A Survey on Quantum Computer Simulators

Zakaria Abdelmoiz Dahi ‡§, Enrique Alba †, Rodrigo Gil-Merino ‡, Francisco Chicano † and Gabriel Luque †
‡ *Department Lenguajes y Ciencias de la Computacion, E.T.S.I. Informática, University of Malaga, Spain*
† *ITIS Software, Edificio Ada Byron, University of Malaga, Spain,*
§ *Department IFA, Faculty NTIC, Constantine 2 University, Algeria*
zakaria.dahi@{uma.es, univ-constantine2.dz}, {eat, chicano, gabriel}@lcc.uma.es, gilmerino@uma.es

***Abstract*—Quantum computers are unique systems based on peculiar properties from quantum physics, such as entanglement and superposition that allow them to provide unique computational performances. Quantum computing is meant to be revolutionary in many senses and fields. Some quantum machines have already been devised, but their accessibility or affordability to large commercialisation is yet to come. Meanwhile, a consistent plethora of quantum computer simulators are provided for users to run quantum programs on classical machines. These same programs could be later also executed in real quantum computers. This promising alternative allows practitioners to get beyond this impasse and evolve research in quantum techniques' design. Nonetheless, simulators differ from one another in many overwhelming aspects that harden classifying and profiling them and also choosing the most adequate one given a specific application purpose, programming or quantum computing paradigms. Considering these facts, our work presents a literature review of the existing quantum computer simulators. As far as we know, we are the first to perform such literature review: analyse and include 149 simulators in it and consider up to 10 comparison metrics including 21 programming languages/frameworks and also web, desktop and hybrid simulators. Our work offers several contributions by: `1)` providing a clear and encompassing repository that will allow users making appropriate choices of simulators, `2)` providing the research community with an up-to-date listing of advances in quantum computer simulation and `3)` opening new perspectives on how to build better future quantum computer simulators.**

***Index Terms*—Quantum Computer Simulators, Quantum Computing, Quantum Computers.**

I. Introduction

Quantum computers are computationally-empowered systems that take advantage of unique features such as entanglement and superposition provided by quantum mechanics [5]. They are expected to make a shift in software engineering and also deliver computational advances with endless applications [1]. Their use is awaited to be ground-breaking in so many ways. Not long ago real-world quantum machines did not exist yet and still up till today, the existing ones are either not affordable or not accessible at all for regular users (e.g. Sycamore QPU [1]). Considering that there is a paramount need for devising techniques that unleash quantum machines' power, as an alternative, while waiting for large and commercial quantum computers, classical computers capacities are being leveraged to simulate quantum computers with the help of *Quantum Computer Simulators* (QCS) [3]. Nonetheless, one should not confuse QCSs with quantum simulators like those described by Richard Feynman [2], which are real quantum systems. Also, one should bear in mind that QCSs can run quantum programs, like real quantum devices do, but they might run into limits such as memory and computation time [1].

Nowadays, a substantial variety of QCSs exist, where each one differs in several aspects (e.g. open accessibility, quantum system and language paradigms, number of qubits, etc.). Such massive configurations' variety poses several problems for picking the most adequate QCS considering a given constraint (e.g. application purpose) and also allowing the research community to keep track of the advances made in this field so as to evolve towards better future QCSs. To the best of the authors' knowledge, no work has reviewed QCSs with enough depth to afford the previous challenges. Nonetheless, one could mention the work done in [3] that dedicates a section to QCSs, although that work did not target QCSs. In addition, another interesting listing of QCSs can be found on the internet [(1,2)], but they are simple enumerations of QCSs not providing any in-depth technical details or classification methodology. Moreover, it provides a deprecated listing of QCSs that does not exist any more and whom the affiliated projects have been shutdown (e.g. `Fraunhofer QCS`, `GQC`, `Quantum Walks`, etc.) or even for those still active (e.g. `Davy Wybiral`) or inactive (e.g. `VirtualQC`), the provided links are incorrect or not working. Moreover, some QCSs are even programming languages (e.g. `Q-gol`, `LanQ`, `QCL`, `QWIRE`, `QASM`, `Quipper`, etc.), so one wonder if they should be classified as QCSs. Finally, some QCSs are said to be GUI-based, while actually, much more are. Also, it might happen that the same QCS is cited with new and old versions (e.g. `Quantum Fog`).

In our work we provide a comprehensive taxonomy by considering a substantial set of QCSs, various sets of comparison metrics that are important and relevant to QCSs' engineering. Technically, we conduct a systematic and encompassing literature review of the QCSs advances [4]. This is done by classifying and analysing the existing QCSs. We consider the aforementioned listings as a partial building-bricks of our work. Our contributions stands in being the first to `1)` dedicate a complete work to profiling QCSs, including 149 QCSs and `2)` consider up to 10 comparison metrics, 21 programming languages/frameworks and web, desktop and hybrid simulators.

[1]QCSs List (1): https://quantiki.org/wiki/list-qc-simulators
[2]QCSs List (2): https://qosf.org/project_list/

The remainder of the paper is as follows. In Section II, we provide some definitions and nomenclature to be used throughout our survey. Section III introduces the taxonomy and analysis of the reviewed QCSs. Sections IV and V present some final thoughts and suggestions to build better future QCSs and also list interesting application domains to be explored. Finally, Section VI concludes our paper.

II. Nomenclature and Selection Criteria

The quantum computing field contains a number of terms that are referring to different concepts, but that could, in some cases, be confused with each other. We can cite particularly quantum simulators, quantum computer simulators, and quantum programming languages. Making such distinction turns to be quite important particularly when looking at the quantum technology and software stacks given in [6]. Thus, in order to help introduce a clear and coherent nomenclature to be used in the literature, and also ease the understanding of our work, we will define here and we encourage future definitions of the three previously-cited terms to avoid any words' misuse.

We also present, in this section, the main criteria we applied to decide if a QCS should (or not) be included in our review.

A. Definitions

In this section, we present the definitions of quantum simulators, programming languages and computer simulators.

Definition 1: Quantum Simulators (QSs) are sometimes referred to as a quantum computing paradigm but technically they are task-dedicated quantum computing devices by themselves for studying a given aspect such as the model of quantum many-body mechanics. As an example of QSs, we could cite those pointed in Feynman's work [3].

Definition 2: Quantum Programming Languages (QPLs) are, in general, the set of languages that are based on classical programming paradigms (e.g. procedural, functional, multi-paradigm, etc.) or new ones (e.g. quantum-object, circuit-based, etc.) for quantum-related applications [7].

Definition 3: quantum computer simulators are, unlike quantum simulators, software that leverages classical computers to simulate quantum computers. The QCSs can be seen as a set of software layers that empowers the simulation of real quantum devices such as quantum simulators and this via quantum programming languages [3].

B. QS vs QPL vs QCS

Regarding the above-cited explanations, our work is about quantum computer simulators and not on QSs nor QPLs. Although the three might overlap in some cases, they should not be confused nor interchangeably used even if some works do it by making debatable statements. Indeed, works such as [7] states that quantum simulators cannot replace quantum programming languages. Moreover, the authors classify `QuantumOptics.jl` as a multi-paradigm quantum programming language, but they refer to it as an open quantum system, which is confusing. In addition, most QPLs reviewed in their work are based on classical ones (e.g. Object-oriented syntax, C and C++ compilers, etc.) and therefore it is not clear how a QPL has (or not) been classified as quantum-material. For example, the authors stated that `Quantum Language Q` is "*not a quantum programming language, but its library is written in C++*", although they do classify it as a QPL. Finally, they refer to `LIQUi|>` as a QPL, but it is actually a tool-suite for quantum computing that eventually could include a QPL.

C. QCSs' Sources, Inclusion and Exclusion Criteria

To the best of our knowledge, no guideline exists on what are the components'/software layers (e.g. compiler, circuit mapper/optimizer, etc.) of a typical QCS's. Doing so go beyond the scope of our paper. Also, even if most QCSs are open source, no details are given about their constituents, how they work, etc. So, as a first effort to review the existing QCSs, we have set some preliminary general criteria to decide which QCS should (or not) be included in our survey. Concretely, we integrated all QCSs that:

- Have been used in research-related works (e.g. theses, journal/conference papers, technical reports, etc.) or commonly cited in quantum-related resources (e.g. specialised magazines, fora, etc.).
- Implement basic or advanced qubits' manipulations.
- Implement basic or advanced gates' applications.
- Can be used online, via a desktop installer, or both.

More efforts are needed to establish standards of QCSs to have a clear definition of their essential components, workflow, etc. This work is a first step towards a more fine-selected QCSs and QCSs' implementation norms. Also, we would like to mention that most QCSs have not been included in officially published works, so most of our resources and references will be link-based (see Appendix A). The sources from where the studied QCSs have been extracted are:

- Academic publishers: e.g. Springer, IEEE, Elsevier, etc.
- Quantum corporations: e.g. IBMQ, D-wave, etc.
- Code's hosting platforms/pages: e.g. GitHub, personal web pages, etc.

Using the above cited-criteria, we have analysed 149 QCSs that we later filtered to 140 ones by neglecting those that we judged are not actually QCSs. After this, we further filtered the 140-QCSs-list to 100 QCSs by discarding all those who are not currently accessible (links and projects non-existent).

III. Comparison Criteria and Taxonomy of QCSs

In this section, we provide a review, analysis and taxonomy of 100 quantum computer simulators according to 10 comparison metrics. Three main families of QCSs are analysed in our work: web-based, desktop-based and hybrid (web-and-desktop-based) QCSs, where the first are online web services delivering a QCS and the second are QCSs that require offline pre-installation and execution without internet connection. For all the QCSs' classes, we performed the taxonomy according to 8 metrics: `Full-stack`, `#Qubit(s)`, `#Gate(s)`, `#Shot(s)`, `Application(s)`, `Project Status`, `Open Access` and `Open Source`.

These metrics are key features that are common, important, application-dependent and rule most QCSs' strengths and applicability. In addition, for the desktop and hybrid QCSs, we considered 2 additional comparison metrics: `programming languages` and `GUI-based` (see Figure 1). We use 21 programming languages/frameworks during the classification since it is the number we found after having extracted all the desktop and hybrid QCSs we found. We did not include the two supplementary metrics (language and GUI) for web-based QCSs since several programming languages/frameworks and APIs could be used simultaneously. One could use tools such as `wappalyzer`[3] to extract the languages involved in building a given web-based QCS. In addition, the metric `GUI-based` is not applicable to web-service-based QCSs.

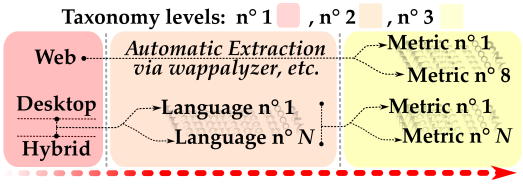

Fig. 1. Taxonomy skeleton

The 10 comparison metrics we use, have been chosen to reflect the QCSs simulation power, their accessibility and their history of activity. The metric `Full-stack` indicates if the QCS has been said (or not) to be a self-contained QCS. `#Qubit(s)` represents the number of qubits that the QCS provides. `#Gate(s)` represents the number of gates that can be used. `#Shot(s)` gives the number of times the circuit can be consecutively executed without being confronted to a given limitation (e.g. token validity, free trial ends, etc.). `Application(s)` tells us on whether the QCS is general-purpose or it has been tailored for a given field of application. `Project Status` reflects if contributions, updates and efforts are still going in the project that supports the QCS. `Open Access` indicates if the QCS is freely available (no payment required) or accessible upon payment. `GUI-based` indicates if the QCS includes a graphical-user interface or not, and `Open Source` shows if the QCS's code is freely available. `Language` indicates what programming language is used to implement the QCS. When reviewing the QCSs, we faced three main obstacles: the links of some QCSs were not working (non-existent), some QCSs could not be installed due to bugs and incompatibilities, other QCSs have a code that is too fuzzy and long to review. If the information could not be obtained or confirmed, we indicate "Unknown". Also, we use "–" for the attributes of a QCS that is found non-existent for a given reason. Indeed, most of the QCSs do not provide tutorials, documentation or insight of their use or implementation.

Figure 2 shows some statistics on the distribution of QCSs according to the platform: web, desktop and hybrid, including the programming languages they are based on. The bars' colour represents a class or language. The 1^{st} and 2^{nd} bars of the same colour indicate how many QCSs are considered before and after filtering, respectively. The latter is performed considering the inclusion/exclusion criteria in Section II-C. One can note that desktop-based are the most widely-spread ones, followed by web-based and then hybrid QCSs. Regarding desktop QCSs, those based on C, C# and C++ are the most popular ones.

The QCSs' list in Table II is organised per programming language (those with more QCSs to those with the least), while Tables I and III do not follow any ordering criterion. In Tables I-III, grey-shaded cells represent the best QCS(s) according to a given metric.

A. Web-based Quantum Computer Simulators

Table I represents the taxonomy of web-based QCSs, where 37.5% and 87.5% QCSs are open access and open source, respectively. One can note that most QCSs are application-tailored and also allow using a relatively high number of qubits and gates, although no much information is provided on their implementation. We also found that `Quirk` is among the top web-based QCSs. It provides a clear graphical QCS, it is easy to use via drag and drop functionality, it puts no limits on the number of qubits or shots to be used, it provides a large set of quantum gates, it provides both a video and written tutorials on how to use the QCS and it is open source.

B. Desktop-based Quantum Computer Simulators

Table II regroups the taxonomy of desktop-based QCSs, where 98.86%, 97.72% and 17.04% are open access, open source and have GUI, respectively. Regarding C/C++ QCSs, we found that `QuEST` is among the best QCSs because it provides many qubits and gates to use, it has a substantial written and video documentation and it does an efficient leveraging of the machines' CPU/RAM/network/GPU capacities. As to Java-desktop-based QCSs, we find that `jQuantum` is a promising simulator to use. Moving to Python-based QCSs, one can state that `Pyquil` is an interesting QCS to use considering the large plethora of applications and documentations it provides. For the remaining programming languages, it is hard to make firm conclusions on the usefulness of one QCS rather than others considering that not much information and insights (e.g. number of qubits, gates, shots, etc.) are given about most QCSs.

C. Hybrid Quantum Computer Simulators

Table III presents a taxonomy of hybrids QCSs, where 100%, 75% and 25% are open access, open source and have GUI, respectively. All these QCSs provide substantial documentation, quantum gates and qubits to be used that suit both industrial and research purposes. Nonetheless, we found that most hybrid QCSs are oriented towards fee-based QCS-services, which restricts their use at some point. Also, although we enumerate only QCSs, one should stress that other actors of the QC community such as `D-wave` and `IonQ` are more oriented towards fully-quantum devices rather

[3]Wappalyzer: https://www.wappalyzer.com/

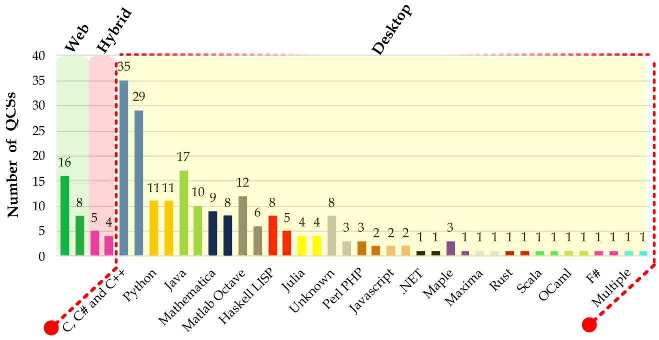

Fig. 2. Some statistics about quantum computer simulators

TABLE I
TAXONOMY OF WEB-BASED QUANTUM COMPUTER SIMULATORS

QCS Name	Full-stack	#Qubit(s)	#Gate(s)	#Shot(s)	Purpose(s)	Project Status	Open Access	Open Source
Factor 15 Circuit	Unknown	4	2	1	Shor's Algorithm	Unknown	Yes	No
Quantum Computing Playground	Unknown	22	21	1	Shor's and Grover's Algorithms, etc.	Unknown	Yes	Yes
Quantum Programming studio	Unknown	Unlimited	34	>= 1	Quantum circuit simulation	Active	Yes	No
Davy Wybiral QCS	Unknown	10	14	Unknown	Qubits manipulations	Unknown	Yes	Yes
Qubit Workbench	Unknown	Free: 4, Non-free: > 100	17	Free: > 9000, Non-free: Unknown	Qubits manipulations	Active	Yes/No	No
Quirk	Unknown	Unlimited	44	Unknown	Multiple	Unknown	Yes	Yes
Quantum Search Applet	Unknown	Unknown	Unknown	Unknown	Shor's algorithm	Unknown	Yes	No
BackupBrain	Unknown	Unlimited	9	>= 1000	Quantum circuit simulation	Active	No	No

than QCSs. Thus, they provide access to hybrid or quantum devices/algorithms and this via both web and desktop toolkits.

IV. DESIRED FEATURES IN FUTURE QCS DESIGN

Considering Section III, it can be seen that each QCS (or its category) has strengths and weaknesses. Thus, better QCSs could be built by summing up the strengths and discarding the weaknesses of all of them. We present here a set of desired features that could achieve this by: `1)` online vs desktop accessibility, `2)` interactions and manipulations, `3)` programming language paradigms, `4)` quantum computing paradigms, `5)` full-stack QCSs and `6)` software engineering principles. Our suggestions can be expanded to further aspects using more in-depth details. However, this goes beyond the scope of this paper. Also, one should keep in mind that future promising QCSs are yet to come such as `cuQuantum SDK` of nvidia, which leverage GPU performances of classical machines and also opens new perspectives in this axis.

A. Hybrid Accessibility and Execution Mode

Most QCSs provide one-handed and very constrained accessibility through webpages, while others require standalone installations. In this case, hybrid QCSs (e.g. IBMQ Experience) appear as a promising alternative that allows users with no internet access to run their programs at any moment using their personal machines, while users that have no access to desktop QCSs could use online QCSs. Moreover, synchronisation between both sides could allow users to switch back-and-forth between online and desktop QCS.

B. Simplified Interaction and Manipulation

Quantum computing users have different backgrounds. Thus, a QCS should provide manipulation tools that go with the users' profiles and expertise levels. For instance, providing graphical online and desktop QCS's features such as drag-and-drop (e.g. IBMQ, QuWire, etc.) could help academics, students and non-experts get familiar with quantum computing for teaching purposes and this without the need of programming skills. Also, automated circuit-based algorithm or automatic-problem formulation could be useful as well to avoid novice users getting too fast into technicality that might jeopardise their use of the QCS.

C. Programming Language and Quantum System Paradigms

Quantum devices are based on paradigms that rule the way they are used/executed. This ranges from problem formulation, problem mapping, algorithm declaration, algorithm mapping, quantum compilation and the used quantum simulator itself. These aspects are more related to advanced users that wish to control sophisticated aspects of the simulation. Therefore, providing unified or a multi-paradigm QCS (e.g. `QRBG`) could have several advantages such as the possibility of executing, with a reasonable change, the quantum program on several quantum computers supporting each a different paradigm. Also, each QCS is based on a particular programming language and paradigm. Thus, the variety of languages and paradigms handled by a given QCS is also a key factor to consider.

D. Full-stack Quantum Computer Simulator

Some QCSs emphasize on certain aspects and purposes of quantum computing rather than others (e.g. optimisation problem-solving, error correction, circuit optimisation, etc.). Such specialisation greatly affects the way the QCS is designed and also the range of its use. It is clear that the application domains and purposes are too large to be all integrated into a single QCS, but gathering the main functionalities could help to unify the research efforts and further comparisons between

TABLE II
TAXONOMY OF DESKTOP-BASED QUANTUM COMPUTER SIMULATORS

Language	QCS Name	Full-stack	#Qubit(s)	#Gate(s)	#Shot(s)	Purpose(s)	Project Status	Open Access	Open Source	GUI
C, C# and C++	**Intel Quantum Simulator (IQS, former qHiPSTER)**	Unknown	> 2	> 4	Unknown	Qubit's manipulations, problem solving, etc.	Active	Yes	Yes	No
	staq	Yes	Unknown	Unknown	>=1	Synthesis, transformation, optimization and compilation of	Active	Yes	Yes	No
	QuEST	Yes	45	29	>=1	Various	Active	Yes	Yes	No
	Scaffold/ScaffCC	No	–	–	–	–	–	Unknown	Yes	No
	Qrack	Unknown	> 32	Unknown	>=1	Quantum bit and gate simulator	Active	Yes	Unknown	No
	QX Simulator	Unknown	> 17	18	>=1	Quantum circuit simulation	Uknwon	Yes	Unknown	Yes
	Quantum++	Unknown	25	Unknown	>=1	General–purpose quantum computing library	Active	Yes	Yes	No
	QMDD	Unknown	Unknown	Unknown	>=1	Efficient representation and manipulation of quantum functionality	Unknown	Yes	Unknown	No
	CHP	Unknown	Unknown	4	>=1	simulator of stabilizer circuits	Inactive	Yes	Unknown	No
	libquantum (C)	–	Unknown	3	>=1	Quantum mechanics and quantum computing	Inactive	Yes	Yes	Yes
	Q++	Unknown	Unknown	Unknown	>=1	Simulating quantum computation	Inactive	Yes	Yes	Unknown
	QCLib	Unknown	20	Unknown	>=1	Classical simulation of realistic quantum computations	Inactive	Yes	Yes	Unknown
	QDD	Unknown	Unknown	Unknown	>=1	Shor's Algorithm	Unknown	Yes	Yes	No
	QGAME	Unknown	Unknown	Unknown	>=1	Quantum Algorithms	Unknown	Yes	Yes	Yes
	qsims	Unknown	Unknown	>= 2	>=1	Quantum computing in addressable optical lattices	Inactive	Yes	Yes	No
	QTM simulator	Unknown	Unknown	Unknown	>=1	Quantum Turing Machine Simulator	Inactive	Yes	Yes	–
	Quantum Computer Simulator	Unknown	Unknown	Unknown	>=1	Quantum computer simulator	Unknown	Yes	Yes	–
	Quantum Construct (qC++) / New quantum toolkit	Unknown	Unknown	Unknown	>=1	Quantum mechanical toolkit and 3D viewer	Unknown	Yes	Yes	Yes
	Quantum Network Computing	Unknown	Unknown	Unknown	>=1	Quantum computer simulations	Unknown	Yes	Yes	Unknown
	Qubiter	Unknown	Unknown	Unknown	>=1	Quantum computer simulations	Inactive	Yes	Yes	No
	QuCoSi	Unknown	Unknown	Unknown	>=1	Quantum computer simulations	Unknown	Yes	Yes	Unknown
	QuIDDPro	Unknown	40	Unknown	>=1	Generic quantum simulation	Inactive	Yes	No	–
	QWalk	Unknown	Unknown	Unknown	>=1	Simulator of quantum walks for one- and two-dimensional lattices	Active	Yes	Yes	Unknown
	Shor's Algorithm Simulation	Unknown	Unknown	Unknown	>=1	Shor's and Grover's algorithms	Active	Yes	Yes	Unknown
	sqct-Single qubit circuit toolkit	Unknown	Unknown	Unknown	>=1	Exact and approximate synthesis of single qubit circuits	Active	Yes	Yes	No
	JKQ–DDSIM	Unknown	> 4	Unknown	> 1000	Quantum simulations	Active	Yes	Yes	No
	QuIDE	Unknown	Unlimited	15		Quantum computer simulations.	Inactive	Yes	Yes	Yes
	QSim / Qsimh	Unknown	>= 1	22	>= 1	Various	Active	Yes	Yes	No
	SimQubit	Unknown	32	12	>= 1	Various: algorithms, etc.	Inactive	Yes	Yes	Yes
Python	**PyQuil /Forest**	Yes	>=6	23	>=1	Various	Active	Yes	No	No
	ProjectQ	Yes	>=22	>=9	>=1	Various	Active	Yes	Yes	No
	PyQu	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Yes	Yes	No
	QCircuits	Unknown	>=3	>=4	>=1	Various	Active	Yes	Yes	No
	qitensor	Unknown	>=1	Unknown	>=1	Study of quantum information and quantum computing	Inactive	Yes	Yes	No
	QuaEC	Unknown	>=1	>=7	>=1	Quantum error correction and fault-tolerance	Inactive	Yes	Yes	No
	Quantum Fog	Unknown	>=1	Unknown	Unknown	Quantum mechanical behavior	Active	Yes	Yes	Unknown
	Qubiter	Unknown	>= 4	>=1	>=1	Quantum circuit simulation	Active	Yes	Yes	No
	QuTiP	Unknown	>=3	32	>=1	Simulation fo dynamics of open quantum systems	Inactive	Yes	Yes	Unknown
	sparse_pauli	Unknown	Unknown	Unknown	Unknown	large, sparse Pauli operators using pairs of sets	Active	Yes	Yes	Unknown
	toqito	Unknown	Unknown	Unknown	Unknown	Study quantum information: states, channels, and measurements.	Active	Yes	Yes	Unknown
Java	**Bloch Sphere Simulator**	Unknown	Unknown	Unknown	Unknown	Bloch Sphere Visualisation	Inactive	Yes	No	Yes
	jQuantum	Unknown	15	7	>=1	Quantum circuit simulations	Active	Yes	Yes	Yes
	jSQ	Unknown	Unknown	Unknown	Unknown	Quantum cryptography	Inactive	Yes	Yes	Unknown
	LibQuantumJava (LQJ)	Unknown	2048		>=1	Quantum computing simulation	Active	Yes	Yes	No
	QuanSuite	Unknown	>= 1	>=1	>=1	Various application suite	Unknown	Yes	Yes	Unknown
	qMIPS101	Unknown	<22	10	>=1	MIPS and quantum circuit simulator		Yes	Yes	Yes
	QuSAnn (and Multiplexor Expander)	Unknown	Unknown	Unknown	Unknown	Code generator for simulated annealing	Unknown	Yes	Yes	Unknown
	Squankum	Unknown	Unknown	Unknown	Unknown	Quantum circuit simulations	Unknown	Yes	Yes	Yes
	Strange	Unknown	>=2	>=3	>=1	Creates Quantum Programs	Active	Yes	Yes	Yes
	Linear Al	Unknown	>= 1	>= 2	>= 1	Quantum information processing	Inactive	Yes	Yes	Yes
Mathematica	**QDENSITY**	Unknown	>=5	>=7	>=1	Teleportation, Shor's and Grover's algorithms	Unknown	Yes	Yes	Unknown
	qmatrix	Unknown	Unknown	Unknown	Unknown	Computation in quantum information theory	Inactive	Yes	Yes	Unknown
	Quantum	Unknown	Unknown	Unknown	Unknown	Various algorithms, applications, etc.	Inactive	Yes	Yes	Yes
	QuantumUtils	Unknown	Unknown	Unknown	Unknown	Various	Active	Yes	Yes	Unknown
	Quantum Information Programs in Mathematica	Unknown	>=4	>=9	>=1	Quantum circuit simulation	Inactive	Yes	Yes	No
	Quantum Turing Machine Simulator	Unknown	Unknown	Unknown	>=1	Quantum Turing Machine	Active	Yes	Yes	No
	QuCalc	Unknown	Unknown	Unknown	>=1	Quantum circuit simulation and problem solving	Inactive	Yes	Yes	Unknown
	QI	Unknown	Unknown	Unknown	>=1	Symbolic analysis of quantum states and operations	Active	Yes	Yes	Unknown
Matlab Octave	**M-fun for QC Progs**	Unknown	Unknown	>=1	>=1	Various	Active	Yes	Yes	No
	QC simulator	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	No	No	Unknown
	QETLAB	Unknown	>=1	Unknown	>=1	Quantum entanglement theory	Active	Yes	Yes	Yes
	QLib	Unknown	>=3	>=4	>=1	Various: entanglement, etc.	Inactive	Yes	Yes	No
	Quantum Octave	Unknown	>=1	>=1	>=1	Various: Teleportation, Shor and Grover algorithms, etc.	Active	Yes	Yes	Unknown
	Qubit4matlab	Unknown	>=20	>= 3	>=1	Quantum information/quantum optics	Inactive	Yes	Yes	Unknown
Haskell LISP	**Quacee**	Unknown	>=2	>=3	>=1	Quantum circuit simulation	Active	Yes	Yes	No
	CS 20c Project	Unknown	Unknown	Unknown	>=1	Quantum Turing machine	Inactive	Yes	Yes	No
	Haskell Simulator of Quantum Computer	Unknown	up to 100	>=4	>=1	Quantum circuit simulation	Active	Yes	Yes	No
	QIO	Unknown	>=1	>=1	>=1	Quantum computation: algorithms, etc.	Active	Yes	Yes	Unknown
	qchas	Unknown	>=1	8	>=1	Quantum Algorithms	Active	Yes	Yes	No
Julia	**QSWalk.jl**	Unknown	Unknown	Unknown	>=1	High-performance analysis of quantum stochastic walks	Active	Yes	Yes	No
	QuantumOptics.jl	Unknown	Unknown	Unknown	>=1	Various	Active	Yes	Yes	No
	QuantumWalk	Unknown	Unknown	Unknown	>=1	Models of quantum continuous and discrete walks	Active	Yes	Yes	No
	Yao.jl	Unknown	Unknown	Unknown	>=1	Empower quantum information simulation	Active	Yes	Yes	No
Unknown	**QCAD**	Unknown	7	11	Unlimited	Quantum circuit design	Inactive	Yes	No	Yes
	Quantum Computer Emulator	Unknown	16	>= 2	>= 1	Various: algorithms, hardware designs of quantum computers, etc.	Active	Yes	No	Yes
	Q-Kit	Unknown	Unlimited	22	Unlimited	Qunatum circuit simulation	Active	Yes	No	Yes
Perl PHP	**Quantum::Entanglement**	Unknown	Unknown	Unknown	Unknown	Shor's algorithm	Inactive	Yes	Yes	Unknown
	Quantum::Superpositions	Unknown	Unknown	Unknown	Unknown	Unknown	Inactive	Yes	Yes	Unknown
Javascript	**quantum-circuit**	Unknown	>= 20	49	>=1	Quantum circuit simulation	Active	Yes	No	No
	jsqis	Unknown	Unknown	Unknown	Unknown	Quantum circuit simulation	Active	Yes	Yes	No
.NET	**Quantum.NET**	Unknown	Unknown	Unknown	>=1	Quantum circuit simulation		Yes	Yes	No
Maple	**OpenQUACS**	Unknown	Unknown	Unknown	>=1	General-purpose universal Quantum Computer Simulator	Unknown	Yes	Yes	No
Maxima	**Qinf**	Unknown	Unknown	Unknown	>= 1	Various	Active	Yes	Yes	No
Rust	**QCGPU**	Unknown	Unknown	Unknown	Unknown	GPU accelerated simulation	Active	Yes	Yes	No
Scala	**VQS - Visual Quantum Simulator**	Unknown	>=4	>=3	>=1	Schrödinger full state Quantum Simulator	Active	Yes	Yes	No
OCaml	**QOCS**	Unknown	(limited) >= 1	>= 3	>= 1	Quantum circuit simulator: e.g. Shor's algorithm	Active	Yes	Yes	No
F#	**LIQUiD**	Yes	up to 30	>= 4	>= 1	Various: error correction, algorithms, etc.	Active	Yes	Yes	No
Multiple	**QRBGS**	Unknown	Unknown	Unknown	Unknown	Random number generation	Unknown	Yes	No	No

TABLE III
TAXONOMY OF HYBRID QUANTUM COMPUTER SIMULATORS

Language	QCS Name	Full-stack	#Qubit(s)	#Gate(s)	#Shot(s)	Purpose(s)	Project Status	Open Access	Open Source	GUI
Python	**Qiskit**	Yes	Up to 5000 (specific)	26	8192	Various	Active	Yes	Yes	Online (Yes), Desktop (No)
Python	**SV1**	Unknown	Up to 34	Unknown	Unlimited upon fees	Various	Active	Yes (Limited)	No	No
Python/Q#	**QDK**	Yes	Up to 30	>= 3	>= 1	Various	Active	Yes (Limited)	Yes	No
Python	**Cirq**	Unknown	20 internal and 30 external	>= 27	>= 1	Various	Active	Yes	Yes	No

the works conducted in the field of quantum computing. In this sense, a likely-to-be-generalised effort has already been done in `Terra`, `Aqua`, `Ignis` and `Aer` packages.

E. Respect of the Software Engineering Principles

To the best of our knowledge, no quantum software engineering principles have been devised yet, but still the QCSs are based on classical software. Thus, it is paramount to apply proper principles of software engineering. Indeed, authors in [3], have stated that most open access quantum computing software have not been devised in respect of software engineering bases. Thus, efforts are encouraged so that future QCSs will be built by considering such an important aspect.

V. Future QCSs' Application Domains

Even if QCSs' power is constrained by the machine they are used on, they still can be applied to several domains. We can cite as a first example, *artificial intelligence and problems' solving*. This stands in devising new hybrid or quantum algorithms that take advantage of the states' superposition and qubits' entanglement to solve intractable optimisation problems in various domains such as machine learning in artificial intelligence.

As a second QCSs' application, one could mention *quantum software engineering*. This includes software testing, hybrid quantum-classical software design, software quality assessment and classical-to-quantum software migration [6]. Finally, an interesting QCSs' application domain is *quantum machines' design*. This stands in evolving the design of quantum systems so they can reach new milestones in quantum computation. Many aspects such as quantum error correction, circuit optimisation and mapping, etc. are related to this axis.

VI. Conclusion

In this paper, we have conducted a systematic and comprehensive review of QCSs by `I)` considering 149 QCSs `II)` performing a comparison over 10 metrics, `III)` including 21 programming languages/frameworks and `IV)` web, desktop and hybrid simulators. Our work can be used to `(1)` make fast, easy and adequate QCSs' selection considering a given specific application, `(2)` allow academics and research community to keep an updated track of QCSs' engineering advances and `(3)` provide propositions for future QCSs' design and applications. We found that C and Python-based QCSs are the most spread, where `Quirk`, `QuEST`, `Pyquil` and hybrid ones are among the most promising QCSs to be used nowadays.

Acknowledgements

This research is partially funded by the Universidad de Málaga, Consejería de Economía y Conocimiento de la Junta de Andaluía and FEDER: grant number UMA18-FEDERJA-003 (PRECOG); Spanish Ministry of Science, Innovation and Universities and FEDER: contract RTC-2017-6714-5 (Eco-IoT); and TAILOR ICT-48 Network (No 952215) funded by EU Horizon 2020 research and innovation programme.

References

[1] Arute, F., Arya, K., Babbush, R., and et al. Quantum supremacy using a programmable superconducting processor. *Nature 574* (October 2019), 505–510.

[2] Feynman, R. Simulating physics with computers. *International Journal of Theoretical Physics volume 21* (June 1982), 467–488.

[3] Fingerhuth, M., Babej, T., and Wittek, P. Open source software in quantum computing. *PLOS ONE 13*, 12 (Dec 2018), e0208561.

[4] Paul, R. ACM SIGSOFT empirical standards version 0.1.0.

[5] Pednault, E., Gunnels, J. A., Nannicini, G., Horesh, L., and Wisnieff, R. Leveraging secondary storage to simulate deep 54-qubit sycamore circuits, 2019.

[6] Piattini, M., Serrano, M., Perez-Castillo, R., Petersen, G., and Hevia, J. L. Toward a quantum software engineering. *IT Professional 23*, 1 (2021), 62–66.

[7] Sunita, G., Maryam, G., and Amir, A. Quantum programming language: A systematic review of research topic and top cited languages. *Archives of Computational Methods in Engineering volume 28* (December 2021), 289–310.

Appendix

This appendix includes the links to the QCSs we have analysed in our work. The links are organised per class (web, desktop and hybrid) and further by programming languages. The QCSs order is the same as they appear in Tables I-III. The list is also maintained online to keep it updated on the long term (see final link in the table).

TABLE IV
Quantum computer simulators' links

Language	QCSs' Link according to QCS order
Web-based	
–	http://web.archive.org/web/20051214071130/http://www.isi.edu/acal/quantum/simulate.html http://www.quantumplayground.net/#/home https://davywybiral.blogspot.com/2012/12/quantum--circuit--simulator.html https://elyah.io/product https://algassert.com/quirk#circuit={\%22cols\%22:[]} https://joanv.me/qucomp/qucompApplet.html https://backupbrain.github.io/quantum-compiler-simulator/ https://quantum-circuit.com/docs,https://quantastica.com/
Desktop-based	
C, C# and C++	https://github.com/iqusoft/intel-qs https://github.com/softwareqinc/staq https://quest.qtechtheory.org/ https://github.com/epiqc/ScaffCC https://vm6502q.readthedocs.io/en/latest/index.html http://quantum-studio.net/ https://github.com/softwareQinc/qpp http://www.informatik.uni-bremen.de/agra/eng/qmdd.php https://www.scottaaronson.com/chp/ http://www.libquantum.de/ http://sourceforge.net/projects/qplusplus/ https://www.quantware.ups-tlse.fr/QWLIB/ http://thegreves.com/david/QDD/qdd.html http://faculty.hampshire.edu/lspector/qgame.html http://qsims.sourceforge.net/ http://web.archive.org/web/20050923134721/http://www.lri.fr/~durr/Attic/qtm/ http://www-imai.is.s.u-tokyo.ac.jp/~tokunaga/QCS/simulator.html https://sourceforge.net/projects/qcplusplus/ https://sourceforge.net/projects/qnc/ http://www.ar-tiste.com/qubiter.html https://sourceforge.net/projects/qucosi/ http://vlsicad.eecs.umich.edu/Quantum/qp/ http://www.cos.ufrj.br/~franklin/qwalk/ https://quantum-algorithms.herokuapp.com/ https://github.com/vadym-kl/sqct https://github.com/iic-jku/ddsim http://www.quide.eu/ https://github.com/quantumlib/qsim https://sourceforge.net/projects/simqubit/
Python	https://pyquil-docs.rigetti.com/en/stable/start.html http://projectq.ch/ https://code.google.com/archive/p/pyqu/ http://www.awebb.info/qcircuits/index.html http://stahlke.org/dan/qitensor/ http://www.cgranade.com/python-quaec/ https://github.com/artiste-qb-net/quantum-fog https://github.com/artiste-qb-net/qubiter http://qutip.org/ https://github.com/bcriger/sparse_pauli https://vprusso.github.io/toqito/
Java	https://eecs.ceas.uc.edu/~cahaymm/blochsphere/ http://jquantum.sourceforge.net/ https://sourceforge.net/projects/simu-quantique/ https://github.com/gbanegas/libQuantumJava http://www.ar-tiste.com/QuanSuite.html http://institucional.us.es/qmipsmaster/ http://www.ar-tiste.com/qusann.html http://jeffwass.github.io/Squankum/ https://github.com/redfx-quantum/strange http://linearal.sourceforge.net/#Home
Mathematica	http://www.pitt.edu/~tabakin/QDENSITY/ https://library.wolfram.com/infocenter/MathSource/1893/ http://homepage.cem.itesm.mx/lgomez/quantum/index.htm https://github.com/QuantumUtils/quantum-utils-mathematica https://quantum.phys.cmu.edu/QPM/ https://github.com/sdiemert/QTMSim https://library.wolfram.com/infocenter/MathSource/657/ https://github.com/iitis/qi
Matlab Octave	http://www.ar-tiste.com/m-fun/m-fun-index.html http://www-m3.ma.tum.de/Software/QCWebHome http://www.qetlab.com/Main_Page https://www.tau.ac.il/~quantum/qlib/qlib.html https://github.com/iitis/quantum-octave http://bird.szfki.kfki.hu/~toth/qubit4matlab.html
Haskell LISP	https://github.com/kat31416/quacee http://web.archive.org/web/20011207175140/www.cs.caltech.edu/~thoth/code.html http://web.archive.org/web/20010803034527/http://www.numeric-quest.com/haskell/QuantumComputer.html http://hackage.haskell.org/package/QIO https://hackage.haskell.org/package/qchas
Julia	https://github.com/iitis/QSWalk.jl https://qojulia.org/ https://github.com/iitis/QuantumWalk.jl https://github.com/QuantumBFS/Yao.jl
Unknown	http://qcad.osdn.jp/ http://www.compphys.org/QCE/ https://sites.google.com/view/quantum-kit/home
Perl PhP	https://metacpan.org/release/AJGOUGH/Quantum-Entanglement-0.32 https://metacpan.org/release/LEMBARK/Quantum-Superpositions-2.02
JavaScript	https://www.npmjs.com/package/quantum-circuit https://github.com/garrison/jsqis
.NET	https://github.com/phbaudin/quantum-computing
Maple	http://web.archive.org/web/20060116174553/http://userpages.umbc.edu/~cmccub1/quacs/quacs.html
Maxima	https://github.com/jlapeyre/qinf
Rust	https://qcgpu.github.io/
Scala	https://github.com/gmenier/VisualQuantumSimulator/wiki/Introduction
OCaml	https://github.com/dillanchang/QOCS
F#	https://tinyurl.com/Liquid-qcs
Multiple	http://random.irb.hr/
Hybrid	
–	https://quantum-computing.ibm.com/ https://aws.amazon.com/fr/braket/ https://azure.microsoft.com/fr-fr/resources/development-kit/quantum-computing/ https://quantumai.google/cirq
Permanent Link to QCSs' List	
https://github.com/Zakaria-Dahi/QCSs-List.git	