IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Mathematical Modelling of Engineering Problems
Vol. 9, No. 4, August, 2022, pp. 879-886
Journal homepage: http://iieta.org/journals/mmep

State-Domain Equations and Their Quantum Computing Solution Based HHL Algorithm

Mohamed El Amine Boudjoghra[1], Sid Ali Sofiane Daimellah[1], Nadjet Zioui[2*], Yousra Mahmoudi[2], Mohamed Tadjine[1]

[1] École Nationale Polytechnique of Algiers, 10 Rue des Frères OUDEK, El Harrach 16200, Algeria
[2] Université du Québec à Trois-Rivières, 3351 Boulevard des Forges, Trois-Rivières, QC G8Z 4M3, Canada

Corresponding Author Email: nadjet.zioui@uqtr.ca

https://doi.org/10.18280/mmep.090404

Received: 31 May 2022
Accepted: 11 August 2022

Keywords:
quantum computing, qubits, HHL algorithm, dynamical systems, quantum state-domain equations

ABSTRACT

Quantum computing is becoming a major new field in engineering. Adaptation of classical computation concepts to this emerging technological reality is underway. In this study, a sampling-based approach is used to adapt state-domain equations to HHL-based formulation to obtain a homologous quantum algorithm for solving the dynamic equations in the state domain. The results obtained by the quantum computing approach are very close to the real response of the system and to the power-series-based exact solution. This work represents a significant contribution to the advancement of dynamic systems modeling and solving in state-domain adapted for use in quantum computers in the future.

1. INTRODUCTION

1.1 Context

Differential equations are crucial in modeling and explaining a very large collection of phenomena and complex problems in science and engineering, especially in automatic control. They have been used in order to introduce state domain equations models and bring an assortment of advanced analysis and control methods to the field. Solving differential equations are at the heart of systems analysis and control and computers have become full-fledged part of the process. With the emergence of quantum computing and the recent advances in quantum computers developments and optimization, a deep work has to be carried out on the several classic problems and concepts in several fields. This is performed in order to benefit from the extolled performances of the quantum computers on one hand. On the other hand, in order to keep up with the advances and to adapt the concepts to the new inevitable quantum reality.

1.2 Problem formulation

Solving systems of equations is a key tool in applied mathematics and appears at the basis of various complex problems such as optimization problems, solving partial differential equations and eigen problems, among others. The classical processes designed for this purpose require an execution time proportional to the number of variables and constraints. Thus, for real-world situations, often incurring large data sets, such a task can be quite hard for classical computers and therefore exploiting the advantages offered by quantum computing seems to be a promising solution.

State equations are one of the most powerful modeling tools in current use in the field of automation and control engineering. They underlie advanced techniques including adaptive control, sliding modes, the trust controller, the Lie algebra-based controller, and many others, in linear as well as non-linear systems, further increasing their versatility (since most systems in nature include nonlinearities).

Quantum versions of differential equations as well as methods of solving systems of equations have been covered widely in the literature. However, no quantum tool as a modeling methodology or for solving state-domain equations for control theory purposes is known. In addition, most studies on solving linear equations involve converting iterative equations into multi-dimensional matrix systems, which requires impractical numbers of qubits [1]. In other words, executing an algorithm on N sampling time periods implies the use of as many qubits, which is usually beyond the limits of the resources available, since quantum computers with a few dozen qubits are still in development.

1.3 Related works

There are several studies that have been carried out in the field of differential and partial differential equations in their quantum versions [2, 3]. The resolution of sparse inhomogeneous linear differential equations and use of high-order methods to propose an improved time-efficient version of quantum simulation algorithm is tackled by Iyer and McCune [3]. Compared to classical counterpart, method provides an exponential speed-up. On the other side, Berry et al. [4] develops a quantum algorithm for systems of linear ordinary differential equations with constant coefficients, which produces a quantum state proportional to the solution in a predefined time. The proposed algorithm is polynomial in the logarithm of the inverse error and offers an exponential improvement over anterior quantum algorithm. Khater et al. [5] studied the method of extended simple equations and an expansion method, and applied them on the two-dimensional nonlinear Kadomtsev-Petviashvili Burgers equation in quantum plasma. The outcome came to be the exact solutions of traveling waves. The authors attest that these methods are

of high efficiency for solving nonlinear partial differential equations. Srivastava and Sundararaghavan proposed an algorithm based on the graph-coloring methodology named "box algorithm" to solve second-order differential equations [6]. Childs et al. [7] proposed a quantum algorithm based on the spectral approaches for linear ordinary differential equations. The proposed algorithm constitutes an attractive substitute to finite difference methods that provides a global approximation of the solution. Kyriienko et al. [8] proposed to solve differential equations in a high-dimensional feature space using a quantum algorithm with spectral method. Zanger et al. explored the use of quantum computers to solve differential equations by using the basis encoding and fixed-point arithmetic approach [9]. Moreover, Kumar et al. [10] combined advanced cuckoo search (CS) algorithm along with adaptive Gaussian quantum behaved particle swarm optimization (AGQPSO) as a hybrid algorithm for solving second order differential equations. Converting these equations into unconstrained/bound constrained optimization problems is the main idea of the proposed algorithm. Other works studied stochastic differential equations (SDEs) [11-13] and Poisson equation [14-16] that constitute other classes of differential equations of great interest. The authors described a quantum Poisson equation solver based on the Harrow-Hassidim-Lloyd (HHL) quantum algorithm in reference [16]. All the works found in the literature are undoubtedly proven contributions to the formulation of differential equations under the gaze of quantum computing, however without vision of orientation of control theory, nor consideration of state domain equations.

The HHL algorithm solves linear systems with exponential speed and outperforms classical methods. This algorithm has been exploited in many important quantum computing algorithms. According to Harrow et al. [17], a quantum computer can estimate the value of a given function of the complete solution $\vec{x}$ of size N in a time that scales logarithmically in N, and polynomially in the number of conditions κ and the anticipated accuracy, in some cases. Accordingly, the authors describe the Harrow-Hassidim-Lloyd (HHL) quantum algorithm for estimating $\vec{x}^t M\vec{x}$ whose algorithmic complexity is polynomial in log(N) and κ. Moreover, Cai et al. [18] conducted an experimental demonstration of the performance of the HHL algorithm by attempting to solve various systems of linear equations of size 2×2. All solutions provided by the HHL algorithm are of high accuracy ranging from 0.825 to 0.993. Lee et al. presented and described a hybrid quantum algorithm for solving systems of linear equations as an improvement of the Harrow-Hassidim-Lloyd (HHL) algorithm [19]. The results produced by the hybrid algorithm are exactly the same as those of the HHL algorithm in most cases. In particular, the hybrid algorithm is shown to be more accurate than the HHL algorithm for some specific systems of linear equations. Sellier and Dimov [20] implements a quantum mechanics based solver to solve for systems of linear equations and Li et al. [21] used the sparsity-independent quantum singular value estimation algorithm to develop a simplified quantum scheme to solve the linear regression equation. The proposed scheme allows reducing the time complexity from $o(Nn^2)$ to $o\left(\sqrt{N}\log(n)\right)$.

Although the HHL algorithm is one of the most widely used algorithms so far for solving systems of equations, it is not suitable as such for a dynamical system. The introduction of the time dimension is a challenge for dynamic systems as it is in control systems.

Additionally, Bernstein and Yang [22] designed and analyzed a quantum algorithm that combines the concept of Extended linearization (XL) with the brute-force search and/or Grover's algorithm to solve systems of m quadratic equations in n variables over a finite field. Moreover, Chang et al. [23] defines a quantum annealing based method to solve general systems of polynomial equations. Its applications for linear regression are demonstrated and an iterative annealing process is defined and proven to be efficient to solve a linear system with a tolerance of 10^{-8}.

1.4 Contribution

This article is the first to focus on quantum state-domain equations and tools devised for the development and implementation of a quantum HHL algorithm for solving state equation systems. Moreover, the proposed method uses less qubits as it suggests the use of an optimized iterative algorithm rather than a matrix relationship. This work represents a significant progress in advanced control theory and should lead to new tools for future state-equations-based developments for quantum computers.

The rest of the document is structured as follows: the basics of quantum mechanics and quantum computing are presented in the second section. The third section recalls the state domain equations and the numerical solution. The fourth section deals with the results of implementing the HHL solution for the state equations along with a discussion of the results. Finally, a conclusion summarizes the research and findings as well as future work.

2. QUANTUM MECHANICS AND COMPUTING

Quantum mechanics is a term widely used but much less understood. It is a mathematical formalism used originally to describe the behavior of the smallest objects known, and by doing so, exposed inadequacies in classical physical theory [24-26]. Quantum theory explains this behavior and thereby proposes a more comprehensive understanding of the nature of the universe. According to quantum mechanics, a fundamental relation defined by the Schrödinger equation presented in relationship (1) governs the dynamic behavior of a particle.

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle \tag{1}$$

The solution of the Schrödinger equation is called the wave function [27], designated ψ. This wave function is defined as being a variable quantity that describes the wave characteristics of a quantum particle and defines the space in which the particle is likely to be found. In general, a given wave function can be expressed as in Eq. (2).

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2}$$

Such as $\sqrt{\alpha^2+\beta^2} = 1$, α and β are complex numbers representing the amplitudes of the state eigenvectors $|0\rangle$ and $|1\rangle$ respectively. Due to the properties of wave quantum particles, complex numbers are used instead of real positive numbers, whereby a particle can be defined as an electron having a specific spin state and spatial orientation, which cannot be described adequately using a single real number.

Finally, it is worth noting that α^2 is the probability of the state $|\psi\rangle$ collapsing to the $|0\rangle$ state.

β^2 is the probability of the state $|\psi\rangle$ collapsing to the $|1\rangle$ state.

The Schrödinger equation is a linear differential equation, meaning that if two wave functions $|\psi_1\rangle$ and $|\psi_2\rangle$ are solutions, the linear combination of both is also a solution, that is, $|\psi_3\rangle = |\psi_1\rangle + |\psi_2\rangle$. This latter property is called superposition. The superposition principle is implicit in the linearity property, insofar as the linear combination of two or more state vectors results in another state vector. When the state $|\psi\rangle$ is in superposition, it simultaneously occupies all possible Eigen state positions, leading to the parallel processing property that lies at the heart of quantum computing power.

Entanglement is another fundamental property of quantum particles. When two particles are entangled, the measurement of the state of one is connected to the state of the other. This connection is stronger than a classical correlation [26]. In fact, the states of the two particles are considered inseparable. Entanglement is useful in computing in a variety of ways, notably for encoding information simultaneously in two different but connected locations.

2.1 Qubit

The transposition of computer bits to their quantum equivalent gives what is known as a qubit. Like bits, qubits are binary entities that exist in the nominal states 0 or 1; however, they can also assume a continuous range of values representing a superposition of states. A qubit is fundamentally a two-level quantum mechanical system. It can be constructed physically in various ways and represented as a two-dimensional complex Hilbert space C^2. A vector in such a Hilbert space can represent the state of the qubit at any given time.

The Hilbert space is a vector space comprising an *inner product*, an operation that allows lengths and angles to be defined, to determine the relative positions of vectors that represent qubit states. The inner product of two vectors $|x\rangle$ and $|y\rangle$ is denoted by $\langle x|y\rangle$.

$|x\rangle$ is known as the ket, which is one of the Dirac notations. If two vectors have the same origin, their inner product will be equal to 0 if they are also orthogonal, and 1 if $|x\rangle = |y\rangle$.

To represent two or more qubits, a tensor product can be used in Hilbert spaces for combined states of the qubits. Methods exist to represent separable states (where qubits are independent of each other) and entangled states (where two-qubit states cannot be separated).

Whereas ordinary bits are always either 0 or 1 at every step during a computation, in quantum computing, this restriction needs to be overcome if the power of quantum rules is to be utilized fully. To achieve this, state is measured only when the output of a quantum computation needs to be extracted. This is because when a measurement is performed, qubits must commit to option $|0\rangle$ or $|1\rangle$. At any other time, the state will remain more complex than can be conveyed by a single binary value; it will be a linear combination of states.

The $|0\rangle$ and $|1\rangle$ states form an orthonormal basis for all other possible states of a qubit since they are both pure exclusive qubit states that do not overlap. The qubit state can be represented using the following 2D vector notation as presented in (3) to (5) [28-30]:

$$|0\rangle = (1 \quad 0)^T \tag{3}$$

$$|1\rangle = (0 \quad 1)^T \tag{4}$$

$$|q\rangle = (\alpha \quad \beta)^T \tag{5}$$

The idea is to represent complex states such as $|q\rangle$ using linear combinations of $|0\rangle$ and $|1\rangle$ as in Eq. (6).

$$|q\rangle = \alpha(1 \quad 0)^T + \beta(0 \quad 1)^T = \alpha|0\rangle + \beta|1\rangle \tag{6}$$

The state vector $|q\rangle$ embeds all the information that could possibly need to be known about the qubit. It is neither entirely $|0\rangle$ nor entirely $|1\rangle$ but rather is described as a linear combination of the two basic states. In quantum mechanics, linear combinations such as this are usually described as superposed. A more comprehensive definition of this state vector representation can be obtained in terms of probabilities.

A qubit state can be represented in the 3D space using the Bloch sphere. With a unit radius, it provides a geometrical representation of a qubit. The north and south poles of the sphere define the orthonormal basis states $|0\rangle$ and $|1\rangle$ respectively, while the surface defines the set of all possible qubit values. In spherical coordinates, this is written as follows:

$$|q\rangle = cos\frac{\theta}{2}|0\rangle + sin\frac{\theta}{2}e^{i\varphi}|1\rangle \tag{7}$$

For various values of the angles θ and φ in expression (7) will result graphically in the Bloch sphere illustrated in Figure 1.

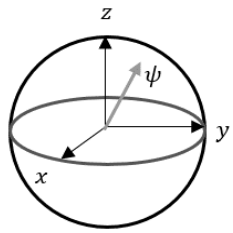


Figure 1. The Bloch sphere representation of the qubit $|q\rangle$ [29]

2.2 Measurement

The concept of measurement needs clarification in probability-based computations. Whereas measurement is straightforward in classical physics, since it presumably does not alter the state of the system, in quantum mechanics, the act of measurement itself has a profound effect on what is observed. To find the probability resulting from measuring a state of the qubit $|q\rangle$ in state $|x\rangle$, the simple rule defined in Eq. (8) is applied:

$$p(|x\rangle) = |\langle x|q\rangle|^2 \tag{8}$$

$\langle x|$ is called the bra notation, which together with the ket notation, forms the Dirac bra-ket. The bra notation is used to represent a row vector, whereas the ket notation is used to represent column vectors. Every ket $|a\rangle$ vector has its corresponding bra $\langle a|$ vector. The conjugate transpose is the transform that links the two vectors. Since $\langle x|$ can represent a possible qubit state, the probability of the qubit whose state vector is $|q\rangle$ having $|0\rangle$ as an output therefore can be calculated as follows:

$$|\langle 0|q\rangle|^2 = |\alpha|^2 \quad (9)$$

Equation or rule (9) says that information is extracted from qubits by calculating probabilities, thus bridging the quantum world and the classical physical world. It is important to note some implications of this rule to develop some intuitive operations in quantum computing. One of these is normalization. A state vector is composed of elements that correspond to probabilities. These elements are called amplitudes. One obvious implication of rule (9) is that the probability of measuring either state $|0\rangle$ or $|1\rangle$ is its amplitude squared. Moreover, the probabilities of all possible output values should add up to 1. Therefore, the magnitude of the state vector needs to be normalized to 1.

2.3 Quantum gates

Qubit unit data need to be processed to convert inputs to outputs for specific tasks. For this purpose, inputs are put through a series of operations called gates, which represent quantum operators that address one or more qubits. By design, the gates are reversible and represented by unitary complex matrices. Pauli gates are three of the best-known operators and have very specific structures. Along with the identity matrix, they form the Pauli Group. Any quantum operator can be expressed using the Pauli gates. The X gate is applied using the following Pauli matrix σ_x as defined in Eq. (10).

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0| \quad (10)$$

Applied to a qubit state, the X gate flips its state. It is equivalent to the classical NOT gate.

The Y gate is is applied using the Pauli matrix σ_y as defined in relationship (11).

$$\sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i|0\rangle\langle 1| + i|1\rangle\langle 0| \quad (11)$$

The Z gate is applied using the Pauli matrix σ_z defined in (12).

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1| \quad (12)$$

The Hadamard or H gate allows shifting from defined qubit states $|0\rangle$ or $|1\rangle$ to a superposition of the two. The corresponding operator is defined by the matrix given in Eq. (13).

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1| \quad (13)$$

Other gates exist, all based on the basic X, Y and Z gates, computed as linear combination of them, such as the Hadamard gate, S, T and P and U gates, in addition to multiple gate operators, such as the controlled NOT, SWAP, Toffoli and controlled SWAP gates.

2.4 Linear systems in quantum computing

Linear systems are widespread throughout the discipline of engineering, where they help the formulation of specific problems by describing the relationships between unknown physical parameters. This formulation can be expressed mathematically as in Eq. (14).

$$A.\vec{x} = \vec{b} \quad (14)$$

In which $A \in C^{N\times N}$ is a matrix that defines the system, $\vec{x} \in C^N$ is the unknown parameter vector, and $\vec{b} \in C^N$ is a known vector. In automation and control, linear systems are used extensively, for example in linear quadratic regulator (LQR) controllers, which require rapid and precise procedures for optimizing performance, or in process result determination by the least-squares method, in which large amounts of input/output data are reduced to a multidimensional A matrix. More reliable optimizations are made possible by designing quantum algorithms to solve linear systems with a time complexity equal to $O\left(\frac{loglog\,(N)\,k^2 s}{\epsilon}\right)$,

where

k is the condition number of the system.

ε is the desired accuracy at the end of the run.

N is the input size.

S is the number of non-zero elements in a column or a row of matrix A.

It is assumed that s is small. On the other hand, the conventional algorithm equivalent to the HHL algorithm has a time complexity equal to $O(N^3)$. Using quantum algorithms, exponential increases in the speed of problem solving can be attained.

It should be kept in mind that a conventional algorithm provides an unambiguous solution, whereas a quantum algorithm provides the expected value of the solution, represented by $\langle\vec{x}\rangle$, in the form of a Hamiltonian matrix M.

2.5 Mathematical formulation of the HHL algorithm

The first step towards solving a linear system using a quantum approach is to encode the problem in quantum language. Vectors $\vec{x}$ and $\vec{b}$ need to be rescaled to quantum format denoted by $|x\rangle$ and $|b\rangle$ respectively, where amplitude b_j represents the value of the jth element of vector $\vec{b}$ and the same applies to the vector $\vec{x}$. After this reshaping of the problem, the focus will be on finding the quantum mechanical solution to an equivalent problem defined by the mathematical formulation (15).

$$A|x\rangle = |b\rangle \quad (15)$$

Matrix A is Hamiltonian and reducible to a sum of eigenvectors scaled by their respective eigenvalues as in Eq. (16).

$$A = \sum_{j=0}^{N-1} \lambda_j |u_j\rangle\langle u_j| \qquad \lambda_j \in R \quad (16)$$

λ_j is the eigenvalue corresponding to eigenvector u_j. Inverted matrix A^{-1} can be defined as in expression (17).

$$A^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1} |u_j\rangle\langle u_j| \quad (17)$$

Similarly, the right-hand term can be written in the A eigenbasis as in Eq. (18).

$$|b\rangle = \sum_{j=0}^{N-1} b_j |u_j\rangle, b_j \in C \tag{18}$$

Relationship (19) defines the readout register state upon exit from the algorithm:

$$|x\rangle = A^{-1}|b\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} b_j |u_j\rangle \tag{19}$$

Three registers initially set to $|0\rangle$ are used. A register denoted n_l is used to store the binary representation of the matrix A eigenvalues. A register denoted n_b is used to store the vector solution.

The Quantum Phase estimation (QPE) algorithm is constructed for the purpose of approximating the eigenvalue of a given eigenvector. Given a unitary $U \in C^{2^m \times 2^m}$ with an eigenvector and an eigenvalue, the role of QPE is to find an approximate value $\tilde{\theta}_j$ as a proxy for the real value θ_j, considering $|0\rangle|\psi\rangle_j$ as the input along with the unitary gate U. This can be sumarrized as in relationship (20).

$$QPE\left(U, |0\rangle|\psi\rangle_j\right) = \left|\tilde{\theta}_j\right\rangle|\psi\rangle_j \tag{20}$$

To use this algorithm in the HHL algorithm, a unitary operator defined as $U = e^{iAt}$ is introduced. After exiting the subroutine, only the eigenvalues of matrix A would remain, since $e^{iAt} = \sum_{j=0}^{N-1} e^{i\lambda_j t}|u_j\rangle\langle u_j|$.

Therefore, the following Eq. (21) stands as valid.

$$QPE\left(e^{iAt}, \sum_{j=0}^{N-1} b_j |0\rangle_{n_l}|u_j\rangle_{n_b}\right) = \sum_{j=0}^{N-1} b_j |\lambda_j\rangle_{n_l}|u_j\rangle_{n_b} \tag{21}$$

3. STATE EQUATIONS AND THEIR NUMERICAL SOLUTION

State domain equations have been extensively used in the literature to solve various engineering control problems [31, 32].

Most dynamic systems can be modeled or approximated using the following state-domain equations:

$$\begin{cases} \dot{x} = Hx + Du \\ y = Cx \end{cases} \tag{22}$$

where

$x \in R^n$ is the state vector.

$u \in R^m$ is the control vector.

$y \in R^p$ is the output vector.

$H \in R^{n\times n}$, $D \in R^{n\times m}$, and $C \in R^{p\times n}$ are the state, the control, and the output matrices, respectively.

System (22) is solved by integrating over time to find x given an initial state $x(0)$ using a method inspired by Berry et al. [4]. In classical computing, finding the solution requires conversion of time to its discrete form $k \triangle t$. The discrete forms of $x(t)$ and $u(t)$ would thus be $x(k \triangle t)$ and $u(k \triangle t)$, which can be denoted by $x(k)$ and $u(k)$, respectively. The Euler approximation of the first derivative of the signal $x(t)$ can be formulated as in Eq. (23). This leads to the formula given in Eq. (24) for computing $x(k+1)$.

$$\dot{x} \approx \frac{1}{\triangle t}\big(x(k+1) - x(k)\big) = H\,x(k) + D\,u(k) \tag{23}$$

$$x(k+1) = (H \times \triangle t + I)\,x(k) + D \times \triangle t\,u(k) \tag{24}$$

If the time goes from 0 to $t = k \triangle t$, then successive iterations of (24) will result in $x(0) = x(t=0)$, $x(1) = (\triangle tH + I)x(0) + \triangle tDu(0)$, $x(2) = (\triangle tH + I)x(1) + \triangle tDu(1)$ …, $x(N) = (\triangle tH + I)x(N-1) + \triangle tDu(N-1)$. This can also be expressed in matrix form, inspired from [3], as follows:

$$\begin{pmatrix} -(I + H\triangle t) & I & 0 & 0 & 0 \\ 0 & -(I + H\triangle t) & I & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & 0 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 & I \\ 0 & 0 & 0 & -(I + H\triangle t) & I \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ \vdots \\ x(N) \end{pmatrix} = D \times \triangle t \begin{pmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ \vdots \\ u(N-1) \end{pmatrix} \tag{25}$$

where, $I \in R^{n\times n}$ is the identity matrix.

N qubits are needed to solve the matrix system (34) and thus solve for x. However, we propose here a method that reduces the number of qubits to two and then uses the HHL algorithm to solve the 2D system of equations. It should be noted that expression (25) comprises a 2x2 matrix used repeatedly, while the remaining elements are zeros. A more compact way of expressing this matrix appears in expression (26), obtained by considering initial conditions to be null, which is the case in most engineering systems.

$$\begin{pmatrix} -(I + H\triangle t) & I \end{pmatrix} \begin{pmatrix} x(k) \\ x(k+1) \end{pmatrix} = D \triangle tu(k) \tag{26}$$

Therefore, only two qubits can be used with repetitive runs over time instead of using N qubits. The core improvement of this technique is that resolving problem (26) is exponentially faster using a quantum computer than with a conventional computer.

Since matrix $(-(I + D\triangle t) \quad I)$ in Eq. (26) is not Hermitian, matrix $\begin{pmatrix} -(I + H\triangle t) & I \\ I & -(I + H\triangle t) \end{pmatrix}$ is used instead in order to avoid errors during the execution of the HHL algorithm, which makes the second side of Eq. (26): $D \triangle t \begin{pmatrix} u(k) \\ u(k+1) \end{pmatrix}$.

4. RESULTS AND DISCUSSION

4.1 Illustrative example

As an illustrative example, the motion of a DC motor is considered. The model of this system can be written in the state domain with an $n = 1$ dimension state Eq. (27).

$$J\frac{d\omega}{dt} = -B_m\omega + \tau_{el} \tag{27}$$

With state vector $x = \omega$, the shaft angular velocity, considered also as system output, and control signal $u = \tau_{el}$, the electromagnetic torque.

J and B_m are respectively the inertia constant and the viscous coefficient constant.

The system parameters are chosen to simplify the computation for illustrative purposes:

$\triangle t$ (i.e., sampling period) considered as 1 s.

B_m/J ratio considered as 2/3.

Discretization by state domain Eq. (27) with initial conditions (null) brings the system to expression (28).

$$\begin{pmatrix} -1/3 & 1 \\ 1 & -1/3 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{28}$$

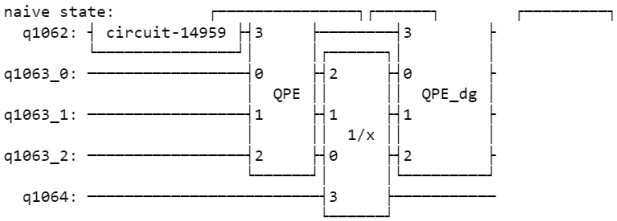
The 4-qubit HHL is developed and run considering matrix A and vector b obtained after replacing the parameters with their numerical values: $A = \begin{pmatrix} -1/3 & 1 \\ 1 & -1/3 \end{pmatrix}$ and $b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

Vector b results from the step response of the system being targeted in this example; in other words, the input to the system is constantly 1.

The DC motor case quantum state equation model was implemented using Qiskit simulator. Figure 2 illustrates a screen shot of the code that was developed and run along with the execution result.

```
import numpy as np
from qiskit.algorithms.linear_solvers.numpy_linear_solver import NumPyLinearSolver
from qiskit.algorithms.linear_solvers.hhl import HHL
from qiskit.quantum_info import Statevector
matrix = np.array([[-1/3, 1],
                   [1, -1/3]])
vector = np.array([1, 1])
naive_hhl_solution = HHL().solve(matrix, vector)
naive_sv = Statevector(naive_hhl_solution.state).data
naive_full_vector = np.array([naive_sv[8], naive_sv[9]])
print('naive state:', naive_hhl_solution.state)
print('naive norm:', naive_hhl_solution.euclidean_norm)
print('naive raw solution vector:', naive_full_vector)
```

(a) Qiskit program



(b) Schematic obtained after one run

Figure 2. Implementing the HHL algorithm for a DC motor (Images obtained using Qiskit [33])

Running this algorithm leads to the following results: Naive norm: 1.5.

For broader analysis and comparison with conventional method results, the DC motor transfer function $F(s)$ was developed and simulated, in Eq. (29). A step input was inserted into the system, giving the response shown in Figure 3.

$$F(s) = \frac{X(s)}{U(s)} = \frac{1}{Js + B_m} \tag{29}$$

$X(s)$ and $U(s)$ are the Laplace transform of the signals $x(t)$ and $u(t)$, respectively.

Figure 3 shows the time response of a first-order system, as a unit electromagnetic torque (1 N.m) leads to an exponential increase in angular velocity w until stabilization at the final value of 1.5 rad/sec.

The quantum simulation gives similar output as the one obtained with Laplace transform, as the final output value stabilizes to 1.5 for both methods.

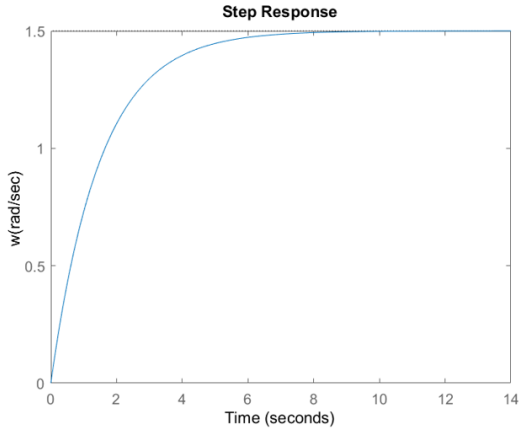


Figure 3. Step response of the DC motor

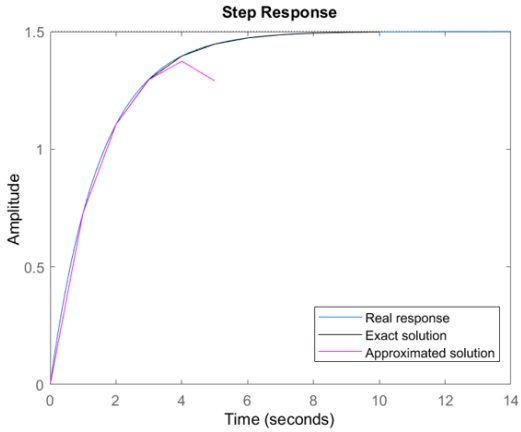


Figure 4. Exact and approximate solutions compared to the system real output

Another alternative would have been to use the exact solution of the problem $x(t) = \frac{3}{2}\left(1 - e^{-\frac{2}{3}t}\right)$ and find its quantum computing version using the power series $x(t) \approx -\frac{3}{2}\left(-\frac{2}{3}t + \frac{1}{2}\left(-\frac{2}{3}t\right)^2 + \frac{1}{3}\left(-\frac{2}{3}t\right)^3 + \frac{1}{3}\left(-\frac{2}{3}t\right)^3 + \cdots\right)$ as suggested in quantum mechanics to approximate the exponential term [15].

The drawback of the power series approach is the high power needed to get closely matching results. In Figure 4, developing to a power of 5 was sufficient to get the first samples to match. However, with increasing time, precision decreases, and the approximation is no longer valid, making more higher-order terms necessary to get close to the real result. Adapting the power series expression at each iteration over time could solve this problem.

The HHL method seems to offer a precise quantum solution to state domain equations for control purposed modelling compared to the exact solution and the quantum classic method involving power series.

5. CONCLUSION

We have examined various aspects of quantum computing, including the quantum mechanics concepts that underlie its power, namely superposition, entanglement, and reversibility, which differentiate quantum computing from classical computing and open the door to new ways of thinking about computations. We summarize the essential elements that make quantum computing possible: qubits, quantum operators, and the bit status measurement process. We compared the performance of quantum algorithms to that of the corresponding classical algorithms. All the concepts introduced were supported in the HHL algorithm.

The quantum HHL algorithm was developed and demonstrated using a theoretical example and the simulation of a DC motor. Running the HHL quantum algorithm yields numerical results like those of the corresponding conventional algorithm, with exponentially higher speed of execution.

The present study was carried out using quantum simulators. Implementation of the algorithm on a real quantum computer can be considered for future work with a real DC motor or even other first-order systems with suitable developed interfaces.

Even though quantum computing might seem extremely advantageous over classical computing, it remains subject to several physical limitations that currently prevent full leveraging of its potential power. While we wait for technological advances to overcome these limitations, theoretical studies will continue to suggest avenues of innovation.

REFERENCES

[1] Hereman, W. (2006). Symbolic computation of conservation laws of nonlinear partial differential equations on multi-dimensions. International Journal of Quantum Chemistry, 106(1): 278-299. http://dx.doi.org/10.1002/qua.20727

[2] Berry, D.W. (2014). High-order quantum algorithm for solving linear differential equations. Journal of Physics A: Mathematical and Theoretical, 17(10): 105301. http://dx.doi.org/10.1088/1751-8113/47/10/105301

[3] Iyer, U.N., McCune, T.C. (2002). Quantum differential operators on K[x]. World Scientific Publ Co Pte Ltd. https://doi.org/10.48550/arXiv.math/0010041

[4] Berry, D.W., Childs, A.M., Ostrander, A., Wang, G. (2017). Quantum algorithm for linear differential equations with exponentially improved dependence on precision. communications in mathematical physics. Communications in Mathematical Physics, 356: 1057-1081. http://dx.doi.org/10.1007/s00220-017-3002-y

[5] Khater, M.M., Seadawy, A.R., Lu, D. (2018). Bifurcations of solitary wave solutions for (two and three)-dimensional nonlinear partial differential equation in quantum and magnetized plasma by using two different methods. Results in Physics, 9: 142-150. http://dx.doi.org/10.1016/j.rinp.2018.02.010

[6] Srivastava, S., Sundararaghavan, V. (2019). Box algorithm for the solution of differential equations on a quantum annealer. Physical Review A, 99: 052355. http://dx.doi.org/10.1103/PhysRevA.99.052355

[7] Childs, A.M., Liu, J.P. (2020). Quantum spectral methods for differential equations. Communications in Mathematical Physics, 375: 1427-1457. http://dx.doi.org/10.1007/s00220-020-03699-z

[8] Kyriienko, O., Paine, A.E., Elfving, V.E. (2021). Solving nonlinear differential equations with differentiable quantum circuits. Physical Review A, 103: 052416. http://dx.doi.org/10.1103/PhysRevA.103.052416

[9] Zanger, B., Mendl, C.B., Schulz, M., Schreiber, M. (2021). Quantum algorithms for solving ordinary differential equations via classical integration methods. Quantum, 5: 502. http://dx.doi.org/10.22331/q-2021-07-13-502

[10] Kumar, N., Shaikh, A.A., Mahato, S.K., Bhunia, A.K. (2021). Applications of new hybrid algorithm based on advanced cuckoo search and adaptive Gaussian quantum behaved particle swarm optimization in solving ordinary differential equations. Expert Systems with Applications, 175: 114646. http://dx.doi.org/10.1016/j.eswa.2021.114646

[11] Lindsay, J.M., Skalski, A.G. (2007). On quantum stochastic differential equations. Journal of Mathematical Analysis and Applications, 330(2): 1093-1114. http://dx.doi.org/10.1016/j.jmaa.2006.07.105

[12] Vissers, G., Bouten, L. (2019). Implementing quantum stochastic differential equations on a quantum computer. Quantum Information Processing, 18: 152. http://dx.doi.org/10.1007/s11128-019-2272-z

[13] An, D., Linden, N., Liu, J.P., Montanaro, A., Shao, C., Wang, J. (2021). Quantum-accelerated multilevel Monte Carlo methods for stochastic differential equations in mathematical finance. Quantum, 5: 481. http://dx.doi.org/10.22331/q-2021-06-24-481

[14] Cao, Y., Papageorgiou, A., Petras, I., Traub, J., Kais, S. (2013). Quantum algorithm and circuit design solving the Poisson equation. New Journal of Physics, 15: 013021. http://dx.doi.org/10.1088/1367-2630/15/1/013021

[15] Arrazola, J. M., Kalajdzievski, T., Weedbrook, C., Lloyd, S. (2019). Quantum algorithm for nonhomogeneous linear partial differential equations. Physical Review A, 100: 032306. http://dx.doi.org/10.1103/PhysRevA.100.032306

[16] Wanga, S., Wanga, Z., Lia, W., Fana, L., Weib, Z., Gu, Y. (2020). Quantum fast Poisson solver: The algorithm and complete and modular circuit design. Quantum Information Processing, 19(6): 170. http://dx.doi.org/10.1007/s11128-020-02669-7

[17] Harrow, A.W., Hassidim, A., Lloyd, S. (2009). Quantum algorithm for linear systems of equations. Physical Review Letters, 103: 150502. https://doi.org/10.1103/PhysRevLett.103.150502

[18] Cai, X.D., Weedbrook, C., Su, Z.E., et al. (2013). Experimental quantum computing to solve systems of linear equations. Physical Review Letters, 110: 230501. http://dx.doi.org/10.1103/PhysRevLett.110.230501

[19] Lee, Y., Joo, J., Lee, S. (2019). Hybrid quantum linear equation algorithm and its experimental test on IBM quantum experience. Scientific Reports, 9: 4778. http://dx.doi.org/10.1038/s41598-019-41324-9

[20] Sellier, J., Dimov, I. (2016). On a quantum algorithm for the resolution of systems of linear equations. Recent Advances in Computational Optimization, pp. 37-53. http://dx.doi.org/10.1007/978-3-319-21133-6_3

[21] Li, K., Dai, H., Jing, F., Gao, M., Xue, B., Wang, P., Zhang, M. (2021). Quantum algorithms for solving linear regression equation. Journal of Physics: Conference Series, 1738: 012063. http://dx.doi.org/10.1088/1742-

6596/1738/1/012063
[22] Bernstein, D.J., Yang, B.Y. (2018). Asymptotically faster quantum algorithms to solve multivariate quadratic equations. In International Conference on Post-Quantum Cryptography, Fort Lauderdale, FL, USA, pp. 487-506. https://doi.org/10.1007/978-3-319-79063-3_23
[23] Chang, C.C., Gambhir, A., Humble, T.S., Sota, S. (2019). Quantum annealing for systems of polynomial equations. Scientific Reports, 9(1): 10258. https://doi.org/10.1038/s41598-019-46729-0
[24] Griffiths, R.B. (2003). Consistent Quantum Theory. Cambridge University Press.
[25] McMahon, D. (2007). Quantum Computing Explained. John Wiley & Sons.
[26] Adedoyin, A., Ambrosiano, J., Anisimov, P., et al. (2018). Quantum algorithm implementations for beginners. arXiv preprint arXiv:1804.03719.
[27] Encyclopedia Britannica. (2021). https://www.britannica.com/science/wave-function
[28] Zioui, N., Mahmoudi, Y., Mahmoudi, A., Tadjine, M., Bentouba, S. (2021). A novel quantum-computing-based quaternions model for a robotic arm position. International Journal of Computational Intelligence in Control, 13(2): 71-77.
[29] Zioui, N., Mahmoudi, Y., Mahmoudi, A., Tadjine, M., Bentouba, S. (2021). A new quantum-computing-based algorithm for robotic arms and rigid bodies' orientation. Journal of Applied and Computational Mechanics, 7(3): 1836-1846. https://dx.doi.org/10.22055/jacm.2021.37611.3048
[30] Fazilat, M., Zioui, N., St-Arnaud, J. (2022). A novel quantum model of forward kinematics based on quaternion/Pauli gate equivalence: Application to a six-jointed industrial robotic arm. Results in Engineering, 14: 100402. https://doi.org/10.1016/j.rineng.2022.100402
[31] Alawad, N.A., Humaidi, A.J., Al-Araji, A.S. (2022) Improved active disturbance rejection control for the knee joint motion model. Mathematical Modelling of Engineering Problems, 9(2): 477-483. https://doi.org/10.18280/mmep.090225
[32] Yalavarthy, U.R.S., Gadi, V.S.K.R. (2022). Modelling, simulation and analysis of indirect space vector control of electric vehicle driven by permanent magnet synchronous motor with fuzzy controller. Mathematical Modelling of Engineering Problems, 9(2): 523-532. https://doi.org/10.18280/mmep.090231
[33] Al-Khazraji, H. (2022). Optimal design of a proportional-derivative state feedback controller based on meta-heuristic optimization for a quarter car suspension system. Mathematical Modelling of Engineering Problems, 9(2): 437-442. https://doi.org/10.18280/mmep.090219

NOMENCLATURE

HHL	Harrow, Hassidim, and Lloyd algorithm for solving systems of equations
SWAP	SWAP gate
LQR	Linear Quadratic Regulator
QPE	Quantum Phase Estimation
$\|0\rangle$	Eigenstate in quantum computing, equivalent to the classical 0 state
$\|1\rangle$	Eigenstate in quantum computing, equivalent to the classical 1 state
$\hbar$	Reduced Planck constant
$\widehat{H}$	Hamiltonian operator
$x \in R^n$	State domain vector
$u \in R^m$	Control vector
$y \in R^p$	Output vector
R	Set of Real numbers
$H \in R^{n \times n}$	State matrix
$D \in R^{n \times m}$	Control matrix
$C \in R^{p \times n}$	Output matrix
$\triangle t$	Sampling time
I	Identity matrix
N	Number of samples
J	Inertia constant of a DC motor
B_m	Viscous friction coefficient
τ_{el}	Electromagnetic torque

Greek symbols

α	Complex coefficient whose squared magnitude represents the probability of the qubit collapsing to state $\|0\rangle$
β	Complex coefficient whose squared magnitude represents the probability of the qubit collapsing to state $\|1\rangle$
λ	Eigen value of a matrix
σ_x, σ_y and σ_z	Pauli matrices
ψ	Wave function in quantum mechanics
$\|\psi\rangle$	Qubit state