Introduction to Quantum Computing

Kalamarakis Theodoros: 2018030022

January 25, 2023

GitHub Repository

https://github.com/kthod/VQLS_algorithm.git

Variational Quantum Linear Solver

Introduction

Solving linear equation systems has been an important task in science and technology. Thus, it is essential to construct rapid and efficient algorithms. The existing classical algorithms, solve the problem in polynomial complexity with respect to number of unknowns variables (and therefore, the number of equations). However, in quantum Computing the suggested HHL algorithm achieves logarithmical escalation, by attempting to prepare a quantum state $|x\rangle$ which is proportional to the actual solution vector $\mathbf{x}$ that satisfies the equation $\mathbf{Ax} = \mathbf{b}$. Although the HHL algorithm can provide an exponential speedup and seems like a major improvement over the classical ones, it still remains a future project due to to its very demanding implementation. In order to treat the problem with the available hardware, an other hybrid Quantum-classical algorithm has been introduced, the **Variational Quantum Linear Solver**. The VQLS manages to be less vulnerable to noise, by using a short-depth quantum circuit, exclusivly for evaluating a cost function. The classical part of the algorithm is the optimizer which is used to minimize the aforementioned cost function.

Description of the VQLS algorithm

We are given a matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ and a complex vector $\mathbf{b} \in \mathbb{C}^{n}$. The goal is to determine a state $|x\rangle$ such that $\mathbf{A}|x\rangle$ is proportional to $\mathbf{b}$, or equivalently:

$$|\boldsymbol{\psi}\rangle := \frac{\mathbf{A}|x\rangle}{\langle x|\,\mathbf{A}^{\dagger}\mathbf{A}\,|x\rangle} \approx |b\rangle$$

where $|b\rangle$ is normalized vector, proportional to $\mathbf{b}$, in a quantum state form. Initially, we have to make a couple of assumptions.

- Matrix $\mathbf{A}$ can be decomposed into a linear combination of L unitaries $A_1, A_2, ..., A_L$ such that:

  $$\mathbf{A} = \sum_{l=1}^{L} c_l A_l$$

  where c_l is a complex number.

- State $|b\rangle$ can be produced by applying a unitary U to state $|0\rangle$ such that.

  $$U|0\rangle = |b\rangle$$

All the unitaries matrices $A_1, A_2, ..., A_L$ and U must be able to be implented with an efficient quantum circuit and alongside the coefficients c_l, they constitute the input of the algorithm.

The algorithm accomplishes its goal, by performing multiple iterations, where in each iteration a potential solution is prepared. This expected solution is formed, using a set of parameters $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ...)$ such that

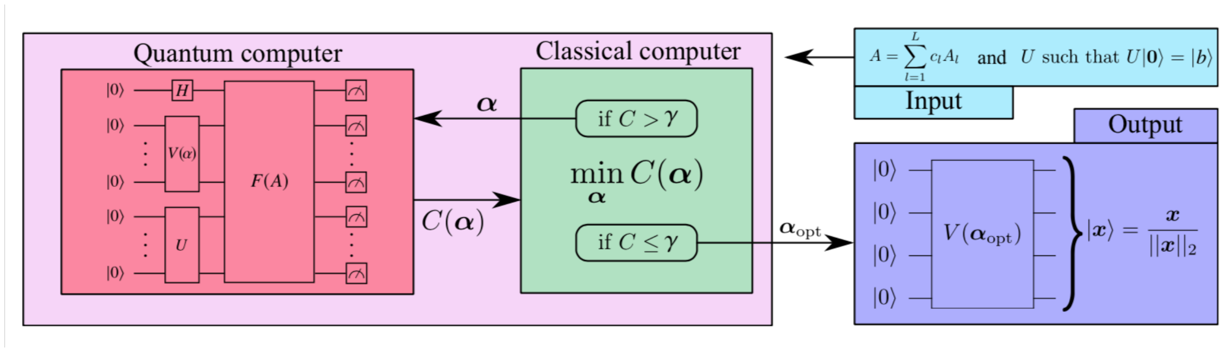
$$|x(\boldsymbol{\alpha})\rangle = V(\boldsymbol{\alpha})\,|\mathbf{0}\rangle$$

where $V(\boldsymbol{\alpha})$ (the, so called, **Ansatz**) is a unitary operator, capable of generating any arbitrary state in Hilbert space, if applyed on state $|\mathbf{0}\rangle$. In each iteration, the parameters $\boldsymbol{\alpha}$ are redefined so that, the expected solution approaches more and more the optimal one, until the approximation error lies within a specific tolerance γ. Responsible for the configuration of the parameters $\boldsymbol{\alpha}$, is a classical optimizer, that aims to minimize a **cost function** according to $\boldsymbol{\alpha}$. The evaluation of this cost function is happening through a quantum circuit.

Hence the VLQS algorithm consists of a quantum and a classical circuit whose interaction is described as follows:

- The quantum circuit receives as input the parameters $\boldsymbol{\alpha}$ and evaluates the cost function on this point ($C(\boldsymbol{\alpha})$).
- The classical optimizer, uses the output value $C(\boldsymbol{\alpha})$ (produced by the quantum circuit), to determine a new set of parameters $\boldsymbol{\alpha}'$ such that, $C(\boldsymbol{\alpha}') < C(\boldsymbol{\alpha})$.
- The quantum circuit is fed with the new set of parameters $\boldsymbol{\alpha}'$ and the process repeats itself until the cost function reaches its global minimum (which is 0)

The cost function is constructed in such a way that its minimum value ($C(\boldsymbol{\alpha}_{opt}) = 0$), corresponds to the optimal solution $|x(\boldsymbol{\alpha}_{opt})\rangle = |x\rangle$
The description above can be visulized, using the following scematic diagram from the cited paper.



Quantum computer
Classical computer
$A = \sum_{l=1}^{L} c_l A_l$ and U such that $U|\mathbf{0}\rangle = |b\rangle$
Input
Output
$|0\rangle$
H
$V(\alpha)$
U
$F(A)$
$\boldsymbol{\alpha}$
$C(\boldsymbol{\alpha})$
if $C > \gamma$
$\min_{\boldsymbol{\alpha}} C(\boldsymbol{\alpha})$
if $C \leq \gamma$
$\boldsymbol{\alpha}_{\text{opt}}$
$V(\boldsymbol{\alpha}_{\text{opt}})$
$|\boldsymbol{x}\rangle = \frac{\boldsymbol{x}}{||\boldsymbol{x}||_2}$

Figure 1: Scematic diagram of the VLQS algorithm. It illustrates the algorithm (light pink box) receiving as input the matrices $\mathbf{A} = \sum_{l=1}^{L} c_l A_l$ and U and returning the optimal parameters $\boldsymbol{\alpha}_{opt}$ which produced the optimal solution $|x\rangle$. Inside the light pink box it is shown the interaction between the quantum circuit (left) and the classical circuit (right) as previously described.

Complexity of the VQLS

The cited paper provides numerical simulations for specific matrices A with fixed conditioned number and with a particular precision. The result of those simulations proves that the algorithm achieves polynomial scaling in n and therefore polylogarithmic scaling in N, since $N = 2^n$. Namely, to guarantee precision 0.3 with $n = 2, ..., 7$ and *condition number* $= 10$, the obtained relation for the running time of the algorithm was $y \sim n^{8.5}$. Any further complexity analysis, other than the experimental one, is difficult due to the heuristic nature of the algorithm. The figure bellow (taken from the original paper) presents the simulation results

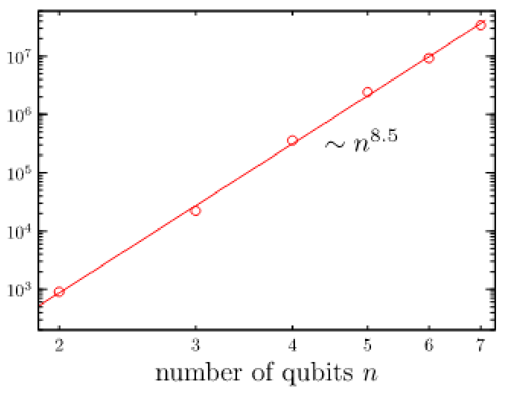
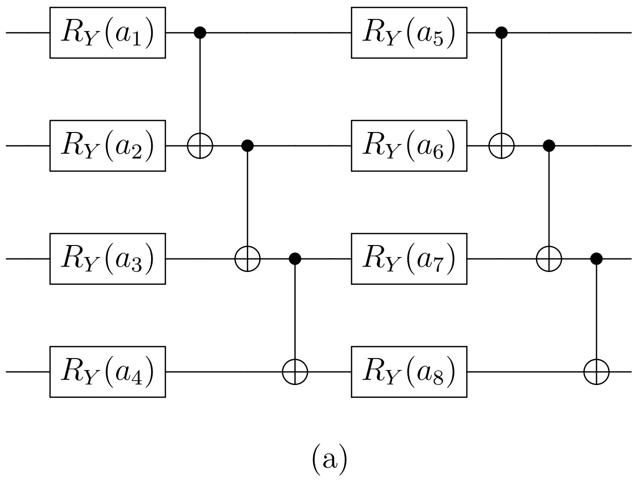


Figure 2: Time-to-solution versus n needed to guarantee precision = 0.3. All matrices had a condition number $\kappa = 10$.

The Ansatz

The Ansatz is the oparator, responsible of preparing the state $|x\rangle$ when it acts on state $|\mathbf{0}\rangle$. It should be capable of generating any arbitrary state on Hilbert space, including the entangled states. A possible way of implementing such an oparator, is by employing single qubits rotation gates (such as R_X, R_Y, R_Z) for "exploring" the space alongside some two-qubit gates to provide the entanglement(such as CX, CZ, CH). Each rotation gate

should correspond to one of the parameters of $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ...)$. One could consider two possible implementation of the Ansatz. Firstly, the **fixed-structure ansatz** in which the configuration of the gates remains the same throughout the execution of the algorithm and secondly, the **variable-structure ansatz** in which the gate's placement is changing. In this report we will focus only on the fixed stucture ansatz.
Two simple variations of the fixed structure ansatz are the following



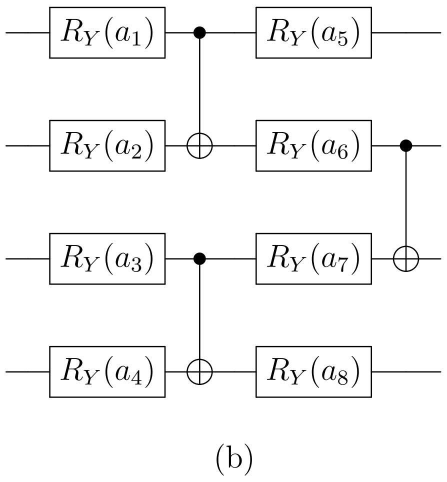


Figure 3: R_Y-C_X ansatz for 4 qubits.(a) linear entanglement (b) alternating entanglement,

Note that in place of the R_Y gates can be settled R_X or $R_X R_Z$ or $R_X R_Y$ etc. and in place of CX gates can be settled CZ or CH. Furthermore it can be proved the order of the CX gates does not matter. One can observe that the two different circuits use the same number of effective parameters (R_Y gates) but circuit (b) uses half the number of CX gates. Hence we consider the alternating entanglement ansatz as the optimal solution, since it is more efficient than the linear entanglement in terms of the number of effective parameters per two-qubit gate.

The Cost function

As it was previously stated, the cost function needs to decrease as the state $|\psi\rangle = |\psi(\boldsymbol{\alpha})\rangle := \mathbf{A}|x(\boldsymbol{\alpha})\rangle$ tends to become proportional to $|b\rangle$ and increase as $|\psi\rangle$ and $|b\rangle$ are close to being orthogonal. A suitable indicator that shows the overlap between $|\psi\rangle$ and $|b\rangle$ is their inner product $\langle\psi|b\rangle$. The general form of a cost function in VQA problems is:

$$\hat{C}_G = \langle\psi|H_G|\psi\rangle$$

A simple Hamiltonian H_G to take advantage of the overlap, is $H_G = \mathbb{I} - |b\rangle\langle b|$. Hence the cost funcion is:

$$\hat{C}_G = \langle\psi|(\mathbb{I} - |b\rangle\langle b|)|\psi\rangle = \langle\psi|\psi\rangle - |\langle b|\psi\rangle|^2$$

which takes values within the set $\hat{C}_G \in [0, \langle\psi|\psi\rangle]$ (taking into account that $|b\rangle$ is normalized). However a potential $|\psi\rangle$ with small norm relative to the tolerance γ could cause a premature termination of the algorithm, since the value of the cost function would be cosidered "small enough" at any point.

To tackle this issue, it is preferable to use a normalized version of this cost function

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

Global and Local cost function

If we proceed to write C_G more explicitly, using that $|b\rangle = U|\mathbf{0}\rangle$ and $|\psi\rangle = \mathbf{A}|x\rangle = \sum_{l=1}^{L} c_l A_l V(\boldsymbol{\alpha})|\mathbf{0}\rangle$, then we end up having the following expression:

$$C_G = 1 - \frac{\langle\psi|b\rangle\langle b|\psi\rangle}{\langle\psi|\psi\rangle} = 1 - \frac{\left(\sum_{l=1}^{L} c_l^* \langle\mathbf{0}|V^\dagger(\boldsymbol{\alpha})A_l^\dagger U|\mathbf{0}\rangle\right)\left(\sum_{l=1}^{L} c_l \langle\mathbf{0}|U^\dagger A_l V(\boldsymbol{\alpha})|\mathbf{0}\rangle\right)}{\left(\sum_{l=1}^{L} c_l^* \langle\mathbf{0}|V^\dagger(\boldsymbol{\alpha})A_l^\dagger\right)\left(\sum_{l=1}^{L} c_l A_l V(\boldsymbol{\alpha})|\mathbf{0}\rangle\right)} =$$

$$= 1 - \frac{\sum_{l=1}^{L}\sum_{l'=1}^{L} c_{l'} c_l^* \langle\mathbf{0}|V^\dagger(\boldsymbol{\alpha})A_l^\dagger U|\mathbf{0}\rangle\langle\mathbf{0}|U^\dagger A_{l'} V(\boldsymbol{\alpha})|\mathbf{0}\rangle}{\sum_{l=1}^{L}\sum_{l'=1}^{L} c_{l'} c_l^* \langle\mathbf{0}|V^\dagger(\boldsymbol{\alpha})A_l^\dagger A_{l'} V(\boldsymbol{\alpha})|\mathbf{0}\rangle}$$

The above is called **global cost function** and even though it works fine in most of the cases, it can still be vulnerable to barren plateaus[1] when it comes to a large number of qubits ($n \approx 50$). For this reason, an alternative cost funcion has been proposed, the **local cost function** :

$$\text{The local Hamiltonian is: } H_L = \mathbb{I} - \frac{1}{n}\sum_{j=1}^{n} |0_j\rangle\langle 0_j| \otimes \mathbb{I}_{\bar{j}}$$

$$C_L = 1 - \frac{\sum_{l=1}^{L}\sum_{l'=1}^{L} c_{l'}c_l^* \langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U \left(\frac{1}{n}\sum_{j=0}^{n-1}(|0_j\rangle\langle 0_j| \otimes \mathbb{I}_{\bar{j}})\right) U^\dagger A_{l'} V(\boldsymbol{\alpha}) |\mathbf{0}\rangle}{\sum_{l=1}^{L}\sum_{l'=1}^{L} c_{l'}c_l^* \langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger A_{l'} V(\boldsymbol{\alpha}) |\mathbf{0}\rangle} =$$

$$= 1 - \frac{1}{n}\frac{\sum_{l=1}^{L}\sum_{l'=1}^{L}\sum_{j=0}^{n-1} c_{l'}c_l^* \langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U(|0_j\rangle\langle 0_j| \otimes \mathbb{I}_{\bar{j}}) U^\dagger A_{l'} V(\boldsymbol{\alpha}) |\mathbf{0}\rangle}{\sum_{l=1}^{L}\sum_{l'=1}^{L} c_{l'}c_l^* \langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger A_{l'} V(\boldsymbol{\alpha}) |\mathbf{0}\rangle}$$

With $|0_j\rangle\langle 0_j|$ being the projection operator $|0\rangle\langle 0|$ applyed on qubit j and $\mathbb{I}_{\bar{j}}$ the identity on all qubits except j. Essentially, $|\mathbf{0}\rangle\langle \mathbf{0}|$ has been replaced with $\frac{1}{n}\sum_{j=0}^{n-1}(|0_j\rangle\langle 0_j| \otimes \mathbb{I}_{\bar{j}})$. Now instead of applying the cost function across all qubits (which is why barren plateaus occur) we have divided the problem up into multiple single-qubit terms, and summed all the results up.

It needs to be confirmed now, that the local cost function is indeed valid for our problem. It can be proved That

$$C_L \leq C_G \leq nC_L$$

which suggests that when $C_G \to 0$ so does the C_L. The same is true for the opposite way, when $C_L \to 0$ so does the C_G. This alone is enough to approve C_L for our purpose.

Evaluation of the cost function using quantum circuit

As it has already been mentioned, the algorithm requires a quantum circuit to receive as input the parameters $\boldsymbol{\alpha}$ and return the value of the cost function on point $\boldsymbol{\alpha}$, $C_G(\boldsymbol{\alpha})$ (or $C_L(\boldsymbol{\alpha})$).

[1]Barren plateau landscapes correspond to gradients that vanish exponentially in the number of qubits. In other words barren plateau describes a situation where the cost function landscape is flat and its gradient is almost 0, which prevents the optimization algorithm from making significant progress arround that area

Having defined our two different cost functions, we are left with the task of calculating the following quantities:

- if we use the global cost funtion we need to calculate the terms:

  $$\beta_{ll'} = \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle$$

  and

  $$\gamma_{ll'} = \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U | \mathbf{0} \rangle \langle \mathbf{0} | U^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle$$

- whereas, if we wish to use the local cost function the terms are:

  $$\beta_{ll'} = \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle$$

  and

  $$\delta_{ll'}^{(j)} = \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U (|0_j\rangle \langle 0_j| \otimes \mathbb{I}_{\bar{j}}) U^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle$$

The latter ($\delta_{ll'}^{(j)}$), can be rewritten more conviniently by noticing that $|0_j\rangle \langle 0_j| = \frac{1}{2}(\mathbb{I}_j + Z_j)$[1]. The reformed term will be :

$$\begin{aligned}
\delta_{ll'}^{(j)} &= \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U \left(\frac{1}{2}(\mathbb{I}_j + Z_j) \otimes \mathbb{I}_{\bar{j}} \right) U^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle = \\
&= \frac{1}{2} \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U U^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle + \frac{1}{2} \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U (Z_j \otimes \mathbb{I}_{\bar{j}}) U^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle = \\
&\overset{2}{=} \frac{1}{2} \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle + \frac{1}{2} \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U (Z_j \otimes \mathbb{I}_{\bar{j}}) U^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle = \\
&= \frac{1}{2} \beta_{ll'} + \frac{1}{2} \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U (Z_j \otimes \mathbb{I}_{\bar{j}}) U^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle
\end{aligned}$$

or equivalently, the factor $\frac{1}{2}$ can be ignored:

$$\delta_{ll'}^{(j)} = \beta_{ll'} + \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U (Z_j \otimes \mathbb{I}_{\bar{j}}) U^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle$$

The Hadamard Test

In order to calculate terms similar to the ones above, we employ a quantum circuit called the Hadamard test.

[1] Z_j indicates the operator Z applyed on qubit j

[2] since U is unitary $UU^\dagger = \mathbb{I}$

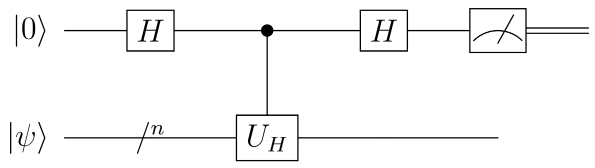


Figure 4: The Hadamard Test. Calculates the Real part of the expected value of U_H with respect to state $|\psi\rangle$

The Hadamard test in Figure 4 can be used to calculate the term $\mathrm{Re}\left[\langle\psi|U_H|\psi\rangle\right]$, where U_H is some unitary operator and $|\psi\rangle$ is an arbitrary state. The first qubit used to control the U_H is called the **achilla** and if it is mesured, the probability of being 0 is $\frac{1}{2}(1+\mathrm{Re}\left[\langle\psi|U_H|\psi\rangle\right])$ and the probability of being 1 is $\frac{1}{2}(1-\mathrm{Re}\left[\langle\psi|U_H|\psi\rangle\right])$. Their difference is

$$P(0)-P(1)=\frac{1}{2}(1+\mathrm{Re}\left[\langle\psi|U_H|\psi\rangle\right])-\frac{1}{2}(1-\mathrm{Re}\left[\langle\psi|U_H|\psi\rangle\right])=\mathrm{Re}\left[\langle\psi|U_H|\psi\rangle\right]$$

For the sake of completeness, the proof is the following:

$$|0\rangle\otimes|\psi\rangle \xrightarrow{H\text{ on first bit}} \frac{1}{\sqrt{2}}(|0\rangle\otimes|\psi\rangle+|1\rangle\otimes|\psi\rangle) \xrightarrow{\text{applying control }U_H} \frac{1}{\sqrt{2}}(|0\rangle\otimes|\psi\rangle+|1\rangle\otimes U_H|\psi\rangle)$$
$$\xrightarrow{H\text{ on first bit}} \frac{1}{2}(|0\rangle\otimes|\psi\rangle+|1\rangle\otimes|\psi\rangle+|0\rangle\otimes U_H|\psi\rangle-|1\rangle\otimes U_H|\psi\rangle)$$
$$=\frac{1}{2}(|0\rangle\otimes(\mathbb{I}+U_H)|\psi\rangle+|1\rangle\otimes(\mathbb{I}-U_H)|\psi\rangle)$$

Thus

$$\begin{aligned}P(0)&=\frac{1}{4}\langle\psi|(\mathbb{I}+U_H)(\mathbb{I}+U_H^\dagger)|\psi\rangle=\frac{1}{4}\langle\psi|(\mathbb{I}^2+U_H+U_H^\dagger+U_HU_H^\dagger)|\psi\rangle=\\&=\frac{1}{4}\langle\psi|(2\mathbb{I}+U_H+U_H^\dagger)|\psi\rangle=\frac{1}{4}\Big(2\langle\psi|\,\mathbb{I}\,|\psi\rangle+\langle\psi|U_H|\psi\rangle+\langle\psi|U_H^\dagger|\psi\rangle\Big)=\\&=\frac{1}{4}\Big(2+\langle\psi|U_H|\psi\rangle+(\langle\psi|U_H^\dagger|\psi\rangle)^*\Big)=\frac{1}{2}(1+\mathrm{Re}\left[\langle\psi|U_H|\psi\rangle\right])\end{aligned}$$

$P(1)$ is calculated similarly

To obtain the Imaginary part, we simple add a $S^\dagger$ after the first hadamard gate on the ancilla qubit

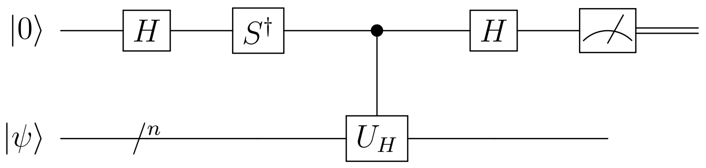


Figure 5: The Hadamard Test. Calculates the Imaginary part of the expected value of U_H with respect to state $|\psi\rangle$

$$|0\rangle \otimes |\psi\rangle \xrightarrow{H \text{ on first bit}} \frac{1}{\sqrt{2}}(|0\rangle \otimes |\psi\rangle + |1\rangle \otimes |\psi\rangle) \xrightarrow{S^\dagger \text{ on first bit}} \frac{1}{\sqrt{2}}(|0\rangle \otimes |\psi\rangle - i\,|1\rangle \otimes |\psi\rangle)$$
$$\xrightarrow{\text{applying control } U_H} \frac{1}{\sqrt{2}}(|0\rangle \otimes |\psi\rangle - i\,|1\rangle \otimes U_H\,|\psi\rangle)$$
$$\xrightarrow{H \text{ on first bit}} \frac{1}{2}(|0\rangle \otimes |\psi\rangle + |1\rangle \otimes |\psi\rangle - i\,|0\rangle \otimes U_H\,|\psi\rangle + i\,|1\rangle \otimes U_H\,|\psi\rangle)$$
$$= \frac{1}{2}(|0\rangle \otimes (\mathbb{I} - iU_H)\,|\psi\rangle + |1\rangle \otimes (\mathbb{I} + iU_H)\,|\psi\rangle)$$

Thus

$$P(0) = \frac{1}{2}(1 + \operatorname{Im}[\langle\psi|U_H|\psi\rangle])$$

Same procedure for $P(1)$

Having explained the functionality of the Hadamard test, what remains to be seen how to make use of it in order to compute the terms of the VLQS ploblem, starting off with the terms $\beta_{ll'}$.

$$\beta_{ll'} = \langle\mathbf{0}|\,V^\dagger(\boldsymbol{\alpha})A_l^\dagger A_{l'}V(\boldsymbol{\alpha})\,|\mathbf{0}\rangle$$

By setting $|\psi\rangle = V(\boldsymbol{\alpha})\,|\mathbf{0}\rangle$ and $U_H = A_l^\dagger A_{l'}$, we obtain that $\beta_{ll'} = \langle\psi|U_H|\psi\rangle$ which is similar to the Hadamard test form. Hence $\beta_{ll'}$ can be calculated using the circuit bellow:

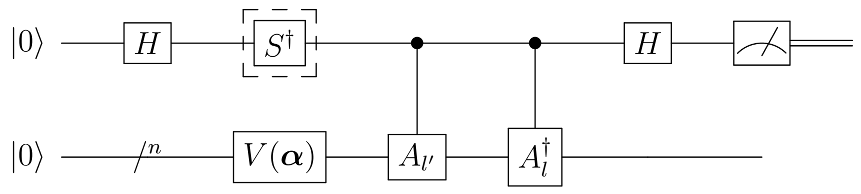


Figure 6: The Hadamard Test for evaluating the terms $\beta_{ll'} = \langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger A_{l'} V(\boldsymbol{\alpha}) |\mathbf{0}\rangle$. The $S^\dagger$ gate in dashed box indicates, the gate's involvment only when calculating the Imaginary part of $\beta_{ll'}$.

In succesion, if we proceed we the the global cost function we have to evaluate the terms

$$\gamma_{ll'} = \langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U |\mathbf{0}\rangle \langle \mathbf{0}| U^\dagger A_{l'} V(\boldsymbol{\alpha}) |\mathbf{0}\rangle$$

which can be accomplished in three different ways:

- The terms $\langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U |\mathbf{0}\rangle$ and $\langle \mathbf{0}| U^\dagger A_{l'} V(\boldsymbol{\alpha}) |\mathbf{0}\rangle$ will be evaluated separately by using two different Hadamard Tests with the $|\psi\rangle = |0\rangle$ and by controlling each one of the inner unitaries

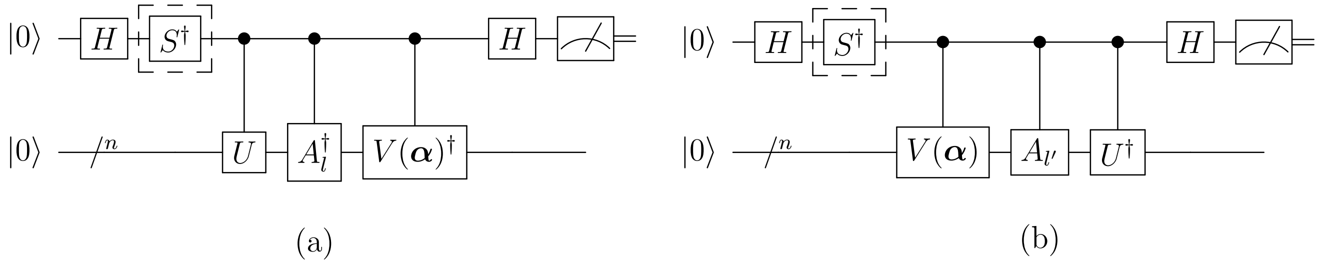


Figure 7: Two different Hadamard Tests for evaluating the terms $\langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U |\mathbf{0}\rangle$ (circuit (a) on the left) and $\langle \mathbf{0}| U^\dagger A_{l'} V(\boldsymbol{\alpha}) |\mathbf{0}\rangle$ (circuit (b) on the right)

- For the second way, we write the second term as

$$\langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U |\mathbf{0}\rangle = (\langle \mathbf{0}| U^\dagger A_l V(\boldsymbol{\alpha}) |\mathbf{0}\rangle)^\dagger = (\langle \mathbf{0}| U^\dagger A_l V(\boldsymbol{\alpha}) |\mathbf{0}\rangle)^*$$

  Thus, instead of using two different Hadamard Tests, it is only required to use the first one (Figure 7 circuit (b)) twice.

- The thrid method, wich is probably the optimal one, is using a different subroutine called the **Hadamard-Overlap Test**. Its advantage over the method with the regular Hadamard Test is that it doesn't require to control the ansazt or the U at the expense of using $2n+1$ qubits onstead of $n+1$. Further explanation of the Hadamard-Overlap Test will not be held in this report

Alternatly if we proceed with the local cost function we have to evaluate the terms

$$\delta_{ll'}^{(j)} = \beta_{ll'} + \langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U (Z_j \otimes \mathbb{I}_{\bar{j}}) U^\dagger A_{l'} V(\boldsymbol{\alpha}) |\mathbf{0}\rangle$$

For the Hadamard Test of this term evaluation we set

$$|\psi\rangle = V(\boldsymbol{\alpha}) |\mathbf{0}\rangle \quad \text{and} \quad U_H = A_l^\dagger U (Z_j \otimes \mathbb{I}_{\bar{j}}) U^\dagger A_{l'}$$

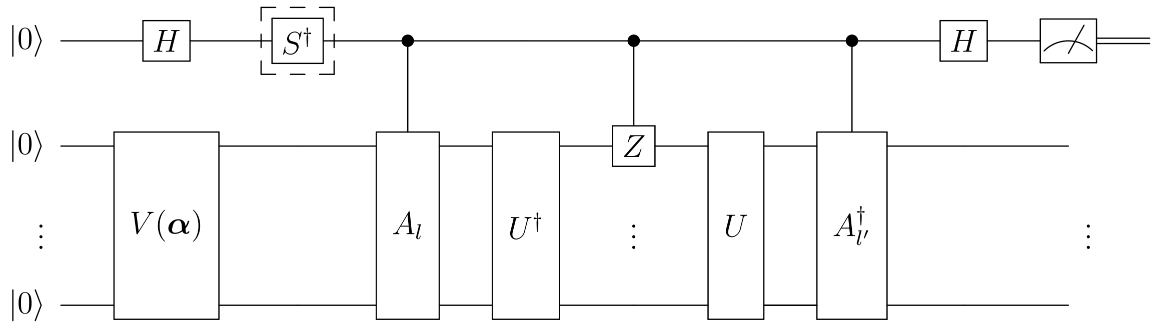


Figure 8: The Hadamard Test for evaluating the terms $\delta_{ll'}^{(j)} = \beta_{ll'} + \langle \mathbf{0}| V^\dagger(\boldsymbol{\alpha}) A_l^\dagger U (Z_j \otimes \mathbb{I}_{\bar{j}}) U^\dagger A_{l'} V(\boldsymbol{\alpha}) |\mathbf{0}\rangle$.

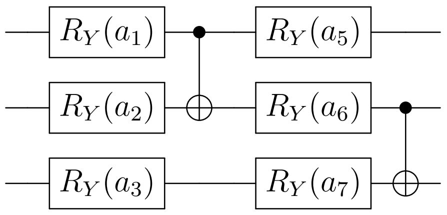
One can notice that the operators U and $U^\dagger$ do not need to be controlled. This is because if the ancilla bit is 0 the operator $Z_j \otimes \mathbb{I}_{\bar{j}}$ will not apply and therefore U and $U^\dagger$ will act successivly. But since U is unitary $UU^\dagger = \mathbb{I}$. Thus, it makes no difference whether we control them or not.

Qiskit Implementation

Using the theoritical analysis above, we proceed to implement the VQLS algorithm in qiskit. The algorithm should solve the 3-qubits linear equations system $\mathbf{A}\mathbf{x} = \mathbf{b}$ where

$$\mathbf{A} = 0.4H_2 + 0.3Z_1 + 0.3X_3 \quad \text{and} \quad |b\rangle = H_1 \otimes H_2 \otimes H_3 \, |\mathbf{0}\rangle$$

For the ansatz we have chosen a circuit similar to the one in Figure 3 (a). Even though we explained that the circuit with alternating entanglement is more efficient than the one with the linear entanglement, for the 3-qubits case, an alternating entanglement circuit like the following



would reduce the number of effective parameters (a_3 and a_7 would count as one parameter). So the implemented ansatz is

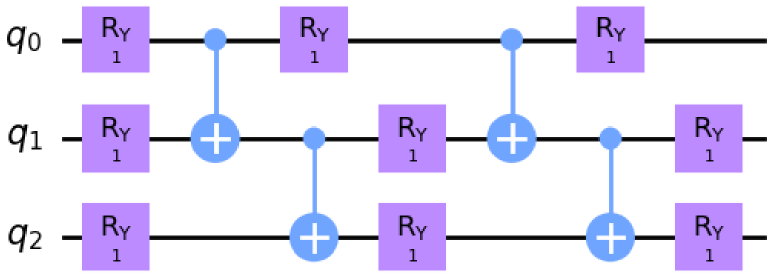


Figure 9: Linear entanglement ansatz implemented in qiskit with 9 effective parameters (all initialized at 1) and 4 CNOTs gates.

For the cost function we use the global cost function and the terms $\gamma_{ll'}$ are evaluated with the second method that was described above (Figure 7 (b)).

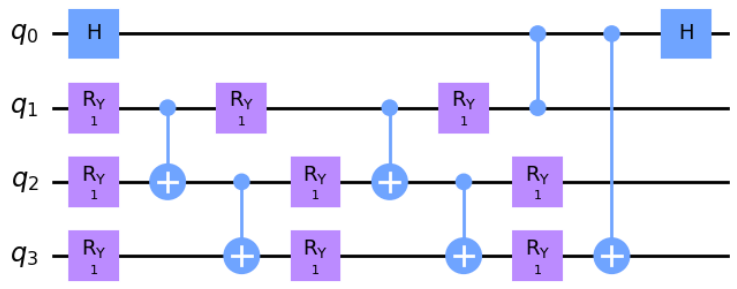


Figure 10: Hadamard Test for evaluating the terms $\beta_{ll'}$. With $A_l = Z_1$ and $A_{l'} = X_3$

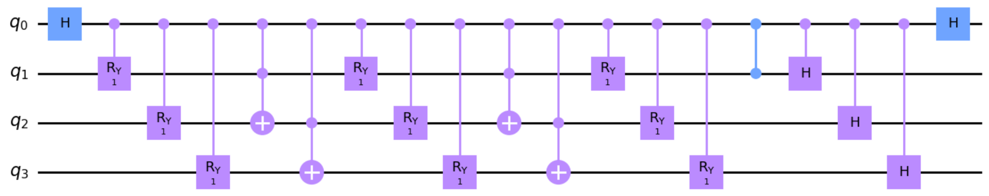


Figure 11: Hadamard Test for evaluating the terms $\gamma_{ll'}$. With $A_l = Z_1$

Note how in the last circuit, U and $V(\boldsymbol{\alpha})$ are also controlled alongside A_l. The eventual cost function is :

$$C_G = 1 - \frac{\sum_{l=1}^{L} \sum_{l'=1}^{L} c_{l'} c_l^* \gamma_{ll'}}{\sum_{l=1}^{L} \sum_{l'=1}^{L} c_{l'} c_l^* \beta_{ll'}}$$

Our goal is to minimize it until it reaches the value 0. To do so we employ the COBYLA optimizer which is gradient-free. Since we used the global cost function, a gradient-free optimizer is convinient because it is less impacted by barren plateaus. The final result of the optimization process is the following:

```
     fun: 0.00012637933571213456
   maxcv: 0.0
 message: 'Maximum number of function evaluations has been exceeded.'
    nfev: 200
  status: 2
 success: False
       x: array([ 3.31224364,  1.54576436,  4.69185342, -0.94657516,  1.98048994,
         0.04202637,  0.11539815,  2.38380735,  3.18573665])
```

`fun` is the final value of the cost function which is close to 0, as desired. `x` is the optimal point $\boldsymbol{\alpha}_{opt}$ which corresponds to the solution $|x\rangle = V(\boldsymbol{\alpha}_{opt})|\mathbf{0}\rangle$. Finally, we attempt to confirm that the obtained solution $|x\rangle$ is valid, by calculating the inner product $\langle x\mathbf{A}|b\rangle$ and expecting it to be 1

```
<xA|b> = (0.9998736206641057-0j)
```

As we can see, it is indeed close to 1 with a very low deviation.

Referances

VQLS Algorithm

https://arxiv.org/pdf/1909.05820.pdf

https://qiskit.org/textbook/ch-paper-implementations/vqls.html

https://pennylane.ai/qml/demos/tutorial_vqls.html

Ansatz

https://arxiv.org/pdf/2111.13730.pdf

Cost Function

https://pennylane.ai/qml/demos/tutorial_local_cost_functions.html