-Zurek mechanism and critical dynamics on a programmable rydberg simulator, *Nature* 568 (2019) 207–211.
- [26] S. Ebadī, T. T. Wang, H. Levine, A. Keesling, G. Semeghini, A. Omran, D. Bluvstein, R. Samajdar, H. Pichler, W. W. Ho, S. Choi, S. Sachdev, M. Greiner, V. Vuletić, M. D. Lukin, Quantum phases of matter on a 256-atom programmable quantum simulator, *Nature* 595 (2021) 227–232. URL: <https://doi.org/10.1038/s41586-021-03582-4>. doi:10.1038/s41586-021-03582-4.
- [27] 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, *Nature Reviews Physics* 3 (2021) 625–644.
- [28] G. A. Quantum, Collaborators*†, F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, S. Boixo, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. Dunsworth, E. Farhi, A. Fowler, B. Foxen, C. Gidney, M. Giustina, R. Graff, S. Habegger, M. P. Harrigan, A. Ho, S. Hong, T. Huang, W. J. Huggins, L. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, C. Jones, D. Kafri, K. Kechedzhi, J. Kelly, S. Kim, P. V. Klimov, A. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, M. Lindmark, E. Lucero, O. Martin, J. M. Martinis, J. R. McClean, M. McEwen, A. Megrant, X. Mi, M. Mohseni, W. Mruczkiewicz, J. Mutus, O. Naaman, M. Neeley, C. Neill, H. Neven, M. Y. Niu, T. E. O'Brien, E. Ostby, A. Petukhov, H. Putterman, C. Quintana, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, D. Strain, K. J. Sung, M. Szalay, T. Y. Takeshita, A. Vainsencher, T. White, N. Wiebe, Z. J. Yao, P. Yeh, A. Zalcman, Hartree-fock on a superconducting qubit quantum computer, *Science* 369 (2020) 1084–1089. URL: <https://doi.org/10.1126/science.abb9811>. doi:10.1126/science.abb9811.
- [29] L. W. Anderson, M. Kiffner, P. K. Barkoutsos, I. Tavernelli, J. Crain, D. Jaksch, Coarse-grained intermolecular interactions on quantum processors, *Phys. Rev. A* 105 (2022) 062409. URL: <https://link.aps.org/doi/10.1103/PhysRevA.105.062409>. doi:10.1103/PhysRevA.105.062409.
- [30] R. P. Feynman, Simulating physics with computers, *International Journal of Theoretical Physics* 21 (1982) 467–488.
- [31] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, J. M. Martinis, Quantum supremacy using a programmable superconducting processor, *Nature* 574 (2019) 505–510. URL: <https://doi.org/10.1038/s41586-019-1666-5>. doi:10.1038/s41586-019-1666-5.
- [32] F. Leymann, J. Barzen, The bitter truth about gate-based quantum algorithms in the nisq era, *Quantum Science and Technology* 5 (2020) 044007. URL: <https://dx.doi.org/10.1088/2058-9565/abae7d>. doi:10.1088/2058-9565/abae7d.
- [33] D. Kielpinski, C. Monroe, D. J. Wineland, Architecture for a large-scale ion-trap quantum computer, *Nature* 417 (2002) 709–711. URL: <https://doi.org/10.1038/nature00784>. doi:10.1038/nature00784.
- [34] D. Willsch, M. Willsch, C. D. Gonzalez Calaza, F. Jin, H. De Raedt, M. Svensson, K. Michielsen, Benchmarking advantage and d-wave 2000q quantum annealers with exact cover problems, *Quantum Information Processing* 21 (2022) 141. URL: <https://doi.org/10.1007/s11128-022-03476-y>. doi:10.1007/s11128-022-03476-y.
- [35] B. Weder, J. Barzen, M. Beisel, F. Leymann, Analysis and rewrite of quantum workflows: Improving the execution of hybrid quantum algorithms., in: *CLOSER*, 2022, pp. 38–50.
- [36] D. Vietz, J. Barzen, F. Leymann, B. Weder, Splitting quantum-classical scripts for the generation of quantum workflows, in: *International Conference on Enterprise Design, Operations, and Computing*, Springer, 2022, pp. 255–270.
- [37] Y. Cao, J. Romero, A. Aspuru-Guzik, Potential of quantum computing for drug discovery, *IBM Journal of Research and Development* 62 (2018) 6–1.
- [38] S. S. Cranganore, V. D. Maio, I. Brandic, T. M. A. Do, E. Deelman, Molecular dynamics workflow decomposition for hybrid classic/quantum systems, in: *18th IEEE International Conference on e-Science, e-Science 2022*, Salt Lake City, UT, USA, October 11-14, 2022, IEEE, 2022, pp. 346–356.
- [39] R. Orús, S. Mugel, E. Lizaso, Quantum computing for finance: Overview and prospects, *Reviews in Physics* 4 (2019) 100028.
- [40] J. Clarke, N. Thomas, J. Roberts, R. Pilliarisetty, Z. Yoscovits, R. Caudillo, H. George, K. Singh, D. Michalak, P. Amin, et al., Quantum computing within the framework of advanced semiconductor manufacturing, in: *2016 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2016, pp. 13–1.
- [41] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, *Phys. Rev. Lett.* 103 (2009) 150502. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.103.150502>. doi:10.1103/PhysRevLett.103.150502.
- [42] R. Ur Rasoool, H. F. Ahmad, W. Rafique, A. Qayyum, J. Qadir, Z. Anwar, Quantum computing for healthcare: A review, *Future Internet* 15 (2023) 94.
- [43] J. R. McClean, J. Romero, R. Babbush, A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, *New Journal of Physics* 18 (2016) 023023. URL: <https://dx.doi.org/10.1088/1367-2630/18/2/023023>. doi:10.1088/1367-2630/18/2/023023.
- [44] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, et al., Variational quantum algorithms, *Nature Reviews Physics* 3 (2021) 625–644.
- [45] A. Peruzzo, J. McClean, P. Shadbolt, 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, *Nature Communications* 5 (2014) 4213.
- [46] 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, 2021. URL: <https://arxiv.org/abs/2111.05176>. doi:10.48550/ARXIV.2111.05176.
- [47] 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, *Physics Reports* 986 (2022) 1–128.
- [48] M. De Stefano, F. Pecorelli, D. Di Nucci, F. Palomba, A. De Lucia, Software engineering for quantum programming: How far are we?, *Journal of Systems and Software* 190 (2022) 111326.
- [49] B. Weder, J. Barzen, F. Leymann, D. Vietz, Quantum software development lifecycle, in: *Quantum Software Engineering*, Springer, 2022, pp. 61–83.

- [50] T. Atkinson, A. Karsa, J. Drake, J. Swan, Quantum program synthesis: Swarm algorithms and benchmarks, in: Genetic Programming: 22nd European Conference, EuroGP 2019, Held as Part of EvoStar 2019, Leipzig, Germany, April 24–26, 2019, Proceedings 22, Springer, 2019, pp. 19–34.
- [51] M. G. Davis, E. Smith, A. Tudor, K. Sen, I. Siddiqi, C. Iancu, Towards optimal topology aware quantum circuit synthesis, 2020 IEEE International Conference on Quantum Computing and Engineering (QCE) (2020) 223–234. URL: <https://api.semanticscholar.org/CorpusID:227221311>.
- [52] Pegasus workflow management system, <https://pegasus.isi.edu/>, [Online].
- [53] V. De Maio, D. Kimovski, Multi-objective scheduling of extreme data scientific workflows in fog, Future Generation Computer Systems 106 (2020) 171–184.
- [54] R. Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, E. Deelman, A characterization of workflow management systems for extreme-scale applications, Future Generation Computer Systems 75 (2017) 228–238.
- [55] Orquestra, <https://zapata.ai/EarlyAccess/>, [Online].
- [56] Covalent: A unified platform for accelerated computing, <https://www.covalent.xyz/>, [Online].
- [57] Dagshub: the home for data science collaboration, <https://dagshub.com/>, [Online].
- [58] S. Leontica, F. Tennie, T. Farrow, Simulating molecules on a cloud-based 5-qubit ibm-q universal quantum computer, Communications Physics 4 (2021) 112. URL: <https://doi.org/10.1038/s42005-021-00616-1>.
- [59] X.-D. Zhang, X.-M. Zhang, Z.-Y. Xue, Quantum hyperparallel algorithm for matrix multiplication, Scientific Reports 6 (2016) 24910. URL: <https://doi.org/10.1038/srep24910>.
- [60] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, Nature 549 (2017) 195–202. URL: <https://doi.org/10.1038/nature23474>.
- [61] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, L. Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. 91 (2019) 045002. URL: <https://link.aps.org/doi/10.1103/RevModPhys.91.045002>.
- [62] Y. Liu, S. Arunachalam, K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, Nature Physics 17 (2021) 1013–1017. URL: <https://doi.org/10.1038/s41567-021-01287-z>.
- [63] A. Zlokapa, S. Boixo, D. Lidar, Boundaries of quantum supremacy via random circuit sampling, 2020. [arXiv:2005.02464](https://arxiv.org/abs/2005.02464).
- [64] M. Weigold, J. Barzen, F. Leymann, M. Salm, Encoding patterns for quantum algorithms, IET Quantum Communication 2 (2021) 141–152. URL: <https://doi.org/10.1049/qtc2.12032>.
- [65] R. LaRose, B. Coyle, Robust data encodings for quantum classifiers, Phys. Rev. A 102 (2020) 032420. URL: <https://link.aps.org/doi/10.1103/PhysRevA.102.032420>.
- [66] A. Callison, N. Chancellor, Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond, Phys. Rev. A 106 (2022) 010101. URL: <https://link.aps.org/doi/10.1103/PhysRevA.106.010101>.
- [67] K. Shashidhar, M. Bruynooghe, F. Catthoor, G. Janssens, Functional equivalence checking for verification of algebraic transformations on array-intensive source code, in: Design, Automation and Test in Europe, 2005, pp. 1310–1315 Vol. 2. doi:10.1109/DATE.2005.163.
- [68] M. Zhang, L. Dong, Y. Zeng, N. Cao, Improved circuit implementation of the hhl algorithm and its simulations on qiskit, Scientific Reports 12 (2022) 13287.
- [69] T. M. A. Do, L. Pottier, S. Caño-Lores, R. Ferreira da Silva, M. A. Cuendet, H. Weinstein, T. Estrada, M. Taufer, E. Deelman, A lightweight method for evaluating in situ workflow efficiency, Journal of Computational Science 48 (2021) 101259. URL: <https://www.sciencedirect.com/science/article/pii/S187750320305573>.
- [70] M. Taufer, S. Thomas, M. Wyatt, T. M. Anh Do, L. Pottier, R. F. da Silva, H. Weinstein, M. A. Cuendet, T. Estrada, E. Deelman, Characterizing in situ and in transit analytics of molecular dynamics simulations for next-generation supercomputers, in: 2019 15th International Conference on eScience (eScience), 2019, pp. 188–198. doi:10.1109/eScience.2019.00027.
- [71] A. Kitao, N. Go, Investigating protein dynamics in collective coordinate space, Current Opinion in Structural Biology 9 (1999) 164–169.
- [72] A. Barducci, M. Bonomi, M. Parrinello, Metadynamics, WIREs Computational Molecular Science 1 (2011) 826–843. URL: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.31>. doi:10.1002/wcms.31.
- [73] S. Wei, H. Li, G. Long, A full quantum eigensolver for quantum chemistry simulations, Research 2020 (2020). URL: <https://spj.science.org/doi/abs/10.34133/2020/1486935>. doi:10.34133/2020/1486935.
- [74] S. Lloyd, M. Mohseni, P. Rebentrost, Quantum algorithms for supervised and unsupervised machine learning, 2013. URL: <https://arxiv.org/abs/1307.0411>. doi:10.48550/ARXIV.1307.0411.
- [75] V. Giovannetti, S. Lloyd, L. Maccone, Quantum random access memory, Phys. Rev. Lett. 100 (2008) 160501. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.100.160501>. doi:10.1103/PhysRevLett.100.160501.
- [76] M. A. Nielsen, I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press, 2010. doi:10.1017/CBO9780511976667.
- [77] S. Aaronson, Bqp and the polynomial hierarchy, 2009. URL: <https://arxiv.org/abs/0910.4698>. doi:10.48550/ARXIV.0910.4698.
- [78] T. Johnston, B. Zhang, A. Liwo, S. Crivelli, M. Taufer, In situ data analytics and indexing of protein trajectories, Journal of Computational Chemistry 38 (2017) 1419–1430. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.24729>. doi:10.1002/jcc.24729.
- [79] T. Johnston, B. Zhang, A. Liwo, S. Crivelli, M. Taufer, In situ data analytics and indexing of protein trajectories, Journal of Computational Chemistry 38 (2017) 1419–1430. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.24729>. doi:10.1002/jcc.24729.
- [80] W. Ritz, Über eine neue methode zur lösung gewisser variationsprobleme der mathematischen physik., Journal für die reine und angewandte Mathematik 135 (1909) 1–61. URL: <http://eudml.org/doc/149295>.
- [81] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, R. Halavati, M. Y. Niu, A. Zlokapa, E. Peters, O. Lockwood, A. Skolik, S. Jerbi, V. Dunjko, M. Leib, M. Streif, D. V. Dollen, H. Chen, S. Cao, R. Wiersema, H.-Y. Huang, J. R. McClean, R. Babbush, S. Boixo, D. Bacon, A. K. Ho, H. Neven, M. Mohseni, Tensorflow quantum: A software framework for quantum machine learning, 2021. [arXiv:2003.02989](https://arxiv.org/abs/2003.02989).
- [82] R. Babbush, D. W. Berry, J. R. McClean, H. Neven, Quantum simulation of chemistry with sublinear scaling in basis size, npj Quantum Information 5 (2019) 92. URL: <https://doi.org/10.1038/s41534-019-0199-y>. doi:10.1038/s41534-019-0199-y.
- [83] Quantum Information and Computation 12 (2012). URL: <http://dx.doi.org/10.26421/QIC12.11-12>. doi:10.26421/qic12.11-12.
- [84] E. Farhi, J. Goldstone, S. Gutmann, A quantum approximate optimization algorithm, 2014. [arXiv:1411.4028](https://arxiv.org/abs/1411.4028).
- [85] N. Wiebe, A. Kapoor, K. M. Svore, Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning, Quantum Info. Comput. 15 (2015) 316–356.
- [86] P. Rebentrost, M. Mohseni, S. Lloyd, Quantum support vector machine for big data classification, Phys. Rev. Lett. 113 (2014) 130503. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.113.130503>. doi:10.1103/PhysRevLett.113.130503.
- [87] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature 567 (2019) 209–212.
- [88] L. Clinton, T. Cubitt, B. Flynn, F. M. Gambetta, J. Klassen, A. Montanaro, S. Piddock, R. A. Santos, E. Sheridan, Towards near-term quantum simulation of materials, Nature Communications 15 (2024) 211.
- [89] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, J. M. Gambetta, Validating quantum computers using randomized model circuits, Physi-

- cal Review A 100 (2019) 032328.
- [90] E. A. Sete, W. J. Zeng, C. T. Rigetti, A functional architecture for scalable quantum computing, in: 2016 IEEE International Conference on Rebooting Computing (ICRC), IEEE, 2016, pp. 1–6.
- [91] N. Quetschlich, L. Burgholzer, R. Wille, Mqt predictor: Automatic device selection with device-specific circuit compilation for quantum computing, 2023. [arXiv:2310.06889](https://arxiv.org/abs/2310.06889).
- [92] M. Salm, J. Barzen, U. Breitenbücher, F. Leymann, B. Weder, K. Wild, The NISQ Analyzer: Automating the Selection of Quantum Computers for Quantum Algorithms, in: Proceedings of the 14th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2020), Springer International Publishing, 2020, pp. 66–85. doi:10.1007/978-3-030-64846-6_5.
- [93] J. Garcia-Alonso, J. Rojo, D. Valencia, E. Moguel, J. Berrocal, J. M. Murillo, Quantum software as a service through a quantum api gateway, IEEE Internet Computing 26 (2022) 34–41. doi:10.1109/MIC.2021.3132688.
- [94] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, M. Van den Nest, Measurement-based quantum computation, Nature Physics 5 (2009) 19–26.
- [95] H. Zhang, A. Wu, Y. Wang, G. Li, H. Shapourian, A. Shabani, Y. Ding, Oneq: A compilation framework for photonic one-way quantum computation, in: Proceedings of the 50th Annual International Symposium on Computer Architecture, 2023, pp. 1–14.
- [96] W. Hasselbring, M. Wojcieszak, S. Dustdar, Control flow versus data flow in distributed systems integration: Revival of flow-based programming for the industrial internet of things, IEEE Internet Computing 25 (2021) 5–12. doi:10.1109/MIC.2021.3053712.
- [97] K. A. Britt, T. S. Humble, High-performance computing with quantum processing units, J. Emerg. Technol. Comput. Syst. 13 (2017). URL: <https://doi.org/10.1145/3007651>. doi:10.1145/3007651.
- [98] J. van de Wetering, Zx-calculus for the working quantum computer scientist, arXiv preprint arXiv:2012.13966 (2020).
- [99] H. Ohashi, Y. Watanobe, Convolutional neural network for classification of source codes, in: 2019 IEEE 13th international symposium on embedded multicore/many-core systems-on-chip (MCSoc), IEEE, 2019, pp. 194–200.
- [100] Y. Chen, Overview of research on code annotation evolution and classification, in: ICETIS 2022; 7th International Conference on Electronic Technology and Information Science, VDE, 2022, pp. 1–4.
- [101] F. Zilk, K. Staudacher, T. Guggemos, K. Füllinger, D. Kranzlmüller, P. Walther, A compiler for universal photonic quantum computers, in: 2022 IEEE/ACM Third International Workshop on Quantum Computing Software (QCS), IEEE, 2022, pp. 57–67.
- [102] X. Zhang, T. Wang, Elastic and reliable bandwidth reservation based on distributed traffic monitoring and control, IEEE Transactions on Parallel and Distributed Systems 33 (2022) 4563–4580.
- [103] S. Sanchez, A. Bonnie, G. Van Heule, C. Robinson, A. DeConinck, K. Kelly, Q. Snead, J. Brandt, Design and implementation of a scalable hpc monitoring system, in: 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2016, pp. 1721–1725.
- [104] K. Jałowiecki, P. Lewandowska, Łukasz Paweła, Pyqbench: a python library for benchmarking gate-based quantum computers, 2023. [arXiv:2304.00045](https://arxiv.org/abs/2304.00045).
- [105] T. Lubinski, S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaie, C. H. Baldwin, K. Mayer, T. Proctor, Application-oriented performance benchmarks for quantum computing, IEEE Transactions on Quantum Engineering 4 (2023) 1–32. doi:10.1109/TQE.2023.3253761.
- [106] M. Schuld, F. Petruccione, Supervised Learning with Quantum Computers, 1st ed., Springer Publishing Company, Incorporated, 2018.
- [107] T. Fösel, M. Y. Niu, F. Marquardt, L. Li, Quantum circuit optimization with deep reinforcement learning, 2021. [arXiv:2103.07585](https://arxiv.org/abs/2103.07585).
- [108] J.-H. Bae, P. M. Alsing, D. Ahn, W. A. Miller, Quantum circuit optimization using quantum karnaugh map, Scientific reports 10 (2020) 15651.
- [109] M. Ostaszewski, E. Grant, M. Benedetti, Structure optimization for parameterized quantum circuits, Quantum 5 (2021) 391. URL: <https://doi.org/10.22331/q-2021-01-28-391>. doi:10.22331/q-2021-01-28-391.
- [110] L. Domingo, G. Carlo, F. Borondo, Taking advantage of noise in quantum reservoir computing, Scientific Reports 13 (2023) 8790.
- [111] D. F. Locher, L. Cardarelli, M. Müller, Quantum Error Correction with Quantum Autoencoders, Quantum 7 (2023) 942. URL: <https://doi.org/10.22331/q-2023-03-09-942>. doi:10.22331/q-2023-03-09-942.
- [112] V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. Brock, A. Ding, L. Frunzio, et al., Real-time quantum error correction beyond break-even, Nature 616 (2023) 50–55.
- [113] M. Beisel, J. Barzen, F. Leymann, F. Truger, B. Weder, V. Yussupov, Configurable readout error mitigation in quantum workflows, Electronics 11 (2022). URL: <https://www.mdpi.com/2079-9292/11/19/2983>. doi:10.3390/electronics11192983.
- [114] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017. [arXiv:1707.06347](https://arxiv.org/abs/1707.06347).
- [115] S. Guadarrama, A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman, K. Wang, E. Gonina, N. Wu, E. Kokkiopoulou, L. Sbaiz, J. Smith, G. Bartók, J. Berent, C. Harris, V. Vanhoucke, E. Brevido, TF-Agents: A library for reinforcement learning in tensorflow, <https://github.com/tensorflow/agents>, 2018. URL: <https://github.com/tensorflow/agents>, [Online; accessed 16-Aug-2023].
- [116] V. D. Maio, A. Aral, I. Brandic, A roadmap to post-moore era for distributed systems, in: C. Georgiou, E. M. Schiller, A. Ali-Eldin, A. Iosup (Eds.), ApPLIED '22: Proceedings of the 2022 Workshop on Advanced tools, programming languages, and PPlatforms for Implementing and Evaluating algorithms for Distributed systems, Salerno, Italy, 25 July 2022, ACM, 2022, pp. 30–34.
- [117] A. Aral, M. Erol-Kantarci, I. Brandic, Staleness control for edge data analytics, Proc. ACM Meas. Anal. Comput. Syst. 4 (2020) 38:1–38:24. URL: <https://doi.org/10.1145/3392156>. doi:10.1145/3392156.
- [118] J. R. Johansson, P. D. Nation, F. Nori, QuTiP: An open-source python framework for the dynamics of open quantum systems, Computer Physics Communications 183 (2012) 1760–1772.
- [119] M. Huber, J. I. de Vicente, Structure of multidimensional entanglement in multipartite systems, Phys. Rev. Lett. 110 (2013) 030501. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.110.030501>. doi:10.1103/PhysRevLett.110.030501.
- [120] J. Bub, QUANTUM INFORMATION AND COMPUTATION, Handbook of the Philosophy of Science, North-Holland, Amsterdam, 2007, pp. 555–660. URL: <https://www.sciencedirect.com/science/article/pii/B9780444515605500099>.
- [121] Front Matter, Academic Press, 2022, p. iii. URL: <https://www.sciencedirect.com/science/article/pii/B9780128229422010013>.
- [122] D. Grier, L. Schaeffer, The Classification of Clifford Gates over Qubits, Quantum 6 (2022) 734. URL: <https://doi.org/10.22331/q-2022-06-13-734>. doi:10.22331/q-2022-06-13-734.
- [123] J. A. Smolin, D. P. DiVincenzo, Five two-bit quantum gates are sufficient to implement the quantum fredkin gate, Phys. Rev. A 53 (1996) 2855–2856. URL: <https://link.aps.org/doi/10.1103/PhysRevA.53.2855>. doi:10.1103/PhysRevA.53.2855.
- [124] T. Monz, P. Schindler, J. T. Barreiro, M. Chwalla, D. Nigg, W. A. Coish, M. Harlander, W. Hänsel, M. Hennrich, R. Blatt, 14-qubit entanglement: Creation and coherence, Phys. Rev. Lett. 106 (2011) 130506. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.106.130506>. doi:10.1103/PhysRevLett.106.130506.
- [125] W. Dür, G. Vidal, J. I. Cirac, Three qubits can be entangled in two inequivalent ways, Phys. Rev. A 62 (2000) 062314. URL: <https://link.aps.org/doi/10.1103/PhysRevA.62.062314>. doi:10.1103/PhysRevA.62.062314.
- [126] H. Buhrman, R. Cleve, J. Watrous, R. de Wolf, Quantum fingerprinting, Phys. Rev. Lett. 87 (2001) 167902. URL: <https://link.aps.org/doi/10.1103/PhysRevLett.87.167902>. doi:10.1103/PhysRevLett.87.167902.
- [127] D. Kocprzyk, Quantum machine learning for data scientists, 2018. [arXiv:1804.10068](https://arxiv.org/abs/1804.10068).

Appendix A. Mathematical prerequisites for quantum computing

Definition Appendix A.1 (Hermitian Matrices/Operators). : Let $A \in M_{n,n}(\mathbb{C})$, i.e., A is a square matrix with complex entries a_{ij} . The matrix A is said to be Hermitian (self-adjoint) if A is invertible and the matrix elements satisfy the condition,

$$a_{ij} = \overline{a_{ji}}$$

Where \bar{a} denotes its complex conjugate. Hence, a Hermitian matrix is equivalent to its transpose complex conjugate (also represented in quantum computing as a \dagger subscript). A succinct matrix representation used in quantum computing is, $A = A^\dagger := \overline{A^T}$.

Definition Appendix A.2 (Unitary Matrices/Operators). : Let $U \in M_{n,n}(\mathbb{C})$. A Unitary matrix U satisfies the following condition,

$$U^{-1} = U^\dagger := \overline{U^T}$$

Hence, the inverse of a Unitary matrix is its transposed complex conjugate (also, known as *Hermitian conjugate*)

Tensor product: Let A be an $m \times n$ matrix and B be a $p \times q$ matrix,

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \ddots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1q} \\ \vdots & \ddots & & \vdots \\ b_{p1} & \dots & b_{pq} \end{bmatrix}.$$

The tensor product C of matrix A and B is an $mp \times nq$ dimensional matrix of the form,

$$C = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ \vdots & \ddots & & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} \equiv \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \dots & a_{12}b_{11} & \dots & a_{1n}b_{1q} \\ \vdots & & & \ddots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \dots & a_{m2}b_{11} & \dots & a_{mn}b_{pq} \end{bmatrix}.$$

Qubit tensor product state: The total degrees of freedom of a qubit composite is given by the Tensor product between qubit states. Consider a set of degrees of freedom associated with an n dimensional Hilbert space,

$$\mathbb{H}_1 = \text{span}\{|0\rangle, |1\rangle, \dots, |n-1\rangle\},$$

and another set of degrees of freedom associated with an m dimensional Hilbert space,

$$\mathbb{H}_2 = \text{span}\{|0\rangle, |1\rangle, \dots, |m-1\rangle\}.$$

Thus, all possible superposition states of these two qubit-conjoined Hilbert space is given by the tensor product,

$$\mathbb{H} = \mathbb{H}_1 \otimes \mathbb{H}_2 \quad (\text{A.1})$$

This tensor-product space is spanned by orthonormal basis vectors,

$$\{|j\rangle \otimes |k\rangle := |j, k\rangle : j = 0, 1, \dots, n-1; k = 0, 1, \dots, m-1\}$$

Thus, an arbitrary state vector $|\Phi\rangle$ in this composite Hilbert space $\mathcal{H}_1 \otimes \mathcal{H}_2$ can be expanded in its computational basis (superposition state) reads,

$$|\Phi\rangle = \sum_{j=0}^{n-1} \sum_{k=0}^{m-1} \gamma_{j,k} |j\rangle \otimes |k\rangle, \quad (\text{A.2})$$

where, the coefficients (amplitudes) $\gamma_{j,k} \in \mathbb{C}$. Using the tensor (Kronecker) product machinery, one can now formally define a quantum register. The Hilbert space of an n -qubit initialized quantum register $\mathbb{H}^{\otimes n}$ is the n -fold tensor product state of a single qubit Hilbert space $\mathbb{H} = \mathbb{C}^2$, i.e.,

$$(\mathbb{C}^2)^{\otimes n} := \underbrace{\mathbb{C}^2 \otimes \dots \otimes \mathbb{C}^2}_{n\text{-copies}} \quad (\text{A.3})$$

with the basis span of Eq.(B.6) being,

$$\mathcal{B} := \{|i_0\rangle \otimes \dots \otimes |i_n-1\rangle := |i_0, \dots, i_{n-1}\rangle : i_0, \dots, i_{n-1} \in \{0, 1\}\}$$

The tensor product basis can be regarded as column vectors for e.g., $|0\rangle \otimes \dots \otimes |0\rangle := |0, 0, \dots, 0\rangle = [1, 0, \dots, 0]^T, \dots, |1\rangle \otimes \dots \otimes |1\rangle := |1, 1, \dots, 1\rangle = [0, 0, \dots, 1]^T$

Theorem 1. Schmidt decomposition: Let $\{\mathbb{H}_1, \mathbb{H}_2, \dots, \mathbb{H}_n\}$ be Hilbert spaces of dimensions p_1, p_2, \dots, p_n respectively. Assume that $p_n \geq p_{n-1} \geq \dots \geq p_1$. For any state in this composite (multi-partite) system, i.e, $|\xi\rangle \in \mathbb{H}_1 \otimes \mathbb{H}_2 \dots \otimes \mathbb{H}_n$, there exist orthonormal states $\{|\phi_1\rangle, \dots, |\phi_{p_1}\rangle\} \subset \mathbb{H}_1, \dots, \{|\psi_1\rangle, \dots, |\psi_{p_n}\rangle\} \subset \mathbb{H}_n$ such that for real non-negative scalar coefficients $\lambda_i \in \mathbb{R}$,

$$|\xi\rangle = \sum_{i=1}^r \lambda_i |\phi_i\rangle \otimes \dots \otimes |\psi_i\rangle.$$

Where, λ_i are the Schmidt coefficients, and $|\phi_i\rangle, \dots, |\psi_i\rangle$ are the corresponding entangled states of the composite (multi-partite) system. The number of non-zero Schmidt coefficients r determines the degree of entanglement between the multiple subsystems.

Appendix B. Quantum Programming Model

Appendix B.1. Quantum information processing and quantum computation

Development of quantum software stacks, quantum compilers, and development in the domains of *quantum software engineering* requires at least a basic understanding of quantum physics and quantum information processing. This section is an introduction to the *programming model* for quantum computation, which is especially intended for computer scientists, computer software engineers, and computational scientists who intend to migrate parts of their classical software onto quantum devices.

Appendix B.2. Qubit

The basic unit of information, reflecting an on and off state of classical computers are the classical bits which can take the values 0 and 1 only. The basic building blocks of quantum computing are known as the quantum bits or in short qubits.

In the case of a single qubit the associated Hilbert space is $\mathbb{H} = \mathbb{C}^2$. Thus, a qubit is a linear combination of orthonormal (vectors) basis states (superposition), denoted in the Dirac notation as $|0\rangle$ and $|1\rangle$. Thus the state of a qubit is a vector in a two-dimensional complex vector space [76]. Thus, $|\Psi\rangle$ can be expanded in the orthonormal basis states as,

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle, \quad (\text{B.1})$$

where,

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (\text{B.2})$$

$\alpha, \beta \in \mathbb{C}$. These complex coefficients are also called probability amplitudes. Since probabilities must add to 1 (normalization condition), the inner-product of the state $|\psi\rangle$ with itself (norm squared of the vector) must be equal to 1. Hence, it is equivalent to the condition that

$$\langle \psi | \psi \rangle := \|\psi\|^2 = 1, \quad (\text{B.3})$$

where the state $\langle \psi | \in \mathbb{H}^*$ belongs to the dual vector space. This is equivalent to saying that the complex coefficients satisfy a unit norm condition, i.e.,

$$|\alpha|^2 + |\beta|^2 = 1. \quad (\text{B.4})$$

An actual measurement process determines the value of the qubit, which is obtained via the amplitude probability squared. The measurement procedure yields a *classical* result either corresponding to a 0 or 1 bit value. Thus, the measurement of $|0\rangle$ is obtained by squaring the probability amplitude of its complex coefficient, i.e. $|\alpha|^2$, whereas $|1\rangle$ is measured by computing $|\beta|^2$. From Eq.(2) it is clear that the state vector $|\psi\rangle$ is constrained on a unit sphere $\mathbf{S}^2 = \{\mathbf{x} \in \mathbb{R}^3 : \|\mathbf{x}\| = 1\}$. Thus the geometric representation of a qubit is the so-called Bloch-sphere representation (The Bloch sphere was generated using the QuTiP [118] package).

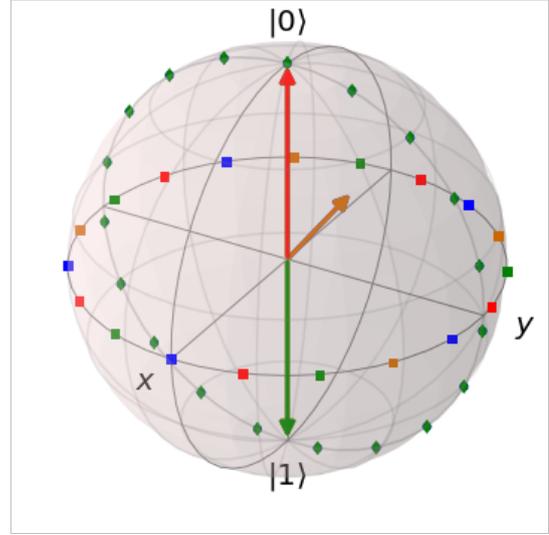


Figure B.9: Bloch sphere representation of the qubit. The red-colored vector (arrow) is the state $|0\rangle$, while the green-colored vector is state $|1\rangle$. The orange vector indicates an intermediate state and the colored points correspond to the qubit rotations (infinitely many possibilities to rotate the vector) on the Bloch sphere.

In the Bloch sphere representation, see Fig.B.9, the state vector $|\psi\rangle$ can be expressed in terms of the spherical polar coordinate basis reads,

$$|\psi\rangle = \cos \frac{\vartheta}{2} |0\rangle + e^{i\varphi} \sin \frac{\vartheta}{2} |1\rangle, \quad (\text{B.5})$$

where, $0 \leq \vartheta \leq \pi$ and $0 \leq \varphi \leq 2\pi$. Thus, the state (Bloch) vector $|\psi\rangle$ can assume any of the infinitely many orientations on the Bloch-sphere. This signifies the tremendous power of information processing using quantum, since, the qubit on the Bloch sphere can exist as an infinite coherent superposition of all the states on the unit-sphere simultaneously! This is in stark contrast to classical information processing wherein, the classical bits can only assume either of the two Boolean values 0 or 1 at a particular time instant [76].

Appendix B.3. Quantum registers

In classical computers, multiple bits are combined to form a (classical) register. In the same way, a sequence of n initialized qubits cascaded together forms the storage-device, and is called the *quantum register* or a *qubit register*. Thus, an arbitrary state vector $|\Psi\rangle$ of the composite n -qubit *quantum register* is a tensor product state living in a very huge Hilbert space $(\mathbb{C}^2)^{\otimes n}$. It allows an expansion in its computational basis (orthonormal) states as,

$$|\Psi\rangle = \alpha_{0,0,\dots,0} |0\rangle \otimes \dots \otimes |0\rangle + \alpha_{0,0,\dots,1} |0\rangle \otimes \dots \otimes |1\rangle + \alpha_{1,1,\dots,1} |1\rangle \otimes \dots \otimes |1\rangle. \quad (\text{B.6})$$

Where, the symbol \otimes corresponds to the mathematical operation of a *Tensor* product (see Appendix A). From here on, we will use a more compact notation for $|i_0\rangle \otimes |i_1\rangle \otimes \dots \otimes |i_{n-1}\rangle = |i_0, i_1, \dots, i_{n-1}\rangle$.

Since a single qubit Hilbert (state) space is $\mathbb{H} = \mathbb{C}^2$, correspondingly, an n -qubit quantum register corresponds to an n -fold tensor product of the single qubit state space, i.e.,

$$(\mathbb{C}^2)^{\otimes n} := \underbrace{\mathbb{C}^2 \otimes \dots \otimes \mathbb{C}^2}_{n\text{-copies}} \quad (\text{B.7})$$

Introducing a more succinct notation, the n -qubit quantum register, Eq.(B.6) can be written as,

$$|\Psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle \quad (\text{B.8})$$

where, i_0, i_1, \dots, i_{n-1} in $|i_0, \dots, i_{n-1}\rangle$ represent the binary notation of i . The complex coefficients, $\alpha_{i_0, \dots, i_{n-1}} \equiv \alpha_i \in \mathbb{C}$. The normalization condition,

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1,$$

Appendix B.4. Quantum Entanglement and Quantum Parallelism

This subsection delves into two pivotal phenomena of quantum physics, namely *quantum entanglement*, or just *entanglement* and *quantum parallelism*, which play fundamental roles in the fields of quantum computation quantum information processing, and quantum communication protocols [119].

- **Quantum entanglement:** In quantum computers, a pair or multiple qubits can be correlated with each other over long distances owing to the property of *quantum entanglement*. In an entangled quantum system, the state of one qubit cannot be described independently of the state of the other irrespective of their spatial separation. Thus, these states are not *separable* (no tensor product representation) and cannot be written as a *tensor product* state. The degree of entanglement between the composite systems can be quantified using the *Schmidt decomposition* (see Appendix A for definition). Manipulating the state of one of the qubits instantaneously changes the state of the other one in a predictable way. Thus, quantum entanglement enables the users to create complex quantum circuits (cf. Appendix C), wherein an operation on a single qubit instantaneously affects the state of another qubit that it is correlated with. Entanglement enables quantum computers to perform parallel computations by adding additional qubits resulting in an exponential increase in its number-crunching capabilities.
- **Quantum parallelism:** Entanglement also influences the superposition of multiple qubits by allowing them to be in a joint superposition state, leading to another intrinsic feature of quantum processors, namely, *quantum parallelism* [120, 121]. It is evident from Eq.(B.6) that a quantum register with n -qubits is in a quantum superposition of 2^n states, i.e., *all classical alternatives at once*. Hence, one can simultaneously manipulate all the 2^n possibilities that exist in this *huge* extended vector space.

Appendix B.5. Quantum Logic Gates, Quantum Circuit Model and Quantum Algorithms

Quantum (logic) gate architectures are the quantum analogs of classical logic gates. A quantum gate \hat{U} is represented as a unitary operator (matrix) is a generalization of rotation on complex vector spaces. An n -qubit quantum register The unitary operator transformations are inner-product (or norm) preserving. By definition, these unitary operator inverse is its own transposed complex conjugate, also called *Hermitian conjugate* (see Appendix A), i.e.,

$$\hat{U}\hat{U}^\dagger = \hat{U}^\dagger\hat{U} = \mathbf{1}.$$

The U^\dagger quantum gate reverses the computation by undoing the gate operator. Thus, in quantum devices, there exists another unique feature of reversible computation using the inverse Unitary operators that is not possible in classical architectures.

Appendix B.5.1. Quantum Circuit Model:

The initialized quantum register together with the qubit operation producing single or multi-qubit quantum gate sequences form the basic building blocks of a quantum circuit. There are multiple quantum gates that one uses to perform computations on the state vector. Some of the most frequently used single qubit gates in quantum information processing and computation are the so-called, PAULI gates (operators) $\{I, X, Y, Z\}$, the HADAMARD gate H and the P -gate (see Appendix B).

A *quantum algorithm* is a collection of unitary quantum gates that are assembled to successively perform one or many *unitary transformations* (computations) on a quantum register, in order to achieve a specific computational task. In short, these perform targeted rotations (operations) on a single qubit or a quantum register by mapping it onto another state on the Bloch-sphere. Thus quantum gates perform linear unitary transformations (manipulation) on an input quantum register $|R\rangle$ and map them onto an output quantum register $|Q\rangle$ as follows,

$$\hat{U}|R\rangle = |Q\rangle := \sum_{i=1}^N \alpha_{i_0, \dots, i_{N-1}} |i_0, \dots, i_{N-1}\rangle \quad (\text{B.9})$$

The result of the quantum algorithm U is obtained by measuring the quantum register $|Q\rangle$ with a probability $|\alpha_{i_0, \dots, i_{N-1}}|^2$. Due to the probabilistic nature of the *quantum processing units* (QPUs), over different executions of U followed by a measurement to determine the result yields different bit-strings according to their probabilities. This is to say that a single execution of a quantum algorithm is like performing a random experiment. Thus, an algorithm is typically executed multiple times, producing a probability density function (probability distribution) of results rather than a single value. The most probable result in this statistical sample space corresponds to the actual *result* of the quantum algorithm.

Appendix C. Unitary Quantum Logic Gates

Single Qubit Gate operations.

X gate: The Pauli-X gate is the quantum analog of the classical NOT gate. It performs a bit flip (NOT) operation $|x\rangle \mapsto |\neg x\rangle$.

The matrix representation of the X-gate reads,

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

and the quantum circuit and truth table is,



Z gate: The Pauli-Z gate is the sign Flip gate following the operation, $|x\rangle \mapsto (-1)^x |x\rangle$.

The matrix representation of the Z-gate reads,

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

with the quantum circuit and truth table being,

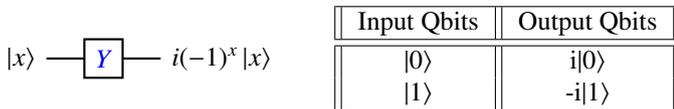


Y gate: The Pauli-Y gate performs a rotation by π around the y-axis.

The matrix representation of the Y-gate reads,

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix},$$

with the quantum circuit and truth table being,

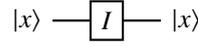


I gate: The Identity gate I leaves all states unchanged,

The matrix representation of the Identity gate reads,

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

with the quantum circuit and truth table being,



Input Qbits	Output Qbits
$ 0\rangle$	$ 0\rangle$
$ 1\rangle$	$ 1\rangle$

One of the most important and fundamental single qubit gate is the HADAMARD gate. The Hadamard gate prepares a superposition state, i.e.,

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} := |+\rangle \quad (C.1)$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} := |-\rangle \quad (C.2)$$

P gate: The phase (shift) gate P are single qubit gates that leaves the basis state $|0\rangle$ unchanged while induces a phase on state $|1\rangle$, i.e.,

$$P(\varphi)|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \quad (C.3)$$

$$P(\varphi)|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = e^{i\varphi} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = e^{i\varphi}|1\rangle \quad (C.4)$$

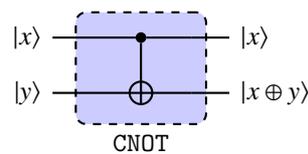
Two Qubit Gate operations.

CNOT gate: The CNOT or the Controlled-X gate is a 2-qubit gate which flips the second qubit (target qubit) if and only if the first qubit (control qubit) is state $|1\rangle$. It is the quantum analogue of the classical XOR gate and maps the state $|x, y\rangle \mapsto |x \oplus y\rangle$, where the symbol \oplus denotes the XOR logic operation.

The matrix representation of the controlled-X gate reads,

$$|0\rangle\langle 0| \mathbb{I} + |1\rangle\langle 1| X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The quantum circuit and the corresponding truth (logic) table is shown below,



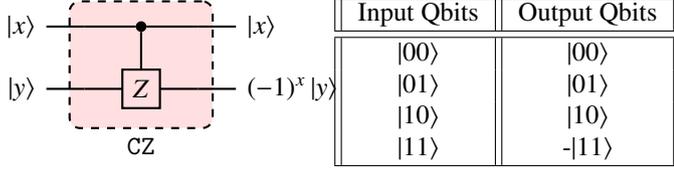
Input Qbits	Output Qbits
$ 00\rangle$	$ 00\rangle$
$ 01\rangle$	$ 01\rangle$
$ 10\rangle$	$ 11\rangle$
$ 11\rangle$	$ 10\rangle$

CZ gate: The controlled Z gate is a 2-qubit gate flips the sign of the state $|11\rangle$, while leaving the other states unaffected, i.e., $|x, y\rangle \mapsto (-1)^x |x, y\rangle$

The matrix representation of the controlled-Z reads,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix},$$

and the corresponding quantum circuit and truth table are also presented below,

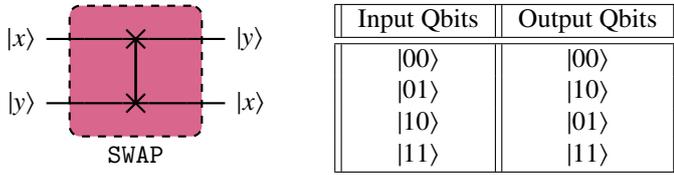


SWAP gate: The 2-qubit SWAP gate swaps the qubit states and maps a state $|a, b\rangle \mapsto |b, a\rangle$.

The permutation matrix representation reads,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (C.5)$$

while the quantum circuit representation and truth table is,



Pauli Group: Pauli matrices generate a discrete group closed under multiplication, called the Pauli group P_n . The set P_n consists of n-fold tensor product of the Pauli operators (Pauli strings) multiplied by a factor $\gamma \in \{\pm 1, \pm i\}$ accounting to 16-elements. An example of the Pauli group for $n = 2$, are the 2-fold tensor product of the Pauli gates, $\{\gamma I \otimes I, \gamma I \otimes X, \gamma I \otimes Y, \gamma I \otimes Z, \gamma X \otimes I, \gamma X \otimes X, \dots, \gamma Z \otimes Z\}$.

Definition Appendix C.1. The *normalizer* of a subgroup H of a group (or semigroup) G is defined as:

$$N_G(H) = \{g \in G | gHg^{-1} = H\}$$

Clifford gates: The Clifford group on n qubits, C_n , are the set of unitary operations that normalize the Pauli group P_n . That is, $U \in C_n$ if $UpU^\dagger \in P_n, \forall p \in P_n$. The *Clifford gates* are unitary operators in $\bigcup_{n \geq 1} C_n$. A quantum circuit constructed merely out of Clifford gates is called the Clifford circuit [122]. It performs qubit operations on some designated set of initialized qubits, while preserving the state of remaining the ancilla qubits.

The Clifford gate set consists of three gates, namely, the CNOT (controlled-NOT), the Hadamard gate H and the Phase gate P .

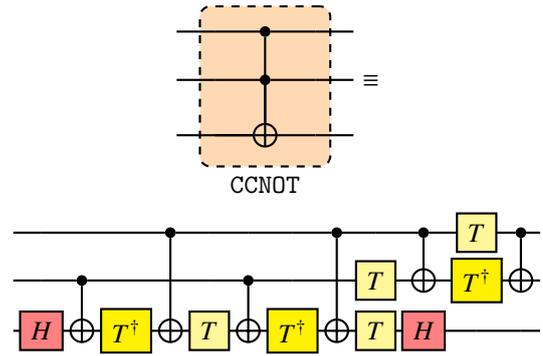
Appendix C.1. Three and Multi-qubit gate operations

TOFFOLI gate: The CCNOT (controlled-controlled NOT gate) or the TOFFOLI gate is a three qubit universal reversible quantum gate. If the first two qubits are in state $|1\rangle$, then it flips the last qubit state, i.e., $|x, y, z\rangle \mapsto |x, y, z \oplus (x \wedge y)\rangle$

The Toffoli gate in its matrix form reads,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

The three qubit Toffoli gate has an equivalent quantum circuit representation solely in terms of two qubit gates H, T and T^\dagger . Also, the truth table of this quantum logic gate are presented below,



Input Qbits	Output Qbits
000>	000>
001>	001>
010>	010>
011>	011>
100>	100>
101>	101>
110>	111>
111>	110>

FREDKIN gate: The CSWAP (controlled-swap gate) is a three qubit universal reversible quantum gate. If and only if the first qubit state is state $|1\rangle$, it leaves the the first qubit unchanged and swaps the last two bits.

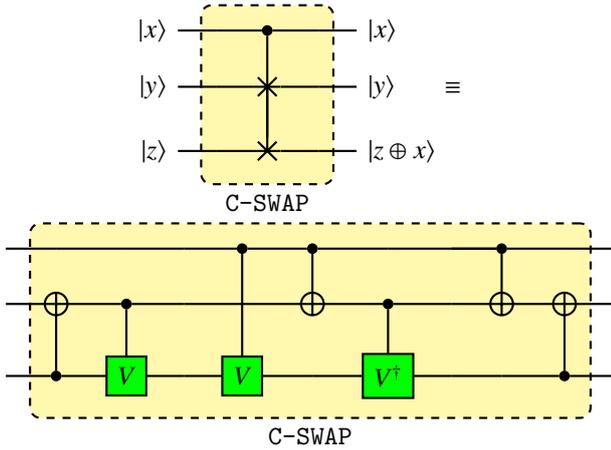
The Fredkin gate in its permutation matrix form reads,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The three qubit Fredkin gate has an equivalent quantum circuit that can be solely constructed in terms of CNOT, V & V^\dagger qubit gates [123]. Here,

$$V = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}^{\frac{1}{2}},$$

is the square-root of the Pauli X gate. Infact, the FREDKIN gate is just the TOFFOLI gate with two CNOTs on its either sides. The corresponding truth table for the FREDKIN gate is presented below,



Input Qbits	Output Qbits
$ 000\rangle$	$ 000\rangle$
$ 001\rangle$	$ 001\rangle$
$ 010\rangle$	$ 010\rangle$
$ 011\rangle$	$ 011\rangle$
$ 100\rangle$	$ 100\rangle$
$ 101\rangle$	$ 110\rangle$
$ 110\rangle$	$ 101\rangle$
$ 111\rangle$	$ 111\rangle$

Appendix D. Engineering complex quantum circuits

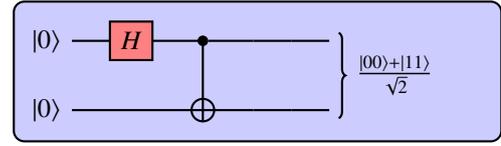
Entangled states: Separable quantum state can be expanded in its computational basis as, $|\Psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle \otimes \dots \otimes |\psi_{n-1}\rangle$. Whereas, an *Entangled* quantum state $|\zeta\rangle$ cannot be decomposed into tensor product states, i.e., $|\zeta\rangle \neq |\xi_0\rangle \otimes |\xi_1\rangle \otimes \dots \otimes |\xi_{n-1}\rangle$. The most simple and maximally entangled quantum

states can be achieved by entangling 2-qubits in 4 different manners, also known as the Bell states or EPR (*Einstein-Podolski-Rosen*) states,

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, |\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}},$$

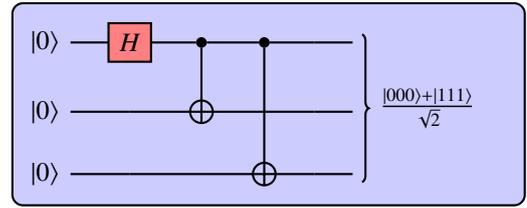
$$|\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, |\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}},$$

We demonstrate the preparation of the $|\Phi^+\rangle$ state via a quantum circuit.



In the 3-qubit case, there exists non bi-separable classes of entangled states in quantum computing are for e.g., the 3-qubit Greenberger-Horne-Zeilinger (GHZ) state [124],

$$|\mathbf{GHZ}\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$



another highly important entangled 3-qubit state that is inequivalent to the GHZ state is the W state [125],

$$|\mathbf{W}\rangle = \frac{1}{\sqrt{3}}(|001\rangle + |010\rangle + |100\rangle)$$

Appendix E. Matrix Size

The symmetric bipartite matrix B_{IJ} can be partitioned into a block matrix form.

$$B_{IJ} = \begin{pmatrix} 0^{n \times n} & E_{IJ} \\ E_{IJ}^T & 0^{n \times n} \end{pmatrix} \quad (\text{E.1})$$

The diagonal entries of the bipartite block matrix contains the zero matrix. The off-diagonal entries E_{IJ} ($n \times n$ matrix) and its transpose E_{IJ}^T contain as entries the Euclidean distances d_{ij} as defined in Eq.(??). The Euclidean distance (metric) is a function defined on vector space \mathbb{V} ,

$$d : \mathbb{V} \times \mathbb{V} \mapsto \mathbb{R}.$$

Therefore, the block matrix E_{IJ} takes values only over the field \mathbb{R} . The matrix representation is given by,

$$E_{IJ} = \begin{pmatrix} d_{ij}^{11} & \dots & d_{ij}^{1n} \\ \vdots & \ddots & \vdots \\ d_{ij}^{n1} & \dots & d_{ij}^{nn} \end{pmatrix}. \quad (\text{E.2})$$

For given two segments I and J , the Euclidean metric between C_α carbon atoms i and j reads,

$$d_{ij} = d(i, j) = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2 + (i_z - j_z)^2}. \quad (\text{E.3})$$

Theorem 2. Input bipartite distance matrix B_{IJ} is a **Hermitian** matrix.

Proof. Let $B_{IJ} \in \mathbb{R}^{2n} \times \mathbb{R}^{2n}$, with $n \in \mathbb{N}$. Note that from Equation E.1,

$$B_{IJ}^\dagger = \begin{pmatrix} 0^{n \times n} & \bar{E}_{IJ} \\ \bar{E}_{IJ}^T & 0^{n \times n} \end{pmatrix}^T = \begin{pmatrix} 0^{n \times n} & E_{IJ} \\ E_{IJ}^T & 0^{n \times n} \end{pmatrix}^T = B_{IJ}$$

Hence, B_{IJ} is a **real symmetric matrix**. Since, every real symmetric matrix is an Hermitian matrix (operator), B_{IJ} is an Hermitian operator. \square

Thus, the set of eigenvalues Λ of the bipartite distance matrix all belong in \mathbb{R} . Thus, the Hermitian bipartite matrix \hat{B}_{IJ} qubit encoding (via mapping on to Pauli operators) without any further manipulation.

Appendix F. C-SWAP test methodology for C_α atom distance estimation

First introduced in the context of quantum fingerprinting [126]. The procedure for engineering the SWAP test circuit involves, (i) Entangling qubit registers consisting of the mapped classical data with an ancillary/helper qubit $|0\rangle$ and, (ii) estimating the inner product between two different states through repeated measurements of the ancillary qubit. Thus, this quantum algorithm measures the so-called *fidelity*, which is nothing but the inner product or overlap between two different quantum states. The fidelity F between two normalized quantum states $|\phi\rangle, |\psi\rangle$ is mathematically expressed as, $F(\phi, \psi) = |\langle \phi | \psi \rangle|^2 \in [0, 1]$.

Hence, the higher the fidelity, the closer are the quantum states to each other, i.e, $F(\phi, \psi) = 1$, meaning the quantum states are parallel while $F(\phi, \psi) = 0$ meaning orthogonal quantum states.

From an application point of view, this quantum routine has been integrated into larger quantum circuits to achieve the target tasks. For example, it has been used as a primary engine for speeding-up matrix multiplications boiling down to merely $O(N^2)$ [59] time-complexity. Moreover it has been extensively used in the domains of *quantum machine learning* and *big-data* analysis (also sometimes called *quantum big data*) due to its exponential speedup offered in calculating distances between huge amounts of vector (tensor) datasets [74, 85, 127].

Circuit Description. We initialize an ancillary (helper) qubit $|0\rangle$, two quantum registers $|\psi\rangle$ and $|\phi\rangle$. The initial state of combined tensor product is

$$|\Psi_A\rangle = |0\rangle \otimes |\phi\rangle \otimes |\psi\rangle, \quad (\text{B.1})$$

Encoded into the states $|\phi\rangle$ and $|\psi\rangle$ are classical data atom coordinates using the AMPLITUDE encoding method. In *amplitude encoding*, a classical vector or tensor, $X = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^N$ is mapped onto the quantum device by implementing the following algorithm written as a pseudocode:

Algorithm 1 Amplitude Encoding Schema

Input: Classical data, $X = [x_0, x_1, \dots, x_{p-1}]^T \in \mathbb{R}^p$, number of qubits n .

Result: Quantum data with coefficients of $X \in \mathbb{R}^p$ encoded as amplitudes of the state-vector $X \mapsto |Q_X\rangle = \frac{1}{\|x\|^2} \sum_{i=0}^{n-1} x_i |i\rangle$

- 1: $p \leftarrow \text{LEN}(X)$
 - 2: **if** $\lceil \log_2(n) \rceil - p = 0$ **then**
 - 3: $Q_X \leftarrow X$ {Calculate magnitude (norm) squared of X }
 - 4: $Q_X \mapsto |Q_X\rangle = \frac{1}{\|x\|^2} \sum_{i=0}^{n-1} x_i |i\rangle$
 - 5: **end if**
 - 6: **if** $\lceil \log_2(n) \rceil - p = (k-1)$ **then**
 - 7: $Q_X \leftarrow \left[\underbrace{x_0, x_1, \dots, x_{p-1}}_{p\text{-entries}}, \underbrace{0, \dots, 0}_{(k-1)\text{-entries}} \right]^T$ {Pad the vector X with $(k-1)$ 0's to convert to dimension n .}
 - 8: $Q_X \mapsto |Q_X\rangle = \frac{1}{\|x\|^2} (\sum_{i=0}^{p-1} x_i |i\rangle + \sum_{i=p}^{n-1} 0 |i\rangle)$ {Calculate magnitude (norm) squared of X }
 - 9: **end if**
 - 10: **return** $|Q_X\rangle$
-

Thus, an n -dimensional classical data vector (tensor) can be efficiently encoded into the wavefunction (state vector), requiring merely $\lceil \log_2(n) \rceil$ qubits [106]. Here, we describe the methodology to adapt the SWAP test quantum circuit [126, 106, 127] for C_α atoms distance matrix calculations.

Appendix F.0.1. Problem Size and Qubit Mapping

Input matrix size: While on classic architectures the amount of atoms and segments we can process is dependent on the

amount of RAM available in the system, in the quantum machines we are limited by the amount of qubits of the machine. As per the limitation of the target machines, we limit our input size, i.e. the length of amino acid segments used to generate the distance or bipartite matrix. For a chosen segment of length k (consisting of k atoms), the Euclidean distance matrix D (within the same segment) is a $k \times k$ symmetric matrix with diagonal entries as zeros. The constructed bipartite distance matrices B_{IJ} (between two separate segments) is a $2k \times 2k$ dimensional matrix with a $k \times k$ block matrices E_{IJ} (cf. Appendix E).

Exploiting symmetries of the input CVs leads to a significant dimensional reduction on the matrix sizes, making it feasible to encode smaller input sizes onto quantum devices. In our case, it suffices to calculate only $\frac{k(k+1)}{2}$ unique entries of D and k^2 entries for the block matrix of E_{IJ} of the $2k \times 2k$ sized B_{IJ} (since the other block is just the transpose of the matrix cf. Appendix E) respectively. Due to such dimensional reduction properties intrinsic to our MD system, it becomes viable to cut-down the input system size for our Target Task I and Target Task II on the NISQ hardware.

The *quantum state preparation* for the C_α atom coordinates done via pseudo code 1 is depicted in the *Data Encoding* block in Figure F.10.

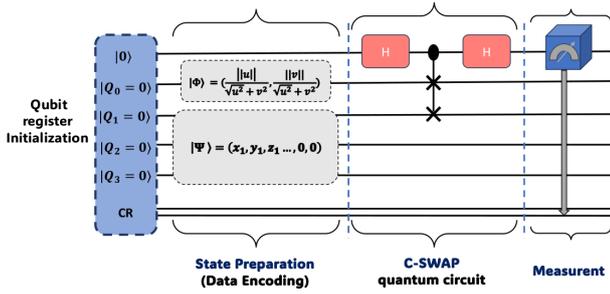


Figure F.10: C-SWAP quantum subroutine implementation for distance matrix generation.

Let the position vectors (classical data) corresponding to two different atoms be denoted as \vec{u}, \vec{v} . Then the qubits can be initialized with the state amplitudes (coefficients), in such a way that:

$$|\phi\rangle = \frac{1}{\sqrt{W}} (||\vec{u}|| |0\rangle - ||\vec{v}|| |1\rangle), \quad (\text{B.2a})$$

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|u, 0\rangle + |v, 1\rangle), \quad (\text{B.2b})$$

with $||\vec{u}||, ||\vec{v}||$ being the Euclidean norm of the coordinates and $W = ||\vec{u}||^2 + ||\vec{v}||^2$. The corresponding amplitude-encoded vectors read,

$$|u\rangle = \sum_{i=0}^{N-1} \frac{u_i}{||\vec{u}||} |i\rangle, \quad (\text{B.3a})$$

$$|v\rangle = \sum_{i=0}^{N-1} \frac{v_i}{||\vec{v}||} |i\rangle \quad (\text{B.3b})$$

Since each qubit has two possible states, the number of coordinates of the classical vector data must necessarily be 2^n [106].

The real-time coordinates of C_α atoms whose positions vectors evolves as a function of time $t, \{x_i(t), y_i(t), z_i(t)\}_{1 \leq i \leq n} \in \mathbb{R}^3$. These three-dimensional position vector \vec{u} must be padded with a 0 as the fourth coordinate. This leads to a vector of the form $(x(t), y(t), z(t), 0)$ with $2^2 = 4$ coordinates. This allows a suitable encoding scheme onto a 2-qubit quantum register.

In Figure F.10, the 3-qubit quantum register $|\psi\rangle = |Q_1, Q_2, Q_3\rangle$, initialized as $|0, 0, 0\rangle$ is then encoded with the concatenated atom-pair coordinates values¹⁰ $(x_1, y_1, z_1, x_2, y_2, z_2, 0, 0) \in \mathbb{R}^8$. Here, it is clear that 0 padding is required for embedding, since three qubits are only capable of storing a vector of dimension $2^3 = 8$.

This is followed by an application of the HADAMARD gate H (cf. Appendix B) on the ancillary qubit. Implementing controlled swap operation, is performed using the three-qubit FREDKIN gate (cf. Appendix B) on the other two registers. The ancillary qubit works like a control bit. The total state of the system after these two gate operations is,

$$|\Psi_B\rangle = \frac{1}{\sqrt{2}} (|0, \psi, \phi\rangle + |1, \phi, \psi\rangle). \quad (\text{B.4})$$

The application of another Hadamard gate on the ancillary qubit $|0\rangle$ yields

$$\frac{1}{2} |0\rangle (|\phi, \psi\rangle + |\psi, \phi\rangle) + \frac{1}{2} |1\rangle (|\phi, \psi\rangle - |\psi, \phi\rangle).$$

On applying the Hadamard gate, the probability of measuring state '0', i.e., of control qubit yields,

$$Pr(0) = \frac{1}{2} + \frac{1}{2} |\langle \phi | \psi \rangle|^2.$$

Euclidean distances $d(i, j)$ between C_α atoms can be obtained using Eqs. (B.2, B.3).

$$d(\vec{u}, \vec{v})^2 = 2W |\langle \phi | \psi \rangle|^2 = 4W (Pr(0) - 0.5). \quad (\text{B.5})$$

Our simulations comprised of a similar strategy as put-forth in [126]. Here, we perform a single execution for each C_α atom pair using the SWAP test subroutine. Hence, a total number of n repeated circuit executions were required for calculating the distances between n atom pairs using the quantum architecture.

Appendix G. The Variational Quantum Eigensolver Machinery

Appendix G.1. The mathematics of VQE

The theoretical groundwork for VQE starts with the variational Rayleigh-Ritz functional. Given a Hamiltonian (Hermitian operator) \hat{H} and an initial trial wavefunction with respect

¹⁰For the sake of brevity, we drop the time-dependence of the atom coordinates.

to some vector-valued parameter $\boldsymbol{\theta}$ is $|\Psi(\boldsymbol{\theta})\rangle$ (ansatz wavefunction). The Rayleigh-Ritz variational principle [80] sets an optimized upper bound for the ground state energy E_0 (lowest possible expectation/average value) associated with the Hamiltonian¹¹, E_0 , i.e.,

$$E_0 := \langle \hat{H} \rangle_{\boldsymbol{\theta}} \leq \frac{\langle \Psi(\boldsymbol{\theta}) | \hat{H}(\boldsymbol{\theta}) | \Psi(\boldsymbol{\theta}) \rangle}{\langle \Psi(\boldsymbol{\theta}) | \Psi(\boldsymbol{\theta}) \rangle}. \quad (\text{F.1})$$

The VQE machinery finds a parameterization of the wavefunction $|\Psi\rangle$, such that the expectation value of the Hermitian operator \hat{H} is minimized and approaches closer to the lowest eigenvalue E_0 after successive iterative optimization steps [46].

A PQC consisting of an initialized qubit register and a set of unitary quantum gates, can only perform a series of unitary transformations and measurements. In order to execute such a minimization (optimization) task as described in Eq.(F.1) using quantum circuits, the user must define a so-called ansatz wavefunction (trial eigenvector) $|\Psi(\boldsymbol{\theta})\rangle$. An initial generic parametrized unitary quantum gate $U(\boldsymbol{\theta})$ applied onto an initialized qubit register state, say $U(\boldsymbol{\theta})|0\rangle^{\otimes N} = |\Psi(\boldsymbol{\theta})\rangle$ ($\forall \boldsymbol{\theta} \in (-\pi, \pi]$) generates the ansatz wavefunction.

The Hamiltonian \hat{H} (Hermitian Matrices in general) can be encoded onto the Pauli operators multiplied by weights (linear combination of elements in the Pauli group), i.e.,

$$\hat{H} = \sum_{\alpha} w_{\alpha} \hat{P}_{\alpha}, \quad \forall \hat{P}_{\alpha} \in \mathcal{P}_n. \quad (\text{F.2})$$

Here, w_{α} are the set of weights (coefficients) and \hat{P}_{α} are Pauli strings in \mathcal{P}_n respectively [46]. In the Pauli decomposed version, Eq.(F.3) Thus the VQE optimization problem, designed using the quantum circuit reads,

$$E_{VQE} = \min_{\boldsymbol{\theta}} \langle \mathbf{0} | U^{\dagger}(\boldsymbol{\theta}) \hat{H} U(\boldsymbol{\theta}) | \mathbf{0} \rangle = \min_{\boldsymbol{\theta}} \sum_a^{\mathcal{P}} w_a \langle \mathbf{0} | U^{\dagger}(\boldsymbol{\theta}) \hat{P}_a U(\boldsymbol{\theta}) | \mathbf{0} \rangle. \quad (\text{F.3})$$

The iterative optimization of Eq.(F.3) is similar to the one that one encounters in machine learning, thus, is also known as the cost (loss) function in Hybrid systems.

Thus, the quantum expectation values need to be executed on a quantum device. By contrast, operations like summation of the expectation values in Eq.(F.3) and the iterative optimization, for e.g., gradient-descent (parameter update à la machine learning) of each of the terms in $E_{VQE} = \min_{\boldsymbol{\theta}} \sum_a w_a E_{p_a}$, is carried out using classical optimization algorithms. This clearly depicts the workload sharing pipeline between classical and quantum devices in hybrid frameworks.

¹¹The expectation value of a matrix \hat{O} with respect to a vector $|\phi\rangle$ is defined as $\frac{\langle \phi | \hat{O} | \phi \rangle}{\langle \phi | \phi \rangle}$.