RIGOLETTO: A Workflow Definition Language for Hybrid Quantum-Classical Scientific Applications

Vincenzo De Maio
HPC research group
TU Wien
Vienna, Austria
Email: vincenzo@ec.tuwien.ac.at

Dominik Bork
Business Informatics Group
TU Wien
Vienna, Austria
Email: dominik.bork@tuwien.ac.at

Ivona Brandic
HPC research group
TU Wien
Vienna, Austria
Email: ivona@tuwien.ac.at

***Abstract*—Scientific workflows constitute a pivotal resource for the scientific computing community, as they provide a standard interface to define, execute, and analyze the results of scientific computations. The advent of the Post-Moore era exposes the scientific computing research community with the challenge of scaling HPC facilities, to address the demands of modern scientific applications. While quantum computing, on the one hand, promises noteworthy theoretical speedups for different scientific applications, executing a whole scientific workflow on quantum hardware is infeasible since not all computations can achieve a speedup on quantum hardware. As a consequence, computational scientists defined the concept of a hybrid quantum-classical workflow, which is capable of exploiting the capabilities of quantum and classical hardware. Since the adaptation of classical scientific workflows to the new hybrid quantum-classical scenario can be challenging for scientists, we propose RIGOLETTO, a workflow definition language allowing scientists to define where to execute each task on a hybrid quantum-classical system. As a proof-of-concept, we report on two scientific workflows and identify open challenges in the definition of hybrid quantum-classical workflows.**

I. Introduction

Computational scientists provide scientists from different disciplines, as well as policy-makers, with important tools to understand scientific phenomena and/or support decision-making in different areas, as shown by the recent COVID-19 pandemic [1]. Among the many tools and frameworks for scientific computing, scientific workflows are crucial. Scientific workflows allow the definition of scientific simulations in high-level languages, improving accessibility to non-computer scientists and ensuring the reproducibility of scientific simulations. Since scientific computations require a cumbersome amount of computational resources, workflows are executed on HPC infrastructures. Workflow execution is managed by workflow management systems, that provide interfaces between the workflows' tasks and available computational resources.

With the advent of the Post-Moore era [2], [3], computational scientists are now facing the challenges of scaling the execution of scientific computations beyond the limits of classical computational hardware. As a consequence, the scientific community is investigating methods to seamlessly integrate non-Von Neumann-hardware in scientific workflows' execution, with the ultimate goal of exploiting the capabilities of different hardware setups to provide computational resources required by scientific workflows. This form of computing is called Post-Moore computing.

Among non-Von Neumann-computing, quantum computing stands out due to its theoretical speedup for different scientific computations and its natural modeling of different scientific problems [4]. However, since not all tasks can guarantee a speedup when executed on quantum devices, it is necessary to enable coordinated execution of both classical and quantum computation in hardware-software ecosystems, also known as *hybrid quantum-classical systems*. Scientific workflows, including both classical and quantum, are defined as *hybrid quantum-classical workflows* [5].

Scientific workflows are currently defined by employing high-level workflow definition languages (WDL), such as SWEL (Scientific Workflow Execution Language) [6] for data-intensive workflows, and CWL (Common Workflow Language) [7] for scientific workflows. However, to enable their execution on hybrid quantum-classical workflows, it is necessary to extend classical WDLs and provide specific language constructs to define (1) if tasks can be executed on quantum; (2) for which type of machine to optimize input code; and (3) execution specific-parameters of quantum computation.

In this work, we propose RIGOLETTO, a workflow definition language for hybrid classical-quantum applications. First, we identify the desired requirements for a WDL targeting hybrid classical-quantum applications and the main parameters of the target domain, based on which, we define the properties of RIGOLETTO. After, we provide examples of RIGOLETTO use on two scientific workflow applications, showing their execution on two types of machines.

We focus on universal quantum computers, more precisely on superconducting IBM machines and on ion-traps AQT machines. Also, we focus on hybrid quantum-classical applications with quantum tasks implemented using IBM Qiskit.

The paper is structured as follows: first, we describe the background and the related work in Section II. In Section III, we describe the main requirements, the properties, and the metamodel of RIGOLETTO, while in Section IV we provide two real-world applications of RIGOLETTO. In Section V, we describe the remaining challenges and future developments for RIGOLETTO. Finally, we describe threats to validity in Section VI and conclude our paper in Section VII.

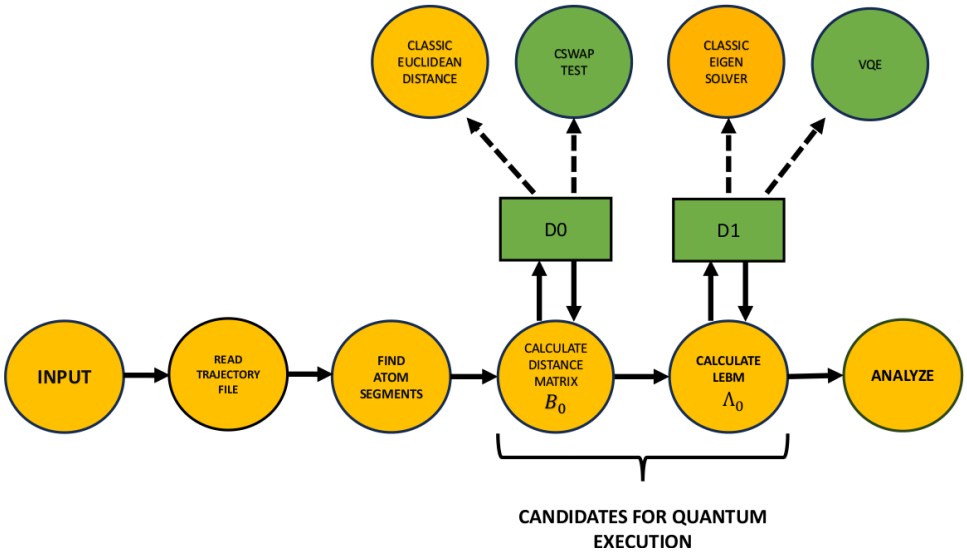


Fig. 1: A Hybrid Quantum-Classical Workflow, from [5].

II. Background

In this section, we briefly introduce the relevant background to this research, primarily scientific workflows, workflow management systems, hybrid quantum-classical workflows, and related works.

A. Scientific Workflows

Scientific Applications in different domains (i.e., finance, biology, chemistry, engineering) can be decomposed into a set of elementary *tasks* (i.e., aggregate data from different sources, average a set of samples, apply a method to a specific dataset). These tasks can then be combined to define *workflows*, which can be represented as directed acyclic graphs (DAGs) [8], [9], [10] where nodes represent the tasks and edges represent data and control dependencies between tasks.

Workflow models representing scientific applications are called *scientific workflows*. Workflows and tasks can be stored in public repositories (i.e., Pegasus workflow gallery[1]), allowing re-use of validated code, *repeatability* of simulations (i.e., the possibility to easily repeat the setup and execution of a simulation to increases confidence in the simulation's results), and reproducibility of computation (i.e., the possibility to reproduce and verify computation results), 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.

B. Workflow Management Systems (WMS)s

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

[1]https://pegasus.isi.edu/workflow_gallery/

C. Hybrid Quantum-Classical Workflows

Following from our definition in [5], a hybrid quantum-classical workflow is composed of both classical and quantum tasks, that are executed respectively on classical and quantum hardware.

An example of Hybrid Quantum-Classical Workflow is given in Fig. 1, coming from our previous work [5]. Yellow nodes represent classical tasks, while green nodes are quantum tasks. Rectangles are instead *decision nodes*, that, based on the logic codified in the WDL, decide whether to execute the classical or the quantum task.

While the execution of classical tasks has been widely discussed in the scientific workflow literature, we identify three specific execution models for hybrid quantum-classical tasks:

- **Circuit Execution:** this execution models a single sampling from a quantum circuit execution. It can be used to model sampling from a probability distribution modeled by the underlying circuit;
- **Task Execution:** in this execution model, multiple samplings from the same circuit are performed. The number of samplings is sometimes referred to as 'shots'. The goal of this execution model can be to calculate an expectation value or the histogram of the probability distribution. It can be used to model executions of quantum subroutines, such as Grover, Shor, or Deutsch-Jozsa.
- **Hybrid Execution:** represents computations involving continuous interaction between classical and quantum hardware, in contrast to what happens with task execution, where interaction happens only in the state preparation and the post-processing. Examples of this

model are Variational Quantum Algorithms [11], where a parametrized quantum circuit with a vector of parameters $\vec{\Theta}$ is used to represent the solution to a specific problem. $\vec{\Theta}$ values are found through iterative optimization on classical hardware, with the goal of finding the values of $\vec{\Theta}$ that bring us (close to) the optimal solution.

D. Related Work

Potential applications of quantum computing to current scientific workflows' performance are described [26], [27], [28]. Also, its native modeling of many scientific phenomena has been described in [29], [30].

Different frameworks for the development of quantum applications have been proposed, either for low-level languages such as QASM [12] or Q-Pragma [13], or high-level frameworks employing Python, such as Qiskit [19], PennyLane [20], and TKET [21]. Each framework provides its binding to different quantum hardware and its transpiler, as well as support to HPC infrastructures (i.e., GPU) to speed up classical computation. However, none of these frameworks provide methods for the definition of hybrid quantum-classical workflows.

Many WDLs have been proposed in the literature. Examples are DAX for Pegasus [8] or AWDL for ASKALON [9], which are based on XML. Other WMS, such as Makeflow [14] and SnakeMake [15], rely on Make syntax. Alternatively, well-known languages such as Python and YAML are used for workflow definition. Tools such as Airflow [22] employ Python for workflow definition, while Pachyderm [16] and Nextflow [17] employ a YAML-based language. Each of them supports different scientific applications, such as molecular dynamics [31], astrophysics [32], and bioinformatics [33]. However, these WMSs support only the classical execution and do not allow the execution of hybrid quantum-classical workflows. Other WMS, such as Python-based Orquestra [24] and Covalent [23], provide features for the definition and execution of workflows on hybrid quantum-classical systems.

Proposals and open challenges for hybrid quantum-classical workflows have been discussed in [5], [34]. Both works, however, do not consider the design of a *workflow definition languages*. The de-facto standard for workflow definition languages is CWL [7], [25], which is based on a YAML- or JSON-style syntax. CWL allows defining all steps of a workflow as well as each intermediate result required by them. At the time of writing, however, CWL does not provide language constructs to integrate quantum computations in hybrid quantum-classical workflows. The same applies to similar WDL proposals, such as GWDL [18] and Snakemake [15].

Table I compares RIGOLETTO with state-of-the-art proposals for scientific workflows. We can see, that languages used on typical WMS platform are mostly targeting classical tasks and classical HPC infrastructures. Platforms that include quantum tasks, such as Orquestra and Covalent, are based on Python, which might pose limits to the integration of new features and systems that do not provide Python APIs. More generic WDLs, such as CWL and GWDL, instead, do not provide abstractions and models for the execution of quantum tasks.

In this work, following the inspiration of CWL, we describe the properties and requirements for a generic WDL for the definition of hybrid quantum-classical workflows, that we name *RIGOLETTO*. We provide first usage examples of RIGOLETTO on two real-world use cases and identify open issues and challenges in the design of a hybrid quantum-classical WDL.

III. RIGOLETTO

In this section, we describe the requirements and the main characteristics of our proposed language, RIGOLETTO (wo**R**kflow def**I**nition lan**G**uage f**O**r quantum-c**L**assical sci**E**n**T**ific applica**T**i**O**ns), as a first step towards a workflow definition language for hybrid quantum-classical workflows. In Fig. 3 we describe how RIGOLETTO should be integrated into existing WMSs: the user defines (1) the structure of a hybrid quantum-classical workflow and (2) the definition of individual tasks, which can be provided using frameworks such as Qiskit or PennyLane. Both workflow structure and tasks' implementations are provided as input to the WMS, to a component called Classic-Quantum Mapper, that is responsible for deciding on which computational node (either classical or quantum) to execute which task, based on the information on the available hardware coming from the hardware catalog. The WMS will then communicate with a hybrid quantum-classical middleware, which provides APIs to execute and monitor tasks on the target hybrid quantum-classical system.

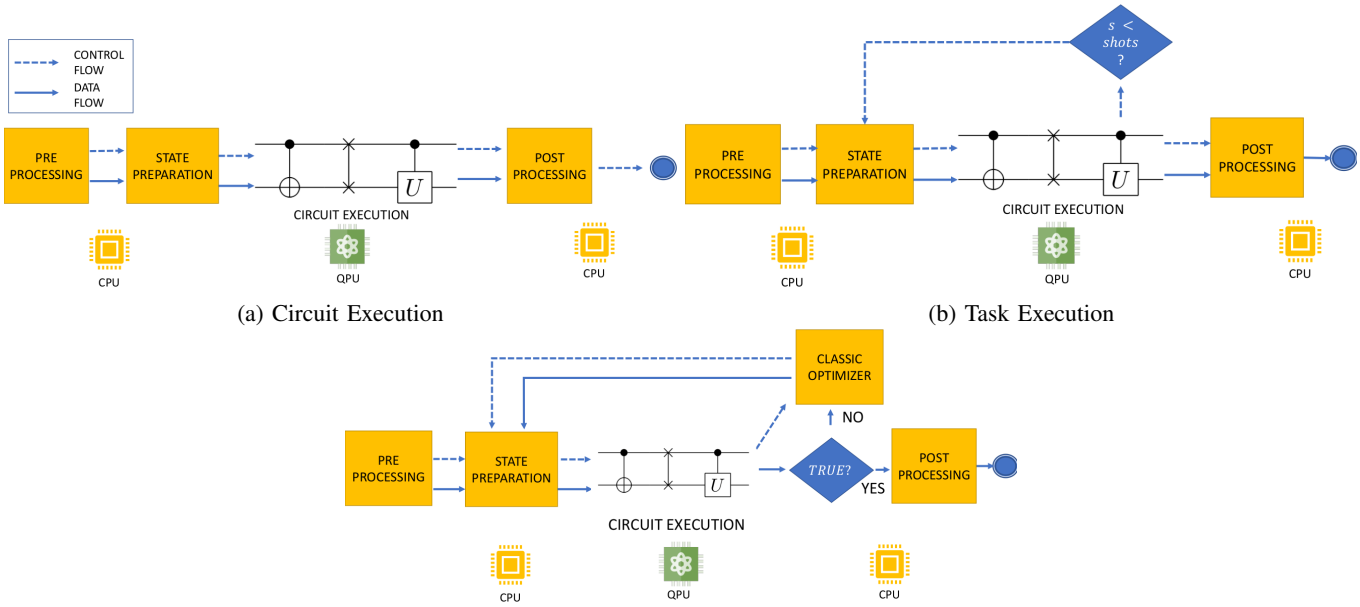
A. Requirements

The main requirements of a hybrid classical-quantum workflow definition languages that we identify are:

- **Expressivity**: RIGOLETTO must be capable of modeling the execution models defined in Section II-B, as well as all the parameters required by each execution model (e.g., number of shots required, target backend);
- **Interoperability**: different Cloud platforms provide access to quantum hardware, e.g., IBM Quantum, Amazon BraKet, and Google Quantum Platform. RIGOLETTO should allow the definition of workflows that can be executed on these different platforms, and it should provide means to define the aggregation and post-processing of output produced by these different platforms;
- **Low redundancy**: RIGOLETTO has to work in cooperation with existing workflow definition languages. As a consequence, the resulting specification should be capable of allowing interaction between classical and quantum hardware without interfering with the underlying WMS.

B. Language Properties

In the following, we describe the properties of RIGOLETTO. Based on the previously defined requirements, we notice that the circuit execution model is equivalent to the task execution model with a single execution. Therefore, we define two



(a) Circuit Execution

(b) Task Execution

(c) Hybrid Execution

Fig. 2: Hybrid Execution Models, from [5].

	Workflow Definition	Classical Tasks	HPC Infrastructure	Quantum Tasks	Quantum Execution Models	Quantum Execution Parameters	Quantum Simulation	Quantum Hardware
Low level languages								
QASM [12]				✓		✓	✓	✓
Q-Pragma [13]	✓	✓	✓	✓	✓	✓		✓
XML-based								
Pegasus WMS [8]	✓	✓	✓					
ASKALON [9]	✓	✓	✓					
Make-based								
Makeflow [14]	✓	✓	✓					
SnakeMake [15]	✓	✓	✓					
YAML-based								
Pachyderm [16]	✓	✓	✓					
Nextflow [17]	✓	✓	✓					
GWDL [18]	✓	✓	✓					
Python-based								
Qiskit [19]			✓	✓	✓	✓	✓	✓
PennyLane [20]			✓	✓	✓	✓	✓	✓
TKET [21]				✓	✓	✓	✓	✓
Airflow [22]	✓	✓	✓					
Covalent [23]	✓	✓	✓	✓			✓	
Orquestra [24]	✓	✓	✓	✓	✓	✓	✓	✓
JSON-based								
CWL [25]	✓	✓	✓					
RIGOLETTO	✓	✓	✓	✓	✓	✓	✓	✓

TABLE I: State-of-the-Art Frameworks for Definition of Hybrid Quantum-Classical Applications.

JSON objects: `task` and `hybrid`, modeling, respectively, the task execution model and the hybrid execution model of Section II-B. We describe each object in the following sections.

1) Task: For each task, firstly, we define whether we allow execution on quantum hardware. Afterward, it is important to define hardware-dependent parameters, i.e., the provider and the type of the quantum machine (usually accessed through the Cloud), execution related-parameters such as the number of shots, and how to process the output of the execution (i.e., if we are interested in the expectation or in the histogram of the samples). We define the parameters as follows:

- `function-name`: name of the function that implements the target task;
- `quantum`: boolean attribute, defining whether the task can be executed on quantum devices. Ideally, this value should be set by the underlying framework, depending on the application requirements and/or the availability of quantum resources;
- `provider`: enumeration of the available providers of quantum hardware. This value depends on the software

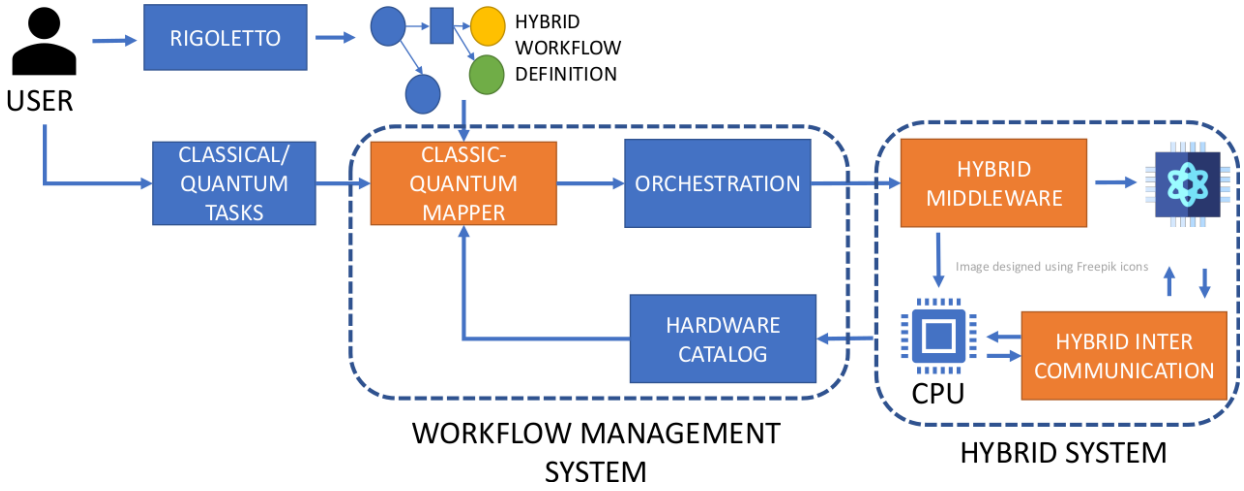


Fig. 3: Usage of RIGOLETTO within Hybrid Quantum-Classical WMS.

bindings that are available in the system, i.e., IBM, BraKet, AQT, Qandela. In the current version of RIGOLETTO, we allow only three values: IBM, AQT, and Aer. While the first two are Cloud providers, the latter is a local quantum simulator.

- `type`: enumeration of the available quantum hardware. This property is used by the system to select the most appropriate compiler for the underlying quantum hardware;
- `backend-name`: array containing the unique identifiers of the preferred backend to execute the quantum task;
- `token`: alphanumeric token, used for authentication purposes by the quantum hardware provider;
- `shots`: integer value determining the number of times that the quantum task is executed. Please note that the circuit model can be modeled by setting the `shots` value to 1;
- `postprocess`: this property can take two possible values: `expectation` and `samples`, respectively, modeling whether the output of the execution should be obtained as the expectation value or the histogram of all samples should be collected.

2) Hybrid: A hybrid object is used to model continuous interactions between classical and quantum hardware. In the current RIGOLETTO version, we only support the execution of variational quantum algorithms, such as Variational Quantum Eigensolver and Quantum Approximate Optimization Algorithms. Therefore, the selected properties are related to the execution parameters of variational quantum algorithms, namely the `ansatz` and the `optimizer` [11]. We extend the `task` object, adding the following properties:

- `ansatz`: enumeration of the available parametrized quantum circuits. In the current version, we focus on the parametrized quantum circuits provided by Qiskit;
- `optimizer`: enumeration of classical optimizers, that will be used to optimize the parameters of the circuits in different interactions;
- `opt_iters`: number of iterations of the optimizer, used as termination criteria for the variational quantum algorithms.

Notably, for each optimizer, other parameters can be considered, such as tolerance threshold, learning rate for gradient-based optimizers, and size of the trust regions for non-gradient-based optimizers. These parameters will be considered in future versions of RIGOLETTO.

C. Metamodel

In the following, we present a metamodel describing the abstract syntax of our RIGOLETTO workflow definition language. Many different ways to represent the abstract syntax (i.e., the metamodel) of a modeling language have been proposed in the literature [35]. Due to its ease of understanding and graphical compactness, we use a diagrammatic way of representing the RIGOLETTO metamodel using a UML class diagram.

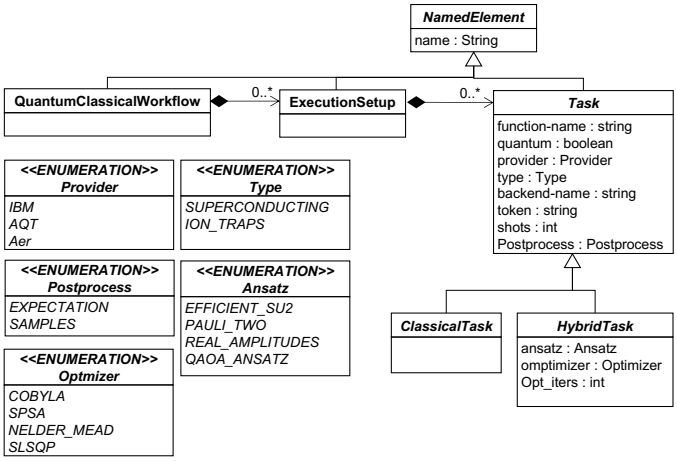


Fig. 4: RIGOLETTO metamodel.

Fig. 4 represents the RIGOLETTO metamodel. A `QuantumClassicalWorkflow` is composed of many `ExecutionSetups` that, themselves, are composed of

`Tasks`. Each `Task` is specialized into `ClassicalTasks` and `HybridTasks`; the latter one extending the properties of the `Task` class by specific properties specifying hybrid quantum-classical workflows. Furthermore, several enumerations are given to allow the valid specification of the `Provider`, `Postprocess`, `Optimizer`, `Type`, and `Ansatz`.

IV. Use Cases

In this section, we describe examples of usage of RIGOLETTO. We focus on two use cases, coming from molecular dynamics and machine learning applications, coming from our earlier work, respectively [4] and [36].

A. Molecular Dynamics Simulation

The target Molecular dynamics (MD) workflow described in Fig. 5 employs Metadynamics [37], i.e., well-chosen collective variables (CVs) are computed to capture important molecular motions in different regions of interest. A CV can be defined as a function of the atomic coordinates in one frame of the MD simulation that helps to reconstruct the free-energy surface for enhanced sampling. Since trajectories are reduced to time series of a few such CVs, analysis of MD simulation can be performed more efficiently. A CV can be the interatomic distance or can involve complex mathematical operations on many atoms. The CV that we will use in this work is the Largest Eigenvalue of the Bipartite Matrix (LEBM), that Johnston *et al.* [38] showed to be an efficient measurement to monitor changes in the conformation of two amino-acid segments I and J.

To capture the structural changes between two segments of amino acids segments, we focus only on the positions of α-Carbon (Cα) backbone atoms. Those backbone atoms are then used to form a bipartite matrix, whose maximum eigenvalue is a proxy for discovering structural changes in the molecular system. Formally, given two amino acid segments I and J, if d_{ij} is the Euclidean distance between C_α atoms i and j, then the symmetric bipartite matrix $B_{IJ} = [b_{ij}]$ is defined as:

$$b_{ij} = \begin{cases} d_{ij}, & \text{if } i \in I \text{ and } j \in J \\ d_{ij}, & \text{if } i \in J \text{ and } j \in I \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

In the next sections, we define the decomposition of the MD simulation in the target workflow described in Fig. 5 and the definition of quantum tasks using RIGOLETTO.

1) Workflow Decomposition: In the first step, the user inputs the parameters of the MD simulation, including a trajectory file, which models the structure of the target molecule. The trajectory file is then modeled in the second step. From the trajectory file, atom segments are extracted in step 3. Afterward, in step 4.1, we need to transform the classical data into a format that quantum machines can process. This process is called data encoding. Different algorithms for data encoding are available [34], and the selection of the most appropriate encoding is specific to the target problem. In this case, we apply amplitude encoding. We use the encoded data to prepare a quantum state, and we apply it to a quantum algorithm to calculate interatomic distance in step 4.2. Based on the values of interatomic distances, we build matrix B_{IJ}, which is then encoded in a quantum state in step 5.1. In step 5.2, we apply Variational Quantum Eigensolver to calculate the LEBM of B_{IJ}. More details on the selected algorithms and the design choices are provided in [4]. Results are then postprocessed using error mitigation.

2) Definition with RIGOLETTO: In this workflow, we identified two quantum tasks: (1) the calculation of distances d_{ij}, which are stored in the bipartite matrix B_{IJ}, and the calculation of the CVs, in this case, the eigenvalues of B_{IJ}, respectively described in Fig. 6 and Fig. 7. After specifying the function implementing the quantum task, in both cases, we allow quantum execution by setting the `quantum` attribute to `true`. For the distance calculation, we set AQT as a quantum device provider, and consequently set the type to `ion-traps`, since AQT provides ion traps machines and simulators. Please note that the value of `type` is not dependent on the provider, but on the type of machines and simulator that the providers expose. Finally, we set the name of the backend where we want to execute the function[2] Finally, we set the access token (as requested by the provider) and the number of shots, which we set to 250 in the case of `dist-calc` since AQT limits the number of shots that can be performed on their machines.

Concerning the calculation of eigenvalues, the function `eigenvalues` that performs the calculation employs a variational quantum algorithm, VQE, that follows the hybrid execution model described in Section II-C. As a consequence, in Fig. 7, we set up the additional parameters required for hybrid execution, namely, the `ansatz`, the `optimizer` and the `opt_iter`, respectively to `EfficientSU2`, `COBYLA`, and 1000 iterations. Afterward, we collect the expectation value of the quantum task.

B. Hybrid Quantum-Classical Machine Learning

We focus on a hybrid quantum-classical workflow that models the training of a Quantum Neural Network (QNN). QNNs have come up as one of the leading algorithms that could achieve a quantum advantage on NISQ technology, due to their simple architecture and training process [11], which is why we choose them for our studies [36], [39]. Similar to classical Neural Networks, a QNN is modeled as a parametrized quantum circuit $U(\vec{\Theta})$, whose gates can be compared to the nodes of the neural network, and the input set of parameters $\vec{\Theta}$ that is applied to the gates can be seen ad the weights of the neural network. The goal of the training is to find the $\vec{\Theta}^*$ that minimizes the loss function. This is done through a combination of classical optimization of $\vec{\Theta}$ and execution on quantum hardware of $U(\Theta)$.

[2]In the future, the language should be able to retrieve a list of available machines from the provider and automatically select the most appropriate for the execution.

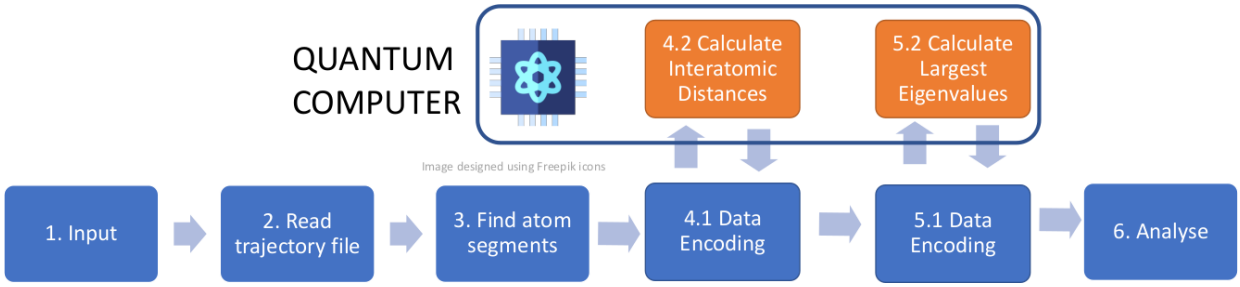


Fig. 5: A Hybrid Quantum-Classical Molecular Dynamics Simulation Workflow, from [5].

```
{
    "task":
    [
    {
      "function-name": "dist-calc",
       "quantum": true,
       "provider": "AQT",
       "type": "ion-traps",
       "backend-name": "name",
       "token": "ANY",
      "shots": 250,
      "postprocess": "expectation"
    }
    ]
}
```

Fig. 6: Example Specification of Quantum Task Execution for Molecular Dynamics Simulations.

```
{
"hybrid":
    [
    {
       "function-name": "eigenvalues",
      "quantum": true,
       "provider": "IBM",
       "type": "superconducting",
       "backend-name": "name",
       "token": "ANY",
      "shots": 1000,
      "postprocess": "expectation",
       "ansatz": "EfficientSU2",
       "optimizer": "cobyla",
       "opt_iters": 1000
    }
    ]
}
```

Fig. 7: Example Specification of Hybrid Execution for Molecular Dynamics Simulations.

The complete workflow structure is described in Fig. 8. We describe its decomposition in different tasks and how we define the execution of quantum tasks with RIGOLETTO.

1) Workflow Decomposition: In the first step, the application takes as input a classical dataset. Afterward, the dataset is pre-processed, i.e., typical operations applied to improve the quality of the dataset (e.g., removing null values) are performed; Then, in step 3, we apply different dimensionality reduction methods, such as LDA or PCA. This is particularly important, considering the limited amount of qubits available on quantum machines. When working with classical data on a quantum computer, a feature encoding method V is required to encode the data onto a quantum state (usually initially in state $|0\rangle$), as $|\psi\rangle = V|0\rangle$. This is done in step 4.1. Then, in step 4.2, $U(\vec{\theta})$ is prepared with initial values for the parameters $\vec{\theta}$. The expectation value for the observable O is then calculated as shown in equation (2).

$$f(x, \vec{\theta}) = \langle\psi| U^{\dagger}(\vec{\theta}) O U(\vec{\theta}) |\psi\rangle \tag{2}$$

After executing the circuit, the parameters $\vec{\theta}$ are optimized on a classical computer, using a cost function $C(f(x, \vec{\theta}), y)$ and an optimizer. The parameters are then transferred to the quantum computer, where the circuit is executed again. The process is repeated until a certain termination criterion is reached (e.g., maximum number of iterations, tolerance threshold). The different choices for feature map, ansatz, and optimizer can highly impact the results. Finally, in step 5, data are post-processed to mitigate the noise.

2) Definition with RIGOLETTO: In the current version of RIGOLETTO, we support only parameters that are common to all variational quantum algorithms, namely, the ansatz, the optimizer, and the number of iterations of the optimizer, as shown in Fig. 9. The data encoding, that applies a quantum feature map as described in [40], is currently handled in the `training` function, implemented in the target application. Concerning the ansatz, in the current implementation, the ansatz class must be part of the `qiskit.circuit.library` package. The same applies to the optimizer, whose class must be part of `qiskit.algorithms.optimizers`. In the example, we use `EfficientSU2` as ansatz and `COBYLA` as optimizer. The number of iterations is set to 1000.

V. Outlook

In this section, we describe the lessons we learned and possible improvements and open challenges in the design of RIGOLETTO.

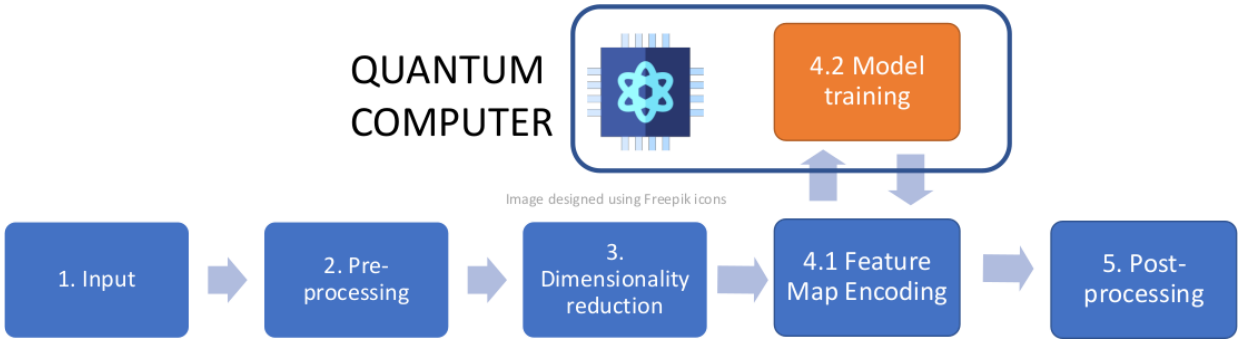


Fig. 8: Hybrid Quantum-Classical Machine Learning Workflow.

```
{
"hybrid":
    [
    {
        "function-name": "training",
        "quantum": true,
        "provider": "IBM",
        "type": "superconducting",
        "backend-name": "name",
        "token": "ANY",
        "shots": 1000,
        "postprocess": "expectation",
        "ansatz": "EfficientSU2",
        "optimizer": "cobyla",
        "opt_iters": 1000
    }
    ]
}
```

Fig. 9: Example Specification of Hybrid Execution for QNN Training.

A. Customizability

The applications that we selected could be implemented using the classes provided by IBM Qiskit, making the design of RIGOLETTO more accessible for us. However, the design of more complex quantum applications may require users to define their own classes and methods to address specific applications' needs. Future versions of RIGOLETTO should allow users to specify new classes and methods to integrate them into the workflow execution. To this end, we would need to define an execution engine of hybrid quantum-classical workflows, allowing users to specify paths and modules for their classes.

B. Integration with WMSs

We developed RIGOLETTO considering two specific use cases, that we implemented from scratch. However, due to the availability of different scientific applications for existing WMS, integration of RIGOLETTO with existing WMS would allow the re-use of existing classical scientific workflows and enhance them by introducing quantum computations. Integration of RIGOLETTO with different WMS would require the implementation of different source-to-source compilers or transformators, allowing to transform RIGOLETTO specifications into the language of the target WMS. A first step toward this integration could be the transformation into CWL [7], which is considered as the reference for WDL. Also, target WMS should be extended to include the required language abstractions, i.e., different quantum hardware and data transformation. We provide an in-depth discussion of these challenges in [5].

C. Quantum Framework Agnosticism

Since we developed target use cases using Qiskit framework, the model of RIGOLETTO is strongly influenced by the structure of Qiskit API. However, a plethora of quantum development frameworks are available on the market, e.g., Qiskit, PennyLane, TKET. Each framework has its own way of implementing the language constructs that are necessary for the execution of hybrid quantum-classical workflows. In the current version, RIGOLETTO is built upon the abstractions provided by Qiskit, since our applications are based on this framework. In future releases, RIGOLETTO should be able to interoperate with different frameworks, allowing to standardize execution of a wider range of applications.

D. Integrated Development Environment

In the current stage, the definition of hybrid quantum-classical workflows is performed using a classical text editor, that is validated using a JSON validator to ensure that the syntax is respected. In future developments, we plan to provide an IDE to support the developers in the design of hybrid quantum-classical workflow. Additionally, we could provide a graphical interface for hybrid quantum-classical workflows, similar to Sequanix [41] for Snakemake. In this respect, we could build a language server [42] for the RIGOLETTO workflow definition language that would allow easy integration into arbitrary IDEs and widely-used editors like VS Code that could realize textual or blended modeling of hybrid quantum-classical workflows.

VI. Threats to Validity

This research is not free from limitations and threats to validity. In the following, we briefly elaborate on the most

critical threats and the mitigation strategies we correspondingly applied.

- *Target Framework:* RIGOLETTO is currently tightly coupled with Qiskit abstractions. The identified properties might not be applied to other frameworks or may require extensions to integrate different applications.
  $\Longrightarrow$ In our future research, we plan to extend RIGOLETTO to also natively support applications designed using other target frameworks.
- *Selected Applications:* We focused on two different applications, namely, molecular dynamics and machine learning. Different applications might require different abstractions and execution models.
  $\Longrightarrow$ In our future research, we plan to extend the use case base evaluation expecting to shed light on further requirements for our metamodel.
- *Data Encoding:* In the current stage, data encoding is handled in the application, and it is not supported explicitly by RIGOLETTO.
  $\Longrightarrow$ In our future research, we plan to integrate the data encoding into an IDE for RIGOLETTO.

VII. Conclusion and Future Work

In this paper, we present the foundations of RIGOLETTO, a JSON-based language for the definition of hybrid quantum-classical workflows. First, we define hybrid quantum-classical workflows, and define the requirements and RIGOLETTO properties, together with the metamodel. Afterward, we show its application to two real use cases and identify challenges and threats to the validity of this study.

In the future, we plan to extend RIGOLETTO by integrating different quantum frameworks and applying RIGOLETTO to different hybrid quantum-classical workflows. Moreover, we plan to develop tool support to enable the efficient use of RIGOLETTO.

Acknowledgments

This work is partially funded through the projects: "Transprecise Edge Computing" (Triton), Austrian Science Fund (FWF): P 36870-N; Trustworthy and Sustainable Code Offloading (Themis), Austrian Science Fund (FWF): 10.55776/PAT1668223; and by the Flagship Project "High-Performance Integrated Quantum Computing" (HPQC) #897481 Austrian Research Promotion Agency (FFG) funded by the European Union – NextGenerationEU.

References

[1] D. Machi, P. Bhattacharya, S. Hoops, J. Chen, H. Mortveit, S. Venkatramanan, B. Lewis, M. Wilson, A. Fadikar, T. Maiden, *et al.*, "Scalable epidemiological workflows to support covid-19 planning and response," in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 639–650, IEEE, 2021.

[2] V. De Maio, A. Aral, and I. Brandic, "A roadmap to post-moore era for distributed systems," in *Proceedings of the 2022 Workshop on Advanced tools, programming languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems*, pp. 30–34, 2022.

[3] Y. Yang and I. Valov, "Rebooting computing in post moore era," *Advanced Intelligent Systems*, vol. 4, no. 8, 2022.

[4] S. S. Cranganore, V. De Maio, I. Brandic, T. M. A. Do, and E. Deelman, "Molecular dynamics workflow decomposition for hybrid classic/quantum systems," in *2022 IEEE 18th International Conference on e-Science (e-Science)*, pp. 346–356, 2022.

[5] S. S. Cranganore, V. De Maio, I. Brandic, and E. Deelman, "Paving the way to hybrid quantum–classical scientific workflows," *Future Generation Computer Systems*, vol. 158, pp. 346–366, 2024.

[6] R. Salado-Cid, A. Vallecillo, K. Munir, and J. R. Romero, "Swel: A domain-specific language for modeling data-intensive workflows," *Business & Information Systems Engineering*, pp. 1–24, 2023.

[7] M. Kotliar, A. V. Kartashov, and A. Barski, "Cwl-airflow: a lightweight pipeline manager supporting common workflow language," *Gigascience*, vol. 8, no. 7, p. giz084, 2019.

[8] "Pegasus workflow management system." https://pegasus.isi.edu/. [Online].

[9] M. Wieczorek, R. Prodan, and T. Fahringer, "Scheduling of scientific workflows in the askalon grid environment," *Acm Sigmod Record*, vol. 34, no. 3, 2005.

[10] "Dagshub: the home for data science collaboration." https://dagshub.com/. [Online].

[11] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, pp. 625–644, Sept. 2021.

[12] L. S. Bishop, "Qasm 2.0: A quantum circuit intermediate representation," in *APS March Meeting Abstracts*, vol. 2017, pp. P46–008, 2017.

[13] A. Gazda and O. Koska, "A pragma based c++ framework for hybrid quantum/classical computation," *Science of Computer Programming*, vol. 236, p. 103119, 2024.

[14] M. Albrecht, P. Donnelly, P. Bui, and D. Thain, "Makeflow: a portable abstraction for data intensive computing on clusters, clouds, and grids," in *Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, (New York, NY, USA), Association for Computing Machinery, 2012.

[15] J. Köster and S. Rahmann, "Snakemake—a scalable bioinformatics workflow engine," *Bioinformatics*, vol. 28, no. 19, pp. 2520–2522, 2012.

[16] J. A. Novella, P. Emami Khoonsari, S. Herman, D. Whitenack, M. Capuccini, J. Burman, K. Kultima, and O. Spjuth, "Container-based bioinformatics with Pachyderm," *Bioinformatics*, vol. 35, pp. 839–846, 08 2018.

[17] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature biotechnology*, vol. 35, no. 4, pp. 316–319, 2017.

[18] M. S. Mahmud, S. Abdullah, and S. Hosain, "Gwdl: a graphical workflow definition language for business workflows," in *Recent Progress in Data Engineering and Internet Technology: Volume 1*, pp. 205–210, Springer, 2013.

[19] A. Cross, "The ibm q experience and qiskit open-source quantum computing software," in *APS March meeting abstracts*, vol. 2018, pp. L58–003, 2018.

[20] A. Asadi, A. Dusko, C.-Y. Park, V. Michaud-Rioux, I. Schoch, S. Shu, T. Vincent, and L. J. O'Riordan, "Hybrid quantum programming with pennylane lightning on hpc platforms," *arXiv preprint arXiv:2403.02512*, 2024.

[21] S. Sivarajah, S. Dilkes, A. Cowtan, W. Simmons, A. Edgington, and R. Duncan, "t—ket⟩: a retargetable compiler for nisq devices," *Quantum Science and Technology*, vol. 6, p. 014003, Nov. 2020.

[22] S. Haines, "Workflow orchestration with apache airflow," in *Modern Data Engineering with Apache Spark: A Hands-On Guide for Building Mission-Critical Streaming Applications*, pp. 255–295, Springer, 2022.

[23] "Covalent: A unified platform for accelerated computing." https://www.covalent.xyz/. [Online].

[24] "Orquestra." https://zapata.ai/EarlyAccess/. [Online].

[25] M. R. Crusoe, S. Abeln, A. Iosup, P. Amstutz, J. Chilton, N. Tijanić, H. Ménager, S. Soiland-Reyes, B. Gavrilović, C. Goble, *et al.*, "Methods included: standardizing computational reuse and portability with the common workflow language," *Communications of the ACM*, vol. 65, no. 6, pp. 54–63, 2022.

[26] Y. Liu, S. Arunachalam, and K. Temme, "A rigorous and robust quantum speed-up in supervised machine learning," *Nature Physics*, vol. 17, no. 9, pp. 1013–1017, 2021.

[27] F. Løvholt, S. Lorito, J. Macias, M. Volpe, J. Selva, and S. Gibbons, "Urgent tsunami computing," in *2019 IEEE/ACM HPC for Urgent Decision Making (UrgentHPC)*, pp. 45–50, 2019.

[28] L. Lin, "Lecture notes on quantum algorithms for scientific computation," 2022.
[29] W.-L. Chang, J.-C. Chen, W.-Y. Chung, C.-Y. Hsiao, R. Wong, and 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*, vol. 20, no. 3, pp. 354–376, 2021.
[30] M. Swan, F. Witte, and R. P. dos Santos, "Quantum information science," *IEEE Internet Computing*, vol. 26, no. 1, pp. 7–14, 2022.
[31] A. Reinhardt, P. Y. Chew, and B. Cheng, "A streamlined molecular-dynamics workflow for computing solubilities of molecular and ionic crystals," *The Journal of Chemical Physics*, vol. 159, no. 18, 2023.
[32] K. Bousselmi, S. B. Hamida, and M. Rukoz, "Bi-objective cso for big data scientific workflows scheduling in the cloud: case of ligo workflow," in *15th International Conference on Software Technologies*, pp. 615–624, SCITEPRESS-Science and Technology Publications, 2020.
[33] S. Bansal and H. Aggarwal, "A multiobjective optimization of task workflow scheduling using hybridization of pso and woa algorithms in cloud-fog computing," *Cluster Computing*, pp. 1–32, 2024.
[34] B. Weder, U. Breitenbücher, F. Leymann, and K. Wild, "Integrating quantum computing into workflow modeling and execution," in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pp. 279–291, IEEE, 2020.
[35] D. Bork, D. Karagiannis, and B. Pittl, "A survey of modeling language specification techniques," *Inf. Syst.*, vol. 87, 2020.
[36] S. Herbst, V. D. Maio, and I. Brandic, "Streaming iot data and the quantum edge: A classic/quantum machine learning use case," in *Euro-Par 2023: Parallel Processing Workshops - Euro-Par 2023 International Workshops, Limassol, Cyprus, August 28 - September 1, 2023, Revised Selected Papers, Part I* (D. Zeinalipour, D. B. Heras, G. Pallis, H. Herodotou, D. Trihinas, D. Balouek, P. Diehl, T. Cojean, K. Fürlinger, M. H. Kirkeby, M. Nardelli, and P. di Sanzo, eds.), vol. 14351 of *Lecture Notes in Computer Science*, pp. 177–188, Springer, 2023.
[37] A. Barducci, M. Bonomi, and M. Parrinello, "Metadynamics," *WIREs Computational Molecular Science*, vol. 1, no. 5, pp. 826–843, 2011.
[38] T. Johnston, B. Zhang, A. Liwo, S. Crivelli, and M. Taufer, "In situ data analytics and indexing of protein trajectories," *Journal of Computational Chemistry*, vol. 38, no. 16, pp. 1419–1430, 2017.
[39] S. Herbst, V. D. Maio, and I. Brandic, "On optimizing hyperparameters for quantum neural networks," 2024.
[40] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.
[41] D. Desvillechabrol, R. Legendre, C. Rioualen, C. Bouchier, J. Van Helden, S. Kennedy, and T. Cokelaer, "Sequanix: a dynamic graphical interface for snakemake workflows," *Bioinformatics*, vol. 34, no. 11, pp. 1934–1936, 2018.
[42] D. Bork and P. Langer, "Language server protocol: An introduction to the protocol, its use, and adoption for web modeling tools," *Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model.*, vol. 18, pp. 9:1–16, 2023.