A Quantum Computer Amenable Matrix Equation Solver with Applications to Computational Electromagnetics

Christopher D. Phillips*, Student Member, IEEE and Vladimir I. Okhotavski*, Member, IEEE
*Department of Electrical and Computer Engineering, University of Manitoba, Winnipeg, MB R3T 2N2, Canada

Abstract—Quantum computation offers a promising alternative to classical computing methods in many areas of numerical science. Algorithms that make use of the unique way in which quantum computers store and operate on data often feature dramatic improvements in performance over their classical counterparts. In this work, we study problems involving the solution of matrix equations, for which there currently exists no efficient, general quantum procedure. We present a generalization of the Harrow/Hassidim/Lloyd algorithm by providing an alternative unitary for eigenphase estimation. This unitary, which we have adopted from research into quantum walks, has the advantage of being well defined for any matrix equation. This allows the matrix solution procedure to be directly implemented on quantum hardware for any well-conditioned matrix equation. To analyze the performance of this procedure, we use both classical simulators and genuine quantum systems, and we consider a practical method of moments problem from computational electromagnetics.

Index Terms—Harrow/Hassidim/Lloyd (HHL), matrix methods, numerical analysis, quantum computing, simulation.

I. INTRODUCTION

Matrix equation formulations are a powerful tool for the numerical analysis of physical systems, and their usage is widespread across many areas of engineering [1]–[3]. Here we limit our study, when necessary, to the field of computational electromagnetics. Unfortunately, the solution of practical matrix equations often demands an exorbitant supply of computational resources, with many present-day simulation problems already stressing the limits of available computing systems. For a matrix equation involving $N$ unknowns, the most sophisticated algorithms of the current day provide $O\left(N\right)$ or $O\left(N\log N\right)$ runtime and memory complexity [4]–[6], depending on whether certain restrictions can be applied to the problem. Thus, as simulations naturally grow more complex with time, the requisite computational power must grow at least proportionately.

For such computationally intensive problems, quantum computers are growing in popularity as a potential remedy, as they are capable of leveraging procedures of dramatically reduced complexity relative to their classical counterparts [7], [8], [11]. This is chiefly due to the manner in which quantum computers store and operate on data. Values of interest are stored as the amplitudes of possible states for a register of qubits, meaning that a 1-qubit system (using a 2-state qubit) can store two values using the amplitudes of the $|0\rangle$ and $|1\rangle$ states; a 2-qubit system can store four values using the $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ states; and so on. In general, the storage capacity of a quantum computer scales exponentially with the size of the system. Quantum operators then act directly on the system’s qubits, allowing the effects of operators to also scale exponentially with system size.

On the topic of matrix equation solvers, the Harrow/Hassidim/Lloyd (HHL) algorithm [9] has become well established as the leading quantum algorithm. For a system of condition number $\kappa$, this algorithm is capable of providing computational complexity $O\left(\kappa^2 \log \left(N\right) / \epsilon\right)$ in the calculation of a solution of precision $\epsilon$. Thus, when $\kappa$ and $1/\epsilon$ scale logarithmically with $N$, the procedure offers an exponential speedup over classical algorithms.

A detailed description of the HHL algorithm is given in section [11] but we provide a brief overview of the procedure here for the sake of motivation. In what follows, note that the notation $|\psi\rangle$ indicates a vector $\tilde{\psi}$ with its components encoded onto the basis states of a quantum system as $|\psi\rangle = \sum_{j=0}^{N-1} \psi_j |j\rangle$, where $\tilde{\psi}$ has $N$ components. We typically suppress normalization factors where their meaning is irrelevant. For simplicity, $|\psi\rangle$ is understood to also refer to the vector $\tilde{\psi}$ itself. We begin with a matrix equation $A |x\rangle = |b\rangle$, where $A$ is an $N \times N$ matrix, and $|x\rangle$ and $|b\rangle$ are $N$-dimensional vectors. If a quantum system is initialized to the state $|b\rangle$, then its state will be a superposition of the eigenvectors of $A$:

$$|b\rangle = \sum_{j=0}^{N-1} \beta_j |u_j\rangle,$$

(1)

where $|u_j\rangle$ is an eigenvector of $A$, and $\beta_j$ is the component of $|b\rangle$ along $|u_j\rangle$. The solution vector $|x\rangle$ can then be obtained by applying the inverse of each eigenvalue $\lambda_j$ of $A$ to its corresponding eigenvector in the superposition $|b\rangle$:

$$\sum_{j=0}^{N-1} \frac{\beta_j}{\lambda_j} |u_j\rangle = A^{-1} |b\rangle = |x\rangle.$$

(2)

This application of inverse eigenvalues, conditioned on the presence of the corresponding eigenvectors, is the basic function of HHL. By using quantum phase estimation (QPE) [10], it is possible to extract and apply these eigenvalue factors in an efficient manner. However, this phase estimation requires the implementation of a problem-dependent unitary in terms of basic quantum gates. Generally the unitary $e^{iA}$ is employed for this purpose, but the process of decomposing this unitary into a sequence of basic quantum gates is nontrivial. Therefore
the process of implementing this operation on real quantum hardware presents a bottleneck for the procedure. In this work we consider a particular method, inspired by research into quantum walks, which produces a unitary with a known decomposition into basic gates.

Quantum walks are a popular area of study in quantum computing, as they have shown success in aiding the construction of efficient algorithms [15]. Reference [16] gives an excellent introduction to the concept of quantum walks. In essence, they are stochastic processes that, when evolved and measured, give information about the system in which they have evolved. The structure of the evolution and the properties measured can be adjusted to change the information obtained. We are interested in the walk procedure developed in [17], which considers the problem of element-distinctness. This problem is quite far removed from our current study, but it is important for the fact that it introduces a particular walk operator based on Grover diffusion operators. Following this research, [18] further investigated diffusion-based quantum walks defined by probabilistic maps, and considered the spectra of the walk operators. Finally, [11] expanded on these findings from the perspective of Hamiltonian simulation, and showed that the walk operators could be defined from a Hamiltonian to produce an operator with a spectrum closely related to that of the Hamiltonian itself. Reference [12] explored methods by which these walk operators could be implemented in practice.

The method developed in [11] uses QPE to extract the eigenvalues of the walk operator, which can then be related to the eigenvalues of the Hamiltonian. This is the critical component from which we can develop a general procedure for solving matrix equations. By treating a Hermitian system matrix as the Hamiltonian, we can apply QPE using the walk operator to extract the eigenvalues of the system matrix. From there, the HHL algorithm can proceed as usual.

Using the Qiskit SDK [13], we have developed a program [14] which closely follows the herein described procedure. In the interest of expositional clarity, as well as to aid those seeking to investigate the program itself, we at times make reference to the specific functions of this program. At these points, the program is referred to simply as “the program”.

II. THE HHL ALGORITHM

Here we describe the basic HHL algorithm which serves as a foundation for our matrix solution procedure. For brevity, we study only a primitive form of the HHL algorithm. More sophisticated formulations can be used to improve performance by, for example, ignoring ill-conditioned portions of the system matrix.

A. Initialization

Given a matrix \( A \in \mathbb{C}^{N \times N} \) and a vector \( |b\rangle \in \mathbb{C}^N \), the HHL algorithm is a quantum procedure to determine \( |x\rangle \) such that

\[
A \ |x\rangle = |b\rangle .
\] (3)

Due to the nature of quantum computation, several restrictive properties are required of this system. It is important to note, however, that all of these properties can be satisfied for any input system by means of a simple preparation procedure. Section V considers such preparation in depth. The restrictions are as follows: The vector \( |b\rangle \) is to be encoded onto a quantum system, and as such we require that \( N \) be a power of two and that \( |b\rangle \) be normalized. Additionally, we require that \( A \) is Hermitian. This ensures that its eigenvectors form a basis for \( \mathbb{C}^N \) [20], and also aids us in the definition of the QPE unitary.

The first step of the HHL procedure is the preparation of three registers:

\[
|b\rangle |0\rangle^{\otimes n_p} |0\rangle .
\] (4)

The first register, which we refer to as the vector register, is initialized to \( |b\rangle \), and hence it requires \( n = \log_2 N \) qubits. The second register is an \( n_p \)-qubit phase register that is used to store the phase estimates produced by QPE. The value of \( n_p \) is chosen by the user, and it should be large enough to admit sufficient resolution in the phase estimation. The final register is termed the ancilla register, and it consists of a single qubit. Once the eigenvalues of \( A \) have been estimated, the state of this register can be rotated to effect the application of the inverse eigenvalue factors.

The initial system state can be expressed in terms of the eigenvectors of \( A \) as

\[
|b\rangle |0\rangle^{\otimes n_p} |0\rangle = \sum_{j=0}^{N-1} \beta_j |u_j\rangle |0\rangle^{\otimes n_p} |0\rangle ,
\] (5)

where \( |u_j\rangle \) is the \( j \)-th eigenvector of \( A \), and \( \beta_j \) is the complex amplitude of \( |b\rangle \) along \( |u_j\rangle \). The next step of the procedure is to apply QPE to this superposition of eigenvectors to determine the corresponding eigenphases.

B. Quantum Phase Estimation

Quantum phase estimation is a procedure for the estimation of the eigenphases of a unitary. It begins with the observation Fig. 1: Diagram of quantum phase estimation. To emphasize the action of the eigenphase being read onto the phase register \( |p\rangle \), the vector register is shown to have the state of a single eigenvector. In general, it will be prepared in a superposition of all eigenvectors of \( U \). In the state \( |\phi_{j,a}\rangle \), the second subscript \( a \) indicates the \( a \)-th bit of the eigenphase \( \phi_j \) corresponding to \( |u_j\rangle \). This final state assumes \( \phi_j \) can be represented exactly by \( n_p \) bits. When \( \phi_j \) cannot be represented thusly, the final state of the phase register will have probabilities which spike sharply in the vicinity of the most accurate approximation to \( \phi_j \).
that the eigenvalues of unitary operators lie on the complex unit circle, and hence they can be expressed in the form

$$\mu_j = e^{2\pi i \phi_j},$$

(6)

where $\mu_j$ is an eigenvalue of some unitary $U$, and $\phi_j \in [0, 1)$ is termed the corresponding eigenphase. The fundamental action of QPE is to use one register, initialized to an eigenvector $|u_j\rangle$, of $U$, to write a binary representation of the corresponding eigenphase onto an external phase register:

$$|u_j\rangle |0\rangle^{\otimes n_p} \xrightarrow{\text{QPE}} |u_j\rangle |\hat{\phi}_j\rangle.$$  

(7)

Here $\hat{\phi}_j$ represents an $n_p$-bit approximation of $\phi_j$. If $\phi_j$ can be exactly represented by $n_p$ bits, then the phase register will have the final state $|\hat{\phi}_j\rangle$ exactly. Otherwise, the state of the phase register will have maximal amplitude at the best $n_p$-bit approximation of $\phi_j$, with the amplitudes of its other basis states decaying rapidly in the vicinity of this best approximation. We do not explore the specifics of the QPE procedure here, but a summary diagram is given in Fig. 1.

Since QPE is itself a linear operator, its effect when applied to a superposition of eigenvectors of $U$ is as follows:

$$\sum_{j=0}^{N-1} \beta_j |u_j\rangle |0\rangle^{\otimes n_p} \xrightarrow{\text{QPE}} \sum_{j=0}^{N-1} \beta_j |u_j\rangle |\hat{\phi}_j\rangle.$$  

(8)

That is, the approximated eigenphase states appear on the phase register entangled with their corresponding eigenvectors. Fig. 2 provides an illustration of this behavior.

In the case of our current system, we require a unitary with eigenvalues and eigenvectors which can be easily related to those of $A$. For the purposes of the HHL algorithm, the QPE unitary is typically defined as the exponential of the system matrix $[21]$:

$$U = e^{iAt},$$

(9)

where $t$ is some constant. This operator is useful due to the close relationships between its eigenvalues and eigenvectors, and those of $A$. However, it is interesting to note that this operator also corresponds to the simulation of a physical system subjected to a process described by the Hamiltonian $A$. The evolution of such a system is given by Schrödinger’s equation:

$$i\hbar \frac{\partial |\psi\rangle}{\partial t} = A |\psi\rangle.$$  

(10)

When evolved from an initial state $|\psi(0)\rangle$ for a time $t$, the system’s final state is given by

$$|\psi(t)\rangle = e^{-iAt/\hbar} |\psi(0)\rangle.$$  

(11)

Thus, the operator in (9) can be considered to be a time-evolution operator for the Hamiltonian $A$, up to the factor of $-1/\hbar$.

Different values of $t$, and even combinations of different values, can be used to improve the performance of the calculation, but from this point on we will use $t = 1$ for notational convenience. Then the unitary $U = e^{iA}$ has eigenvalues $e^{i\lambda_j}$, and hence the eigenvalues of $A$ are related to the eigenphases of $U$ by

$$e^{i\lambda_j} = e^{2\pi i \phi_j} \implies \lambda_j = 2\pi \phi_j + 2\pi l,$$

(12)

for some integer $l$. Recall that $\phi_j$ is on the range $[0, 1)$. If we assume that $A$ has eigenvalues on the range $[-\pi, \pi]$ (this can be achieved in general by a simple rescaling of the system matrix) then we see that $\phi_j \in [0, \frac{2}{3})$ correspond to $\lambda_j \in [0, \pi]$, and $\phi_j \in (\frac{2}{3}, 1)$ correspond to $\lambda_j \in (-\pi, 0)$. Thus, $\lambda_j$ can be computed as

$$\lambda_j = \begin{cases} 2\pi \phi_j, & \phi_j \in [0, \frac{2}{3}] \\ 2\pi (\phi_j - 1), & \phi_j \in (\frac{2}{3}, 1). \end{cases}$$

(13)

Of course, we do not compute the eigenphases exactly. Instead, we transform the phase register to some superposition where the amplitude of a basis state $|k\rangle$ grows large when $k/N_p \approx \phi_j$. Here we have defined $N_p = 2^{n_p}$. Then we in fact need to calculate potential approximations of the eigenvalues as

$$\lambda_k = \begin{cases} \frac{2\pi k}{N_p}, & k \leq \frac{N_p}{2} \\ 2\pi \left(\frac{N_p}{2p} - 1\right), & k > \frac{N_p}{2}. \end{cases}$$

(14)

From the above descriptions, we see that the effect of QPE on the current system state is

$$\sum_{j=0}^{N-1} \beta_j |u_j\rangle |0\rangle^{\otimes n_p} \xrightarrow{\text{QPE}} \sum_{j=0}^{N-1} \sum_{k=0}^{N_p-1} \beta_j \rho_{jk} |u_j\rangle |k\rangle |0\rangle,$$

(15)

where $\rho_{jk}$ is the amplitude of the phase state $|k\rangle$ resulting from the action of QPE on eigenvector $|u_j\rangle$. Thus, $\rho_{jk}$ should spike sharply when $|k\rangle$ nears an accurate approximation of the
j-th eigenphase of $U$, and remain near zero otherwise. With the eigenvalue extraction procedure defined, we can apply the third step of the procedure: ancilla rotation.

C. Ancilla Rotation and Uncomputation

In this step, the ancilla register is used to impose the inverse eigenvalue factors on the system. We use the rotation operator

$$R_y \left( 2 \arccos \left( \frac{C}{\lambda_k} \right) \right) |0\rangle = \frac{C}{\lambda_k} |0\rangle + \sqrt{1 - \frac{C^2}{\lambda_k^2}} |1\rangle,$$  \hspace{1cm} (16)

where $C$ is a constant chosen to ensure that all arguments of the arcsin function obey its domain restrictions. We recommend $C = 2\pi/N_p$, as this corresponds to the minimum possible magnitude of the denominator $\lambda_k$ in [14], excepting $k = 0$. The $k = 0$ case should be explicitly omitted from the calculation, as a nontrivial contribution from this state would indicate an eigenvalue near zero, and therefore an ill-conditioned matrix. By using a controlled version of this rotation operator, with the phase register acting as the control register, it is possible to entangle the $\lambda_k$ rotation with the $|k\rangle$ state of the phase register. Applying this controlled rotation to the ancilla register for all $k \in \{1, \ldots, N_p - 1\}$, the system is transformed to the state

$$\sum_{j=0}^{N-1} \sum_{k=1}^{N_p-1} \beta_j \rho_{jk} |u_j\rangle |k\rangle \left( \frac{C}{\lambda_k} |0\rangle + \sqrt{1 - \frac{C^2}{\lambda_k^2}} |1\rangle \right),$$  \hspace{1cm} (17)

where we have omitted the $k = 0$ case on the assumption that $\rho_{j0} \approx 0$.

Note that the state (17) is approximately the following ideal state:

$$\sum_{j=0}^{N-1} \sum_{k=0}^{N_p-1} \beta_j \rho_{jk} |u_j\rangle |k\rangle \left( \frac{C}{\lambda_j} |0\rangle + \sqrt{1 - \frac{C^2}{\lambda_j^2}} |1\rangle \right).$$  \hspace{1cm} (18)

This is true since $\rho_{jk}$ spikes sharply when $\lambda_k$ approaches $\lambda_j$, with the spike narrowing as more accurate approximations become available. With the knowledge that the QPE error is obscured by this step, we can see from this that the system is in state (18) exactly. Acting on this state with an inverse QPE will then “uncompute” the phase register, leaving us in the state

$$\sum_{j=0}^{N-1} \beta_j |u_j\rangle |0\rangle^{\otimes n_p} \left( \frac{C}{\lambda_j} |0\rangle + \sqrt{1 - \frac{C^2}{\lambda_j^2}} |1\rangle \right).$$  \hspace{1cm} (19)

This is the final state of the system, from which our solution can be extracted. Selecting those states with $|0\rangle$ in all qubits of the phase register and the ancilla register, we have

$$C \sum_{j=0}^{N-1} \beta_j |u_j\rangle.$$  \hspace{1cm} (20)

This state, when divided by $C$, is exactly the decomposition of $A^{-1} |b\rangle$ in terms of the eigenvectors of $A$, and hence (20) gives our final solution $|x\rangle$.

III. The Walk Operator

A walk operator is a unitary operator that is used during a quantum walk procedure, with each application of the operator representing a “step” made by the system. As mentioned above, we do not here describe a quantum walk procedure per se. Rather, we simply adopt a walk operator from a formerly studied quantum walk procedure [17, 18]. The operator chosen is useful because its implementation is well defined for any arbitrary system matrix.

The walk operator itself has several constituents which must be defined before the operator can be stated succinctly. To begin, we define the vectors

$$|\phi_j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle \sqrt{\frac{N}{X}} A_{jk}^*,$$  \hspace{1cm} (21)

for $j \in \{0, \ldots, N - 1\}$. Here $X$ is a constant which must satisfy $X \geq N |A_{jk}^\max|$, where $|A_{jk}|^\max = \max_{j,k} (|A_{jk}|)$. This factor is required for the purposes of ancilla rotation, as discussed in section IV. The $|\phi_j\rangle$ vectors are useful to us as carriers of information about the system matrix, and they form foundational elements of the walk operator. However, they are possessed of some critical shortcomings. First, they are not normalized. While they could be explicitly normalized through appropriate j-dependent factors, subsequent derivations serve to show that this scheme would destroy crucial operator properties. In particular, the result of (21) would no longer hold. The second concern regarding the $|\phi_j\rangle$ vectors is their preparation. Producing unitary operators to generate these states is a nontrivial problem, and even once computed, these operators would also carry j-dependent normalization factors.

Both of these shortcomings can be remedied with relative ease by introducing an ancillary qubit and studying the set of augmented states

$$|\phi_j^a\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle \left( \sqrt{\frac{N}{X}} A_{jk}^* |0\rangle + \sqrt{1 - \frac{N}{X}} |A_{jk}| |1\rangle \right).$$  \hspace{1cm} (22)

These states are normalized, and they can be easily produced through unitary transformations (see section IV). Note that the $|\phi_j\rangle$ states appear when the ancilla is in the $|0\rangle$ state. Hence, the $|1\rangle$ state of the ancilla can be interpreted as a failure state.

Storage of each $|\phi_j^a\rangle$ state requires an $(n + 1)$-qubit register. We define two such registers, and refer to them by the names $|r_1\rangle$ and $|r_2\rangle$. The ancilla-augmented basis states for these registers are indicated by a superscript $a$, e.g. $|j^a\rangle$. Note that there are $2^{n+1} = 2N$ augmented basis states. When necessary, we always assume that the ancillary qubit is appended to the relevant register. In this procedure, there are two basic operators which act on these registers. The first is the swap operator, which swaps the content of $|r_1\rangle$ and $|r_2\rangle$:

$$S = \sum_{j=0}^{2N-1} \sum_{k=0}^{2N-1} |k^a\rangle |j^a\rangle \langle k^a| \langle j^a|.$$  \hspace{1cm} (23)
The second operator is that which produces $|\phi^a_j\rangle$ on $|r_2\rangle$ when $|r_1\rangle$ is in the state $|j\rangle$ with ancilla state $|0\rangle$:

$$T = \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right).$$

(24)

Here the $|\zeta^a_j\rangle$ states correspond to failure states. A forthcoming analysis of the $T^\dagger ST$ operator shows that they must have ancilla state $|1\rangle$, but otherwise their definition is insignificant. These are proper quantum states, and hence they must be normalized.

These prerequisites are sufficient for us to state the walk operator:

$$W = iS (2TT^\dagger - I).$$

(25)

The process of implementing $W$ is further explored in section IV. For the remainder of this section, we take it as given that $W$ can be implemented. As stated above, this walk operator is particularly interesting to us because it has eigenvalues and eigenvectors closely related to those of $A$ itself. In deriving these relationships between $W$ and $A$, we first calculate some useful relationships for the constituent operators $T$ and $S$. To begin, consider the product $T^\dagger T$:

$$T^\dagger T = \left[ \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right) \right]$$

$$\cdot \left[ \sum_{k=0}^{N-1} \left( |k, 0\rangle \langle k, 0| + |k, 1\rangle \langle k, 1| \right) \right]$$

$$= \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \left( |j, 0\rangle \langle j, 0| \langle k, 0\rangle \langle k, 0| + |j, 1\rangle \langle j, 1| \langle k, 0\rangle \langle k, 0| \right)$$

$$+ |j, 1\rangle \langle j, 1| \langle k, 1\rangle \langle k, 1|.$$ 

(26)

Recalling that the $|\phi^a_j\rangle$ and $|\zeta^a_j\rangle$ states are normalized and that the basis states are orthonormal, we have

$$T^\dagger T = \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right) = I.$$ 

(27)

Note that despite the above, $TT^\dagger$ does not yield the identity operator, and hence $T$ is not a unitary transformation as currently stated. The second relationship of interest is the following:

$$T^\dagger ST = \left[ \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right) \right]$$

$$\cdot \left[ \sum_{k=0}^{N-1} \left( |k, 0\rangle \langle k, 0| + |k, 1\rangle \langle k, 1| \right) \right]$$

$$= \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \left( |j, 0\rangle \langle j, 0| \langle k, 0\rangle \langle k, 0| + |j, 1\rangle \langle j, 1| \langle k, 0\rangle \langle k, 0| \right)$$

$$+ |j, 1\rangle \langle j, 1| \langle k, 1\rangle \langle k, 1|.$$

(28)

If this operator acts on an arbitrary input vector $|\psi, 0\rangle$, we have

$$T^\dagger ST |\psi, 0\rangle$$

$$= \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \left( |j, 0\rangle \langle j, 0| \langle k, 0\rangle \langle k, 0| + |j, 1\rangle \langle j, 1| \langle k, 0\rangle \langle k, 0| \right)$$

$$+ |j, 1\rangle \langle j, 1| \langle k, 1\rangle \langle k, 1|.$$ 

(29)

This elimination of the $|\zeta^a_j\rangle$ contributions is a crucial result, and hence we require that these states have ancilla state $|1\rangle$. The inner products are

$$\langle \phi^a_k|, k\rangle$$

$$= \left[ \frac{1}{\sqrt{N}} \sum_{p=0}^{N-1} \left( \sqrt{\frac{X}{A_{jp}}} \right) \langle 0| \right.$$

$$+ \left. \sqrt{1 - \frac{N}{X}} |A_{jp}| \langle 1| \right) \langle p| \right] |k\rangle |0\rangle$$

$$= \sqrt{\frac{A_{kj}}{X}}.$$ 

(30)

Then we have

$$T^\dagger ST |\psi, 0\rangle$$

$$= \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} |j, 0\rangle \sqrt{\frac{A_{kj}}{X}} \left( \sqrt{\frac{A_{kj}}{X}} \right)^* |k, 0\rangle |0\rangle.$$ 

(33)

The statement that $\sqrt{A_{kj}} \left( \sqrt{A_{kj}} \right)^* = A_{kj}$ is true, but it presents an issue for implementation when $A_{kj}$ lies on the negative real axis. This problem is further discussed in section IV. By the above, we see that $T^\dagger ST$, when applied to a vector with ancilla state $|0\rangle$, acts as a rescaled version of $A$. We introduce the shorthand

$$\hat{A} = \frac{1}{X} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} A_{jk} |j, 0\rangle \langle k, 0|.$$ 

(34)
This transformation is a $2N \times 2N$ matrix, and it can therefore be applied directly to the $2N$-dimensional augmented states. This simplifies the statement of (33):

$$T^1 ST |\psi, 0\rangle = \hat{A} |\psi, 0\rangle.$$  \hspace{1cm} (35)

Now we can begin the analysis of the walk operator’s eigenvalues and eigenvectors. We begin with the assertion that the eigenvalues of $A$ must lie on the range $[-1, 1]$. By the definition of $X$, this is in fact always satisfied regardless of the choice of $A$, as we show in section 11. With this established, consider now the effect of applying $W$ to the vector $T |u_j^0\rangle = T |u_j, 0\rangle$ (recall that $|u_j\rangle$ is an eigenvector of $A$):

$$WT |u_j^0\rangle = iST |u_j^0\rangle.$$  \hspace{1cm} (36)

Note that we have continued to assume that the input vector is provided with ancilla state $|0\rangle$. Meanwhile, if $W$ is applied to $ST |u_j^0\rangle$, we obtain

$$WST |u_j^0\rangle = 2i\lambda_j ST |u_j^0\rangle - iT |u_j^0\rangle,$$ \hspace{1cm} (37)

where $\lambda_j = \lambda_j/X$ is the eigenvalue of $\hat{A}$ corresponding to $|u_j^0\rangle$. Thus, if $W$ is applied to any superposition of $T |u_j^0\rangle$ and $ST |u_j^0\rangle$, the result will be a simple superposition of the same two vectors:

$$W (T |u_j^0\rangle + \gamma ST |u_j^0\rangle) = -i\gamma T |u_j^0\rangle + i \left( 1 + 2\lambda_j \gamma \right) ST |u_j^0\rangle,$$  \hspace{1cm} (38)

where $\gamma$ is an arbitrary factor for the relative phase and magnitude of the two contributions. Then $(T |u_j^0\rangle + \gamma ST |u_j^0\rangle)$ is an eigenvector $|v_j^0\rangle$ of $W$, with eigenvalue $\mu_j$, if $\gamma = i\mu_j$ and $i\mu_j^2 = i \left( 1 + 2i\lambda_j \mu_j \right)$. Solving this expression for the eigenvalues, we find two solutions:

$$\mu_j^\pm = i\lambda_j \pm \sqrt{1 - \lambda_j^2}. \hspace{1cm} (39)$$

The eigenvectors $|v_j^0\rangle$ corresponding to these eigenvalues can be normalized by computing the inner products of their un-normalized counterparts:

$$\langle v_j^0 | v_j^0 \rangle = \left( |u_j^0\rangle T^{-1} - i (\mu_j^0)^* | u_j^0\rangle T^1 S^T \right) (T |u_j^0\rangle + i\mu_j^0 ST |u_j^0\rangle)$$

$$= |u_j^0\rangle T^{-1} T |u_j^0\rangle + i\mu_j^0 T |u_j^0\rangle + i |u_j^0\rangle T^1 S^T |u_j^0\rangle$$

$$- i (\mu_j^0)^* |u_j^0\rangle T^1 S^T |u_j^0\rangle + |\mu_j^0|^2 |u_j^0\rangle T^1 S^T ST |u_j^0\rangle$$

$$= |u_j^0\rangle |u_j^0\rangle + i\mu_j^0 |u_j^0\rangle \hat{A} |u_j^0\rangle$$

$$- i (\mu_j^0)^* |u_j^0\rangle \hat{A}^T |u_j^0\rangle + |\mu_j^0|^2 |u_j^0\rangle$$

$$= 1 + i\mu_j^0 \lambda_j - i (\mu_j^0)^* \lambda_j + 1$$

$$= 2 \left( 1 - \lambda_j \right).$$ \hspace{1cm} (40)

Therefore, $W$ has the normalized eigenvectors

$$|v_j^0\rangle = \frac{1 + i\mu_j^0 S}{\sqrt{2 \left( 1 - \lambda_j \right)}} T |u_j^0\rangle.$$ \hspace{1cm} (41)

While the above analysis does not constitute a comprehensive investigation into the eigenvalues and eigenvectors of $W$, this simplified derivation is, as shown in what follows, sufficient for our purposes. For a detailed analysis, see [18]. To demonstrate the sufficiency of the above analysis, consider the following superposition of eigenvectors of $W$:

$$\frac{1 + i\lambda_j \mu_j^-}{\sqrt{2 \left( 1 - \lambda_j^2 \right)}} | v_j^+ \rangle + \frac{1 + i\lambda_j \mu_j^+}{\sqrt{2 \left( 1 - \lambda_j^2 \right)}} | v_j^- \rangle.$$  \hspace{1cm} (42)

Expanding the eigenvector expressions and simplifying, we find

$$\frac{1}{2 \left( 1 - \lambda_j^2 \right)} \left[ \left( 1 + i\lambda_j \mu_j^- \right) \left( 1 + i\mu_j^+ S \right) T |u_j^0\rangle + \left( 1 + i\lambda_j \mu_j^+ \right) \left( 1 + i\mu_j^- S \right) T |u_j^0\rangle \right]$$

$$= \frac{1}{2 \left( 1 - \lambda_j^2 \right)} \left[ 2 + i \left( \mu_j^+ + \mu_j^- \right) S \right.$$

$$\left. + i\lambda_j \left( \mu_j^+ + \mu_j^- \right) - 2\lambda_j \mu_j S \right] T |u_j^0\rangle$$

$$= \frac{1}{2 \left( 1 - \lambda_j^2 \right)} \left[ 2 - 2\lambda_j S - 2\lambda_j S + 2\lambda_j \right] T |u_j^0\rangle$$

$$= T |u_j^0\rangle.$$ \hspace{1cm} (43)

This result gives us the necessary components to build a matrix equation solver. By applying $T$ to the initial right-hand side vector $|b, 0\rangle$, we can transform the vector into a predictable superposition of the $|v_j^0\rangle$’s:

$$T |b, 0\rangle = \sum_{j=0}^{N-1} \beta_j T |u_j^0\rangle$$

$$= \sum_{j=0}^{N-1} \beta_j \frac{1}{\sqrt{2 \left( 1 - \lambda_j^2 \right)}} \left[ \left( 1 + i\lambda_j \mu_j^- \right) |v_j^+\rangle + \left( 1 + i\lambda_j \mu_j^+ \right) |v_j^-\rangle \right].$$ \hspace{1cm} (44)

Then an application of QPE to this state using $W$ as the requisite unitary yields the phase corresponding to $\mu_j$, when $|u_j^0\rangle$ appears in the initial state. From here it is possible to extract $\lambda_j$ and apply an HHL-style ancilla rotation. After applying inverse QPE, the final step of the procedure is to uncompute by applying $T^1$, thereby leaving only the rotated initial state, as in the result of HHL.

IV. CONSIDERATIONS FOR PRACTICAL IMPLEMENTATION

The analysis we have provided so far represents a complete theoretical formulation of the procedure, but practical implementation presents several nontrivial problems. In particular, the implementation of $W$, the initial application of $T$, and the eigenvalue extraction require elaboration. Here we present a recommended approach to the resolution of these sticking points.

Implementation of $W$ requires, in addition to the straightforward operators $i$ and $S$, the more involved operator $2TT^1 - I$. This operator is a sum of conditional reflectors\footnote{A brief introduction to the concept of reflectors is given in appendix A.} about the $|0_j^0\rangle$
and $|\zeta^p_j\rangle$ states, as the following calculation shows. We begin by expanding and simplifying the $TT^\dagger$ portion of the operator:

$$TT^\dagger = \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right) \sum_{k=0}^{N-1} \left( |k, 0\rangle \langle k, 0| + |k, 1\rangle \langle k, 1| \right)$$

$$= \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \left( |j, 0\rangle \langle k, 0| \langle j, 0| + |j, 1\rangle \langle j, 1| \right)$$

$$= \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right)$$

$$= \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right)$$

$$= \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right)$$

Then the full operator $2TT^\dagger - I$ is

$$2 \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right)$$

$$= \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right)$$

$$= \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right)$$

$$= \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right)$$

$$= \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right)$$

Note that the swap operations involved in the walk operator mean that, at intermediate stages, the system stores essential data on the $|1\rangle$ state of the ancilla, and hence the reflections about the $|\zeta^p_j\rangle$’s cannot be ignored.

The implementation of the state preparation operators is straightforward. For the $B_j$ operators, we begin by preparing a uniform superposition:

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k, 0\rangle .$$

Next, we rotate the ancilla:

$$\frac{1}{\sqrt{X}} \sum_{k=0}^{N-1} \left( |k, 0\rangle + \sqrt{1 - X} |A_{jk}\rangle |1\rangle \right) .$$

It is at this point that the role of $X$ becomes clear. In order to ensure a valid rotation, we select $X$ such that $X/N \geq |A_{jk}|_{\text{max}}$. Note that the calculation of $A_{jk}^*$ requires some additional consideration. In particular, the result of $A_{jk}^*$ requires that

$$\sqrt{A_{jk}^*} \left( \sqrt{A_{jk}} \right)^* = A_{jk} .$$

In most cases, this can be satisfied by taking

$$\sqrt{A_{jk}^*} = |A_{jk}|e^{-i \arg(A_{jk})/2} ,$$

and using the principal square root for $|A_{jk}|$. However, if $A_{jk} \in (-\infty, 0)$, then $A_{jk} = A_{kj}$, and we have

$$\sqrt{A_{kj}^*} \left( \sqrt{A_{jk}} \right)^* = \left( |A_{jk}|e^{-i \pi/2} \right)^* = |A_{jk}| .$$

For $j \neq k$, we can recover the correct sign by taking the negative square root in \[53\] whenever $k > j$. That is, if $A_{jk}$ is a negative real number, we use

$$\sqrt{A_{kj}^*} = \text{sign}(j - k) \sqrt{A_{jk}}|e^{-i \arg(A_{jk})/2}|.$$

While this prescription is effective for $j \neq k$, it does not suffice when $j = k$. To address this case, we simply prevent negative elements from appearing on the diagonal of $A$ by adding an appropriate multiple of the identity matrix. Proper treatment of this modification is considered in section \[43\].

The $B_j$ operators can be implemented by simply switching the ancilla state of the operand register to $|1\rangle$. This gives $|\zeta^p_j\rangle = |0\rangle \otimes |1\rangle$, which is sufficient for our purposes.

The initial application of $T$ is also of concern. The operation is defined such that, when the $|r_1\rangle$ ancilla qubit is in the $|0\rangle$ state, the $|\phi^p_j\rangle$ states are produced on $|r_2\rangle$ regardless of the initial state of the register. This is problematic to implement in practice, but since the only direct application of $T$ occurs at the beginning of the calculation, we can safely assume that $|r_2\rangle$ begins with all qubits in the $|0\rangle$ state. Then $T$ can be effectively applied by conditional application of the state preparation operators:

$$T = \sum_{j=0}^{N-1} \left( |j, 0\rangle \langle j, 0| + |j, 1\rangle \langle j, 1| \right).$$
Since the initial state must be supplied with ancilla state $|0\rangle$, it is also possible to ignore the applications of $B'_j$.

The last point of concern is the new procedure for eigenvalue extraction. Where the canonical HHL unitary gives the eigenphase relationship $e^{2\pi i \phi_j} = e^{i \lambda_j}$, the walk unitary has $e^{2\pi i \phi_j} = i \lambda_j \pm \sqrt{1 - \lambda_j^2}$, and hence the extraction procedure must be modified. Recalling that $\lambda_j \in [-1, 1]$, this extraction is straightforward if we take the imaginary components:

$$\sin(2\pi \phi_j) = \lambda_j \implies \lambda_j = X \sin(2\pi \phi_j). \quad (57)$$

V. TREATMENT OF ARBITRARY MATRIX EQUATIONS

Recall the following list of restrictions that have been imposed on our system:

1) $|b\rangle$ must be normalized.
2) $A$ must be Hermitian.
3) $M$ must be a power of two.
4) The eigenvalues of $A$ must lie on the interval $[-X, X]$.
5) No negative real numbers may appear on the diagonal of $A$.

In this section, we develop a method by which an arbitrary matrix equation can be modified to satisfy these constraints. To keep our notation consistent, we begin from the following initial problem: Given $A_0 \in \mathbb{C}^{M \times M}$ and $|b_0\rangle \in \mathbb{C}^M$, find $|x_0\rangle$ such that

$$A_0 |x_0\rangle = |b_0\rangle. \quad (58)$$

This arbitrary input system provides the foundational elements for an appropriately restricted system.

The first restriction is that $|b\rangle$ is normalized. This is easily satisfied by dividing $|b_0\rangle$ by the magnitude of $|b_0\rangle$:

$$\frac{1}{||b_0||_2} A_0 |x_0\rangle = \frac{1}{||b_0||_2} |b_0\rangle. \quad (59)$$

Here we assume that $||b_0||_2 \neq 0$. Since the zero vector is always a valid solution when $||b_0||_2 = 0$, this case is trivial and need not be considered for our purposes.

The second assumption is that the system matrix $A$ must be Hermitian. This is satisfied in general by expanding the problem to the following $2M \times 2M$ system:

$$\frac{1}{||b_0||_2} \begin{bmatrix} 0 & A_0 \\ A_0^\dagger & 0 \end{bmatrix} \begin{bmatrix} 0 \\ x_0 \end{bmatrix} = \frac{1}{||b_0||_2} \begin{bmatrix} b_0 \\ 0 \end{bmatrix}. \quad (60)$$

Third, we consider that the size of our quantum system must be a power of 2, as the algorithm is constructed for quantum systems using two-state qubits. Therefore, we define $n = \lceil \log(2M) \rceil$ and $N = 2^n$, and once again expand our system, this time to

$$\frac{1}{||b_0||_2} \begin{bmatrix} 0 & A_0 & 0 & 0 \\ A_0^\dagger & 0 & 0 & 0 \\ 0 & 0 & I_{N-2M} \end{bmatrix} \begin{bmatrix} 0 \\ x_0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{||b_0||_2} \begin{bmatrix} b_0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (61)$$

where $I_{N-2M}$ is the identity matrix of dimension $N - 2M$, and we have appended $N - 2M$ zeros to each vector. Now each vector can be represented by an $n$-qubit state.

The final problem is the restriction of the eigenvalues of $A$ to the range $[-X, X]$. Let $\lambda_{\text{max}}$ be the magnitude of the dominant eigenvalue of $A$. Then we have

$$\lambda_{\text{max}} \leq N |A_{jk}|_{\text{max}} \leq X, \quad (62)$$

by the discussion following (51). Thus $[-\lambda_{\text{max}}, \lambda_{\text{max}}] \subseteq [-X, X]$, and hence the spectrum of $A$ lies entirely on the required range. That is, for this procedure, no rescaling of the system is required to satisfy the eigenvalue restrictions.

Every element on the diagonal of the system matrix is now either 0 or a positive real number, and hence no negative real elements appear on its diagonal. This accounts for all imposed restrictions, leaving us with the following scheme for the preparation of an appropriate system:

$$A = \frac{1}{||b_0||_2} \begin{bmatrix} 0 & A_0 & 0 & 0 \\ A_0^\dagger & 0 & 0 & 0 \\ 0 & 0 & I_{N-2M} \end{bmatrix},$$

$$|b\rangle = \frac{1}{||b_0||_2} \begin{bmatrix} b_0 \\ 0 \\ 0 \end{bmatrix},$$

$$|x\rangle = \begin{bmatrix} x_0 \\ 0 \end{bmatrix}. \quad (63)$$

Of course, it is possible that $A_0$ is already Hermitian, in which case some efficiency can be gained by not performing the full expansion as stated above. Instead, only the size restriction must be satisfied. To this end, we redefine $n = \lceil \log(N) \rceil$—keeping the definition $N = 2^n$—and expand the system as follows:

$$A = \frac{1}{||b_0||_2} \begin{bmatrix} A_0 & 0 \\ 0 & I_{N-M} \end{bmatrix},$$

$$|b\rangle = \frac{1}{||b_0||_2} \begin{bmatrix} b_0 \\ 0 \end{bmatrix},$$

$$|x\rangle = \begin{bmatrix} x_0 \\ 0 \end{bmatrix}. \quad (64)$$

Now it is possible for the system matrix to have negative real values on the diagonal. This can be amended by using a shifted system:

$$(A + dI) |x\rangle = |b\rangle, \quad (65)$$

where $d$ is an upper bound on the magnitude of the offending values on the diagonal of $A$. For the purposes of the walk operator, $A + dI$ should be used as the system matrix, but note that this has the effect of shifting all eigenvalues of $A$ by $d$. Hence, in order to directly apply the inverse of $A$ to the initial state, the eigenvalue extraction of (57) must be modified:

$$\lambda_j = X \sin(2\pi \phi_j) - d. \quad (66)$$

So long as $X$ is calculated from the shifted system matrix, the eigenvalues are still appropriately bounded, and $A$ still requires no additional rescaling to satisfy the eigenvalue restrictions.
corresponding to each of the two eigenvectors from the ancillae of $T_A$. Initial Application of HHL rotation, and uncomputation.

The first stage of the procedure is the application of $T_A$, followed by quantum phase estimation, $\mu$-stages: the initial application of $T_A$. The phase register is in the state $|0\rangle$.

Algorithm 1 Implementation of $T$

1: for $j = 0 \ldots N - 1$ do
2: Compute the operator $B_j$
3: Apply $B_j$ to $|r_2\rangle$ with the control condition that $|r_1\rangle$ is in the state $|j, 0\rangle$
4: end for

The first stage of the procedure is the application of $T_A$, which maps $|r_2\rangle$ to a superposition of the $|\phi^+_j\rangle$ and $|\phi^-_j\rangle$ states defined in section III. A circuit diagram for our suggested implementation of $T_A$ is shown in Fig. 4 and a pseudocode description of the operator is given in algorithm II. In the program, the “T” function of the “QWOps” module implements this operator. The definition of $T$ depends on the state preparation operators $B_j$, and as such we have also provided a circuit diagram for an arbitrary $B_j$ in Fig. 5.

Algorithm 2 Implementation of $B_j$

1: for $k = 0 \ldots n - 1$ do
2: Apply $H$ to qubit $k$
3: end for
4: for $k = 0 \ldots N - 1$ do
5: $\theta_{jk} \leftarrow \arccos \left( \sqrt{|A_{jk}|} N / X \right)$
6: $\omega_{jk} \leftarrow -\arg(A_{jk}) / 2$
7: if $\arg(A_{jk}) = \pi$ and $j < k$ then
8: $\omega_{jk} \leftarrow -\omega_{jk}$
9: end if
10: Apply $XP(\omega_{jk}) X R_y(2\theta_{jk})$ to the ancilla qubit with the control condition that the rest of the register is in the state $|k\rangle$
11: end for

VI. SUMMARY

Here we summarize the preceding analysis in the form of a self-contained procedure. We assume that the supplied matrix equation has been properly prepared according to $\mathbb{V}$.

Fig. 3 provides a large-scale pictorial description of the full procedure. Notice that this procedure consists of four main stages: the initial application of $T_A$, quantum phase estimation, HHL rotation, and uncomputation.

A. Initial Application of $T$

Fig. 3: Structural outline of the full procedure. Dashed boxes indicate the state of the system after every major intermediate stage. The $|r_1\rangle$ and $|r_2\rangle$ registers are the ancilla-augmented vector registers on which the walk operator and its associated constituents act. The phase register $|p\rangle$ provides space for the estimation of the eigenphases of $W$. The ancilla $|a\rangle$ is used for the application of the HHL rotation described in section III. We commonly refer to $|a\rangle$ as the “HHL ancilla” to distinguish it from the ancillae of $|r_1\rangle$ and $|r_2\rangle$. The phase amplitudes $\rho_{jk}$ and $\rho_{jk}'$ are used to distinguish the amplitudes of phase estimates corresponding to each of the two eigenvectors $|v_j\rangle$. The eigenvalue estimates themselves require no such distinction, as both $\mu_j^+$ and $\mu_j^-$ have the same imaginary part.
| $b, 0 \rangle = \sum_{j=0}^{N-1} \alpha_j |j, 0 \rangle$
| \hline
| $\sum_{j=0}^{N-1} \beta_j |w_j^2 \rangle$
| \hline
| $|r_2 \rangle: |0 \ldots 0, 0 \rangle$
| \hline

\begin{align*}
|b, 0 \rangle = \sum_{j=0}^{N-1} \alpha_j |j, 0 \rangle \\
= \sum_{j=0}^{N-1} \beta_j |w_j^2 \rangle
\end{align*}

As a result, we have only provided a final state for the total system.

---

**Quantum Phase Estimation**

With $|r_1 \rangle$ and $|r_2 \rangle$ prepared in the desired superposition of the walk operator's eigenvectors, an application of QPE using $W$ as the unitary writes the eigenphases of $W$ onto the phase register $|p\rangle$. An expansion of this step, in the typical QPE format, is given in Fig. 6. A circuit diagram for the walk operator itself is shown in Fig. 7 and the corresponding pseudocode is given in algorithm 3. The walk operator is implemented in the program by the “W” function of the “QWOps” module. Note that we have made use of our suggested $j$-independent form of the $|\phi_j^p \rangle$ states, wherein

\begin{equation}
|b, 0 \rangle = \sum_{j=0}^{N-1} \alpha_j |j, 0 \rangle
\end{equation}

---

**Fig. 4:** Diagram of the initial application of $T$. Here state preparation operators are applied to $|r_2 \rangle$, conditioned on the state of $|r_1 \rangle$. Note that, due to the presumption that $|r_1 \rangle$ is initially prepared with an ancilla state of $|0 \rangle$, the $|r_1 \rangle$ ancilla is ignored in this calculation. Since the resulting system is highly entangled, individual qubit states cannot be separated. As a result, we have suppressed the phase and HHL ancilla registers, as they are not subject to the effects of $T$.

**Fig. 5:** Diagram of the state preparation operator $B_j$ applied to a register with all qubits initialized to the $|0 \rangle$ state. Here $\theta_j = \arccos \left( \frac{1}{\sqrt{N}} \right)$ and $\omega_{jk} = \pm \arg (A_{jk}) / 2$, where the positive solution for $\omega_{jk}$ is chosen if $\arg (A_{jk}) = \pi$ and $j > k$. Thus, the $R_y$ gates rotate according to the magnitudes of the elements of $A$, with the factor of $X P X$ subsequently enforcing the correct phase on the $|0 \rangle$ state of the ancilla.

\begin{align*}
[b, 0 \rangle |0 \ldots 0, 0 \rangle &= \sum_{j=0}^{N-1} \beta_j |u_j, 0 \rangle |0 \ldots 0, 0 \rangle \\
\tilde{T} \rightarrow \sum_{j=0}^{N-1} \frac{\beta_j}{\sqrt{2 \left(1 - \lambda_j^2 \right)}} &\left[ \left( 1 + i \lambda_j \mu_j \right) |v_j^+ \rangle + \left( 1 + i \lambda_j \mu_j \right) |v_j^- \rangle \right].
\end{align*}

Here we have suppressed the phase and HHL ancilla registers, as they are not subject to the effects of $T$.

\begin{align*}
|b, 0 \rangle |0 \ldots 0, 0 \rangle &= \sum_{j=0}^{N-1} \beta_j |u_j, 0 \rangle |0 \ldots 0, 0 \rangle \\
\tilde{T} \rightarrow \sum_{j=0}^{N-1} \frac{\beta_j}{\sqrt{2 \left(1 - \lambda_j^2 \right)}} &\left[ \left( 1 + i \lambda_j \mu_j \right) |v_j^+ \rangle + \left( 1 + i \lambda_j \mu_j \right) |v_j^- \rangle \right].
\end{align*}

Here we have suppressed the phase and HHL ancilla registers, as they are not subject to the effects of $T$.

**B. Quantum Phase Estimation**

With $|r_1 \rangle$ and $|r_2 \rangle$ prepared in the desired superposition of the walk operator’s eigenvectors, an application of QPE using $W$ as the unitary writes the eigenphases of $W$ onto the phase register $|p\rangle$. An expansion of this step, in the typical QPE format, is given in Fig. 6. A circuit diagram for the walk operator itself is shown in Fig. 7 and the corresponding pseudocode is given in algorithm 3. The walk operator is implemented in the program by the “W” function of the “QWOps” module. Note that we have made use of our suggested $j$-independent form of the $|\phi_j^p \rangle$ states, wherein

\begin{align*}
|b, 0 \rangle |0 \ldots 0, 0 \rangle &= \sum_{j=0}^{N-1} \beta_j |u_j, 0 \rangle |0 \ldots 0, 0 \rangle \\
\tilde{T} \rightarrow \sum_{j=0}^{N-1} \frac{\beta_j}{\sqrt{2 \left(1 - \lambda_j^2 \right)}} &\left[ \left( 1 + i \lambda_j \mu_j \right) |v_j^+ \rangle + \left( 1 + i \lambda_j \mu_j \right) |v_j^- \rangle \right].
\end{align*}

Here we have suppressed the phase and HHL ancilla registers, as they are not subject to the effects of $T$. 
Algorithm 3 Implementation of $W$

1: for $j = 0 \ldots N - 1$ do
2: \hspace{1em} Compute the state preparation operator $B_j$
3: \hspace{1em} Apply the reflector $B_j (2 |0⟩⟨0| - I) B_j^\dagger$ to $|r_2⟩$ with the control condition that $|r_1⟩$ is in the state $|j, 0⟩$
4: end for
5: Compute the state preparation operator $B'$
6: \hspace{1em} Apply the reflector $B' (2 |0⟩⟨0| - I) (B')^\dagger$ to $|r_2⟩$ with the control condition that the $|r_1⟩$ ancilla is in the state $|1⟩$
7: for $j = 0 \ldots N - 1$ do
8: \hspace{1em} Apply a swap operation to the $j$-th qubits of $|r_1⟩$ and $|r_2⟩$
9: end for
10: Apply a swap operation to the ancillae of $|r_1⟩$ and $|r_2⟩$
11: Apply $iI$ to any qubit of $|r_1⟩$ or $|r_2⟩$

Algorithm 4 Implementation of the HHL rotation

1: for $k = 0 \ldots N_p - 1$ do
2: \hspace{1em} $\tilde{\lambda}_k \leftarrow X \sin (2\pi k / N_p) - d$
3: \hspace{1em} if $\tilde{\lambda}_k \neq 0$ then
4: \hspace{2em} Apply the rotation $R_y \left[ 2 \arccos \left( C/\tilde{\lambda}_k \right) \right]$ to $|a⟩$, with the control condition that $|p⟩$ is in the state $|k⟩$
5: \hspace{1em} end if
6: end for

With the phase register holding the eigenphases of $W$, extraction of the corresponding eigenvalues can proceed according to the analysis of sections [V] and [VI]. Specifically, the eigenvalues are calculated according to

$$\lambda_j = X \sin (2\pi \phi_j) - d,$$

where $d = 0$ can be used if the system was expanded to satisfy the Hermitian system matrix property. Of course, the exact values of $\phi_j$ are not known in general, and in practice approximate eigenvalues must be calculated from the state of the phase register as follows:

$$\tilde{\lambda}_k = X \sin \left( 2\pi \frac{k}{N_p} \right) - d.$$

Once these eigenvalue approximations have been calculated, a rotation operator can be used to apply an eigenvalue factor corresponding to the inverse of the system matrix for every possible state of the phase register, as detailed in section [VI]. A circuit diagram for this procedure is given in Fig. 8, and the relevant pseudocode is provided in Algorithm 4. This rotation is implemented in the program by the “HHLRotation” function of the “QAlgs” module.

Fig. 7: Diagram of the walk operator $W$. Here $S$ represents a one-to-one exchange of qubits states between $|r_1⟩$ and $|r_2⟩$. The operator $iI$ can be applied to any qubit in $|r_1⟩$ or $|r_2⟩$ to produce the desired effect.
suitable for fully quantum analysis, we are at present limited
the expected states. Therefore, when considering a problem
primitive gates being, in our experience, unlikely to yield
∼ propagate rapidly, with calculations involving more than
This final state is analogous to (19).

First, the quantum systems currently available to the public are
quantum systems. Section VII-A constructs and analyzes such a problem, but it is possible to construct a more
satisfying test using classical simulation. By using a classical
computer to simulate a quantum system, we gain access to more qubits and eliminate gate errors. Such simulation is
extremely inefficient, but we have found that as many as 14
qubits can be simulated for this procedure using available classical
systems. Section VII-B considers a larger, less restricted
test problem using classical simulation.

A. A Fully Quantum Problem

Being limited to 7 qubits, we must consider only the
smallest of possible problems. Thus we choose a problem of two unknowns, requiring a single qubit for each of the
main portions of |r1⟩ and |r2⟩. Both of these registers require
ancillary qubits, and a third ancilla is also required for the
eigenvalue rotation. Then we currently stand at 5 qubits
accounted for, with 2 left for the phase register. These four
possible phase states admit no error in the phase estimation,
and hence we must choose a problem such that the eigenphases
can be exactly represented by two bits.

By (66), the eigenvalues of the system matrix must satisfy
\[ \lambda_j = X \sin (2\pi \phi_j) - d = \begin{cases} -d, & \phi_j = 0, \frac{1}{2} \\ X - d, & \phi_j = \frac{1}{2} \\ -X - d, & \phi_j = \frac{3}{4} \end{cases} \] (73)
The system
\[ A = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix}, \quad d = 3, \quad X = 2 \] (74)
satisfies the constraints of our procedure, and has eigenvalues
\[ \lambda_0 = -3 = -d, \quad \lambda_1 = 1 = X - d. \] (75)

Thus, this system provides a suitable candidate for our anal-
ysis. Note that the system is Hermitian, and no expansion

Upon completion of this rotation, the system has the state
\[ \sum_{j=0}^{N-1} \sum_{k=0}^{N_p-1} \beta_j \] 
\[ = \sum_{j=0}^{N-1} \sum_{k=0}^{N_p-1} \frac{1}{\sqrt{2}} \left(1 - \lambda_j^2\right)^{\frac{1}{2}} \] 
\[ \left[ \rho_{jk}^+ \left(1 + i\lambda_j \mu_j^+ \right) |v_j^+ \rangle + \rho_{jk}^- \left(1 + i\lambda_j \mu_j^+ \right) |v_j^- \rangle \right] |k\rangle \] 
\[ \otimes \left( C \lambda_k \left|0\right\rangle + \sqrt{1 - C^2 \lambda_k^2} \left|1\right\rangle \right) . \] (71)

D. Uncomputation

After rotation, |p⟩, |r2⟩, and the ancilla of |r1⟩ are uncom-
puted by applying inverse QPE and T operations. Ideally, these
registers are returned to the |0⟩ state, and any other state can
be interpreted as a failure of the procedure. This leaves |r1⟩
entangled with |a⟩. When |a⟩ takes the state |0⟩, the state of
|r1⟩ is proportional (up to the error introduced through QPE)
to the solution state |x⟩. Meanwhile, if |a⟩ is in the state |1⟩,
the system is in a failure state. That is, the overall final state
of the system is
\[ C |x, 0\rangle \otimes |0\rangle \otimes |0\rangle + |\text{failure} \rangle . \] (72)
This final state is analogous to [19].

VII. RESULTS

In developing suitable tests for the practical performance of
this procedure, two important limitations must be considered:
First, the quantum systems currently available to the public are
quite small, with the largest system we have access to offering
only 7 qubits [19]. Second, errors on these systems can
propagate rapidly, with calculations involving more than \( \sim 800 \)
primitive gates being, in our experience, unlikely to yield
the expected states. Therefore, when considering a problem
suitable for fully quantum analysis, we are at present limited

\footnote{This is the ibmq jalaka v1.0.23 system, which is one of the IBM
Quantum Falcon Processors.} to very short, small calculations. Section VII-A constructs
and analyzes such a problem, but it is possible to construct a more
satisfying test using classical simulation. By using a classical
computer to simulate a quantum system, we gain access to more qubits and eliminate gate errors. Such simulation is
extremely inefficient, but we have found that as many as 14
qubits can be simulated for this procedure using available classical
systems. Section VII-B considers a larger, less restricted
test problem using classical simulation.
is required. For a right-hand side vector, we use an equal superposition of both eigenvectors of $A$:

$$|b\rangle = \frac{1}{2} \left( | -1 \rangle + | 1 \rangle \right) = | 0 \rangle. \quad (76)$$

For this problem of known form, some simplification is possible in the various operators of the solution procedure. This allows the size of the final circuit to be significantly reduced. The system matrix defining the walk operator is

$$A + dI = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}. \quad (77)$$

Therefore, all $B_j$ operators apply the same ancilla rotation:

$$R_y \left( 2 \arccos \sqrt{1 \cdot \frac{2}{2}} \right) = R_y (0) = I. \quad (78)$$

Since all elements are real, no phase rotation is necessary, and all $B_j$ operators can be implemented by simply applying a Hadamard gate to $|r_2\rangle$. This elimination of the complicated phase and $R_y$ gates, both of which also require additional control considerations, greatly reduces the complexity of each $B_j$ operator.

This simplification of the $B_j$ operators carries through the $T$ and $W$ operators. Without this reduction, the full program produces a circuit consisting of 15,728 basic gates. This figure results from transpilation using the available $CX$, $I$, $R_z$, $\sqrt{X}$, and $X$ basis gates. After simplification, the final circuit requires only 2,696 gates. This is a notable reduction, but the circuit is still much too large considering the errors incurred by each gate application. We remedy this by dividing the circuit into many smaller sub-circuits, and running each of these components as its own calculation. By initializing each component based on a simulated result of the previous component (rather than continuing from an imprecise result as a direct calculation would), the compounding effect of gate errors can be eliminated. We can then verify the validity of our solution by confirming that the results of each component agree with the simulated results for the same component.

In total, the circuit was divided into 120 components as listed in appendix [B]. The results of the calculation are shown in Fig. 9. We see that good agreement is maintained between the simulated and quantum procedures until the more complex later stages, when gate counts approach and exceed 800. At this point, calculations quickly become unreliable. Note that the sum of the number of gates required for all sub-circuits is significantly larger than the number of gates reported for the direct circuit. This is due to the reinitialization of the system, which adds an unpredictable overhead cost to each operation. The average relative error of the final simulated solution compared to a direct classical solution is $1.12 \times 10^{-10}$, which is well within the tolerances prescribed by the program.

### B. Simulated Solution

With space available for a slightly larger system, we consider a practical, though still trivial, problem. We consider a calculation of the charge per unit length on a transmission line consisting of two long, conducting strips, with known potentials, radiating in free space. This problem can be modeled as a method of moments discretization of a cross-section of the line, yielding a matrix equation for the element charges $|Q\rangle$ as a function of their potentials $|V\rangle$:

$$B |Q\rangle = |V\rangle. \quad (79)$$

Elements of the matrix $B$ are given by [22]:

$$B_{jk} = \begin{cases} -\frac{l_j}{\pi \varepsilon_0} \ln (l_j) - 1.5, & j = k \\ -\frac{l_j}{\pi \varepsilon_0} \ln (d_{jk}), & j \neq k. \end{cases} \quad (80)$$

where $l_j$ is the length of the $j$-th one-dimensional boundary element and $d_{jk}$ is the separation between the centroids of elements $j$ and $k$. By using elements of uniform size $l$, we can ensure that $B$ is Hermitian. We give each strip unit width, and orient both strips parallel to each other with unit separation.

The calculated charge density for each element is shown in Fig. 10 and the solution vector and its error are given in Table...
TABLE I: Comparison of Simulated Quantum and Direct Classical Solution Vectors

| Element Index | Quantum | Classical | Relative Error |
|---------------|---------|----------|---------------|
| 0             | 0.0383  | 0.0071   | 0.0315        |
| 1             | -0.0383 | -0.0071  | 0.0315        |
| 2             | -0.0383 | -0.0071  | 0.0315        |
| 3             | -0.0383 | -0.0071  | 0.0315        |

For this calculation, four unknowns and seven phase qubits were used. We see that the quantum results are within ~3% of the correct solution, showing that our procedure produces a reliable approximation. For more accurate solutions, a more complex system involving additional phase qubits would need to be simulated.

VIII. Conclusion

By combining the known HHL algorithm with a unitary adopted from quantum walk research, we have developed a method for the solution of any well-conditioned matrix equation which is suitable for direct implementation on quantum hardware. While the method itself is suitable for such practical implementation, available quantum systems are far too small and unreliable to support the analysis of any nontrivial problem. This situation is expected to improve rapidly, however, with IBM projecting the release of a 1,000 qubit system by the end of 2023 [23]. They also anticipate a dramatic decrease in errors [24]. Google has also indicated plans to develop a commercial-grade system by 2029 [25].

It is also important to note that this procedure does not, in itself, offer any advantage over classical methods. Implementation of the T and W operators requires O (N^2) unique rotation operators, which surpasses the complexity of efficient classical algorithms. In the future, however, we hope to use this general procedure as a foundation for the development of more refined methods for particular classes of problems.

Appendix A

Reflection Operators

Here we briefly describe the nature of reflection operators. In particular, we consider operators of the form 2 |v⟩ ⟨v| − I, where |v⟩ is some normalized vector. This operator, when applied to another vector |w⟩ of the same dimension, reflects |w⟩ about |v⟩. As an initial example, consider the two-dimensional case where |v⟩ is the unit vector along the y axis, |y⟩, and |w⟩ = a |x⟩ + b |y⟩ is arbitrary. Then the effect of the reflection operator is as follows:

\[
\begin{align*}
(2 |y⟩ ⟨y| - I) (a |x⟩ + b |y⟩) &= (2a |y⟩ ⟨y| x⟩ + 2b |y⟩ ⟨y| y⟩) - (a |x⟩ + b |y⟩) \\
&= 2b |y⟩ - (a |x⟩ + b |y⟩) \\
&= -a |x⟩ + b |y⟩.
\end{align*}
\]

Then the arbitrary reflector 2 |v⟩ ⟨v| − I has the effect of negating the perpendicular component of the operand vector, while leaving the parallel component unchanged.

Appendix B

Quantum Circuit Components

This section lists the individual operations which constitute each component of the divided circuit described in section VII-A. Operations are given in Qiskit-style syntax similar to that used in the program. Operators are listed first, with the operand registers listed on the following line. We assume that C, X, and d have been defined. Note that each of these components also includes the operations necessary to initialize it to the ideal state resulting from the precise application of all previous components. These operations, applied in sequence to an appropriately initialized system, constitute the solution procedure for the relevant system.
150  XGate().control(1,None,'1')
149  [reg_phase[1]]+ [reg_r1[0]]
148  SGate().inverse().control(1,None,'1')
147  XGate().control(1,None,'1')
146  [reg_phase[1]]+ [reg_r1[0]]
145  XGate().control(1,None,'1')
144  [reg_phase[1]]+ [reg_r1[0]]
143  XGate().control(1,None,'1')
142  [reg_phase[1]]+ [reg_r1[0]]
141  XGate().control(1,None,'1')
140  [reg_phase[1]]+ [reg_r1[0]]
139  XGate().control(1,None,'1')
138  [reg_phase[1]]+ [reg_r1[0]]
137  XGate().control(1,None,'1')
136  [reg_phase[1]]+ [reg_r1[0]]
135  XGate().control(1,None,'1')
134  [reg_phase[1]]+ [reg_r1[0]]
133  XGate().control(1,None,'1')
132  [reg_phase[1]]+ [reg_r1[0]]
131  XGate().control(1,None,'1')
130  [reg_phase[1]]+ [reg_r1[0]]
129  XGate().control(1,None,'1')
128  [reg_phase[1]]+ [reg_r1[0]]
127  XGate().control(1,None,'1')
126  [reg_phase[1]]+ [reg_r1[0]]
125  XGate().control(1,None,'1')
124  [reg_phase[1]]+ [reg_r1[0]]
123  XGate().control(1,None,'1')
122  [reg_phase[1]]+ [reg_r1[0]]
121  XGate().control(1,None,'1')
120  [reg_phase[1]]+ [reg_r1[0]]
119  XGate().control(1,None,'1')
118  [reg_phase[1]]+ [reg_r1[0]]
117  XGate().control(1,None,'1')
116  XGate().control(1,None,'1')
115  XGate().control(1,None,'1')
114  XGate().control(1,None,'1')
113  XGate().control(1,None,'1')
112  XGate().control(1,None,'1')
111  XGate().control(1,None,'1')
110  XGate().control(1,None,'1')
109  XGate().control(1,None,'1')
108  XGate().control(1,None,'1')
107  XGate().control(1,None,'1')
106  XGate().control(1,None,'1')
105  XGate().control(1,None,'1')
104  XGate().control(1,None,'1')
103  XGate().control(1,None,'1')
102  XGate().control(1,None,'1')
101  XGate().control(1,None,'1')
100  XGate().control(1,None,'1')
99  XGate().control(1,None,'1')
98  XGate().control(1,None,'1')
97  XGate().control(1,None,'1')
96  XGate().control(1,None,'1')
95  XGate().control(1,None,'1')
94  XGate().control(1,None,'1')
93  XGate().control(1,None,'1')
92  XGate().control(1,None,'1')
91  XGate().control(1,None,'1')
90  XGate().control(1,None,'1')
89  XGate().control(1,None,'1')
88  XGate().control(1,None,'1')
87  XGate().control(1,None,'1')
86  XGate().control(1,None,'1')
85  XGate().control(1,None,'1')
84  XGate().control(1,None,'1')
83  XGate().control(1,None,'1')
82  XGate().control(1,None,'1')
81  XGate().control(1,None,'1')
80  XGate().control(1,None,'1')
79  XGate().control(1,None,'1')
78  XGate().control(1,None,'1')
77  XGate().control(1,None,'1')
76  XGate().control(1,None,'1')
75  XGate().control(1,None,'1')
74  XGate().control(1,None,'1')
73  XGate().control(1,None,'1')
72  XGate().control(1,None,'1')
71  XGate().control(1,None,'1')
70  XGate().control(1,None,'1')
69  XGate().control(1,None,'1')
68  XGate().control(1,None,'1')
67  XGate().control(1,None,'1')
66  XGate().control(1,None,'1')
65  XGate().control(1,None,'1')
64  XGate().control(1,None,'1')
63  XGate().control(1,None,'1')
62  XGate().control(1,None,'1')
61  XGate().control(1,None,'1')
60  XGate().control(1,None,'1')
59  XGate().control(1,None,'1')
58  XGate().control(1,None,'1')
57  XGate().control(1,None,'1')
56  XGate().control(1,None,'1')
55  XGate().control(1,None,'1')
54  XGate().control(1,None,'1')
53  XGate().control(1,None,'1')
52  XGate().control(1,None,'1')
51  XGate().control(1,None,'1')
50  XGate().control(1,None,'1')
49  XGate().control(1,None,'1')
48  XGate().control(1,None,'1')
47  XGate().control(1,None,'1')
46  XGate().control(1,None,'1')
45  XGate().control(1,None,'1')
44  XGate().control(1,None,'1')
43  XGate().control(1,None,'1')
42  XGate().control(1,None,'1')
41  XGate().control(1,None,'1')
40  XGate().control(1,None,'1')
39  XGate().control(1,None,'1')
38  XGate().control(1,None,'1')
37  XGate().control(1,None,'1')
36  [reg_phase[0]]+ [reg_r2[0]]
35  CNOT().power(0.5).inverse()
34  CNOT()
33  [reg_phase[0]]
32  [reg_phase[0]]
31  [reg_phase[0]]
30  [reg_phase[0]]
29  [reg_phase[0]]
28  [reg_phase[0]]
27  [reg_phase[0]]
26  [reg_phase[0]]
25  [reg_phase[0]]
24  [reg_phase[0]]
23  [reg_phase[0]]
22  [reg_phase[0]]
21  [reg_phase[0]]
20  [reg_phase[0]]
19  [reg_phase[0]]
18  [reg_phase[0]]
17  [reg_phase[0]]
16  [reg_phase[0]]
15  [reg_phase[0]]
14  [reg_phase[0]]
13  [reg_phase[0]]
12  [reg_phase[0]]
11  [reg_phase[0]]
10  [reg_phase[0]]
9  [reg_phase[0]]
8  [reg_phase[0]]
7  [reg_phase[0]]
6  [reg_phase[0]]
5  [reg_phase[0]]
4  [reg_phase[0]]
3  [reg_phase[0]]
2  [reg_phase[0]]
1  [reg_phase[0]]
0  [reg_phase[0]]

ACKNOWLEDGMENT

We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do
not reflect the official policy or position of IBM or the IBM Quantum team.

REFERENCES

[1] T. R. Chandrupatla and A. D. Belegundu, Introduction to Finite Elements in Engineering, 4th ed., Harlow, England: Pearson, 2012.

[2] W. C. Gibson, The Method of Moments in Electromagnetics, Boca Raton, FL, USA: Chapman Hall/CRC, 2008.

[3] M. N. Özisik, H. R. B. Orlande, M. J. Colaço, and R. M. Cotta, Finite Difference Methods in Heat Transfer, 2nd ed., Boca Raton, FL, USA: CRC Press, 2017.

[4] W. C. Chew, J. M. Jin, E. Michielsen, and J. Song, Fast and Efficient Algorithms in Computational Electromagnetics, Norwood, MA, USA: Artech House, 2001.

[5] Z. Chen, S. Zheng, and V. I. Okhmato, “Tensor train accelerated solution of volume integral equation for 2-D scattering problems and magneto-quasi-static characterization of multiconductor transmission lines,” IEEE Trans. Microw. Theory Tech., vol. 67, no. 6, pp. 2181–2196, June 2019, DOI: 10.1109/TMTT.2019.2908368.

[6] S. Omar and D. Jiao, “A linear complexity direct volume integral equation solver for full-wave 3-D circuit extraction in inhomogeneous materials,” IEEE Trans. Microw. Theory Tech., vol. 63, no. 3, pp. 897–912, Mar. 2015.

[7] L. K. Grover, “A Fast Quantum Mechanical Algorithm for Database Search,” in Proc. 28th Annu. ACM Symp. Theory Comput., pp. 212-219, July 1996, DOI: 10.1145/237814.237866.

[8] E. Gerjuoy, “Shor’s factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers,” Amer. J. Phys., vol. 73, no. 6, pp.521–540, May 2005, DOI: 10.1119/1.1891170.

[9] A. W. Harrow, A. Hassidim, and S. Lloyd, “Quantum algorithm for linear systems of equations,” Phys. Rev. Lett., vol. 103, no. 15, p. 150502, Oct. 2009, DOI: 10.1103/PhysRevLett.103.150502.

[10] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, “Quantum algorithms revisited,” Proc. R. Soc. Lond. A., vol. 454, no. 1969, pp.339–354, Jan. 1998, DOI: 10.1098/rspa.1998.0164.

[11] A. M. Childs, “On the relationship between continuous- and discrete-time quantum walk,” Commun. Math. Phys., vol. 294, no. 2, pp.581–603, Mar. 2010, DOI: 10.1007/s00220-009-0930-1.

[12] D. W. Berry and A. M. Childs, “Black-box Hamiltonian simulation and unitary implementation,” Quantum Info. Comput., vol. 12, no. 1-2, pp.29–62, Jan. 2012.

[13] M. S. Anis et al., “Qiskit: An Open-source Framework for Quantum Computing,” 2021, DOI: 10.5281/zenodo.2573505.

[14] C. D. Phillips and V. I. Okhmato, “QMES,” 2021. [GitHub repository]. Available: https://github.com/philli69/QMES

[15] S. E. Venegas-Andraca, “Quantum walks: a comprehensive review,” Quantum Inf. Process., vol. 11, no. 5, pp. 307–327, July-Aug. 2012, DOI: 10.1007/s11128-012-0432-5.

[16] J. Kempe, “Quantum random walks: An introductory overview,” Contemporary Physics, vol. 44, no. 4, pp. 307-327, July-Aug. 2003, DOI: 10.1080/00107151031000110776.

[17] A. Ambainis, “Quantum walk algorithm for element distinctness,” in Proc. 45th Annu. IEEE Symp. on Found. of Comput. Sci., pp. 22-31, Oct. 2004, DOI: 10.1109/FOCS.2004.54.

[18] M. Szegedy, “Quantum Speed-Up of Markov Chain Based Algorithms,” in Proc. 45th Annu. IEEE Symp. on Found. of Comput. Sci., pp. 32-41, Oct. 2004, DOI: 10.1109/FOCS.2004.53.

[19] IBM Quantum. https://quantum-computing.ibm.com/, 2021.

[20] R. A. Horn and C. R. Johnson, “Hermitian Matrices, Symmetric Matrices, and Congruence,” in Matrix Analysis, 2nd ed., New York, NY, USA: CUP, 2013, ch. 4, sec. 4.2, p. 234.

[21] B. C. Hall, “The Matrix Exponential,” in Lie Groups, Lie Algebras, and Representations: An Elementary Introduction, 2nd ed., Cham, Switzerland: Springer, 2015, ch. 2, sec. 2.1, p. 31.

[22] M. N. O. Sadiku, “Moment Methods,” in Numerical Techniques in Electromagnetics, 2nd ed., Boca Raton, FL, USA: CRC Press, 2001, ch. 5, sec. 5.4, p. 312.

[23] A. Cho, “IBM promises 1000-qubit quantum computer—a milestone—by 2023,” Science, Sep., 2021. Accessed on: Dec. 2, 2021. [Online]. Available: https://www.science.org/content/article/ibm-promises-1000-qubit-quantum-computer-milestone-2023

[24] J. Council, “The Decade of Quantum Computing Is Upon Us, IBM Exec Says,” WSJ, Mar., 2021. Accessed on: Dec. 2, 2021. [Online]. Available: https://www.wsj.com/articles/the-decade-of-quantum-computing-is-upon-us-ibm-exec-says-11614802190

[25] S. Castellanos, “Google Aims for Commercial-Grade Quantum Computer by 2029,” WSJ, May, 2021. Accessed on: Dec. 2, 2021. [Online]. Available: https://www.wsj.com/articles/google-aims-for-commercial-grade-quantum-computer-by-2029-11621359156