Quantum Computer Simulators

Julia Wallace*

Department of Computer Science, Old Library,
University of Exeter,
EX4 4PT, United Kingdom

Fax: +44 1392 264067
E-mail: J.Wallace@exeter.ac.uk
http://www.dcs.ex.ac.uk/~jwallace

Abstract

This paper offers the first available overview of progress in the field of quantum computer simulation. As research effort in quantum computation grows, together with the number of field researchers, it seems certain that quantum computer simulators will play a major role in this research.

Quantum computer simulators are discussed together with their role in the study of quantum computation. Currently available simulators are described and compared, and recommendations are given for the most appropriate simulators to use for specified purposes. In addition, a generalised, extensible framework of metrics for comparing quantum computer simulators is introduced and demonstrated.

An awareness of the functions and features of existing simulators is essential for researchers in the field, to avoid duplication of effort and to build on the foundations of previous work.

Keywords: quantum computer simulators, simulator metrics

1 Introduction

This paper details the results of an investigation into quantum computer simulators. These are defined as computer programs executed on a classical machine to simulate the actions of a quantum computer. The simulations effectively involve the use of machines that work according to the laws of classical Newtonian physics to simulate machines that work according to the laws of quantum mechanics.

---

* Julia Wallace is supported by Engineering and Physical Sciences Research Council (EPSRC) award no. 98318343

As quantum computer hardware is not available outside research laboratories at present, it is very useful to be able to create and develop quantum algorithms and test these via simulation on current, classical, computer hardware. Currently, a variety of quantum computer simulators exist, these vary in the complexity of simulations possible, in the representations used for quantum data structures, in the implementations of the quantum algorithms used by the simulators and in the accuracy of the simulators themselves. There is now a need for a comparative review of quantum computer simulators, which prompted this paper giving the first available overview of progress in the field of quantum computer simulation.

The paper begins discussing why there is a need for quantum computer simulators, explaining the problems that are inherent in attempting to simulate a quantum system by classical means and introducing a variety of quantum computer simulators. The paper will also briefly describe Shor's polynomial time quantum factoring algorithm (Shor, 1994), the most commonly simulated quantum algorithm. It aims to provide a general comparison of the quantum computer simulators and offer recommendations as to the most appropriate simulators to use for specific purposes and those simulators that seem suited to further development.

To assist in the assessment and comparison of quantum computer simulators, it would be useful to have a framework of simulator metrics in addition to a general review. Metrics would allow quick comparisons to be made between simulators, provide an easy framework to express results and allow comparison of simulators that have yet to be developed against those that have already been assessed. The paper discusses the identification and development of a framework of suitable metrics for comparing quantum computer simulators, provides descriptions of the metrics themselves and discusses how they can be used to assess simulators.

This paper describes the development of a generalised assessment methodology using the metrics. This was developed because it is possible to use quantum computer simulators for a variety of purposes. The framework of metrics developed and described in this paper can be adapted and extended to suit any comparison between simulators. The methodology is demonstrated by assessing a selection of simulators. Finally, the limitations of the simulator review work are discussed; this is followed by suggestions for further work.

2 The need for Quantum Computer Simulators

Feynman (1982) observed that classical systems cannot effectively model quantum mechanical systems. He proposed that the only way to effectively model a quantum mechanical system would be by the use of another quantum mechanical system. Feynman's observation suggests that it should be possible to use computers based on the

laws of quantum mechanics instead of the laws of classical physics to model quantum mechanical systems.

As research into quantum computation has progressed, significant progress has been made in developing the techniques necessary to produce quantum computer hardware. However, this hardware is not currently available outside research laboratories, and is inadequate in any case. The systems that have been constructed are insufficient for detailed exploration of some of the quantum algorithms that have been proposed. It is therefore useful to explore quantum algorithms by means of a quantum computer simulator which enables investigations to take place which would otherwise be impossible given the current state of quantum computer hardware.

3 Problems with Quantum Computer Simulators

A quantum computer simulator is an attempt to model a quantum mechanical system on a classical system and the quantum computer simulator must keep track of exponentially many computations in order to model the quantum mechanical machine accurately. The exponential increase in the number of computations increases with each additional simulated qubit and results in an exponential slowdown that is noticeable even in simulations of systems involving a relatively small number of qubits.

A simulation of a quantum computation is exponential in both space and time. If it were possible to construct an efficient quantum computer simulator then it would no longer be necessary to construct a quantum computer via production of quantum computer hardware. The quantum computer simulator itself would effectively be a quantum computer.

4 Quantum Computer Simulators

Details of the quantum computer simulators investigated for this review are given in Table 1. Although the primary aim of each simulator is to model the operation of a quantum computer, several simulators have been developed with secondary aims. For example, Qubiter aims to show the use of Quantum Bayesian Nets, QDD aims to show how Binary Decision Diagrams (BDDs) can be used to represent the quantum state, and both QCL and OpenQubit use a complex-number representation of the quantum state. In addition, QCL is an attempt to develop a high level, architecture independent programming language for quantum computers[1]. OpenQubit is a project which aims to write a quantum computer simulator to demonstrate Shor's algorithm and its efficiency on a quantum computer, and then to extend this code to a more general API that will allow the implementation of other quantum algorithms.

[1] I.e. the language has been developed with the idea that it will run on quantum hardware as well as being a simulation language.

Table 1: Quantum Computer Simulators

Simulator	Brief Description
QuaSi (Jun 2000)	QuaSi is a quantum computer simulator that simulates Deutsch's algorithm, Shor's algorithm and Grover's algorithm. Source code is not available but the author is currently working on an applet version for use over the Web. http://iaks-www.ira.uka.de/home/matteck/QuaSi/
OpenQUACS (May 2000)	OpenQUACS is an OpenSource general-purpose Quantum Computer Simulator written in the Maple Programming language. It comes as a pre-compiled Maple library or Maple source and has a full tutorial included. http://www.gl.umbc.edu/~cmccub1/quacs/quacs.html
Universal Quantum Computation Simulator (Jul 1999*)	The Quantum Computer Simulator enables simulation of a not-yet-realised quantum computer on a classical computer. It incorporates a sophisticated, easy-to-operate graphical user interface (GUI), enabling easy simulation of quantum algorithms by exchanging unitary elements with Mathematica. http://www.qc-simulator.com/
Quantum Turing Machine Simulator v1.2 (Jul 1999)	The Quantum Turing Machine Simulator (QTS) contains Mathematica software, ready-to-use Quantum Turing Machine Models, skeleton code to construct Quantum Turing Machines (QTMs) and performance measurements to estimate run times of QTMs. http://www.h-star.com/conresearch.html
Hayward's Shor's Algorithm Simulation (Jul 1999)	Program which simulates the operation of a quantum computer performing Shor's algorithm. Consists of four files which contain a simple complex number class for storing state information, a generic quantum register class which can be made to simulate any quantum memory register, Shor's algorithm itself and a library of useful functions used by Shor's algorithm. http://www.imsa.edu/~matth/cs299/
QDD v0.2 (Mar 1999*)	C++ library for quantum computer simulation demonstrating the use of binary decision diagrams, includes an implementation of Shor's algorithm. Version 0.2 released September 1999, version 0.3 in development. http://home.plutonium.net/~dagreve/qdd.html
Eqcs-0.0.5 (Mar 1999*)	Eqcs is a library allowing clients to simulate a quantum computer. Includes a test driver for the library. Still very much under development but includes a program showing the creation of a controlled NOT gate. http://home.snafu.de/pbelkner/eqcs/index.html
Quantum Computer Emulator (QCE) (1999)	QCE is a software tool that emulates various hardware designs of quantum computers. QCE provides an environment to debug and execute quantum algorithms under realistic conditions. It consists of a GUI and the simulator itself, is available for Windows '98/NT4 and is distributed with implementations of the Deutsch-Josza algorithm and Grover's algorithm. QCE is no longer considered to be under development, however, changes are made to the GUI at intermittent periods. http://rugth30.phys.rug.nl/compphys/qce.htm
OpenQubit 0.2.0 (Dec 1998*)	C++ quantum computer simulator which aims to demonstrate Shor's algorithm, and its efficiency on a quantum computer. The aim of the project is to extend this code to a more general application program interface (API). The project also aims to simulate Grover's algorithm on the same simulator. Current development version is NewSpin 0.3.3a. http://www.openqubit.org/

QCL v0.3 (Jul 1998)	QCL (Quantum Computation Language) is a high level, architecture independent programming language for quantum computers, includes program files for simulation of an implementation of Shor's algorithm and files for simulating other aspects of quantum computation. http://tph.tuwien.ac.at/~oemer/qcl.html
Finite State Machine (FSM) Simulation (Jul 1998)	The aim of this simulator is to show the simulation of any deterministic FSM (the FSM is made reversible) on a quantum computer in a space-efficient manner. By constructing a superposition of input strings of length k or less, it is possible to ask questions about the FSM (such as which inputs reach particular nodes). The answers can be found using a search algorithm (e.g. Grover's algorithm). http://xxx.soton.ac.uk/abs/quant-ph/?9807026
CS20c (Jun 1998)	Java library for simulation of a quantum computer, includes an implementation of Shor's algorithm. http://www.cs.caltech.edu/~amchilds/
AST Quantum Algorithm Simulator (Jun 1998)	Java simulation of the algorithmic steps of a model of Shor's algorithm. The simulator can be used to investigate Shor's method of factoring for numbers up to 10 digits in length (Crick, 1998).
Qubiter 1.0 (May 1998*)	Qubiter demonstrates the use of quantum Bayesian nets. It takes as input an arbitrary unitary matrix and returns as output an equivalent sequence of elementary operations (these are quantum computer operations like controlled NOTs and qubit rotations). These sequences can be represented graphically by qubit circuits. http://www.ar-tiste.com/qubiter.html
Be++ (Feb 1998*)	Be++ is a quantum computer simulator that runs under BeOS. Still very much under development, the first release of the software simulates the controlled NOT gate. http://home.worldnet.fr/~kubernan/
Shor's algorithm (Mathematica Notebook Simulation) (1998)	Mathematica notebook simulation showing the steps taken by a quantum computer factoring an integer using Shor's algorithm. Shows graphically what happens in each quantum register for each stage of the algorithm and obtains multiple samples of the discrete Fourier transform of register 1 by repeating Shor's algorithm $O(log(q))$ times to deduce the period, r. http://www.telospub.com/catalog/PHYSICS/Explorations.html
Mathematica Notebook Simulations (1998)	Mathematica notebooks to show: animations of quantum systems, basic tools for Dirac notation, simulation of quantum error correction, simulation of Feynman's quantum computer, analysis of interference effects, one time pad cryptosystem, simulating bugs in quantum computers, simulation of quantum cryptography, RSA-public key cryptosystem and simulation of quantum teleportation. These notebooks are generally not quantum computer simulators but illustrate issues that are relevant to quantum computation and the construction of quantum computer simulators. http://www.telospub.com/catalog/PHYSICS/Explorations.html
Quantum (Jun 1997)	Quantum circuit simulator for simulating quantum circuits on a parallel machine. Simulates more complex circuits than expected will be built within the next few years. Takes input in the form of a mathematical description of a circuit and then simulates the action of the circuit, includes an implementation of Shor's algorithm. http://www.themilkyway.com/quantum/

LGP2 and QC Simulator (1997*)	Quantum computer simulator and a specialised genetic programming system (LGP/LGP2) which can be used to discover better-than-classical quantum algorithms. The quantum computer simulator is used to evaluate the fitness of evolving quantum algorithms. The composite system demonstrates the development of quantum algorithms by genetic programming. Development of quantum algorithms is non-trivial and this research aims to address this problem by using automatic programming techniques to automatically generate new algorithms. http://hampshire.edu/lspector/code.html
Quantum Fog (1997)	A tool for investigating and discussing quantum measurement problems graphically in terms of quantum network diagrams called Bayesian nets (like Qubiter). It can calculate one- and two-variable conditional probability distributions, and draw a picture of every Feynman path that contributes to a physical situation. http://www.ar-tiste.com/
QULIB (shor.gz) (Oct 1996)	C++ library for the simulation of quantum computers on an abstract functional level, includes simulation of Shor's algorithm (this library is used within QCL). http://tph.tuwien.ac.at/~oemer/
Q-gol 3 (1996*)	An attempt to write a high-level programming language to allow researchers to describe algorithms designed to run on quantum computers, includes an implementation of Shor's algorithm as well as allowing visual quantum circuit design. http://www.ics.mq.edu.au/~gregb/q-gol/index.html
Factor 15 Circuit (1996)	An HTML form linked to a cgi-bin script (which uses the parallel quantum computer simulator) that performs a sample simulation of a factor 15 circuit (an implementation of Shor's algorithm where $n = 15$) showing the results of adding inaccuracies to the operation of the quantum computer. The user can select the error model as well as the magnitude of the error angle/variance and the initial random seed for the positive/negative and gaussian error models. http://www.isi.edu/acal/quantum/simulate.html
Parallel Quantum Computer Simulator (1996)	The parallel quantum computer simulator allows the simulation of circuits that are three to four orders of magnitude larger than any current proposed experimental realisations of a quantum computer. The simulator is modelled directly on the cold trapped ion quantum computer scheme proposed by Cirac and Zoller. The simulator takes as input the description of a quantum circuit specified in terms of logic gates. The simulator implements one, two and three bit controlled NOT gates as well as rotation gates. Circuits have been created to allow simulation of both Shor's and Grover's algorithms. http://www.isi.edu/acal/quantum/quantum_intro.html

*indicates that the simulator is still under development

A large number of quantum computer simulators provide implementations of Shor's algorithm. As a result of this, Shor's algorithm therefore became a natural point of comparison between quantum computer simulators in this research. It was used as a basis for exploring and comparing many of those simulators that it was possible to observe in operation. In view of the importance of this algorithm to the quantum computer simulator investigation, it will be helpful to describe its basic features.

5 Shor's Quantum Factoring Algorithm

Shor's quantum factoring algorithm is based on Simon's work (Simon, 1994) and a result from number theory (Miller, 1976). Simon described the construction of an oracle problem that takes polynomial time on a quantum computer but requires exponential time on a classical computer. Miller showed that factorisation can be reduced to the problem of finding the order of an element (i.e., given x and n, find r, such that $x^r \equiv 1 \pmod{n}$. r is called the order of the element x). Finding the order of an element is also known as finding the period, r, of the function $f_{x,n}(a) = x^a \bmod n$. Shor's algorithm replaces the call to the oracle in Miller's reduction by a call to an efficient quantum algorithm that finds the order of an element.

Shor's algorithm is the most commonly implemented quantum algorithm for quantum computer simulators. As a result of this, there may be several elements of this algorithm that differ from Shor's original description, depending on the implementation. Differences can result from minor variations in interpretation or a stronger adherence to a particular description of Shor's algorithm. Specific examples include whether non-coprime values of x are allowed as random numbers and differences in the methods used to calculate the factors themselves once the value of the period, r, has been calculated.

6 Simulating Shor's Algorithm

Detailed investigations of Shor's algorithm were carried out using QCL, QULIB, QDD, OpenQubit CS20c and Hayward's Shor's Algorithm Simulation[2]. The investigations involved executing multiple simulations of each simulator's implementation of the algorithm, using a range of values for n (the number to be factored). An investigation also took place into the effects of varying the value of the random number, x, when n = 33. The motivation behind this part of the investigation resulted from the observation that although calculations which use a particular value of x may produce the desired prime factors of n (i.e. factors which are both correct and non trivial), this may not be by the most efficient means. Detailed results from these investigations can be found in (Wallace, 2000) which also provides more extensive information about the simulators than is possible to present in this paper.

7 Comparison of Simulators

One distinguishing characteristic of the different quantum computer simulators is the representation of the quantum state. QDD uses a Binary Decision Diagram (BDD) representation of the quantum state. This contrasts with the complex-number

[2] These simulators all provided implementations of Shor's algorithm that could be used to obtain a reasonably sized set of results over a variety of inputs.

representation used by QCL, QULIB, OpenQubit and Hayward's Shor's Algorithm Simulation. Using complex numbers to represent probability amplitudes is currently the most common choice of quantum state representation. QCL is unique among the simulators in that, as a programming language, it is designed to work with any qubit-based quantum computer architecture as well as being a quantum computer simulation language. However, OpenQUACS is a recent attempt at a general-purpose Quantum Computer Simulator.

The Quantum Fog and Qubiter simulators are similar to QCL and OpenQubit as they provide an exact simulation of quantum behaviour, but these use quantum Bayesian Nets to represent the quantum state. Bayesian Nets are used in Quantum Fog and Qubiter because they are appropriate for working with the conditional probabilities encountered in entangled quantum states. However, Quantum Fog is a tool for writing quantum computer programs in a high level visual language, rather than a "bit level" quantum computer simulator like other simulators e.g. QCL and OpenQubit. A tool such as Qubiter is then used to translate this high-level language to qubit-level instructions.

BDDs, which are used by QDD to represent the quantum state, are suited to describing Boolean functions. The use of BDDs to model the underlying quantum state allows QDD to model relatively large quantum states (hence the large range of values of n which QDD can factor). However, use of the BDD representation restricts QDD to operating as a "digital" quantum computer, QCL and OpenQubit for example, support an "analogue" quantum computer model.

The FSM simulator aims to address the question of using quantum parallelism to simulate the execution of a program over a variety of inputs. This might imply that quantum computers could be useful tools for software validation. The Quantum Turing Machine (QTM) Simulator was designed to allow QTMs to be built that follow the step operator approach and perform model calculations. It is also hoped that it might serve to help demonstrate mathematical and physical principles involved in the machine model of quantum information theory.

The Parallel Quantum Computer Simulator was designed for a detailed investigation into the effects of errors that may occur in quantum computations. The Parallel Quantum Computer Simulator is the only simulator described in this review that is based on an actual physical experimental realisation of a quantum computer (Cirac & Zoller, 1995). Because the Parallel Quantum Computer simulator is based on a model of a quantum computer that has been experimentally realised, it was possible for its developers to compare experimental results with those generated by the simulator.

LGP2 and its quantum computer simulator is the only simulator system described within this review which, instead of aiming to simulate existing algorithms, aims to evolve new quantum algorithms by using automatic programming techniques. It enables quantum

algorithms to be produced by genetic programming and uses a quantum computer simulator and a specialised genetic programming system to discover better-than-classical quantum algorithms.

The majority of the quantum computer simulators only simulate a single quantum algorithm, often Shor's quantum factoring algorithm together with associated algorithms that are needed within Shor's algorithm such as the quantum Fourier transform[3]. Bing-Parks (1999) has developed a simulator which simulates and provides a visualisation of the abstract representations and mathematical derivations of Grover's algorithm (Grover, 1997). The exceptions to this "one algorithm per simulator" rule are the Parallel Quantum Computer Simulator (which provides circuits to simulate both Shor's and Grover's algorithms), the Universal Quantum Computation Simulation by Senko Corporation, Quantum Computer Emulator (QCE), and QuaSi. Simulating more than one algorithm using the same simulator is a significant step forward and has become more common in simulators that have been developed recently in 1999 and 2000.

The results of the simulator comparison, and the investigation into simulations of implementations of Shor's algorithm, show that there are several simulators that appear to provide a good basis for further development[4], these are QCL, OpenQubit, QDD and the CS20c Java simulator. In addition, the Parallel Quantum Computer Simulator is particularly appropriate for investigating the effects of errors in quantum computations and algorithms. It could also be used for designing circuits to run on a physical version of the model used by the simulator, as well as for general purpose quantum circuit and quantum algorithm design. This simulator is unique as it is possible to verify the results generated by the simulator using results obtained experimentally using cold trapped ion quantum computers.

The simulators discussed within this review will be of use to different groups of people. A student studying quantum computing, for example, may find it helpful to examine QCL (a generic Quantum Computation Language that is not tied to a specific architecture). Conversely, researchers who are involved with practical experimentation as well as simulation may find simulators based on a particular experimental model useful for verifying, extending and enhancing their investigations.

Finally, inconsistencies between the results obtained by some simulators highlight the need to compare results between simulators and, where possible, to compare simulator results with results obtained experimentally. Errors can be made in software development, these will affect simulator results. Errors may also occur in experimental

---

[3] However, some simulations of Shor's algorithm do not simulate the quantum Fourier transform.

[4] These recommendations relate only to those simulators that could be observed in operation. Simulators such as Qubiter, Q-gol, QuaSi, QCE etc. were not considered.

design, these will affect the results obtained from physical realisations of quantum computers.

8 Identification of Simulator Metrics

To assist in the assessment and comparison of quantum computer simulators, this paper will now discuss the development of a framework of simulator metrics. An extensive amount of research has taken place into identifying the type of factors that can be used for effective software metrics. Using general guidelines e.g. Mills in (Ashrafuzzaman, 1995) and (Lively, 1998), a large number of metrics have been identified and attempts have been made to classify these metrics into groups for assessing software, e.g. McCall's Software Quality Factors in (Zin & Foxley, 1996) and Hewlett Packard's FURPS in (Lively, 1998).

The simulator metrics were devised after analysing the factors it was felt necessary to consider while reviewing and comparing the quantum computer simulators. This required involved identifying factors that played a significant part in the simulator testing process. In addition it was necessary to consider a general evaluation of usability characteristics together with factors that might be relevant when selecting one simulator in preference to another. The metrics that were identified are as follows:

- performance - assessing the simulators in action, etc.
- size - size of the simulator and memory requirements, etc.
- quality - failure rates, bugs found, output produced for test cases, etc.
- functionality - range of problems handled, limitations, customisability without re-coding, etc.
- ease of use - ease of set-up, error messages, help provided, etc.
- portability - number of platforms supported, source code, software development tools needed, extensibility, etc.
- currency - age of simulator, whether used in real/actual quantum computing research, etc.

The investigation concentrated solely on features that can be compared on a "marks out of 10" basis, e.g. ease of use. General simulator features that should be recorded but cannot be compared on this basis, e.g. the programming languages that the simulators are coded in, are not considered within this paper.

9 Simulator Metrics

Comparative index measures will be applied to each of the quantum computer simulators with a view to assessing the metrics which follow. The items following each metric are examples of the factors measured by the metric.

- **performance**

    - speed of execution for a benchmark set of problems
- **size**
    - source code lines
    - size of compiled executable
    - memory requirements while running
- **quality**
    - rate of failure in benchmark tests
    - count of bugs uncovered during tests
    - inability to handle specific exceptions
    - verifiable, consistent and correct output for all test cases
- **functionality**
    - ability to handle increasingly large problems
    - limitations on the size of simulation
    - range of problems that can be handled
    - number of quantum algorithms simulated
    - quantum computer constructs simulated
    - customisability without re-coding
    - method of generating output
    - display of output in an understandable form
    - extent to which it simulates an "actual" general purpose quantum computer
    - results obtained for the amount of time spent
- **ease of use**
    - preparation for simulation
    - settings for a run saved in a file for re-use later
    - error messages meaningful/helpful
    - help/manual
    - author support
    - ease of understanding
- **portability**
    - availability of source code (or availability of executable for a variety of platforms)
    - number of platforms supported
    - availability of software development tools (editor/compiler/libraries etc.)
    - ease of customising/extending the simulator
    - compatibility with other simulators
- **currency**
    - currency of model of quantum operations
    - representation of "state of the art" thinking in its underlying algorithms
    - previous application to real/actual quantum research work
    - comparison with experimental results
    - prediction of experimental results

10 Scoring and Weighting the Metrics

Each metric was given a score ranging from 0 to 10 for each of the simulators. The importance of any particular metric depends on the purpose for which the simulator will be used, so to account for this, a weighting of 0.0 to 1.0 was applied to each score to reflect its overall importance, taking into account the intended use of the simulator. The overall score for each metric is given by: *score* * *rating*. The overall rating for a simulator is calculated by adding the overall scores for each metric, i.e.: Σ *overall metric score* = *overall rating*.

There are several ways in which the framework of metrics that have been described could be scored and weighted, the simplest of these is to score each metric and then apply a weighting. In order to establish the weightings for each metric, it was necessary to decide the primary use for the quantum computer simulator. Examples of possible primary uses include using simulators to develop general quantum algorithms, to test a particular quantum algorithm or to look in detail at the operation of a quantum computer. It may also be necessary to take into account the target user group.

Once the primary use of the simulator has been established, the list of metrics need to be ranked in order of their importance and relevance to this purpose. Weightings from 0.0 to 1.0 are then allocated to each metric in the ranked list. The overall score for each metric (and hence the overall rating of the simulator with respect to that particular use) can then be calculated.

Scores given for each particular metric are subjective, they depend on the person using the framework of metrics to assess the simulators and the purpose for which they are doing the assessment. The exception to this are scores for those items that are solely based on performance measures for test cases that can be duplicated.

The metric scoring method that has been detailed is extensible and is a generalised approach, as it allows easy assessment of simulators for purposes other than that for which the original metric scorings were given. This is possible because the relative weightings for a metric can be changed as necessary to reflect the importance of that metric to the new purpose. These new weightings can then be applied to the original scores to give a new overall rating for that simulator with respect to the new use, without the need to re-score the metrics.

11 Methodology and Design

The methodology for assessing the simulators is to allocate scores to each metric for every simulator (this will only need to be done once for each simulator unless new metrics are added or the structure of the scoring system is changed significantly), then:

- decide on the primary use of the simulator (may need to determine target user group if this is considered an important factor)
- make ranked list of metrics with respect to their importance to the primary use of the simulator
- allocate weightings to the ranked metrics
- calculate overall score for each metric
- calculate overall rating for each simulator
- compare overall ratings

Recommendations can then be made having used a comparative assessment method to determine the best simulator for the job. In order to demonstrate the methodology that has been developed, this paper describes the assessment of a variety of simulators using this scheme. For our example, the primary use for the simulators was selected to be the development of general quantum algorithms.

To demonstrate the methodology, it was decided to "estimate" the mark for every metric for each of the simulators (based on the knowledge obtained about each simulator from previous research). The metrics will then be prioritised with respect to the intended use of the simulator and appropriate weightings will be set for the ranked metrics. Weighted scores for each metric will be calculated. The weighted scores will be totalled to give the overall rating for each simulator.

Only those simulators that could be observed in operation were assessed, these were QCL, QDD v0.2, QULIB, OpenQubit 0.2.0, CS20c, AST, Mathematica Simulations (Shor's algorithm), Eqcs-0.0.5 and Hayward's Shor's Algorithm Simulation. Although it was possible to observe the Universal Quantum Computation Simulator and the Quantum Computer Emulator in action, these simulators were not assessed using the metrics due to problems installing and running the software.

The prioritised list of metrics for the purpose of general quantum algorithm simulation is detailed below. The user group selected for our demonstration is quantum algorithm developers. Weightings were applied to the seven metrics based on their relative importance to this user group. The list follows and is ordered with the metrics in descending order of importance. The name of the metric is followed by its weighting (in brackets) and then by the abbreviation used for the metric:

1. functionality, (1.0), FUNC
2. quality, (0.9), QUAL
3. performance, (0.8), PERF
4. currency, (0.7), CURR
5. ease of use, (0.6), EASE
6. size (0.5), SIZE (This is the only metric scored inversely, i.e. a high score indicates a small, fast simulator and a lower score indicates a larger, slower simulator.)
7. portability, (0.4), PORT

12 Results

The results of applying the framework of metrics to the simulators are presented in Table 2. Tables 3 and 4 show the initial scores for the simulators and the scores with the weightings applied. It can be seen that QCL has the highest overall rating when the primary use of the simulator will be to develop general-purpose quantum algorithms and that Eqcs-0.0.5 has the lowest rating.

Table 2: Final Rating (Maximum Rating = 49)

Simulator	Rating
QCL	35.7
QDD v0.2	35.4
QULIB	34.9
OpenQubit 0.2.0	33.9
CS20c	33.4
Hayward's Shor's Algorithm Simulation	33.2
AST	33.0
Mathematica (Shor's algorithm)	31.1
Eqcs-0.0.5	29.2

Table 3: Initial Scores

Simulator	PERF	SIZE	QUAL	FUNC	EASE	PORT	CURR
CS20c	6	4	8	7	6	8	8
QDD v0.2	9	2	8	7	7	8	8
QCL	7	1	9	9	8	7	7
QULIB	8	5	9	6	7	7	7
OpenQubit 0.2.0	8	3	6	7	8	8	8
Mathematica (Shor's algorithm)	5	8	9	5	7	4	6
AST	8	6	7	6	6	7	7
Eqcs-0.0.5	6	5	7	5	6	7	6
Hayward's Shor's Algorithm Simulation	8	5	6	7	7	7	7

Table 4: Scores with Weightings Applied

Simulator	PERF	SIZE	QUAL	FUNC	EASE	PORT	CURR	Rating
CS20c	4.8	2.0	7.2	7.0	3.6	3.2	5.6	33.4
QDD v0.2	7.2	1.0	7.2	7.0	4.2	3.2	5.6	35.4
QCL	5.6	0.5	8.1	9.0	4.8	2.8	4.9	35.7
QULIB	6.4	2.5	8.1	6.0	4.2	2.8	4.9	34.9
OpenQubit 0.2.0	6.4	1.5	5.4	7.0	4.8	3.2	5.6	33.9
Mathematica (Shor's algorithm)	4.0	4.0	8.1	5.0	4.2	1.6	4.2	31.1
AST	6.4	3.0	6.3	6.0	3.6	2.8	4.9	33.0
Eqcs-0.0.5	4.8	2.5	6.3	5.0	3.6	2.8	4.2	29.2

Hayward's Shor's Algorithm Simulation	6.4	2.5	5.4	7.0	4.2	2.8	4.9	33.2
Weighting	**0.8**	**0.5**	**0.9**	**1.0**	**0.6**	**0.4**	**0.7**	

13 Conclusions

Quantum computer simulators are a necessity because, at present, appropriate quantum computer hardware is not available outside research laboratories so it is very difficult to test and analyse quantum algorithms. These simulators are therefore extremely useful tools as they enable exploration and simulation of known quantum algorithms.

Designing and developing implementations of known quantum algorithms for quantum computer simulators has several benefits. Firstly, it increases our knowledge of the quantum algorithms themselves. Secondly, implementing algorithms for simulation may help us to make improvements to these algorithms, to find and fix errors, and aid the design and development of new algorithms. The design and development of new quantum algorithms can also be aided by the use of automatic programming techniques.

The aim of this paper was to provide a comprehensive overview of the quantum computer simulators that are available at the present time. The review also presented a generalised, extensible framework of metrics for comparing quantum computer simulators. A proper assessment of the simulators using the framework of metrics that has been developed should now be conducted, rather than an "estimate" assessment. In addition, a series of common uses for quantum computer simulators can be identified and ranked lists of metrics could be developed for each of these uses. Finally, there is also scope for extending and improving existing simulators and developing new simulators.

Acknowledgements

Thanks to Professor Greve, Dr Spector, Dr Tucci and Mr Ömer for answering questions about their simulators and to all the researchers and institutions that provided me with source code and executables. I would also like to acknowledge the advice of my supervisor Dr Narayanan and thank all those who have contacted me regarding my work.

References

Ashrafuzzaman Mohammad (1995). Putting Metrics into Software Perspectives, Department of Computer Science, University of Saskatchewan, http://www.cs.usask.ca/homepages/grads/moa135/856/metrics/metrics.html

Bing-Parks Terrance D. (1999). A Visualization of a Quantum Mechanical Search Algorithm, MS Thesis, San Jose State University.

Cirac J. I. & Zoller P. (1995). Quantum Computations with Cold Trapped Ions, Physical Review Letters, 74, Number 20, pp. 4091-4094.

Crick David A. (1998). Quantum Computing, Department of Computer Science, University of Exeter, UK, Project III Stage III Report [Unpublished].

Feynman Richard P. (1982). Simulating Physics with Computers, International Journal of Theoretical Physics, Vol. 21, pp. 467-488.

Grover Lov K. (1997). Quantum Mechanics Helps in Searching for a Needle in a Haystack, Physical Review Letters, Vol. 78, Number 2, pp. 325-328, quant-ph/9605043.

Lively Mac (1998). Technical Metrics for Software Chapter 18, Department of Computer Science, Texas A&M University, http://www.cs.tamu.edu/course-info/cpsc689/summer98/lively/volz_431_html/chap-18/

Miller Gary L. (1976). Riemann's Hypothesis and Tests for Primality, Journal of Computer and System Sciences, Vol. 13, pp. 300-317.

Shor Peter W. (1994). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, quant-ph/9508027 v2. (Proceedings of the 35th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Editor Shafi Goldwasser, pp. 124-134, 1994).

Simon Daniel (1994). On the Power of Quantum Computation, Proceedings of the 35th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Editor Shafi Goldwasser, pp. 116-123.

Wallace Julia (2000). Quantum Computer Simulators – A Review, Version 2.1, http://www.dcs.ex.ac.uk/~jwallace/simrevab.htm

Zin Abdullah Mohd & Foxley Eric (1996). Automatic Program Assessment System, Department of Computer Science, University of Nottingham, http://www.cs.nott.ac.uk/Department/Staff/ef/ceilidh/papers/ASQA.html