Variational quantum linear solver with a dynamic ansatz

Hrushikesh Patil, Yulun Wang, and Predrag S. Krstić

Phys. Rev. A **105**, 012423 — Published 25 January 2022

DOI: 10.1103/PhysRevA.105.012423

Variational Quantum Linear Solver with Dynamic Ansatz

Hrushikesh Patil*
Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794

Yulun Wang† and Predrag S. Krstić‡
Institute for Advanced Computational Science, Stony Brook University, NY 11794-5250

(Dated: January 4, 2022)

Variational quantum algorithms have found success in the NISQ era owing to their hybrid quantum-classical approach which mitigate the problems of noise in quantum computers. In our study we introduce the dynamic ansatz in the Variational Quantum Linear Solver for a system of linear algebraic equations. In this improved algorithm, the number of layers in the hardware efficient ansatz circuit is evolved, starting from a small and gradually increasing until convergence of the solution is reached. We demonstrate the algorithm advantage in comparison to the standard, static ansatz by utilizing fewer quantum resources and with a smaller quantum depth on average, in presence and absence of quantum noise, and in cases when the number of qubits or condition number of the system matrix are increased. The numbers of iterations and layers can be altered by a switching parameter. The performance of the algorithm in using quantum resources is quantified by a newly defined metric.

I. INTRODUCTION

Solving a System of Linear Equations (SLE) is the fundamental problem in computation. Nearly 70% of the scientific and engineering numerical problems reduces to solving SLE [1], which can be symbolically expressed as finding unknown vector $\boldsymbol{x}$ of dimension N from a system of equations $A\boldsymbol{x} = \boldsymbol{b}$, where A is a matrix of dimension $M \times N$, and $\boldsymbol{b}$ is a vector of dimension M. In this paper we will consider only square SLEs with $N = M$, since these appear most often in applications. In classical computation, solving of a SLE scales polynomially with N [2]. Well known quantum computing SLE algorithm HHL [3] scales logarithmically with N. However, the application of HHL is severely limited in the Noisy Intermediate-Scale Quantum (NISQ) era due to its considerable quantum depth. The hybrid approaches, which combine quantum and classical computing [4–9], result in a family of algorithms called Variational Hybrid Quantum Classical Algorithms (VHQCA). These have been successfully applied in computational chemistry (VQE) [4]), fidelity estimation [5], quantum machine learning [10], etc. Characteristics of a VHQCA is quantum ansatz of a small quantum depth, combined with a classical, post-processing optimization routines. Despite the promising results, VHQCAs have shown to be also susceptible to damaging effects of noise [11], especially when the number of qubits increases.

Bravo-Prieto et al. [6] proposed a VHQCA for solving a system of linear equations called Variational Quantum Linear Solver (VQLS). This algorithm uses a variational quantum circuit to evaluate the overlap between states $A|x\rangle$ and $|b\rangle$, where the state $|x\rangle$ is a trial wave function which approximates the solution of the SLE. This is encoded iteratively by a variational quantum circuit, ansatz, followed by the classical optimization of the variational parameters until convergence is reached. Authors of VQLS [6] report empirical scaling of VQLS to be polylogarithmic in N and VQLS was used to solve linear systems of size $N = 1024$ with $n = log_2(N) = 10$ qubits on a quantum computer. This is a significant achievement as compared to the maximal system size 8 (i.e. with 3 qubits) achieved by the HHL algorithm [12]. Similar VHQCA's for SLE have been proposed by Huang et al. [7] and Xu et al. [8]. However, since VQLS has polylogarithmic scaling (as $O((log_2(N))^{8.5}\kappa log(1/\epsilon))$ [6], iterative classical methods like conjugate gradient (CG) which scale as $O(Ns\kappa log(1/\epsilon))$ [3] are faster for small system sizes. VQLS is expected to show advantage over CG for very large system sizes, $N > 2 \times 10^{12}$ (> 41 qubits) with sparsity $s = 0.5$, precision $\epsilon = 0.01$ and condition number $\kappa = 1$. However, unlike VQLS the CG method is constrained to positive definite matrices. Hence, for classical iterative solvers like GMRES (Generalized Minimum Residual Method) which are more generalized which scale as $O(N^2)$[13], VQLS would perform better for matrix sizes larger than 2×10^5 (corresponding to > 17 qubits). When we compare VQLS to a direct solver like Gaussian Elimination with $O(N^3)$ complexity we can expect VQLS to show advantage for $N > 750$ (>9 qubits). Although the output of the quantum solver is quantum state represented with $n = log_2(N)$ qubits, a full tomographic reconstruction of the solution vector of dimension N would require $O(N)$ measurements, diminishing the exponential speedups of the VQLS solver. Still, the scaling advantage of the quantum solver is in the applications which require only a small part of the full solution vector, as well as when only the expectation value of a linear operator acting on the solution vector is

* hrushikesh.patil@stonybrook.edu
† yulun.wang@stonybrook.edu
‡ krsticps@gmail.com

wanted.

The primary interest in quantum computing of the SLE is the problems of large dimension. Hence, for a prospective SLE quantum algorithm it is of interest to reach a good performance with increase of N, i.e., the scalability. The scalability of VQLS is limited by the quantum noise and appearance of barren plateaus in the optimization landscape. The latter is a consequence of vanishing gradients, a well-known problem in classical [14] and quantum machine learning algorithms [15]. Recent papers by Mclean et al. [15] and Cerezo et al. [16] demonstrated the appearance of the barren plateaus in VHQCA.

The quantum noise has been a stumbling block in the quantum computer implementations of all quantum and quantum-classical algorithms. Recent studies [11] have shown that quantum noise can induce barren plateaus. The ansatzes for both local and global cost functions are susceptible to noise-induced barren plateaus in the NISQ era applications, though local cost functions are considered more resistant to the barren plateaus [16]. The noise is dependent on the number of gates, and thus, correlated to the quantum depth of an ansatz circuit [17][18]. Hence, a smaller quantum depth will result in a smaller overall damaging effect of the quantum noise.

In the present work, we address the scalability problem of VQLS by developing an algorithm with evolving, dynamic ansatz as inspired in part by the adaptive VQE [19] and NASNet [20]. The idea of using a variable structure ansatz has been explored for Quantum Machine Learning (QML) [21] and for VQE [22] and was termed Quantum Architecture Search [23][24]. Ostaszewski et al. [25], demonstrated that variation of the ansatz structure along with the variational parameters in VQE significantly improved the algorithm performance. Layerwise learning of ansatz layers for quantum neural networks was investigated by Skolik et al. [26], while Rattew et al. [22] developed evolutionary algorithm to grow the VQE ansatz. Our adaptive algorithm does not aim to improve the computational complexity of VQLS. It has the same polylogarithmic complexity as standard VQLS. We aim to improve the performance of VQLS in presence of quantum noise. Our work focuses on varying the ansatz for VQLS by gradual change of the number of layers in ansatz until desired accuracy was reached while keeping used quantum resources at minimum. We evaluate our approach against the standard VQLS algorithm with hardware efficient ansatz [4] by comparing the quantum depths of the standard and dynamic VQLS ansatzes in a number of SLE test examples. We define and use in Section III a new metric termed "total resource cost" for a more stringent comparison between the two approaches to VQLS in both, presence and absence of the quantum noise. To induce the quantum noise, we use IBM Qiskit noise models [27] in the simulators. The motivation behind this study is the reduction of the effective quantum depth in the algorithm, thereby limiting the noise and allowing increase of the SLE dimension N in the VQLS implementations.

The paper is organized as following: Section II describes the methods used in the paper. In Section III we show and discuss the results, while our conclusions are presented in Section IV.

II. METHODS

The cost functions and ansatz circuits involved in the standard VQLS method [6] are reviewed in subsections II A and II B. The new dynamic ansatz, the algorithm for its evolution through iterations, classical optimization as well as relevant quantum noise simulations are presented in subsections II C, II D, and II E. In our work we limited the scope of matrix A and vector b to real numbers.

A. The cost function and the quantum circuits in VQLS

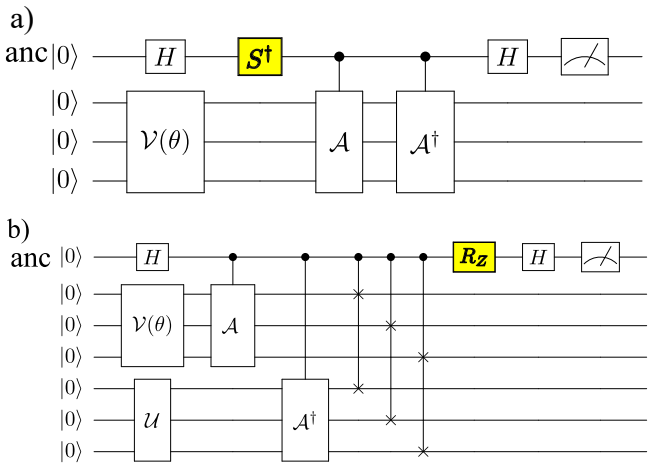

FIG. 1. Quantum circuits for VQLS: a) Hadamard test to calculate denominator, and b) swap test to calculate numerator of the cost function for a system of size $N = 8$. The gates highlighted in yellow (shaded) are present only for calculation of the imaginary terms of the cost function.

The cost function is defined in VQLS [6] as the overlap of states $A|x\rangle$ and $|b\rangle = U|0\rangle$, where U is an unitary operator which is the state initialization circuit as defined in [28]. The trial state is encoded by a variational ansatz $|x\rangle = V(\boldsymbol{\theta})|0\rangle$, with a set of variable parameters, expressed as a vector of rotation angles $\boldsymbol{\theta} = [\theta_0, \theta_1, ..., \theta_m]$. Since matrix A is an arbitrary matrix, it is decomposed into a linear combination of unitary matrices of the form $A = \sum_l c_l P_l$ where c_l are complex numbers and P_l are Pauli matrices [29]. The decomposition process is a preparatory, one-time, process which we illustrate a few examples in the Supplemental Materials (section S1). The equation for the global cost function C_G is defined

in [6] as

$$C_G = 1 - \frac{|\langle b|\psi\rangle|^2}{\langle\psi|\psi\rangle} \tag{1}$$

with $|\psi\rangle = A|x\rangle = AV(\boldsymbol{\theta})|0\rangle$. Although the local cost functions are more resistant to barren plateaus [16], we will use the global cost function for which the quantum circuits are simpler than those for the local cost functions. We note that we compare strategies for relatively small ansatzes (up to eight qubits), for which barren plateaus usually do not present a critical problem.

To calculate the denominator and numerator of the global cost function, Eq.(1), one makes use of the Hadamard and swap tests [6], shown in Fig. 1 on the example of $N = 8$. The $S^\dagger$ and R_Z gates are used only when measuring the imaginary part of the cost function. The main advantage of both tests is that only the ancilla qubit needs to be measured. While the Hadamard test can also be used to calculate the numerator of the cost function, it leads to a more complex quantum circuit with multiple unitaries needed to be controlled. In contrast, swap test requires twice the number of qubits than the Hadamard test but yields a much simpler circuit with fewer controlled unitaries. Swap test can be further simplified by using circuits specified in [30] or by using Hadamard Overlap Test introduced in [6].

B. Static and Dynamic Ansatzes

We discern two algorithms, the Algorithm with Static Ansatz (ASA) and with Dynamic Ansatz (ADA). In ASA, the total number of layers "d" is predefined and fixed during the VQLS run. In ADA the number of layers evolves, depending on the instantaneous value of the cost function. ADA avoids search in an initially large Hilbert space. Instead, the algorithm starts finding the minima in a small subspace spanned by the single layered ansatz, then another layer of ansatz is appended, and the algorithm moves into a larger subspace. This procedure of adding new layers keeps repeating until value of the cost function reaches the pre-defined convergence criterium.

Like in ASA [6], we use in ADA the basic hardware efficient ansatz with layered structure to generate a trial state $|x\rangle$. Each layer consists of a sublayer of rotation gates, followed by a sublayer of entanglers, as illustrated in Fig. 2 for $N = 16$. This structure is repeated for a total number of "d" layers, as needed. The minimum number of layers in ASA estimated to guarantee convergence is given by

$$d_{min} = \frac{2^n}{n} \tag{2}$$

where n is the number of qubits. The reasoning behind the Eq. (2) is that the vector $|x\rangle$ contains 2^n elements, hence at maximum we need one parameter per element. The denominator comes from the fact that hardware efficient ansatz contains n parameters at one layer. The number of layers for ASA is d_{min}. This equation is an estimate of a number of layers needed for convergence in the problems with real matrix A and real vector b, and with choice of hardware efficient ansatz in Fig.2. Depending on the choice of a layered ansatz, the number of layers could be larger than in Eq. 2.

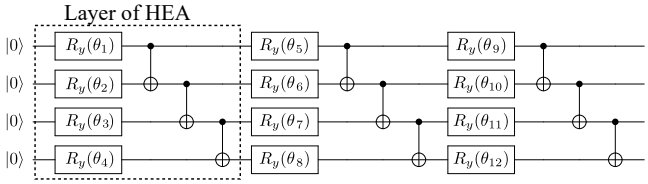

FIG. 2. The layered structure of hardware efficient ansatz for SLE with $N = 16$. This figure represents a choice of ansatz, used in our experiments. If matrix A and vector b contain complex numbers, the ansatz structure must include R_Z gates. In general, the ansatz in the figure may be replaced by any layered ansatz.

The motivation behind ADA is to improve the performance in comparison to the "standard" VQLS with ASA by reducing the total quantum depth of the ansatz, accumulated through iterations and evolution of the number of layers, thereby reducing damaging effects of noise. This is achieved for some types of the SLE systems, as discussed in Section III, thus reducing the quantum noise.

C. Classical Optimization

In the classical part of the VQLS algorithm we apply the gradient-based optimization utilizing the gradient descent method [31] defined by the equation

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \delta \nabla_{\boldsymbol{\theta_t}} C_G(\boldsymbol{\theta_t}) \tag{3}$$

where $\boldsymbol{\theta_t}$ is the value of the parameter vector at iteration t, δ is the step size of the optimizer and C_G is the cost function parameterized by $\boldsymbol{\theta_t}$. The step size is an input to VQLS algorithm. Choosing too large value of δ may result in lack of convergence while too small value results in long runtimes of VQLS. Below we explain how the gradients $\nabla\theta_j$ are calculated in Eq. (3).

In classical computing, backpropagation has been the method of choice for calculating gradients in a neural networks. The ansatz bears some resemblance to the neural network, more specifically to the parameterized connected network of parameters which are "trained" using an optimizer. Still, the ansatz cannot be optimized using backpropagation, since this requires the measurement of intermediate layers of the "neural network", which would cause collapse of the quantum wave function, disabling its transfer to the next layer. Hence, approaches which do not require intermediate measurements like are finite differences, automatic differentiation, and parameter shift rule are choices for gradients calculations in a quantum

ansatz [32]. We use the parameter shift rule which provides exact gradients and is defined by equation,

$$\frac{\partial C_G(\boldsymbol{\theta})}{\partial \theta_i} = \frac{C_G\left([\theta_1, ..., \theta_i + \frac{\pi}{2}, ..., \theta_k]\right)}{2} - \frac{C_G\left([\theta_1, ..., \theta_i - \frac{\pi}{2}, ..., \theta_k]\right)}{2} \quad (4)$$

where $\boldsymbol{\theta} = [\theta_1, ..., \theta_i, ..., \theta_k]$, where k is the iteration dependent number of parameters and $i = 1, 2, ..., k$.

This equation requires two cost function computations for gradient calculation of each parameter. The cost function evaluations are executed on a quantum computer and hence are affected by quantum noise. The time t_L spent in each optimization loop which calculates all gradients can be is calculated from

$$t_L = 2\nu t_c \quad (5)$$

where ν is the total number of variational parameters and t_c is the time for each cost function computation. In case of the hardware efficient ansatz ν is obtained as the number of qubits multiplied by the number of layers, which can be conjectured from Fig. 2.

We performed a benchmarking experiment comparing various simulators for time taken by each of them to calculate the cost function as well as for time taken for gradient calculations and parameter updates. The details are presented in Supplemental Information, section S7. From Table S9 it is clear that Pennylane-qulacs is on average about 2 times faster than Pennylane default and about 3 times faster than Pennylane-qiskit in calculations of the gradients.

D. Evolution of ansatz in ADA

As discussed in subsection II A the idea behind ADA is to start with a small configuration space, and increase it successively, limiting the total number of layers when convergence is reached. Initially, we limit the search space to a subspace smaller than the one for a hardware-efficient ansatz with d_{min} layers (Eq. (2)). After finding the minima in the subspace, a check of overall convergence is performed. If convergence is not reached, a new layer of ansatz is appended to the existing ansatz (Fig. 3). The procedure is continued until the convergence is reached, i.e. value of the cost function goes below a specified threshold $d_t = \frac{\epsilon^2}{\kappa^2}$. To control the speed of appending layers in ADA, we define a hyperparameter which we name the switching parameter (SP). When the difference in costs of two successive optimization steps drops below SP, a new layer is appended. The switching parameter governs the dynamical nature of the ansatz.

For example, one would have potentially lesser layers at the cost of increasing number of optimization steps, but setting the SP low. Conversely, setting the SP too high would add too many layers before the maximum

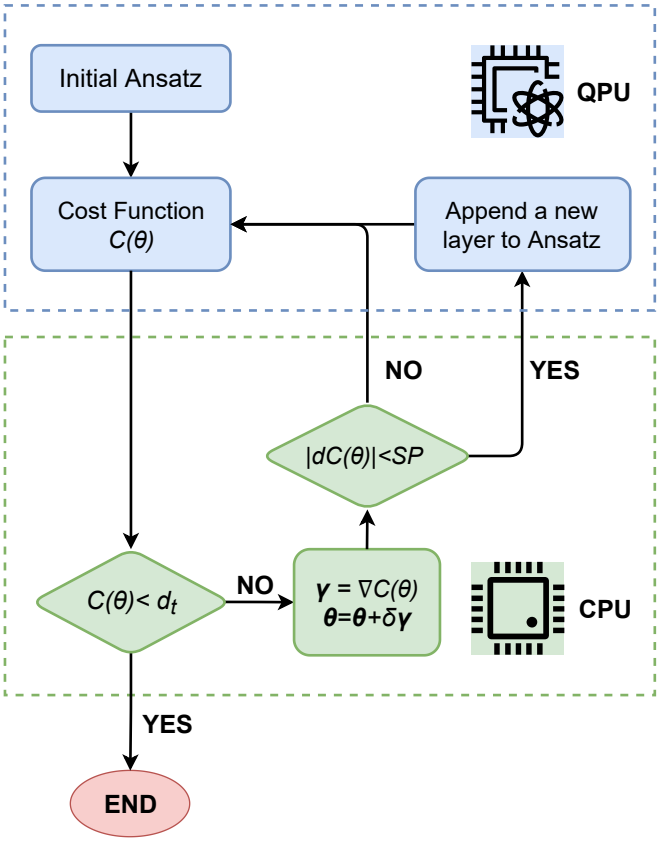

FIG. 3. Flowchart for ADA algorithm

predefined quantum depth is reached. In this case, the ADA performs like the ASA in terms of both the number of iterations and the Total Resource Cost (to be defined in Section III).

A good way to determine value of SP is by following formula,

$$SP = \frac{1 - d_t}{n_{iterd}} \quad (6)$$

where d_t is the stopping threshold (given by the user, which depends on the desired precision (ϵ) of solution $d_t = \frac{\epsilon^2}{\kappa^2}$) and n_{iterd} is the maximum allowed number of iterations which is also set in advance by a user. There is no guarantee that the algorithm will converge by any specified number of iterations. Setting n_{iterd} low will make SP value high, thereby making ADA algorithm to add layers faster. Conversely, setting n_t to a higher value will make SP value low. The idea behind the choice of SP can be understood from the figure 4 SP value is compared with the difference of cost function values between two successive optimization iterations to decide the addition of layer of ansatz. Essentially what the algorithm does is comparison of the slope of the cost function at each iteration with slope of SP. In figure 4 the blue line (solid line) approximates values of cost function against the number of iterations. The other three lines indicate the slopes of different values of SP obtained from the Eq.

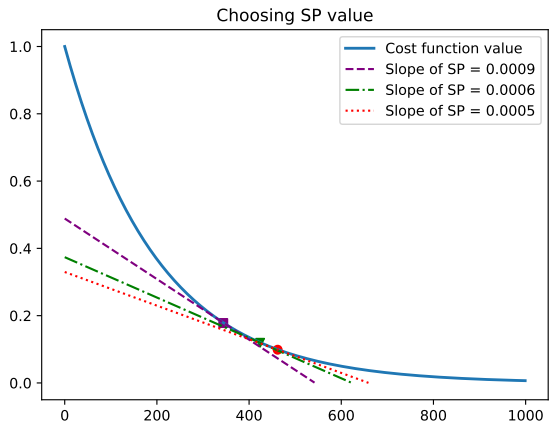

FIG. 4. Effect of different SP values on the number of iterations for addition of a new layer.

6. The points where they are tangent to the blue curve (solid curve) indicate the points where an addition of a new layer is triggered. The $SP = 0.0009$ line will trigger the addition of layer at iteration 344 as compared to $SP = 0.0005$ at iteration 462. After the addition of a layer, the new cost function is calculated with parameters of the newly added layer initialized to zero and with the previous values of the parameters in the rest of the circuit. All parameters in the new cost function are then optimized/reoptimized in the next and subsequent iterations.

E. Noise Simulation

We study the effect of noise on VQLS with ADA and ASA using PennyLane [33] with the access to IBM QASM noise simulator. The simulator uses a noise model generated by the calibration data at "ibmq_16_melbourne" as an approximation to the noise in the real quantum computer. This noise model is contains depolarizing errors on both single and two-qubit gates, thermal relaxation errors for simulating decoherence, and single-qubit readout errors on all individual measurements, as explained in the Supplemental Information (Section S7). We compare the ASA and ADA in presence of noise in Section III E.

III. RESULTS

All experimental results were obtained using Xanadu's PennyLane library [33]. For the simulations we chose the qulacs backend [34] on basis of the results and discussion in subsection II C. For the experiments with noisy simulations we use IBM QASM simulator. The optimization method is utilized with the gradient descent (Eq. (3)) with parameter shift rule as in Eq. (4), and with a step size of 0.05. To reduce the computation time, we set convergence threshold d_t of the cost function to 0.1 and the maximum number of iterations to 6400. Only the cases where convergence is reached are discussed in our analysis. Interestingly, the cases when convergence is not achieved are appearing simultaneously for both ASA and ADA. This approach is used in all experiments.

A. Total Resource Cost (TRC)

To evaluate and compare efficacies of the ADA and ASA , we define a new performance metric, Total Resource Cost (TRC). From Eq. (4), two cost function evaluations are needed per parameter in each iteration of the optimizer. The number of parameters is determined by the number of rotational gates in the ansatz which is proportional to the number of layers in ansatz. These together define the Effective Quantum Depth (EQD) as quantum depth of the circuit accumulated through the iterations. In ADA the EQD is proportional to the sum of all layers in the evolution of the ansatz. The quantum noise in the algorithm implementation is function of the number of gates, which is proportional the EQD.

TRC is defined as the sum of number of layers of each ansatz over iterations. For ASA, this is simply product of the number of iterations needed for convergence, z_{ASA}, and the predefined number of layers in ansatz, d_{ASA},

$$TRC_{ASA} = z_{ASA} d_{ASA} \tag{7}$$

For ADA the TRC is a cumulative sum of number of layers at each optimization step

$$TRC_{ADA} = \sum_{i=0}^{z_{ADA}} d_{ADA}(i) \tag{8}$$

where z_{ADA} is the total number of iterations of VQLS with ADA and $d_{ADA}(i)$ is the number of layers in the iteration "i". For example, assume that ADA has 10 iterations in total with iteration 1 and 2 having 1 layer, 3 and 4 having 2 layers, 5 to 8 having 3 layers and 9 and 10 having 4 layers in total. The $TRC_{ADA} = 2 \cdot 1 + 2 \cdot 2 + 4 \cdot 3 + 2 \cdot 4$. For ASA, even if we have 7 iterations with 4 layers the $TRC_{ASA} = 28$, i.e. ASA has a higher resource cost in this example.

The TRC metric is proportional to the EQD of the variational quantum circuit. Each iteration in ASA contains larger or equal number of parameters than ADA (since $d_{ADA}(i) \leq d_{ASA}$ for all i). Fewer number of parameters means lesser number of gradient calculations, thereby implying more efficient computation in each iteration, but also lesser number of (rotational) gates, which means smaller noise.

Here we compare two ASA examples, which have different number of iterations, z_{ASA}, and different number of layers, d_{ASA}, showing that neither of these parameters defines unambiguously the algorithm efficacy. In

TABLE I. Comparison of arithmetically averaged experimental data for ADA and ASA of VQLS.

Number	TRC		Final Layers		Total iterations	
of qubits	ADA	ASA	ADA	ASA	ADA	ASA
4	1777.8	1949.2	3.4	4	613.3	487.3
5	2385.68	2910.81	4.5	6	572.77	485.13
6	6442.53	7366.93	6.33	8	890.06	920.86

the first example $z^{(1)}_{ASA} = 10$ and $d^{(1)}_{ASA} = 4$. In the second example $z^{(2)}_{ASA} = 20$ and $d^{(2)}_{ASA} = 2$. We also assume that each layer in both cases contains R rotational gates, i.e. R optimization parameters. Since there are two cost function evaluations per parameter, the total number of the cost evaluations is the same for both cases, i.e. $4 \cdot 10 \cdot 2 = 2 \cdot 20 \cdot 2 = 80$, implying the same running time. This is correctly reflected by the equality $TRC^{(1)} = TRC^{(2)} = 40$, but not with either number of iterations or number of layers.

The rationale behind introducing TRC is that, unlike classical computing, quantum computing is heavily affected by quantum noise. The noise is directly proportional to depth of a quantum circuits. Polylogarithmic computational complexity of the quantum algorithm vs. polynomial computational complexity of the classical algorithms does not capture the effect of noise on the algorithm. Even when two algorithms have the same computational complexity, the needed number of operations could be different because of a different quantum circuit depth. Hence, we need a metric that tracks the usage of resources (i.e. number of gates) over all iterations. Since addition of each gate increases noise in the circuit, the TRC measures how the proposed algorithm is noise resilient.

B. Variation of the number of qubits

In this experiment we use 20 randomly generated SLEs for each of the sizes $N = 16, 32$ and 64 (i.e. with 4, 5 and 6 qubits), using the PennyLane software library. The condition numbers of SLE are varied from 1 to 20. About 50% of these tests reach convergence for which we compare TRCs, final number of layers and total number of iterations with both ASA and ADA. The averaged results are shown in Table I. The detailed results can be seen in tables S2-S6 of Supplemental Information (sections S2-S5).

The arithmetic averages of TRC_{ADA} are smaller (up to 20%) than these in TRC_{ASA} for all considered cases. This implies the faster calculations and smaller total noise of ADA. We also compare the number of layers of ansatz in the final iteration, which we term as "final layers". On average ADA finishes with smaller number of final layers than ASA. As the number of qubits increases, the ADA requires fewer number of iterations than ASA to converge.

C. Variation of the matrix condition number

Here we use 25 SLEs of dimension $N = 32$ (5 qubits). Condition number is calculated by taking the ratio of the largest to the smallest eigenvalue of matrix A. The number of layers in ansatzes was 6 (refer Eq. (2)) for ASA while the maximum number of layers was set to 6 for ADA. We varied the condition number in range 1 to 20 (see Table II). For each condition number, the experiments at four systems are performed and these all reach convergence. We calculate TRC for both ASA and ADA, and and for comparison we calculate Average of the Relative TRC (ARTRC) deviation:

$$\frac{\Delta TRC}{TRC} = \frac{TRC_{ADA} - TRC_{ASA}}{TRC_{ADA}} \tag{9}$$

TABLE II. Comparison of ARTRC deviation between ASA and ADA results with respect to the condition number for SLEs of a fixed size 32 (5 qubits).

Condition Number	ARTRC deviation
1	0.0625
3.684	0.41
7.899	0.31
13.572	0.27
20.651	-1.18

The averaged results are presented in Table II, while the details are shown in Supplemental Information (section S3, table S5). The minus sign indicates better performance of ADA.

Better performance is obtained for ASA when the condition numbers are smaller. Up to condition number 5 TRC_{ADA}'s are more than 40% larger. But with further increase of condition number, the performance of the ADA is improving. For poorly conditioned matrices (i.e matrices with a high condition number 20) the ADA results in 18% lower TRC than that of ASA.

D. Variation of the matrix sparsity

The systems of equations with sparse matrices are common occurrence in science and technology. In our experiments, we define sparsity of a matrix as the ratio of the number of zero matrix elements to the total number of matrix elements. We used matrices of size $N = 16$. The maximum number of layers was set to 4. The sparsity was varied in 4 steps, from 0.9375 to 0.75, while the condition number of the matrices was kept low, between 1 and 1.5. For each sparsity, experiments were performed with ten different matrices. The ARTRC deviation is presented in the table III (details in SM, table S6)

The sparsity does not significantly impact performance of VQLS with ADA vs ASA, although there is a weak advantage of the ADA results. This advantage is expected

TABLE III. Comparison of ARTRC deviation between ASA and ADA results with respect to the condition number for systems of a fixed size 32 (5 qubits).

Sparsity	ARTRC deviation
0.9375	-0.04
0.875	0.17
0.8125	-0.012
0.75	-0.14

to increase for the system matrices with larger condition numbers.

E. Comparison of ADA and ASA in presence of quantum noise

In the noisy simulations we used systems of dimension $N = 16$. More details on how we perform the noise simulations can be found in subsection II E.

TABLE IV. Comparison of the TRCs for ADA and ASA in presence of noise.

Condition Number	Average TRC		% of cases
	ADA	ASA	$TRC_{ADA} < TRC_{ASA}$
1	261.1	205.6	30
3.6840	639	625	75
7.8995	542.4	550.4	80
20.6519	699.5	1024.5	100

From Table IV, the TRC of ADA is becoming substantially smaller than of ASA with increase of the condition number and with quantum noise present. Also, in almost 30% of cases, ADA reached convergence with lesser number of final layers than ASA. In all other cases both algorithms reached the convergence with same number of final layers.

F. Variation of the number of layers in ASA

In ASA, the number of layers have to be specified before the algorithm is executed. It is important to have sufficient number of layers so that the ASA is not underparameterized to reach convergence. However, it is not always easy to predict the number of layers as the input system matrix can vary in terms of condition number and sparsity. Hence, to guarantee convergence with ASA, one might over-parameterize the ansatz by adding more than the needed number of layers. The problem with overparameterization is that the extra layers increase the quantum depth of the circuits used to calculate the cost function, thus increasing the total amount of noise. Due to these factors, setting the right number of layers is essential in ASA. This is not a problem with ADA since it changes the number of layers dynamically without specifying it in advance.

In the present experiments, the number of qubits is 8. To guarantee convergence, irrespective of the input matrix, the needed number of layers is 32, for which the vector $|x\rangle$ contains 256 elements (more details in Supplemental Information, section S2). With 8 parameters in each layer, 32 layers contain 256 parameters. We purposely under-parameterize the ASA by setting the maximum number of layers to 8 and 16. This is done to show that for some matrices the needed number of layers in ASA is lesser than the d_{min} of Eq. (2). We utilize VQLS to 21 systems, applying 8 and 16 layers ASA, and also ADA with maximum 8 layers. We found that out of 21 matrices only 6 matrices reached convergence, in each case with $d_t = 0.1$. The convergence was reached with the similar values of TRC's for all three case groups. Another interesting observation is that for ASA with 16 layers the algorithm takes fewer iterations than with ansatz with 8 layers but still converges with only slightly higher TRC. The details are shown in SM Section S8, Table S10.

G. Variation of the switching parameter

TABLE V. Variation of the TRC and Final number of layers in ADA when SP is varied.

	ADA				ASA
SP	10^{-1}	10^{-2}	10^{-3}	10^{-5}	NA
Lowest TRC (%)	33	14	52	14	0
FL[a] (%) ADA $< d_{min}$	0	0	66.67	71.42	NA

[a] FL stands for Final Layers

In these experiment we use 30 systems of $N = 16$ (4 qubits) with varying condition numbers of their matrices A. We consider results for the 21 systems which reached convergence with ADA, with each of the SPs (0.1, 0.01, 0.001, 0.0001) as well as with ASA. Limiting number of layers for ADA is set to 4, which was also the fixed number of layers for ASA. The main objective is to test the effect of the choice of SP on the TRC performance of ADA, which could provide a mean to obtain a good balance between the number of iterations and number of layers in ansatz to reach convergence.

In Table V, first row shows percentage of cases when ADA with particular value of SP (in columns) has the minimal value of TRC among all 105 cases (4x21 converged with ADA plus 21 converged with ASA). The second row indicates percentage of cases when final number of layers with a particular SP value in columns was less than the d_{min}.

As seen in the Table V, some applications of ADA have lesser TRC than ASA. The TRC and final number of layers of ADA and ASA is almost the same when SP is 0.1 (Section S6). The TRC of ADA seems to have a convincing minimum when SP is 0.001. This indicates that there

is an optimum value of switching parameter which can be found out empirically from numerical experiments. For example, the optimum value for above system is obviously between 0.00001 and 0.01. From the Table V one can also see that out of all experimental runs, in 67% cases the final number of layers of ADA is less than that of ASA if SP is 0.001 (Refer SM, section S7 for details).

H. Future directions with ADA

In this work ADA was used to improve performance of the VQLS algorithm. However, we expect that ADA could be also applicable in VQE and Quantum Machine Learning, since ADA algorithm is agnostic. The future improvements of the algorithm could be done by implementing the popular optimizers like ADAM [35] and other momentum based gradient optimizers. The momentum value can help to adjust the switching parameter. It would be also interesting to investigate the performance of ADA with non-gradient based optimizers like are Powell and COBYLA [36]. These have fewer evaluations of the quantum cost function. Another possible improvement can be also made by using optimizers like L-BFGS [37] which use the second order derivatives for faster convergence, but on account of more cost function computations. ADA tends to have fewer parameters than ASA and thus can make use of L-BFGS more feasible for NISQ era. The second order derivative values can be leveraged in ADA for switching decision between the layers.

IV. CONCLUSION

Developing an efficient solver for a system of linear equations is one of the central tasks in computing due to its wide use in many physical and engineering applications. Unlike the classical algorithms, VQLS is an algorithm with polylogarithmic scaling with respect to the system size, which promises to offer the quantum advantage even in the NISQ era. The goal of this study was to show that with a good decision making process within a variational algorithm, in particular with VQLS, we can significantly improve its performance in the NISQ era. In our study the decision making process is the variation of the number of layers in the VQLS ansatz, which is the basis for development of our algorithm with dynamic ansatz, ADA. We computationally compared efficiency of ADA with the standard algorithm with static ansatz, ASA, varying the system size (i.e. the number of qubits), condition number and sparsity of the system matrix, as well as switching parameter and number of layers in the hardware efficient ansatz. In presence of noise, ADA tends to outperforms ASA algorithm, quantified by reduction of the total resource cost. Since TRC is directly dependent on the number or layers in the ansatz as well as of the number of iterations, reduction of the TRC results into reduction of the effective quantum depth of the circuit, accumulated through the iterations. ADA also outperformed ASA when the system is ill-conditioned, i.e. when the system matrix has a large condition number. With the increase of number of qubits, ADA starts outperforming ASA not only measured by TRCs but also by number of iterations. While one can always estimate the number of layers for ASA to guarantee convergence, this may often lead to overparameterization, and thus unnecessary increase of the quantum noise. ADA is not susceptible to the overparameterization problem since it doesn't require specifying number of layers of ansatz in advance. Rather, the algorithm gradually finds the appropriate amount of layers, i.e. the number of optimization parameters, for convergence. Fewer parameters mean fewer cost function evaluations on the quantum computer to calculate gradients. In particular, the values of gradients are less affected by noise. In ADA we define the switching parameter, SP, which controls the addition of new layers in the iteration process. By a proper choice of SP one can adapt the optimum performance of ADA to the noise characteristics of the particular quantum computer. Finally we remark that, ADA is a subset of Quantum Circuit Architecture Search Algorithms. This class of algorithms is useful in optimizing a quantum circuit architecture during the algorithm run.

V. ACKNOWLEDGMENTS

HP acknowledges the useful discussions with Dr. Ji Liu. YW was supported by a graduate student scholarship from IACS. All experiments were performed at Seawulf HPC cluster of SBU, and at XSEDE HPCs Comet and Expanse of SDSC.

[1] G. Dahlquist and A. Bjorck, *Numerical Methods* (Prentice-Hall, 1974).
[2] V. Pan, Complexity of algorithms for linear systems of equations, in *Computer Algorithms for Solving Linear Algebraic Equations*, edited by E. Spedicato (Springer Berlin Heidelberg, Berlin, Heidelberg, 1991) pp. 27–56.
[3] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. **103**, 150502 (2009).
[4] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature **549**, 242 (2017).
[5] M. Cerezo, A. Poremba, L. Cincio, and P. J. Coles, Variational Quantum Fidelity Estimation, Quantum **4**, 248

(2020).
[6] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles, Variational quantum linear solver (2020), arXiv:1909.05820 [quant-ph].
[7] H.-Y. Huang, K. Bharti, and P. Rebentrost, Near-term quantum algorithms for linear systems of equations (2019), arXiv:1909.07344 [quant-ph].
[8] X. Xu, J. Sun, S. Endo, Y. Li, S. C. Benjamin, and X. Yuan, Variational algorithms for linear algebra (2019), arXiv:1909.03898 [quant-ph].
[9] K. Bharti *et al.*, Noisy intermediate-scale quantum (nisq) algorithms (2021), arXiv:2101.08448 [quant-ph].
[10] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature **549**, 195 (2017).
[11] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, Noise-induced barren plateaus in variational quantum algorithms (2021), arXiv:2007.14384 [quant-ph].
[12] J. Wen, X. Kong, S. Wei, B. Wang, T. Xin, and G. Long, Experimental realization of quantum algorithms for a linear system inspired by adiabatic quantum computing, Phys. Rev. A **99**, 012320 (2019).
[13] Y. Saad and M. H. Schultz, Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems, Siam Journal on Scientific and Statistical Computing **7**, 856 (1986).
[14] S. Shalev-Shwartz, O. Shamir, and S. Shammah, Failures of gradient-based deep learning, in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17 (JMLR.org, 2017) p. 3067–3075.
[15] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nat. Commun. **9**, 4812 (2018).
[16] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits, Nat. Commun. **12**, 1791 (2021).
[17] P. J. Salas, Noise effect on grover algorithm, The European Physical Journal D **46**, 365 (2008).
[18] Y. Wang and P. S. Krstic, Prospect of using grover's search in the noisy-intermediate-scale quantum-computer era, Phys. Rev. A **102**, 042609 (2020).
[19] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, An adaptive variational algorithm for exact molecular simulations on a quantum computer, Nat. Commun. **10**, 3007 (2019).
[20] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, Learning transferable architectures for scalable image recognition (2018), arXiv:1707.07012 [cs.CV].
[21] M. Bilkis, M. Cerezo, G. Verdon, P. J. Coles, and L. Cincio, A semi-agnostic ansatz with variable structure for quantum machine learning (2021), arXiv:2103.06712 [quant-ph].
[22] A. G. Rattew, S. Hu, M. Pistoia, R. Chen, and S. Wood, A domain-agnostic, noise-resistant, hardware-efficient evolutionary variational quantum eigensolver (2020), arXiv:1910.09694 [quant-ph].
[23] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao, Differentiable quantum architecture search (2020), arXiv:2010.08561 [quant-ph].
[24] Y. Du, T. Huang, S. You, M.-H. Hsieh, and D. Tao, Quantum circuit architecture search: error mitigation and trainability enhancement for variational quantum solvers (2020), arXiv:2010.10217 [quant-ph].
[25] M. Ostaszewski, E. Grant, and M. Benedetti, Structure optimization for parameterized quantum circuits, Quantum **5**, 391 (2021).
[26] A. Skolik, J. R. McClean, M. Mohseni, P. van der Smagt, and M. Leib, Layerwise learning for quantum neural networks, Quantum Machine Intelligence **3**, 5 (2021).
[27] G. Aleksandrowicz *et al.*, Qiskit: An Open-source Framework for Quantum Computing (2019).
[28] V. Shende, S. Bullock, and I. Markov, Synthesis of quantum-logic circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**, 1000 (2006).
[29] Qiskit, Proving universality (2021).
[30] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles, Learning the quantum algorithm for state overlap, New Journal of Physics **20**, 113022 (2018).
[31] S. Ruder, An overview of gradient descent optimization algorithms (2017), arXiv:1609.04747 [cs.LG].
[32] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A **99**, 032331 (2019).
[33] V. Bergholm *et al.*, Pennylane: Automatic differentiation of hybrid quantum-classical computations (2020), arXiv:1811.04968 [quant-ph].
[34] Y. Suzuki *et al.*, Qulacs: a fast and versatile quantum circuit simulator for research purpose (2020), arXiv:2011.13524 [quant-ph].
[35] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, edited by Y. Bengio and Y. LeCun (2015).
[36] M. J. D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, The Computer Journal **7**, 155 (1964).
[37] R. Fletcher, *Practical methods of optimization*, 2nd ed. (Wiley, 1987).