Simulating Sound Wave Propagation using Quantum Algorithms

MAPS Summer Research Project Report
Supervisor: Dr. Reza Haqshenas
Lau Pak To Ryan

I. INTRODUCTION

In the 1980's, it has been speculated that quantum computers can be used to simulate systems, solving physics and chemistry problems [1]. Over the years, there has been huge developments in quantum computation. In quantum computation, we manipulate qubits which have states $|0\rangle$ and $|1\rangle$ to perform operations. Because of the effect of entanglement and superposition, we can encode and process information far more efficiently than classical bits, which have on(1) and off(0) states. We are now in the process of building a scalable fault-tolerant computation. However there are multiple reports that quantum computers do solve many problems way faster than classical computation and have huge impacts on solving optimisation problems [2], quantum machine learning [3], quantum chemistry [4], addressing global problems such as climate change [5], drug discovery [6] and solving many-body physics problems [7]. This "quantum advantage" provokes us to utilise quantum computation to simulate sound waves via solving the Helmholtz equation. The project aims to initiate and build a foundation for a software based on quantum computation for simulating sound waves for future therapeutic applications [8, 9].

This semi-formal report will have the following structure: We first define the target model in Sec II, then we review the Harrow-Hassidim-Lloyd (HHL) algorithm in Sec III. The methods proposed to solve the problem and results are described in Sec IV. The report will finish with a detailed discussion and future work to be done in Sec V and a conclusion in Sec VI.

II. THE MODEL

We aim to solve a model similar to the one suggested in [10]:

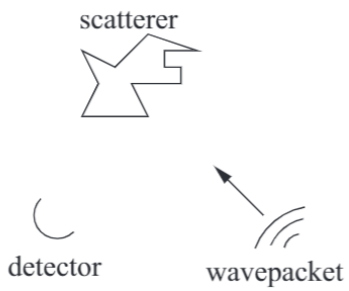

FIG. 1. Taken from [10], we adapt for our model: for a original soundwave from the source (wavepacket), we shoot it across some medium and hitting an object (scatterer). We then detect the pressure using the detector after the scattering.

The model is a sound wave propagation model, where we can use the time-independent Helmholtz equation in the frequency domain:

$$\nabla^2\phi(\mathbf{r}) + k^2\phi(\mathbf{r}) = S(\mathbf{r}) \tag{1}$$

where ϕ is the wave-field as a function of the position vector $\mathbf{r} = (x, y, z)^T$ and k the wavenumber, defined as:

$$k = \frac{2\pi\nu}{c} \tag{2}$$

where ν is the wave frequency and c the speed of sound. $S(\mathbf{r})$ is the source term as a function of $\mathbf{r}$. The source can be, for example, a plane wave or a monopole. When $S(\mathbf{r}) = 0$, Eq (1) will be homogeneous. For simplicity, we look to solve Eq. (1) in the frequency domain in 2-dimensions ($d = 2$), hence the laplacian ∇^2 is written as:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{3}$$

Solving the homogeneous model without scattering can be done algebraically by separation of variables and applying series solutions. The same cannot necessarily be done if a source term and a scatterer is present. Therefore we will look to discretise the laplacian and solve a series of linear equations, which can be solved using the HHL quantum algorithm [11].

III. THE HHL ALGORITHM

This section will describe the HHL algorithm with appropriate details.

A. Theory

1. Motivation

Suppose we have a Hermitian $N \times N$ matrix $\mathbf{A}$ and some unite vector $\mathbf{B}$, we need to solve for $\vec{x}$ which agrees with $\mathbf{A}\vec{x} = \mathbf{B}$. N must be a power of 2 for the algorithm to work. It will be more convenient using the bra-ket notation:

$$A\,|x\rangle = |b\rangle \tag{4}$$

$|b\rangle$ and A are given, where they can be given in the eigenbasis of A, $\{u_j\}$. By spectral decomposition, assuming A is Hermitian with eigenvalues $\{\lambda_j\}$:

$$A = \sum_{j=0}^{N-1} \lambda_j\,|u_j\rangle\,\langle u_j|\,, \lambda_j \in \mathbb{R} \tag{5}$$

Its inverse is trivial:

$$A^{-1} = \sum_{j=0}^{N-1} \lambda_j^{-1} |u_j\rangle \langle u_j| \tag{6}$$

$|b\rangle$ can be given in the eigenbasis of A, $\{u_j\}$ with its eigenvalues $\{\beta_j\}$:

$$|b\rangle = \sum_{j=0}^{N-1} \beta_j |u_j\rangle, \beta_j \in \mathbb{C} \tag{7}$$

To determine $|x\rangle$, we can do it algebraically:

$$|x\rangle = A^{-1}|b\rangle = \sum_{j=0}^{N-1} \lambda_j^{-1} \beta_j |u_j\rangle \tag{8}$$

Notice the last expression of 8: $|x\rangle \propto \lambda_j^{-1}$. This gives the main motivation to the algorithm: we have to manipulate the qubits such that we have some state $|\psi\rangle$ corresponding to $|x\rangle \propto \lambda_j^{-1}$.

2. Qubits used

There are three sets of qubits needed with initial state of $|0\rangle$:

1. One anciliary qubit $|0\rangle_a$ for controlled phase rotations
2. A set of n-qubits, $|0\rangle_b$, to store eigenvalues of A in binary format
3. A memory qubit $|0\rangle_m$ to store first $|b\rangle$ then $|x\rangle$.

3. The Algorithm

A has to be Hermitian. If not, then we have to define:

$$\hat{A} = \begin{pmatrix} 0 & A \\ A^\dagger & 0 \end{pmatrix} \tag{9}$$

The above gives a Hermitian matrix, then we can solve the following:

$$\hat{A}\vec{y} = \begin{pmatrix} \vec{b} \\ 0 \end{pmatrix}, \vec{y} = \begin{pmatrix} 0 \\ \vec{x} \end{pmatrix} \tag{10}$$

For simplicity, we assume A is Hermitian from now on.

a. Loading $|b\rangle$ We first load $|b\rangle$ into our circuit.

b. Quantum Phase Estimation (QPE) The first step is to perform a QPE [12, 13]. The main idea is to turn A into a unitary operator e^{iAt}:

$$U = e^{iAt} := \sum_{j=0}^{N-1} e^{i\lambda_j t} |u_j\rangle \langle u_j| \tag{11}$$

where t is the evolution time of the Hamiltonian. Some requirements should be fulfilled for this conversion to be possible:

1. A has to be an s-sparse matrix.
2. Hamiltonian simulation has to be performed

Eigenvalues of A can thus be estimated. Be aware of the ideal and non-ideal cases, as well as the associated errors. This step encodes, or maps the eigenvalues of A on to the set of qubits b: $|0\rangle_b \mapsto \left|\tilde{\lambda}_j\right\rangle$, where $\tilde{\lambda}_j$ is the binary representation of λ_j.

c. Phase Rotations We then apply controlled phase rotations to the ancillary qubit. We would like to map λ_j into something that has the term proportional to $1/\lambda_j$. The phase rotation matrix $R_y(\theta)$ is defined as:

$$R_y(\theta) = \begin{pmatrix} \cos\theta/2 & -\sin\theta/2 \\ \sin\theta/2 & \cos\theta/2 \end{pmatrix} \tag{12}$$

where θ is:

$$\theta = 2\arcsin\frac{C}{\lambda_j}, C \in \mathbb{C} \tag{13}$$

C is a normalisation constant. Note that there is a requirement for C [11]: $|C| \leq \min|\lambda_j|$. The total rotational operation on the ancillary qubit $|0\rangle_a$ will lead to the following:

$$R_y|0\rangle_a = \frac{1}{\lambda_j}\left(\sqrt{\lambda_j^2 - C^2}|0\rangle_a + C|1\rangle_a\right) \tag{14}$$

d. Inverse QPE (QPE^{-1}) Then we apply QPE^{-1} to uncompute $\left|\tilde{\lambda}_j\right\rangle$, which does the following: $\left|\tilde{\lambda}_j\right\rangle \mapsto |0\rangle_b$. The entire outcome state will be:

$$|\Psi\rangle = \sum_{j=0}^{N-1} \beta_j \left(\frac{1}{\lambda_j}\left(\sqrt{\lambda_j^2 - C^2}|0\rangle_a + C|1\rangle_a\right)\right)|0\rangle_b |u_j\rangle_m \tag{15}$$

e. Measurement In postselection, we select specifically the outcome of state $|1\rangle_a$. The desired output will be:

$$|x\rangle \propto \sum_{j=1}^{N} \frac{\beta_j}{\lambda_j}|u_j\rangle \tag{16}$$

up to some normalisation constant.

B. Implementation

The HHL algorithm can be implemented via IBM's quantum computing platform using Python's Qiskit package[14], where there is an already in-built quantum circuit ready to use. The circuit then can be parsed through simulators and processors[15]. An interesting part of the algorithm offered by IBM's Qiskit is the part of loading $|b\rangle$, where they used an isometry[16]. To get the solution, we aim to get the statevectors of the rotated qubits. We have streamlined the process by developing a software package, which is still in the development phase. Once finished, the open-source Python package will be published.

IV. POPOSED METHODS AND RESULTS

Having our algorithm at hand, we look to get a system of equations into the form as in Eq (4) and solve for ϕ in the frequency domain. We will approximate this operator using a matrix, similar to [10], where we will use the finite difference method.

A. The Finite Difference Method

For some general function $f = f(x, y, z)$, we can use a Taylor expansion to approximate it around some point h_i, which is some value along and away from some general coordinate x_i, where $i = 1, 2, 3$ correspond to the x,y and z coordinates respectively in Cartesian coordinates. The forward difference taylor expansion is:

$$\begin{aligned} f(x_i + h_i) &= \sum_{n=0}^{N} \frac{h_i^n}{n!} f_{x_i}^{(n)}(x_1, x_2, x_3) \\ &= f(x) + h f_{x_i}^{(1)} + \frac{h^2}{2!} f_{x_i}^{(2)} + ... \end{aligned} \tag{17}$$

Note that the superscript (n) expresses the number of times f is partially differentiated with respect to x_i, denoted by the subscript. The backward difference has a similar form:

$$\begin{aligned} f(x_i - h_i) &= \sum_{n=0}^{N} (-)^n \frac{h_i^n}{n!} f_{x_i}^{(n)}(x_1, x_2, x_3) \\ &= f(x) - h f_{x_i}^{(1)} + \frac{h^2}{2!} f_{x_i}^{(2)} - ... \end{aligned} \tag{18}$$

Note that $\mathcal{O}(h^3)$ or above orders may not have physical significance, hence the physical interpretation of f, or in our case ϕ, has to be specified. In addition to this, the truncation errors are the orders we have not added, i.e. the terms with $\mathcal{O}(h^3)$ or above. We now go on to approximate the partial derivatives.

For 1st derivatives, we subtract 17 and 18. After some rearrangements we get the approximation:

$$\frac{\partial f}{\partial x_i} = f_{x_i}^{(1)} = \frac{f(x_i + h_i) - f(x_i - h_i)}{2h_i} \tag{19}$$

For 2nd derivatives, we add 17 and 18. Again, with some rearrangements:

$$\frac{\partial^2 f}{\partial x_i^2} = f_{x_i}^{(2)} = \frac{f(x_i + h_i) - 2f(x_i) + f(x_i - h_i)}{h_i^2} \tag{20}$$

Let us return back to our function from the Helmholtz equation ϕ. Using methods above, we can approximate its first and second order derivatives as shown in the set of equations below. For clarity, we choose $h_1 = h$ and $h_2 = k$.

$$\frac{\partial \phi}{\partial x} = \frac{\phi(x + h, y) - \phi(x - h, y)}{2h} \tag{21}$$

$$\frac{\partial \phi}{\partial y} = \frac{\phi(x, y + k) - \phi(x, y - k)}{2k} \tag{22}$$

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi(x + h, y) - 2\phi(x, y) + \phi(x - h, y)}{h^2} \tag{23}$$

$$\frac{\partial^2 \phi}{\partial y^2} = \frac{\phi(x, y + k) - 2\phi(x, y) + \phi(x, y - k)}{k^2} \tag{24}$$

The above sets of equations are what we will be using to discretise ∇^2 to form a matrix. We can further simplify the problem using boundary conditions [10], namely the Dirichlet boundary condition:

$$\phi = 0 \tag{25}$$

and the Neumann boundary condition:

$$\nabla \phi \cdot \hat{n} = 0 \tag{26}$$

For a good approximation, we will have to "chop up" space into little chunks. It will be too inefficient to do the discretisation by hand, therefore we can obtain the matrix with the help of the Findiff package [17] with boundary conditions applied. For the source term $S(\mathbf{r})$ with an amplitude A_0, we have two choices. One is the plane wave solution:

$$S(\mathbf{r})_{\text{plane}} = A_0 e^{-i\mathbf{k}\cdot\mathbf{r}} \tag{27}$$

another is the monopole source:

$$S(\mathbf{r})_{\text{mono}} = \frac{A_0}{|\mathbf{r}|} e^{-ik|\mathbf{r}|} \tag{28}$$

One could have other choices of source terms.

B. Evaluation of Results

We evaulate some results which we had. So far, the demonstrations are highly influenced by the IBM documentation [18] and textbook [19] and do not differ much. One may look into the provided links for some demonstrations. Unfortunately, the project is far from complete due to time constraints and unforeseen errors. Our software using HHL Qiskit works for calculating euclidean norms, counting gates and gives information of the gates used. Statevector results from the classical and HHL result are compared. The classical result can be obtained via matrix inversion and multiplication or can be done via SciPy [20]. One may compare the classical and HHL results simply by subtracting the two and getting the ratio of the former to the latter. However for the statevector results, we run into problems when comparing the signs of individual components due to a rotation of the global phase whilst encoding $\vec{b}$ using the isometry method [16]. Tests have only been run up to the 16x16 case. We still cannot verify any more than 32×32 without using great computational power. The cases are run in different simulators provided by IBM [15]. We have yet to test the cases on real quantum processors. We are not at the point where we can solve the Helmholtz equation. Suggestions of future work and possible directions which could be taken will be discussed in the next section.

The software is not yet published as it is still in initial developmental stages. Some notebooks within

there shows our progress.

V. FUTURE WORK AND DISCUSSIONS

We discuss the implications of this project and possible improvements in the future.

A. Onto real, physical systems

Note that this is a 2-dimensional problem. For n dimensional problems, [10] suggests the following 3 steps :

1. for each spatial direction, separate the discretised laplacian into n graph Laplacians
2. for these graph Laplacians, write the Laplacians into a matrix, then factor them into incidence matrices
3. vertically stack, link, or concatenate their incidence matrices

B. Higher order approximations

One can use the Lagrange interpolation formula to approximate our partial derivatives [21]. The formula could improve the accuracy for our model by taking higher orders of accuracies [10]. The formula can exactly fit a polynomial set of points of some general function g, $\{p_j, g(p_j) = g_j\}$, with M number of p_j with labelling $j \in -M, -M+1, .., M$. For a simple case, let us assume our scalar function is a function of only the x coordinate $\phi = \phi(x)$. With the Lagrange interpolation formula, the function with arbitrarily high order approximations will be

$$\phi(x) = \sum_{j=-M}^{M} \phi(x_j) \prod_{l=-M, l\neq j}^{M} \left(\frac{x - x_l}{x_m - x_l} \right) \tag{29}$$

The second derivative of Eq (29) will give us the Laplacian of ϕ. Writing explicity with uniform lattice $x_j = jh$ for $j \in \mathbb{Z}$, the point at $x = x_0$ will be:

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{-1}{h^2} \left[2\phi(x_0) \sum_{l=1}^{M} \frac{1}{l^2} \right. \tag{30}$$

$$\left. - \sum_{j=1}^{N} \frac{\phi(x_j) + \phi(x_{-j})}{h^2} \prod_{l=-M, l\neq j}^{M} \left(\frac{l^2}{l^2 - j^2} \right) \right] \tag{31}$$

One can check that at $M = 1$ corresponds to the standard second order approximation of the Laplacian. After such approximations, smoothness and error analysis can be performed, or actual experiments can be run to verify the approximations.

C. Optimisation

There is a large room for improvement for the HHL algorithm. For instance from [22], it is suggested that one could modify the HHL circuit from Qiskit with less gates, depth and width with higher fidelity. However, one would need to check for systems of larger than 4x4, as this paper only provided tests and examples of 2x2 and 4x4. [23] applied a hybrid quantun linear solver using HHL and fast-matrix inversion. Moreover, [24] suggests that the HHL algorithm can be improved via improving the κ dependence by having variable stopping times. One of this being the variable time amplitude amplification (VTAA) scheme proposed by [25]. Another method suggested will be improving on the ϵ dependency of the algorithm. Following the above, [24] suggests also a variational based method for solving linear systems. [26] solved linear systems of equations using a hybrid iterative phase estimation algorithm (HIPEA), which achieves solving such equations when number of qubits are limited.

There may be possible classical methods which can solve our problem. For instance a new emerged artificial intellegnece technology developed by Deep Mind, named AlphaTensor, can discover algorithms for matrix multiplication, and potentially matrix inversion [27]. Such algorithms can then help us solve equations of the form of Eq (8) without the use of quantum algorithms.

D. Complexity analysis and Speedups

Complexity and runtime analysis must be done to verify that quantum algorithms are more efficient than classical ones. A well-known example is Shor's algorithm, which achieves integer factorisation in polynomial time [28]. This algorithm is important in cryptosystems. We introduce some basic ideas of complexity and speedups.

For time complexity, let us use a polynomial case for example: $T(n) = \Theta(n^t), t \in \mathbb{Z}^+$. There is a lower bound and upper bound for such complexty: $\exists c_1, c_2$, s.t. $c_1 n^t < T(n) < c_2 n^t$ where $c_1, c_2 \in \mathbb{R}$ are some constants. The lower and upper bound are defined as:

$$\Omega(n^t) = c_1 n^t < T(n) \tag{32}$$

$$\mathcal{O}(n^t) = T(n) < c_2 n^t \tag{33}$$

Quantum speedups can be grouped into 3 categories:

	Classical case	Quantum case
Quadratic	$\mathcal{O}(n)$	$\mathcal{O}(\sqrt{n})$
	$\mathcal{O}(2^n)$	$\mathcal{O}(2^{n/2})$
Polynomial	$\mathcal{O}(n^{15})$	$\mathcal{O}(n^2)$
	$\mathcal{O}(n^3)$	$\mathcal{O}(n^{3/2})$
Exponential	$\mathcal{O}(2^n)$	$\mathcal{O}(n)$
	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$

TABLE I. Examples of quantum speedups

Another area of analysis will be gate complexity. One may find the following link useful: `https://quantumalgorithmzoo.org/`, which describes the algorithm and its speedups, together with links to relevant works.

E. Considering resources

Simulators such as the IBM Qiskit ones do provide simulations of up to 5000 qubits, but at the moment there may not be enough qubits physically to run the algorithm. We first encounter the storage and loading problem. In our finite difference construction, if one has to discretize the problem finely to maximise the accuracy, the size of $|b\rangle$ and subsequently A, will be, qualitatively speaking, extremely large. Encoding the information of A and $|b\rangle$ classically first will be a memory problem. Then, loading $|b\rangle$ into the quantum circuit will be a loading problem. It is unsure that the method used in [16] is the ideal way to tackle this. Moreover for non-Hermitian matrices, one must convert the matrix to the form of (9). The classical memory problem will be more severe in this case. The subsequent system will be twice as large as the original and on the other hand, more qubits are required to store eigenvalues of $\hat{A}$ and to store $\vec{y}$. The exact growth of the number of qubits with the system parameters are still unknown. As per the previous sections, further resource analysis and optimisation should be conducted. Furthermore, the number of qubits required to solve the problem may not be physically possible at the moment. For instance in the IBM Quantum Scaling roadmap [29], processors of over 1000 qubits are yet to come until after 2023.

VI. CONCLUSION

In this report, we introduced the model we wish to solve. We reviewed the HHL algorithm and its implementations. We proposed the methods one can solve the problem and evaluated the results we had. Unfortunately at this stage, we cannot use our software to solve the Helmholtz equation. However, we gave some future directions and discussed room for improvement at the end.

All in all, we aim to develop a software package which simulates sound wave propagation for medical applications. We hope that this theoretical and computational work can be furthered and verified with actual experiments, and implemented

Although the project is yet to be completed, it does show many gaps of research, delayed to future work.

ACKNOWLEDGMENTS

I would like to thank the Faculty of Mathematical and Physical Sciences at University College London as well as my supervisor, Dr. Reza Haqshenas, for selecting me to this project, which is an enjoyable and rewarding experience. I would like to again express my gratitude to Dr. Haqshenas for his hard work, guidance and discussions during the project. On the other hand, I want to thank all the staff members and colleagues from the UCLQ summer school 2022 for the lessons, discussions and practical work throughout that week of intensive training, which made this project run more smoothly.

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

[2] F. Phillipson and H. S. Bhatia, Portfolio optimisation using the d-wave quantum annealer (2020).

[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature **549**, 195 (2017).

[4] W. J. Huggins, B. A. O'Gorman, N. C. Rubin, D. R. Reichman, R. Babbush, and J. Lee, Unbiasing fermionic quantum monte carlo with a quantum computer, Nature **603**, 416–420 (2022).

[5] V. von Burg, G. H. Low, T. Häner, D. S. Steiger, M. Reiher, M. Roetteler, and M. Troyer, Quantum computing enhanced computational catalysis, Phys. Rev. Research **3**, 033055 (2021).

[6] J. Allcock, A. Vangone, A. Meyder, S. Adaszewski, M. Strahm, C.-Y. Hsieh, and S. Zhang, The prospects of monte carlo antibody loop modelling on a fault-tolerant quantum computer, Frontiers in Drug Discovery **2**, 10.3389/fddsv.2022.908870 (2022).

[7] K. J. Satzinger, Y.-J. Liu, A. Smith, C. Knapp, M. Newman, C. Jones, Z. Chen, C. Quintana, X. Mi, A. Dunsworth, C. Gidney, I. Aleiner, F. Arute, K. Arya, J. Atalaya, R. Babbush, J. C. Bardin, R. Barends, J. Basso, A. Bengtsson, A. Bilmes, M. Broughton, B. B. Buckley, D. A. Buell, B. Burkett, N. Bushnell, B. Chiaro, R. Collins, W. Courtney, S. Demura, A. R. Derk, D. Eppens, C. Erickson, L. Faoro, E. Farhi, A. G. Fowler, B. Foxen, M. Giustina, A. Greene, J. A. Gross, M. P. Harrigan, S. D. Harrington, J. Hilton, S. Hong, T. Huang, W. J. Huggins, L. B. Ioffe, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, T. Khattar, S. Kim, P. V. Klimov, A. N. Korotkov, F. Kostritsa, D. Landhuis, P. Laptev, A. Locharla, E. Lucero, O. Martin, J. R. McClean, M. McEwen, K. C. Miao, M. Mohseni, S. Montazeri, W. Mruczkiewicz, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, T. E. O'Brien, A. Opremcak, B. Pató, A. Petukhov, N. C. Rubin, D. Sank, V. Shvarts, D. Strain, M. Szalay, B. Villalonga, T. C. White, Z. Yao, P. Yeh, J. Yoo, A. Zalcman, H. Neven, S. Boixo, A. Megrant, Y. Chen, J. Kelly, V. Smelyanskiy, A. Kitaev, M. Knap, F. Pollmann, and P. Roushan, Realizing topologically ordered states on a quantum processor, Science **374**, 1237 (2021), https://www.science.org/doi/pdf/10.1126/science.abi8378.

[8] J. E. Katz, R. I. Clavijo, P. Rizk, and R. Ramasamy, The basic physics of waves, soundwaves, and shockwaves for erectile dysfunction, Sexual Medicine Reviews **8**, 100–105 (2020).

[9] G. ter Haar, Therapeutic applications of ultrasound, Progress in Biophysics and Molecular Biology **93**, 111 (2007), effects of ultrasound and infrasound relevant to human health.

[10] P. C. S. Costa, S. Jordan, and A. Ostrander, Quantum algorithm for simulating the wave equation, Phys. Rev. A **99**, 012323 (2019).

[11] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum

algorithm for linear systems of equations., Physical Review Letters **103**, 150502 (2009).

[12] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, Quantum algorithms revisited, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**, 339 (1998).

[13] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

[14] Ibm quantum (2016), `https://quantum-computing.ibm.com/composer/docs/iqx/manage/systems/`.

[15] Ibm quantum compute resources (2016), `https://quantum-computing.ibm.com/composer/docs/iqx/manage/systems/`.

[16] R. Iten, R. Colbeck, I. Kukuljan, J. Home, and M. Christandl, Quantum circuits for isometries, Phys. Rev. A **93**, 032318 (2016).

[17] M. Baer, findiff software package (2018), `https://github.com/maroba/findiff`.

[18] Hhl qiskit 0.39.2 documentation.

[19] T. Q. Team, Solving linear systems of equations using hhl (2022).

[20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods **17**, 261 (2020).

[21] D. T. Colbert and W. H. Miller, A novel discrete variable representation for quantum mechanical reactive scattering via the s-matrix kohn method, J. Chem. Phys. **96**, 1982 (1992).

[22] M. Zhang, L. Dong, Y. Zeng, and N. Cao, Improved circuit implementation of the HHL algorithm and its simulations on QISKIT., Scientific Reports **12**, 13287 (2022).

[23] M. R. Perelshtein, A. I. Pakhomchik, A. A. Melnikov, A. A. Novikov, A. Glatz, G. S. Paraoanu, V. M. Vinokur, and G. B. Lesovik, Solving large-scale linear systems of equations by a quantum hybrid algorithm, Annalen der Physik **534**, 2200082 (2022).

[24] D. Dervovic, M. Herbster, P. Mountney, S. Severini, N. Usher, and L. Wossnig, Quantum linear systems algorithms: a primer, arXiv 10.48550/arxiv.1802.08227 (2018).

[25] A. Ambainis, Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations (2010).

[26] F. Gao, G. Wu, M. Yang, W. Cui, and F. Shuang, A hybrid algorithm to solve linear systems of equations with limited qubit resources, Quantum Information Processing **21**, 111 (2022).

[27] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, and P. Kohli, Discovering faster matrix multiplication algorithms with reinforcement learning, Nature **610**, 47 (2022).

[28] S. Beauregard, Circuit for shor's algorithm using 2n+3 qubits 10.48550/ARXIV.QUANT-PH/0205095 (2002).

[29] J. Gambetta, Ibm's roadmap for scaling quantum technology (2022).