



# Article A Variational Quantum Linear Solver Application to Discrete Finite-Element Methods

Corey Jason Trahan <sup>1,\*</sup>, Mark Loveland <sup>1</sup>, Noah Davis <sup>2</sup> and Elizabeth Ellison <sup>1</sup>

- <sup>1</sup> Information and Technology Laboratory, U.S. Army Engineer Research and Development Center, Vicksburg, MS 39180, USA
- <sup>2</sup> Applied Research Laboratories, The University of Texas at Austin, Austin, TX 78713, USA
- \* Correspondence: corey.j.trahan@erdc.dren.mil

Abstract: Finite-element methods are industry standards for finding numerical solutions to partial differential equations. However, the application scale remains pivotal to the practical use of these methods, even for modern-day supercomputers. Large, multi-scale applications, for example, can be limited by their requirement of prohibitively large linear system solutions. It is therefore worthwhile to investigate whether near-term quantum algorithms have the potential for offering any kind of advantage over classical linear solvers. In this study, we investigate the recently proposed variational quantum linear solver (VQLS) for discrete solutions to partial differential equations. This method was found to scale polylogarithmically with the linear system size, and the method can be implemented using shallow quantum circuits on noisy intermediate-scale quantum (NISQ) computers. Herein, we utilize the hybrid VQLS to solve both the steady Poisson equation and the time-dependent heat and wave equations.

**Keywords:** quantum computing; quantum variational algorithm; finite-element methods; Poisson equation; heat equation; quantum algorithms

# 1. Introduction

Quantum computing has reached a new era where theory is transitioning into practice as quantum computers and simulators become more widespread and available to the scientific community. This transition has encouraged algorithmic exploration, with an intent toward showing "quantum supremacy" or "quantum advantage". Quantum advantage refers to the demonstrated and measured success in processing a real-world problem faster on a quantum computer than on a classic computer. Quantum supremacy [1], on the other hand, refers to the demonstrated and measured ability to process **any problem** faster on a quantum computer, regardless of its real-world applicability [2].

In 2019, Arute et al. [3] claimed to have achieved quantum supremacy using a programmable superconducting processor by "performing a series of operations in 200 s that would take a supercomputer about 10,000 years to complete". In December 2020, a group based out of the University of Science and Technology of China (USTC) led by Jian-Wei Pan claimed quantum supremacy by implementing Gaussian boson sampling on 76 photons with their photonic quantum computer [4]. The paper states that to generate the number of samples the quantum computer generates in 20 s, a classical supercomputer would require 600 million years of computation. Although these supremacy claims have been the source of much recent debate, mostly with respect to whether or not their classical comparisons are the most efficient, it is clear that we are on the threshold of a new age of computation, heralded by today's noisy intermediate-scale quantum (NISQ) hardware.

Today's NISQ computers are limited in scalability because they are (1) subject to noise and thus not fault-tolerant, and (2) they are qubit-limited (usually meaning less than 100 qubits). Regarding the latter, however, the number of qubits on modern day quantum computers is rapidly growing, with IBM projecting a remarkable 1121 qubit system in



Citation: Trahan, C.J.; Loveland, M.; Davis, N.; Ellison, E. A Variational Quantum Linear Solver Application to Discrete Finite-Element Methods. *Entropy* 2023, *25*, 580. https:// doi.org/10.3390/e25040580

Academic Editor: Rosario Lo Franco

Received: 21 February 2023 Revised: 21 March 2023 Accepted: 22 March 2023 Published: 28 March 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https:// creativecommons.org/licenses/by/ 4.0/). 2023. Although this exponential qubit growth is vital in the near term, "shot" noise arising form the Heisenberg uncertainty principle and zero-point thermal fluctuations cause a phenomenon called decoherence, which may ultimately prevent scalability to larger qubit applications. Quantum systems achieve their notable advantage over classical ones via entanglement, a process by which a pure state quantum system develops a probability distribution over multiple classical outcomes. Entanglement gives quantum computers the ability to process and store exponentially more information than a classical computer. Noise, however, introduces errors that cause decoherence in the entanglement and can significantly degrade the performance of NISQ computers [5–8]. In fact, much of today's quantum computing efforts are in noise mitigation [9–14]. In January 2022, for example, a group of scientists from the University of Chicago and Purdue University collaborated on a new promising noise control technique: Instead of directly trying to measure the noise, they constructed a unique "fingerprint" of the noise on a quantum computer as it was seen by a program run on the computer [15]. This approach shows promise for mitigating the noise problem, as well as suggesting ways that users could actually turn noise into an advantage.

Despite these drawbacks, NISQ computers remain promising in application areas such as quantum chemistry, cybersecurity, drug development, financial modeling, traffic optimization, weather forecasting, climate change prediction, artificial intelligence and machine learning. Over the last few years, quantum hardware has become available to the average researcher, mostly through two types of cloud computing. The first type is cloud services providing access to a single company's collection of quantum devices. The Qiskit cloud service offered by IBM Quantum [16] is the premier example of this. On the other hand, there are multi-platform services such as Amazon Braket [17] that work as intermediaries to give users options to access quantum devices owned by multiple vendors. In most cases, cloud computing interfaces for quantum devices are implemented in Python to provide starting points for accessing working quantum devices. Introductory resources for algorithmic understanding and design are also widely available to the public. For example, the IBM Qiskit textbook [18] provides a college-level introduction to quantum information with integrated programming exercises, the Codebook by Xanadu [19] provides an introductory course built around the Pennylane package, allowing for differentiable programming of quantum computers, and QBraid is an online platform for developing quantum software with introductory quantum tutorials [20].

As near-term supremacy does not mean utility, many today utilize these current cloud resources for the investigation of quantum advantages for practical problems. NISQ computers must be restricted to "shallow" circuits for noise control. These circuits have a minimal number of qubits that are more easily controlled. One way of keeping quantum circuits shallow, for example, is by combining quantum and classical algorithms so that only the computationally intensive portion of the problem is implemented on the quantum computer, thereby offering some degree of quantum speed-up or advantage while maintaining shallow circuits amenable to NISQ computers. This type of hybrid set-up is somewhat analogous, for example, to classical GPU acceleration. Recently, hybrid methods such as these have been utilized for near-term acceleration of machine learning and optimization problems [21–29]. A number of quantum algorithms for machine learning are based on the idea of amplitude encoding, which associates the amplitudes of a quantum state with the inputs and outputs of computations [24,30,31]. Since a state of *m* qubits is described by  $2^m$  complex amplitudes, this information encoding can allow for an exponentially compact representation. Intuitively, this corresponds to associating a discrete probability distribution over binary random variables with a classical vector. The goal of algorithms based on amplitude encoding is to formulate quantum algorithms whose resources grow polynomially in the number of qubits *m*, which amounts to a logarithmic time complexity in the number of amplitudes and therefore the dimension of the input.

Many quantum machine learning algorithms are based on variations in the quantum algorithm for linear systems of equations [32] (colloquially called HHL after the paper's

authors) which, under specific conditions, perform a matrix inversion using an amount of physical resources growing only logarithmically in the dimensions of the matrix. One of these conditions is that a Hamiltonian which, entry-wise, corresponds to the matrix can be simulated efficiently, which is known to be possible if the matrix is sparse [33] or low in rank [23]. Quantum matrix inversion can be applied to machine learning methods in which the training reduces to solving a linear system of equations, such as in least squares linear regression [30,31], the least squares version of support vector machines [24], and Gaussian processes [34].

For suitably conditioned linear systems, the HHL algorithm scales logarithmically in n, suggesting the possibility of exponential speed-up over classical systems [32], which holds promise for quantum computers beyond the NISQ era. In today's NISQ machines, however, shot noise has dramatically limited the size of the linear systems directly solvable by the HHL algorithm. To date,  $2 \times 2$  systems have been solved by superconducting qubits [35,36], nuclear magnetic resonance [37], and photonic devices [38,39]. The largest system solved on a gate-based computer was an  $8 \times 8$  problem using NMR [40].

Given today's NISQ limitations of the HHL algorithm, an alternative method for linear system solution has been proposed to gain a quantum advantage: variational hybrid quantum-classical algorithms (VHQCAs). VHQCAs are capable of providing an advantage to Shor's algorithm for factoring [41] and have gained momentum in the fields of chemistry [42–45], simulation [46–50], data compression [51], state diagonalization [52–54], compiling [55,56], quantum foundations [57], fidelity estimation [58], and meteorology [59]. The general VHQCA algorithm reduces the quantum circuit depth by using a classical optimizer and only evaluating the cost/objective function on the quantum computer.

In this study, we continue to investigate quantum advantages in classical problems by utilizing a VHQCA recently introduced by Bravo-Prieto et al. [60,61] called the variational quantum linear solver (VQLS) to obtain finite-element solutions to the Poisson, heat, and wave equations. The quantum/classical hybrid VQLS is a method for solving linear systems on near-term quantum computers which variationally prepares a quantum state  $|x\rangle$  such that  $\mathbf{A} |x\rangle \propto |b\rangle$ . Bravo-Prieto et al. were able to derive a meaningful termination condition for VQLS that allows one to guarantee a desired solution precision with efficient quantum circuits to estimate the variational cost function *C* while providing evidence for the classical hardness of its estimation. Using Rigetti's quantum computer, the VQLS was used for solutions up to a problem size of  $1024 \times 1024$  (10 qubits), which is the largest implementation of a linear system on quantum hardware to date. The time complexity of the VQLS was heuristically found to scale efficiently with the linear solution precision  $\epsilon$ , the matrix condition number  $\kappa$ , and the linear system size N.

# 2. The Variational Quantum Linear Solver

The quantum/classical hybrid VQLS [60,61] algorithm attempts to find a solution to the linear system such that  $\mathbf{A} | x \rangle \propto | b \rangle$  by minimizing a scalar cost function based on the scaled projection of  $\mathbf{A} | x \rangle$  onto  $| b \rangle$ . The solution vector  $| x \rangle$  is approximated with a wave function created through a quantum circuit ansatz. To prepare a linear system for VQLS solution, the matrix  $\mathbf{A}$  must be expressed as a linear combination of universal quantum gates. Additionally, the right-hand side (RHS) of the linear system must be transformed into a normalized quantum state  $| b \rangle$ , which can be generated by unitary operations *U* applied to the ground state of some number of qubits. We now discuss these elements of the VQLS in detail.

#### 2.1. The Variational Ansatz

In the VQLS algorithm,  $|x\rangle$  is prepared by acting on the  $|0\rangle$  state with a trainable gate sequence  $V(\alpha)$ . The ansatz  $V(\alpha)$  can be expressed in terms of *L* gates from a gate alphabet  $A = G_k(\alpha)$  as

$$V(\alpha) = G_{k_1}(\alpha_L) \cdots G_{k_i}(\alpha_i) \cdots G_{k_1}(\alpha_1)$$
<sup>(1)</sup>

Here,  $\vec{k} = (k_L, \dots, k_1)$  identifies the types of gates and their placement in the circuit (i.e., on which qubit they act), while  $\alpha$  represents the continuous parameters over which optimization occurs. All results presented herein are based on a "fixed ansatz", where  $\vec{k}$  is fixed over time and *V* is only optimized over  $\alpha$ . Though it was not investigated in this study, variable ansatz optimization was shown to improve convergence in some cases in [52,62].

Training of the ansatz is performed layer by layer, just as in neural networks. The number of layers is decided by the user. Although the solution function space widens as the layers are increased, over-determined parameter optimization may become difficult and inefficient. The properties of a good ansatz are as follows: (1) the circuit is shallow, minimizing decoherence, (2) it has minimal optimization parameters, and (3) the ansatz should span the space where the solution lives. Of all the layer structures we tested, the ansatz given in [60] (shown in Figure 1) was the most optimal one. This ansatz begins with an initial y rotation (Ry) of each qubit before moving on to the layered portion of the circuit.



Figure 1. A four-qubit example of the fixed ansatz used for this study.

Each layer starts with alternating controlled-z (CZ) gates followed by Ry rotations on the controlled qubits. The CZ gates have the crucial function of entangling the qubits, which allows for an exponentially larger space representation than a purely classical cost evaluation. The Ry gates allow one to "search" the state space by varying the rotational parameters.

In this study, a range of layers were tested for each application of the VQLS. Some general guidelines for choosing the number of layers were found: (1) a greater number of layers was needed, as the problem's dimensionality was increased (resulting in larger linear systems), and (2) a greater number of layers was required, as the number of terms in the Pauli decomposition of the stiffness matrix grew. These two factors greatly limited the size of the finite-element problems we could test at this time to a maximum of 10 nodes (8 internal nodes or 3 qubits).

#### 2.2. Matrix Pauli Decomposition

In order to solve the linear system using the VQLS, the matrix must be represented as a linear combination of Hermitian unitary operators  $\mathbf{A} = \sum_i c_i \mathbf{U}_i$ , representing a system Hamiltonian where  $U_i$  represents the unitaries and  $c_i$  represents complex coefficients. Additional assumptions are that the matrix condition number  $\kappa < \inf$  and  $\|\mathbf{A}\| \leq 1$  and that the  $\mathbf{A}_i$  unitaries can be implemented with efficient quantum circuits. Typically, this decomposition consists of a linear combination of Kronecker products of the Identity and Pauli matrices, as these gates are widely used and recognized. These matrices and gates are defined as follows:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \ \mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \ \mathbf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \ \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$
(2)

For all application matrices herein, a recently proposed algorithm given in [63], which takes a square real symmetric matrix of an arbitrary size and decomposes it into a tensor product of Pauli spin matrices, was used. The routine was given by the authors in Python and is publicly available. The mathematical procedure for generating this decomposition

for a general-sized stiffness matrix, often encountered in discrete finite-element methods, is given in Appendix A.

#### 2.3. Right-Hand Side Preparation

The VQLS requires that the linear system RHS be transformed into a normalized quantum state  $|b\rangle$  generated by some series of unitary operations **U** applied to the ground state of the qubits:

$$|b\rangle = \mathbf{U} |0\rangle \tag{3}$$

Again, we assume that **U** can be efficiently implemented with a quantum circuit. For example, if the boundary conditions are homogeneous, and a reduced linear system is used which includes only the internal domain grid points, then the constant RHS wave function can be created by a quantum circuit which applies a Hadamard gate to each qubit:

$$b\rangle = (\mathbf{H}_0 \mathbf{H}_1 \mathbf{H}_1 \cdots \mathbf{H}_{m-1}) |0\rangle \tag{4}$$

where *m* is the total number of qubits used to represent the reduced system. In general, however, the RHS vector of the linear systems will not be constant, and a vector-specific circuit must be generated. For the applications herein, we utilized the "isometry" package in Qiskit to produce the corresponding quantum state from a specific RHS vector. It is worth noting that more general, non-constant RHSs may lead to deeper, more complex circuitry that may affect the VQLS's efficiency, since this circuit is evaluated in a controlled manner during each cost calculation.

### 3. Computational Details

The VQLS in this study was implemented in Python using IBM's Qiskit [16]. Qiskit is an open source software development kit for working with OpenQASM and the IBM Q quantum processors. For prototypical applications, such as those needed for the early stages of this work, Qiskit offers a quantum computer simulator which allows the user to build and test quantum circuits on a local machine without the need for a quantum computer. The Qiskit package, along with its statevector simulator, can be imported into a Python script in the usual way. For all problems in this study, the Qiskit Aer simulator backend was used.

## 4. Training Algorithm

Scientific Python (SciPy) offers a variety of options for both constrained and unconstrained optimization of scalar objective/cost functions. The purpose of these optimizers is to update the parameters of the VQLS ansantz. Generally speaking, multi-variant objection function optimizers fall into two categories: gradient- and non-gradient-based optimization. Gradient-based methods, such as the Newton conjugate gradient method, use the objective function gradients (i.e., Jacobians or Hessians) to move in a descending direction toward a minima. Non-gradient methods, on the other hand, work by iteratively approximating the actual constrained optimization problem with linear programming problems. During an iteration, an approximating linear programming problem is solved to obtain a candidate for the optimal solution. The candidate solution is evaluated using the original objective and constraint functions, yielding a new data point in the optimization space. This information is used to improve the approximating linear programming problem used for the next iteration of the algorithm. When the solution cannot be improved anymore, the step size is reduced, refining the search. When the step size becomes sufficiently small, the algorithm finishes.

Previous studies have compared gradient- and non-gradient based optimization for a range of VQLS applications using quantum simulators, quantum simulators with shot noise, and fully quantum applications. In particular, in [64], it was shown that once shot noise is included in either the statevector simulator or real quantum application, gradientbased optimizers do not offer much of an advantage over non-gradient optimizers. A popular choice for VQLS applications, for example, is the non-gradient based constrained optimization by linear approximation (COBYLA) method. Due to these previous findings and the complexity of including the objective function gradients, the COBYLA method was used for all applications herein, and gradient-based methods were not investigated.

#### 5. Applications

5.1. Application 1: The Poisson Equation

For the first application, the QVA was used to solve the Dirichlet problem for the 1*D* Poisson equation, given in strong form by

$$-\triangle u(x) = f(x), \ u(x) \in \Omega, \tag{5}$$

where  $u(0) = u_L$  and  $u(1) = u_R$ . The equivalent weak representation of this equation is obtained by taking Equation (5) and multiplying it by an arbitrary test function in the appropriate function space, followed by integrating by parts [65] to give

$$\int_{u_L}^{u_R} \frac{d\phi}{dx} \frac{du}{dx} dx = \int_{u_L}^{u_R} \phi f(x) \, dx \quad \forall \phi \in H^1_0(\Omega) \tag{6}$$

Here,  $\phi(x)$  is the arbitrary test function in the appropriate Hilbert space, and the boundary term from integrating by parts vanishes since the test space  $H_0^1$  has 0 trace. To discretize this equation, the standard Galerkin approximation with linear Lagrange polynomials is used on a uniform 1D grid of N points, where the *i*th nodal location is given by  $x_i = ih$ . Here, h = 1/(N-1) and  $0 \le i \le (N-1)$ . Additionally, we define n = N-2 as the internal node count. This discretization results in the linear system

$$\mathbf{K}\vec{u} = \vec{f} \tag{7}$$

where for linear, Lagrangian basis function support, **K** is the typical tridiagonal "stiffness" matrix,  $\vec{u}$  is the solution vector, and  $\vec{f}$  is the right-hand side. When applying non-homogeneous Dirichlet boundary conditions, it is essential to manipulate this linear system to force the specified solution values on the domain endpoints, giving the following RHS:

$$\vec{f} = \begin{pmatrix} \int_0^1 \phi_1 f(x) \, dx + \int_0^1 \frac{\partial \phi_1}{\partial x} \frac{\partial \phi_0}{\partial x} \, u_L \, dx \\ \int_0^1 \phi_2 f(x) \, dx \\ \int_0^1 \phi_3 f(x) \, dx \\ \vdots \\ \int_0^1 \phi_n f(x) \, dx + \int_0^1 \frac{\partial \phi_n}{\partial x} \frac{\partial \phi_{n+1}}{\partial x} \, u_R dx \end{pmatrix}$$
(8)

For Dirichlet boundary conditions, a reduced system can be solved without the endpoints, since these are known. The reduced matrices were used for all applications herein to increase the grid resolution, since the qubit count was extremely limited. While obtaining the quantum wavefunction for the RHS of the homogeneous Poisson equation is relatively straightforward, heterogeneous boundaries or time-dependent solutions require more complex ways of calculating the RHS wavefunction on the fly. As mentioned, this was accomplished using Qiskit's Isometry package. An example for creating a wavefunction from an arbitrary vector *U* is as follows:

This circuit is shown in Figure 2.



**Figure 2.** A quantum circuit representing  $\vec{f}^T = [0.1, 2, 2, 2, 2, 2, 2, 0.1]$  found using Qiskit's Isometry command.

5.1.1. Poisson Case 1: Parabolic Solution with Homogeneous Boundary Conditions

For the first Poisson test, a manufactured quadratic solution for Equation (5) was used to simplify the RHS preparation. The solution was given by

$$u(x) = a + b(x - x_0)^2$$
(9)

where  $a = g - b(-x_0)^2$ , u(0) = u(1) = g and  $x_0 = 1/2$ . The RHS of Equation (5) then simplifies to a constant -2b. For homogeneous boundary conditions, where g = 0, the reduced RHS of Equation (7) can be written as

$$\vec{f} = h[-2b - 2b - 2b - 2b - 2b]^T$$
(10)

where h is the uniform grid spacing and T is the transpose.

This linear system was solved using the fixed-ansatz VQLS as described with the Pauli decomposition given in Appendix A and the right-hand side preparation detailed in Section 2.3. For the quantum simulator results without shot noise, errors arose only from discretization of spatial derivatives and the VQLS optimization. The number of qubits *m* in the VQLS determines the grid resolution such that the total number of nodes is  $N = 2^m + 2$ . Our attempts at optimization for anything greater than three qubits (eight nodes) took too long to simulate on a serial machine. This low qubit count leads to very coarse grids and noticeable discretization errors. To properly converge the discrete problem, finer grids were needed. All ansatz parameters were initialized randomly between  $-\pi <= \theta_k <= \pi$ , default optimizer tolerances of  $10^{-4}$  were used, and the initial change to the variables in the COBYLA optimizer was set to *rhobeg* =  $\pi$ .

For the two-qubit homogeneous Poisson application, there were four internal and six total finite element nodes. The convergence results for the VQLS for a range of ansatz layers can be seen in Figure 3. These results were averaged over 20 runs, with solid lines indicating the average and variances shown with vertical bars. The two-qubit linear system's stiffness was a  $4 \times 4$  matrix. This figure shows that 2 layers were sufficient to successfully capture the solution to the default tolerance within 100 optimization steps. Note that the total number of optimization parameters  $N_{\theta}$  varied as  $N_{\theta} = m + 2(m-1)(nlayers - 1)$ , so for a two-qubit and two-layer network, there were four parameters to span the solution space. This figure shows that as the number of layers or parameters increased, the optimization converged slower, though still relatively fast when compared with the three-qubit problem. This was expected, however, since the solution test space dimensionality was increasing, and the variational algorithm had to span this space. For all layer cases, full solution convergence was achieved within the COBYLA tolerance using the statevector simulator in less than 100 iterations. Figure 4 plots the wall clock time in seconds versus the number of layers averaged over the 20 runs for the 2 qubit problem. From this figure, it is seen that the time it took to converge the solution was linearly proportional to the number of layers used in the variational ansatz.



**Figure 3.** Two-qubit VQLS cost function results for the reduced Poisson problem with homogeneous Dirichlet boundary conditions. The results were averaged over 20 trial runs. Variances are shown by respective bars.



**Figure 4.** Wall clock time in seconds versus the number of layers for the two-qubit VQLS reduced Poisson problem with homogeneous Dirichlet boundary conditions.

Figure 5 displays the cost function of the 3 qubit statevector solution averaged over 10 runs. For this case, there were 8 internal nodes and 10 total, and the linear system stiffness was an  $8 \times 8$  matrix. While the two-qubit results converged relatively fast for a small number of layers, this was not the case for the three-qubit application. Additionally, it took 4 or more layers for the cost function to converge within 1000 iterations. An interesting note from this figure is that the even-numbered layers performed notably better than the odd layers, with six layers converging in the least amount of time and most accurately. This can be seen more clearly in Figures 6 and 7, which display solution results and the grid root mean square errors averaged over all runs for each layer, respectively. Lastly, Figure 8 displays the time in seconds averaged over all 10 runs for each layer. Since the three-layer run never fully converged within the COBLYA default tolerance, it took the longest. All layers greater than three once again showed a linear increase in time as the layers were incremented.



**Figure 5.** Three-qubit VQLS cost function results for the reduced Poisson problem with homogeneous Dirichlet boundary conditions. The results were averaged over 10 trial runs. Variances are shown by respective bars.

Figure 6 displays the VQLS versus the classical discrete solution for the three-qubit, eight-internal node problem. In this figure, we see the VQLS solution growing in accuracy as the number of ansatz layers is increased, as expected. In the right column, the VQLS solutions are plotted along with the analytic system solution. Note that the VQLS solution here is being compared to the discrete finite-element solution, and thus both include discretization errors which are not shown.



**Figure 6.** Three-qubit (eight-node) VQLS results (filled circles with dashed lines) for reduced Poisson problem with homogeneous Dirichlet boundary conditions. The classical discrete solution is shown with a solid black line.

For the VQLS results in Figures 3–8, a Qiskit statevector simulator was used so that the full wave function was known, eliminating measurement and sample errors from the

convergence figures. For a quantum calculation, however, measurements are necessary, and sampling errors can affect the classical optimizer convergence. Measurements occurred in the Hadamard tests of the cost calculations. Figure 9 shows the COBYLA cost convergence as the number of shots was increased. It was found that to achieve accurate and smooth convergence, at least 100,000 shots were needed for the 2 qubit VQLS system.



**Figure 7.** The root mean squared solution error versus the number of layers for the three-qubit VQLS reduced Poisson problem with homogeneous Dirichlet boundary conditions. Here, the errors were averaged over all 10 runs for each layer.



**Figure 8.** Wall clock time in seconds versus the number of layers for the three-qubit VQLS reduced Poisson problem with homogeneous Dirichlet boundary conditions. The three-layer run does not converge.



Figure 9. The COBYLA cost convergence for a range of shots in the two-qubit VQLS reduced Poisson problem with homogeneous Dirichlet boundary conditions.

Optimization steps

#### 5.1.2. Poisson Case 2: Cubic Solution with Non-Homogeneous B.C.

80

**Optimization steps** 

Next, we consider a non-symmetric cubic Poisson solution with non-homogeneous boundaries, which will further complicate the RHS vector as it modifies f to be

$$f_i = ah^2(6ih - 2) + u_i \quad i = 1, n \tag{11}$$

In this case,  $u_i = u_L$  for i = 1, and  $u_i = u_R$  for i = n. Note that normally, the RHS addition to the 1D case would be  $u_D/h$ , but both sides are multiplied by h in the discrete matrix solution.

The following cubic manufactured solution is used:

$$u(x) = a(-x^3 + x^2 + x + 1)$$
(12)

where a = 1 so that  $u_L = 1$  and  $u_R = 2$ . The two-qubit Qiskit wavefunction simulator was used to calculate the discretized, finite-element VQLS results for 2-6 ansatz layers to investigate the layer count sensitivities to accuracy and convergence. For each layer count, five runs were executed, and the mean and standard deviation of the runs were calculated. Once again, all ansatz parameters were initialized randomly between  $-\pi <= \theta_k <= \pi$ , default tolerances of  $10^{-4}$  were used, and the initial change to the variables in the COBYLA optimizer was set to *rhobeg* =  $\pi$ . Using these parameters, it was found that only two ansatz layers were needed to fully capture the solution, as can be seen in Figure 10. The best cost

convergence was also seen for two layers, shown in Figure 11. The cost curve variances did not show any obvious trend with the layer count.



**Figure 10.** Two-qubit, two-layer solution (filled circles) along with the analytic solution (solid line) of Case 2: the cubic Poisson problem.



**Figure 11.** VQLS mean cost versus iteration or optimization count over a range of layers for the cubic Poisson problem. Variances are shown as curve error bars.

## 5.2. Application 2: The Heat Equation

The Poisson test cases were time-independent and required only one linear solve for the solution. In this section, however, the VQLS results are presented for the 1D time-dependent heat equation

$$\partial_t - \partial_{xx} u = 0 \quad x \in (0, 1)$$

$$u(x_L, t) = u_L$$

$$u(x_R, t) = u_R$$

$$u(x, 0) = u_0$$
(13)

where  $x_L$  and  $x_R$  are the 1D domain endpoints. The weak form of this equation is

$$\int_{x_L}^{x_R} \partial_t u \,\phi \,dx + \int_{x_L}^{x_R} u_x \phi_x \,dx = 0 \quad \forall \phi \in H^1_0(0,1)$$
(14)

Discretizing in time with uniform time steps  $\Delta t$  and using the backward Euler approximation for the time derivative gives

$$\int_{x_L}^{x_R} u^{k+1} \phi \, dx - \int_{x_L}^{x_R} u^k \phi \, dx + \Delta t \int_{x_L}^{x_R} u_x^{k+1} \phi_x \, dx = 0 \tag{15}$$

where *k* is the discrete time step index such that k = 1, *nt* and *nt* is the total number of time steps. This equation is made to hold for all test functions, giving the following finite-element (FE) backward Euler matrix equation:

$$(\mathbf{M} + \Delta t \mathbf{K})\vec{u}^{k+1} = \mathbf{M}\vec{u}^k \tag{16}$$

For linear basis functions on a uniform grid of a spacing h, the matrix operators are

$$\mathbf{M} = \begin{bmatrix} \frac{2h}{3} & \frac{1h}{6} & 0 & 0 & \dots \\ \frac{1h}{6} & \frac{2h}{3} & \frac{1h}{6} & 0 & \dots \\ 0 & \frac{1h}{6} & \frac{2h}{3} & \frac{1h}{6} & 0 \dots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$
(17)

$$\mathbf{K} = \begin{bmatrix} \frac{2}{h} & -\frac{1}{h} & 0 & 0 & \dots \\ -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & 0 & \dots \\ 0 & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & 0 \dots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$
(18)

This equation gives a linear system  $\mathbf{A}\vec{x} = \vec{b}$  at each time step such that

$$\mathbf{A} = \begin{bmatrix} \frac{2h}{3} + \frac{2\Delta t}{h} & \frac{1h}{6} + \frac{-1\Delta t}{h} & 0 & 0 & \dots \\ \frac{1h}{6} + \frac{-1\Delta t}{h} & \frac{2h}{3} + \frac{2\Delta t}{h} & \frac{1h}{6} + \frac{-1\Delta t}{h} & 0 & \dots \\ 0 & \frac{1h}{6} + \frac{-1}{h} & \frac{2h}{3} + \frac{2\Delta t}{h} & \frac{1h}{6} + \frac{-1\Delta t}{h} & 0 \dots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$
(19)

and  $\vec{b} = \mathbf{M}\vec{u}^k$ .

For verification of the FE-VQLS algorithm, a nonlinear solution was fabricated of the form

$$u(x,t) = \frac{1}{\sqrt{4\pi t}} exp(-\frac{(x-0.5)^2}{4t})$$
(20)

on the domain  $[0 \le x \le 1] \times [1 \le t \le 3]$ . A uniform grid was created with  $n = 2^m$  internal spatial grid points, N = n + 2 total spatial grid points, and nt = 11 time points. Figure 12 shows the two-qubit results (dashed lines and open circles) plotted against the analytic



solution (solid line). For these results, three layers were used, and *rhobeg* in the COBYLA method was set to  $\pi/100$ .

**Figure 12.** Analytic solution (solid line) versus two-qubit VQLS-based finite element results (dashed line with open circles) for the time-dependent heat equation at each time step.

The results in this figure show excellent agreement between the FE-VQLS and the analytic solution. It should be noted, however, that in order to obtain these results, the FE-VQLS solution had to be scaled appropriately at each time step, since the quantum results were only proportional to the solution. This could be accomplished by using the boundary conditions if they were non-homogeneous, and the system was not solved in a reduced way. However, since the reduced systems were used herein, the non-homogeneous boundaries were not included, and the ratio of the analytic and FE-VQLS solution of the first internal point was used for the scaling.

At each time step, the previous VQLS ansatz parameters were used to initialize the minimization procedure and speed up convergence. Ideally, the number of COBYLA iterations should decrease in time. This was seen for the two-qubit solution, as shown in Figure 13.



Figure 13. The COBYLA iteration count over time for the two-qubit solution of the heat equation.

## 5.3. Application 3: The Wave Equation

For the last application, we present the VQLS results for a 1D wave equation of the form

$$\partial_{tt}u - \partial_{xx}u = 0 \quad x \in (0, 1)$$

$$u(x_L, t) = u_L$$

$$u(x_R, t) = u_R$$

$$u(x, 0) = u_0$$
(21)

where  $x_L$  and  $x_R$  are the endpoints of the 1D domain. The weak form of this equation is

$$\int_{x_L}^{x_R} \partial_{tt} u \,\phi \,dx + \int_{x_L}^{x_R} u_x \phi_x dx = 0 \quad \forall \phi \in H^1_0(0,1)$$
(22)

Discretizing in time with uniform time steps  $\Delta t$  and using a second-order difference approximation for the time derivative gives

$$\int_{x_L}^{x_R} u^{k+1} \phi \, dx - 2 \int_{x_L}^{x_R} u^k \phi \, dx + \int_{x_L}^{x_R} u^{k-1} \phi \, dx + \Delta t^2 \int_{x_L}^{x_R} u_x^{k+1} \phi_x \, dx = 0$$
(23)

Note here that we have treated the diffusion term implicitly. When applied to all test functions, this yields the matrix equation

$$(\mathbf{M} + \Delta t^2 \mathbf{K})\vec{u}^{k+1} = \mathbf{M}(2\vec{u}^k + \vec{u}^{k-1})$$
(24)

where  $\mathbf{A} = \mathbf{M} + \Delta t^2 \mathbf{K}$  and  $\vec{b} = \mathbf{M}(2\vec{u}^k + \vec{u}^{k-1})$ .

To test the VQLS, a non-separable solution of

$$u(x,t) = sin(x+t) \tag{25}$$

was used on the domain  $[0 \le t \le 1] \times [0 \le x \le 1]$ . A total of m = 2 qubits (n = 4 internal points) were used for the matrix-reduced internal solve with three ansatz layers. The time step was set to  $\Delta t = 0.1s$ . As can be seen in Figure 14, the VQLS results agreed well with the analytic solution, and it is noted that a majority of the differences came from discretization and not from the VQLS procedure. The time-dependent COBLYA iteration count, which essentially represents the time evolution regularity of the ansatz parameters, can be seen in Figure 15. This figure shows a large initial iteration count associated with random sampling and a decrease in iteration count for each linear solve as the solution converged over time.



**Figure 14.** Analytic solution (solid line) versus two-qubit VQLS-based finite element results (dashed line with open circles) for the time-dependent wave equation at each time step.



Figure 15. The COBYLA iteration count over time for the two-qubit solution of the wave equation.

## 6. Discussion

In this study, the variational quantum linear solver recently proposed by Bravo-Prieto et al. [60,61] was used to solve the linear systems obtained from finite-element discretization

of the time-independent Poisson and time-dependent heat and wave equations. Although the results presented focused on these equations, the tools of this effort can generally be used to solve any discretization of a partial differential equation that leads to a matrix solution. The key findings of this effort are that (1) the Qiskit Isometry command can be used to generate wavefunctions for arbitrary vectors, a vital component for solving timedependent right-hand sides, (2) the quantum/classical hybrid variational solver can be used as a potential "accelerator" for discrete finite-element problems, (3) the large number of sampling shots and  $N^2$  matrix gate Hadamard test evaluation requirements greatly affects qubit scalability and thus the finite element grid resolution, and (4) the minimization iteration count decreases over time as the solution converges, reflecting an ansatz parameter regularity. The latter point is particularly useful for initial value problems, where a set of initial ansatz parameters need only be found once and used thereafter.

Regarding scalability of the VQLS, although it was previously found in [61] that this method was scalable for up to  $1024 \times 1024$  (10 qubit)-sized systems, that was certainly not the case for the practical linear systems herein, where the matrix and RHS required deeper circuits. Since each term in the stiffness Pauli decomposition requires a Hadamard test against the other terms, this size of the linear combination (circuit depth) directly affects the efficiency of the VQLS algorithm. For the 2 qubit, 6 node systems, the scalability more closely resembled that presented in Bravo-Prieto's analysis, but the 4 qubit, 18 node systems required 16 Pauli terms in the linear combination, and the VQLS results converged too slowly to be practical.

Future work will include further investigation of (1) new ansatz and optimization options, (2) more efficient methods for creating arbitrary RHS vectors specifically for use in finite-element methods, and (3) the quantum hardware scalability and effect of quantum noise for applications of the FE-VQLS.

**Author Contributions:** Conceptualization, C.J.T.; Methodology, C.J.T. and M.L.; Validation, M.L.; Investigation, C.J.T., M.L., N.D. and E.E.; Data curation, E.E.; Writing—original draft, C.J.T. and M.L.; Writing—review & editing, C.J.T., M.L. and N.D.; Visualization, C.J.T.; Supervision, C.J.T.; Project administration, C.J.T.; Funding acquisition, C.J.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** N.D. would like to acknowledge support from the Applied Research Laboratories at the University of Texas at Austin.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

## Appendix A. Preparation of the Stiffness Matrix

A crucial bottleneck of methods that simulate linear algebra computations with the amplitudes of quantum states is state preparation, which often requires one to initialize a quantum system in a state whose amplitudes reflect the features of the entire dataset. Although efficient methods for state preparation are known for specific cases [66,67], this step easily hides the complexity of the task [68,69].

In order to solve the linear system either directly or variationally on a quantum computer, the stiffness matrix **K** must be represented as a linear combination of Hermitian unitary operators,  $\mathbf{K} = \sum_i c_i \mathbf{U}_i$ , representing a system Hamiltonian where  $U_i$  represents the unitaries and  $c_i$  represents the complex coefficients. Additionally, we assume that the matrix condition number  $\kappa < \inf \text{ and } \|\mathbf{A}\| \le 1$  and that the  $\mathbf{A}_i$  unitaries can be implemented with efficient quantum circuits. Typically, this decomposition consists of a linear combination of Kronecker products of the Identity and Pauli matrices, as these gates are widely used and recognized. These matrices and gates are defined as follows:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \mathbf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$
(A1)

To find this linear combination, the tridiagonal matrix can be expressed recursively. Ignoring the diagonal component  $I_{2^m}$ , suppose that we have an expression for the off-diagonal elements in the *m* qubit case

$$\mathbf{A}_{m} = \sum_{i=0}^{2^{m}-2} (|i\rangle \langle i+1| + |i+1\rangle \langle i|)$$
(A2)

which gives an expression such as

$$|0\rangle \langle 1| + |1\rangle \langle 2| + ... + |2^m - 2\rangle \langle 2^m - 1| + h.c.$$
 (A3)

Now, we write

$$\mathbf{I}_{2} \otimes \mathbf{A}_{m} = \left( \left| 0 \right\rangle \left\langle 0 \right| + \left| 1 \right\rangle \left\langle 1 \right| \right) \otimes \mathbf{A}_{m}$$

$$= \sum_{i=0}^{2^{m}-2} \left( \left| i \right\rangle \left\langle i+1 \right| + \left| i+1 \right\rangle \left\langle i \right| \right) + \sum_{i=2^{m}}^{2^{m+1}-2} \left( \left| i \right\rangle \left\langle i+1 \right| + \left| i+1 \right\rangle \left\langle i \right| \right)$$

$$= \mathbf{A}_{m+1} - \left( \left| 2^{m}-1 \right\rangle \left\langle 2^{m} \right| + \left| 2^{m} \right\rangle \left\langle 2^{m}-1 \right| \right)$$
(A4)

and thus

$$\mathbf{A}_{m+1} = \mathbf{I}_2 \otimes \mathbf{A}_m + \left( \left| 2^m - 1 \right\rangle \left\langle 2^m \right| + \left| 2^m \right\rangle \left\langle 2^m - 1 \right| \right) \tag{A5}$$

In the second line of Equation (A4), tensoring  $|1\rangle \langle 1|$  with the second sum puts a "1" bit ahead of every bit string, which shifts every index in the summand by 2<sup>*m*</sup>. Then, by comparison with Equation (A2), this line is simply  $\mathbf{A}_{m+1}$  but missing a term connecting the two tri-diagonal block submatrices of  $\mathbf{I}_2 \otimes \mathbf{A}_m$ .

As an m = 3 example, it is first easy to find the off-diagonal solution for m = 2, given by

$$\begin{aligned} \mathbf{A}_{2} &= \mathbf{I} \otimes \mathbf{X} + \frac{1}{2} (\mathbf{X} \otimes \mathbf{X} + \mathbf{Y} \otimes \mathbf{Y}) \\ &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{aligned}$$
(A6)

Using this result, Equation (A5) can be written as

$$\mathbf{A}_{3} = \begin{pmatrix} \mathbf{A}_{2} & 0\\ 0 & \mathbf{A}_{2} \end{pmatrix} + (|011\rangle \langle 100| + |100\rangle \langle 011|) \tag{A7}$$

Appendix A.1. Implementing the Recursion Using GHZ States

All that is needed now is an operator representing  $(|2^m - 1\rangle \langle 2^m| + |2^m\rangle \langle 2^m - 1|)$ . This turns out to be closely related to writing an m + 1 qubit GHZ state in terms of Pauli operators. The GHZ state  $|\psi\rangle = |0\rangle^{\otimes (m+1)} + |1\rangle^{\otimes (m+1)}$  operator  $|\psi\rangle \langle \psi|$  has two off-diagonal elements: one in the top left and one in the top right of the corresponding matrix. These elements can be permuting toward the center of the matrix with the operator

$$\mathbf{B}_{m+1} = |0\rangle \langle 2^{m+1} - 1| + |2^{m+1} - 1\rangle \langle 0|$$

$$= (|0\rangle \langle 1|)^{\otimes (m+1)} + (|1\rangle \langle 0|)^{\otimes (m+1)}$$
(A8)

Thus, we have

$$\begin{aligned} (\mathbf{X} \otimes \mathbf{I}_{2^{m}})\mathbf{B}_{m+1}(\mathbf{X} \otimes \mathbf{I}_{2^{m}}) &= (\mathbf{X} \otimes \mathbf{I}_{2^{m}}) \big( (|0\rangle \langle 1|)^{\otimes (m+1)} + (|1\rangle \langle 0|)^{\otimes (m+1)} \big) (\mathbf{X} \otimes \mathbf{I}_{2^{m}}) \\ &= (|10...0\rangle \langle 01...1| + |01...1\rangle \langle 10...0|) \\ &= |2^{m}\rangle \langle 2^{m} - 1| + |2^{m} - 1\rangle \langle 2^{m}| \end{aligned}$$
(A9)

Now, using the results of (GUHNE 2007), the center shift operator can be written as

$$\mathbf{B}_{m} = (|0\rangle \langle 1|)^{\otimes m} + (|1\rangle \langle 0|)^{\otimes m}$$
(A10)

$$=\frac{1}{2^{m-1}}\sum_{t=0}^{\lfloor m/2 \rfloor} (-1)^t \sum_{\pi} \mathbf{S}_{\pi}(\mathbf{X}^{\otimes (m=2t)} \otimes \mathbf{Y}^{\otimes 2t})$$
(A11)

Here, the  $S_{\pi}$  operator permutes *m* subsystems according to a permutation  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, m\}$ , and the sum runs over all unique permutations  $\pi$  on size *m* sets. Using this formula along with Equation (A5) gives an analytic Pauli decomposition of the stiffness matrix.

## Appendix A.2. Preparation of the m = 3 Stiffness Matrix

For the m = 3 qubit stiffness matrix (with  $2^m = 8$  finite element nodes), Equation (A10) becomes

$$\mathbf{B}_3 = \frac{1}{4} (\mathbf{X}\mathbf{X}\mathbf{X} - \mathbf{X}\mathbf{Y}\mathbf{Y} - \mathbf{Y}\mathbf{X}\mathbf{Y} - \mathbf{Y}\mathbf{Y}\mathbf{X})$$
(A12)

and therefore

$$(\mathbf{X} \otimes \mathbf{I}_4)\mathbf{B}_3(\mathbf{X} \otimes \mathbf{I}_4) = \frac{1}{4}(\mathbf{X}\mathbf{X}\mathbf{X} - \mathbf{X}\mathbf{Y}\mathbf{Y} + \mathbf{Y}\mathbf{X}\mathbf{Y} + \mathbf{Y}\mathbf{Y}\mathbf{X})$$
(A13)

Substituting this into Equation (A5) and adding the diagonal factor 2III gives our final three-qubit, eight-node stiffness matrix Pauli decomposition as follows:

$$2\mathbf{I}_8 - \mathbf{A}_3 = 2\mathbf{I}_8 - \left[\mathbf{I}_2 \otimes \mathbf{A}_2 + (\mathbf{X} \otimes \mathbf{I}_4)\mathbf{B}_3(\mathbf{X} \otimes \mathbf{I}_4)\right]$$
(A14)  
= 2III -  $\left[\mathbf{IIX} + \frac{1}{2}(\mathbf{IXX} + \mathbf{IYY}) + \frac{1}{4}(\mathbf{XXX} - \mathbf{XYY} + \mathbf{YXY} + \mathbf{YYX})\right]$ 

# Appendix A.3. Preparation of a General m Qubit Stiffness Matrix

Generalization of the above procedure for an *m* qubit stiffness matrix gives the following recursive procedure with  $A_1 := x$ :

$$\mathbf{A}_{m} = \mathbf{I}_{2} \otimes \mathbf{A}_{m-1} +$$

$$(\mathbf{X} \otimes \mathbf{I}_{2^{m-1}}) \frac{1}{2^{m-1}} \sum_{t=0}^{\lfloor m/2 \rfloor} (-1)^{t} \sum_{\pi} \mathbf{S}_{\pi} (\mathbf{X}^{\otimes (m=2t)} \otimes \mathbf{Y}^{\otimes 2t}) + (\mathbf{X} \otimes \mathbf{I}_{2^{m-1}})$$

The final finite-element stiffness matrix is then

$$\mathbf{K}_m = 2\mathbf{I}_{2^m} - \mathbf{A}_m \tag{A16}$$

# References

- 1. Preskill, J. Quantum computing and the entanglement frontier. *arXiv* **2012**. arXiv:1203.5813. Available online: http://arxiv.org/ abs/1203.5813 (accessed on 20 February 2023).
- 2. Harrow, A.W.; Montanaro, A. Quantum computational supremacy. *Nature* 2017, 549, 203–209. [CrossRef] [PubMed]

- Arute, F.; Arya, K.; Babbush, R.; Bacon, D.; Bardin, J.C.; Barends, R.; Biswas, R.; Boixo, S.; Brandao, F.G.S.L.; Buell, D.A.; et al. Quantum supremacy using a programmable superconducting processor. *Nature* 2019, 574, 505–510. Available online: https: //www.nature.com/articles/s41586-019-1666-5 (accessed on 20 February 2023). [CrossRef] [PubMed]
- 4. Connor, E. The New Light-Based Quantum Computer Jiuzhang Has Achieved Quantum Supremacy. 2020. Available online: https://www.sciencenews.org/article/new-light-based-quantum-computer-jiuzhang-supremacy (accessed on 20 February 2023).
- 5. Head-Marsden, K.; Flick, J.; Ciccarino, C.J.; Narang, P. Quantum information and algorithms for correlated quantum matter. *Chem. Rev.* **2021**, *121*, 3061–3120. [CrossRef]
- 6. Breuer, H.-P.; Petruccione, F. The Theory of Open Quantum Systems; Oxford University Press: Oxford, UK, 2007; ISBN 9780199213900.
- Clerk, A.A.; Devoret, M.H.; Girvin, S.M.; Marquardt, F.; Schoelkopf, R.J. Introduction to quantum noise, measurement, and amplification. *Rev. Mod. Phys.* 2010, *82*, 1155–1208. [CrossRef]
- 8. Lidar, D.A. Lecture notes on the theory of open quantum systems. *arXiv* 2019, arXiv:1902.00967.
- 9. Krantz, P.; Kjaergaard, M.; Yan, F.; Orlando, T.P.; Gustavsson, S.; Oliver, W.D. A quantum engineer's guide to superconducting qubits. *Appl. Phys. Rev.* 2019, *6*, 021318. [CrossRef]
- Kandala, A.; Temme, K.; Córcoles, A.D.; Mezzacapo, A.; Chow, J.M.; Gambetta, J.M. Error mitigation extends the computational reach of a noisy quantum processor. *Nature* 2019, 567, 491–495. [CrossRef]
- 11. McArdle, S.; Yuan, X.; Benjamin, S. Error-mitigated digital quantum simulation. Phys. Rev. Lett. 2019, 122, 180501. [CrossRef]
- 12. Smart, S.E.; Mazziotti, D.A. Quantum-classical hybrid algorithm using an error-mitigating n -representability condition to compute the mott metal-insulator transition. *Phys. Rev. A* **2019**, *100*, 022517. [CrossRef]
- 13. Smart, S.E.; Boyn, J.N.; Mazziotti, D.A. Resolving correlated states of benzyne with an error-mitigated contracted quantum eigensolver. *Phys. Rev. A* 2022, 105, 022405. [CrossRef]
- 14. Endo, S.; Cai, Z.; Benjamin, S.C.; Yuan, X. Hybrid quantum-classical algorithms and quantum error mitigation. *J. Phys. Soc. Jpn.* **2021**, *90*, 032001. [CrossRef]
- 15. Smart, S.E.; Hu, Z.; Kais, S.; Mazziotti, D.A. Relaxation of stationary states on a quantum computer yields a unique spectroscopic fingerprint of the computer's noise. *Commun. Phys.* **2022**, *5*, 8. [CrossRef]
- Aleksandrowicz, G.; Alexander, T.; Barkoutsosa, P.; Bello, L.; Ben-Haim, Y.; Bucher, D.; Cabrera-Hernández, F.; Carballo-Franquis, J.; Chen, A.; Chen, C.; et al. Qiskit: An Open-Source Framework for Quantum Computing. 2019. Available online: https://doi.org/10.5281/zenodo.2562111 (accessed on 10 May 2022)
- 17. Amazon, Amazon Braket. Available online: https://aws.amazon.com/braket/ (accessed on 10 May 2022).
- 18. IBM, Learning Quantum Computation Using Qiskit. Available online: http://qiskit.org/textbook (accessed on 1 July 2021).
- 19. Albornoz, C.; Alonso, G.; Andrenkov, P.A.M.; Asadi, A. Anothers, Xanadu Quantum Codebook. 2021. Available online: https://codebook.xanadu.ai (accessed on 1 July 2022).
- 20. Qbraid. Qbraid: Cloud-Based ide for Quantum Computing. Available online: https://qbraid.com (accessed on 10 July 2022).
- Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* 2017, 549, 195–202. [CrossRef]
- 22. Pudenz, K.L.; Lidar, D.A. Quantum adiabatic machine learning. *Quantum Inf. Process.* **2013**, *12*, 2027–2070. [CrossRef]
- 23. Lloyd, S.; Mohseni, M.; Rebentrost, P. Quantum principal component analysis. Nat. Phys. 2014, 10, 631–633. [CrossRef]
- 24. Rebentrost, P.; Mohseni, M.; Lloyd, S. Quantum support vector machine for big data classification. *Phys. Rev. Lett.* **2014**, *113*, 130503. [CrossRef] [PubMed]
- 25. Schuld, M.; Sinayskiy, I.; Petruccione, F. An introduction to quantum machine learning. *Contemp. Phys.* 2015, 56, 172–185. [CrossRef]
- Altaisky, M.V.; Zolnikova, N.N.; Kaputkina, N.E.; Krylov, V.A.; Lozovik, Y.E.; Dattani, N.S. Towards a feasible implementation of quantum neural networks using quantum dots. *Appl. Phys. Lett.* 2016, 108, 103108. [CrossRef]
- 27. Dunjko, V.; Taylor, J.M.; Briegel, H.J. Framework for learning agents in quantum environments. arXiv 2015, arXiv:1507.08482.
- Alvarez-Rodriguez, U.; Lamata, L.; Escandell-Montero, P.; Martín-Guerrero, J.D.; Solano, E. Supervised quantum learning without measurements. Sci. Rep. 2017, 7, 1–9.
- 29. Lamata, L. Basic protocols in quantum reinforcement learning with superconducting circuits. Sci. Rep. 2017, 7, 1–10.
- 30. Wiebe, N.; Braun, D.; Lloyd, S. Quantum algorithm for data fitting. Phys. Rev. Lett. 2012, 109, 050505. [CrossRef] [PubMed]
- 31. Schuld, M.; Sinayskiy, I.; Petruccione, F. Prediction by linear regression on a quantum computer. *Phys. Rev. A* 2016, *94*, 022342. [CrossRef]
- Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* 2009, 103, 150502. [CrossRef] [PubMed]
- Berry, D.W.; Childs, A.M.; Kothari, R. Hamiltonian simulation with nearly optimal dependence on all parameters. In Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, Berkeley, CA, USA, 17–20 October 2015; pp. 792–809. [CrossRef]
- Zhao, Z.; Fitzsimons, J.K.; Fitzsimons, J.F. Quantum-assisted Gaussian process regression. *Phys. Rev. A* 2019, 99, 052331. [CrossRef]

- 35. Zheng, Y.; Song, C.; Chen, M.C.; Xia, B.; Liu, W.; Guo, Q.; Zhang, L.; Xu, D.; Deng, H.; Huang, K.; et al. Solving systems of linear equations with a superconducting quantum processor. *Phys. Rev. Lett.* **2017**, *118*, 210504. [CrossRef]
- Lee, Y.; Joo, J.; Lee, S. Hybrid quantum linear equation algorithm and its experimental test on ibm quantum experience. *Sci. Rep.* 2019, 9, 4778. [CrossRef] [PubMed]
- Pan, J.; Cao, Y.; Yao, X.; Li, Z.; Ju, C.; Chen, H.; Peng, X.; Kais, S.; Du, J. Experimental realization of quantum algorithm for solving linear systems of equations. *Phys. Rev. A* 2014, *89*, 022313. [CrossRef]
- 38. Cai, X.-D.; Weedbrook, C.; Su, Z.-E.; Chen, M.-C.; Gu, M.; Zhu, M.-J.; Li, L.; Liu, N.; Lu, C.; Pan, J. Experimental quantum computing to solve systems of linear equations. *Phys. Rev. Lett.* **2013**, *110*, 30501. [CrossRef]
- Barz, S.; Kassal, I.; Ringbauer, M.; Lipp, Y.O.; Dakić, B.; Aspuru-Guzik, A.; Walther, P. A two-qubit photonic quantum processor and its application to solving systems of linear equations. *Sci. Rep.* 2014, *4*, 6115. [CrossRef] [PubMed]
- 40. Wen, J.; Kong, X.; Wei, S.; Wang, B.; Xin, T.; Long, G. Experimental realization of quantum algorithms for a linear system inspired by adiabatic quantum computing. *Phys. Rev. A* **2019**, *99*, 012320. [CrossRef]
- 41. Anschuetz, E.; Olson, J.; Aspuru-Guzik, A.; Cao, Y. Variational quantum factoring. In *Quantum Technology and Optimization Problems*; Feld, S., Linnhoff-Popien, C., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 74–85.
- 42. Peruzzo, A.; McClean, J.; Shadbolt, P.; Yung, M.; Zhou, X.; Love, P.J.; Aspuru-Guzik, A.; O'Brien, J.L. A variational eigenvalue solver on a photonic quantum processor. *Nat. Commun.* **2014**, *5*, 4213. [CrossRef] [PubMed]
- Cao, Y.; Romero, J.; Olson, J.P.; Degroote, M.; Johnson, P.D.; Kieferova, M.; Kivlichan, I.D.; Menke, T.; Peropadre, B.; Sawaya, N.P.D.; et al. Quantum chemistry in the age of quantum computing. *Chem. Rev.* 2019, 119, 10856–10915. [CrossRef]
- 44. Higgott, O.; Wang, D.; Brierley, S. Variational quantum computation of excited states. Quantum 2019, 3. 156. [CrossRef]
- 45. Jones, T.; Endo, S.; McArdle, S.; Yuan, X.; Benjamin, S.C. Variational quantum algorithms for discovering hamiltonian spectra. *Phys. Rev. A* **2019**, *99*, 062304. [CrossRef]
- 46. Li, Y.; Benjamin, S.C. Efficient variational quantum simulator incorporating active error minimization. *Phys. Rev. X* 2017, 7, 021050. [CrossRef]
- 47. Kokail, C.; Maier, C.; van Bijnen, R.; Brydges, T.; Joshi, M.K.; Jurcevic, P.; Muschik, C.A.; Silvi, P.; Blatt, R.; Roos, C.F.; et al. Self-verifying variational quantum simulation of lattice models. *Nature* **2019**, *569*, 55–360. [CrossRef]
- 48. Heya, K.; Nakanishi, K.M.; Mitarai, K.; Fujii, K. Subspace variational quantum simulator. arXiv 2019, arXiv:1904.08566.
- 49. Cirstoiu, C.; Holmes, Z.; Iosue, J.; Cincio, L.; Coles, P.J.; Sornborger, A. Variational fast forwarding for quantum simulation beyond the coherence time. *npj Quantum Inf.* **2020**, *6*, 82. [CrossRef]
- 50. Yuan, X.; Endo, S.; Zhao, Q.; Li, Y.; Benjamin, S.C. Theory of variational quantum simulation. *Quantum* 2019, 3, 191. [CrossRef]
- Romero, J.; Olson, J.P.; Aspuru-Guzik, A. Quantum autoencoders for efficient compression of quantum data. *Quantum Sci. Technol.* 2017, 2, 045001. [CrossRef]
- 52. LaRose, R.; Tikku, A.; O'Neel-Judy, É.; Cincio, L.; Coles, P.J. Variational quantum state diagonalization. *npj Quantum Inf.* **2019**, *5*, 57. [CrossRef]
- 53. Bravo-Prieto, C.; Garcí a-Martín, D.; Latorre, J.I. Quantum singular value decomposer. Phys. Rev. A 2020, 101, 062310. [CrossRef]
- 54. Cerezo, M.; Sharma, K.; Arrasmith, A.; Coles, P.J. Variational quantum state eigensolver. npj Quantum Inf. 2022, 8, 113. [CrossRef]
- 55. Khatri, S.; LaRose, R.; Poremba, A.; Cincio, L.; Sornborger, A.T.; Coles, P.J. Quantum-assisted quantum compiling. *Quantum* **2019**, 3, 140. [CrossRef]
- Jones, T.; Benjamin, S.C. Robust quantum compilation and circuit optimisation via energy minimisation. *Quantum* 2022, 6, 628. [CrossRef]
- 57. Arrasmith, A.; Cincio, L.; Sornborger, A.T.; Zurek, W.H.; Coles, P.J. Variational consistent histories as a hybrid algorithm for quantum foundations. *Nat. Commun.* **2019**, *10*, 3438. [CrossRef]
- 58. Cerezo, M.; Poremba, A.; Cincio, L.; Coles, P. J. Variational quantum fidelity estimation. Quantum 2020, 4, 248. [CrossRef]
- 59. Koczor, B.; Endo, S.; Jones, T.; Matsuzaki, Y.; Benjamin, S.Variational-state quantum metrology. *New J. Phys.* 2020, 22, 083038. [CrossRef]
- 60. Bravo-Prieto, C.; LaRose, R.; Cerezo, M.; Subasi, Y.; Cincio, L.; Coles, P.J. Variational Quantum Linear Solver. *arXiv* 2019, arXiv:1909.05820. [CrossRef]
- 61. Bravo-Prieto, C.; LaRose, R.; Cerezo, M.; Subaşı, Y.; Cincio, L.; Coles, P.J. Variational quantum linear solver: A hybrid algorithm for linear systems. *Bull. Am. Phys. Soc.* **2020**, arXiv:1909.05820v2. [CrossRef]
- Cincio, L.; Subaşı, Y.; Sornborger, A.T.; Coles, P.J. Learning the quantum algorithm for state overlap. New J. Phys. 2018, 20, 13022. [CrossRef]
- 63. Pesce, R.M.N.; Stevenson, P.D. H2zixy: Pauli spin matrix decomposition of real symmetric matrices. arXiv 2021, arXiv:2111.00627.
- 64. Pellow-Jarman, A.; Sinayskiy, I.; Pillay, A.; Petruccione, F. A comparison of various classical optimizers for a variational quantum linear solver. *Quantum Inf. Process.* **2021**, *20*, 202. [CrossRef]
- 65. Hughes, T.J. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis;* Courier Corporation: Chelmsford, MA, USA, 2012.
- 66. Soklakov, A.N.; Schack, R. Efficient state preparation for a register of quantum bits. Phys. Rev. A 2006, 73, 012307. [CrossRef]

- 67. Giovannetti, V.; Lloyd, S.; Maccone, L. Quantum Random Access Memory. Phys. Rev. Lett. 2008, 100, 160501. [CrossRef]
- 68. Aaronson, S. Read the fine print. Nat. Phys. 2015, 11, 291–293. [CrossRef]
- 69. Bang, J.; Dutta, A.; Lee, S.W.; Kim, J. Optimal usage of quantum random access memory in quantum machine learning. *Phys. Rev.* A 2019, 99, 012326. [CrossRef]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.