

# VARIATIONAL QUANTUM LINEAR SOLVER

Introduction to Quantum Computing

Kalamarakis Theodoros: 2018030022

*February 8 , 2023*

GitHub Repository

[https://github.com/kthod/VQLS\\_algorithm.git](https://github.com/kthod/VQLS_algorithm.git)

# Introduction

$$A\mathbf{x} = \mathbf{b} \text{ where } A \in \mathbb{C}^{N \times N} \text{ and } \mathbf{x}, \mathbf{b} \in \mathbb{C}^N$$

Existing classical algorithms solves the problem with polynomial scaling in  $N$

# Introduction

$$A\mathbf{x} = \mathbf{b} \text{ where } A \in \mathbb{C}^{N \times N} \text{ and } \mathbf{x}, \mathbf{b} \in \mathbb{C}^N$$

Exponential speedup can be achieved using the quantum algorithm **HHL** which:

- Determines a quantum state  $|\mathbf{x}\rangle$  that is proportional to  $\mathbf{x} : |\mathbf{x}\rangle \sim \mathbf{x}$
- Achieves polylogarithmic scaling in  $N$

# Introduction

$Ax = b$  where  $A \in \mathbb{C}^{N \times N}$  and  $x, b \in \mathbb{C}^N$

Exponential speedup can be achieved using the quantum algorithm **HHL** which:

- Determines a quantum state  $|x\rangle$  that is proportional to  $x : |x\rangle \sim x$
- Achieves polylogarithmic scaling in  $N$

However

- Demanding implementation

# Introduction

$$A\mathbf{x} = \mathbf{b} \text{ where } A \in \mathbb{C}^{N \times N} \text{ and } \mathbf{x}, \mathbf{b} \in \mathbb{C}^N$$

Exponential speedup can be achieved using the quantum algorithm **HHL** which:

- Determines a quantum state  $|x\rangle$  that is proportional to  $\mathbf{x} : |x\rangle \sim \mathbf{x}$
- Achieves polylogarithmic scaling in  $N$

However

- Demanding implementation

Instead a Variational Hybrid Quantum Classical Algorithm can be used

**The Variational Quantum Linear Solver (VQLS)**

- Can be performed in NISQ quantum computers

# Complexity of the VQLS algorithm

- Complexity analysis on heuristic algorithms like the VHQCA is difficult

# Complexity of the VQLS algorithm

- Complexity analysis on heuristic algorithms like the VHQCA is difficult
- However complexity can be determined by performing numerical simulation

# Complexity of the VQLS algorithm

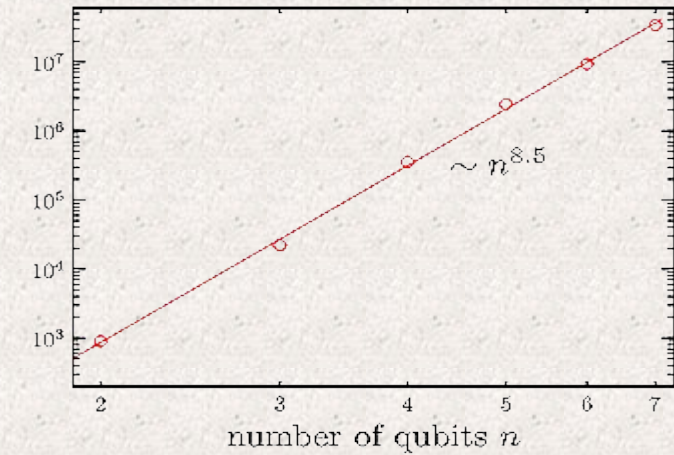
- Complexity analysis on heuristic algorithms like the VHQCA is difficult
- However complexity can be determined by performing numerical simulation
- The obtained relation for running time scaling with  $n$  is :

$$y \sim n^{8.5}$$

$$\text{Since } N = 2^n \Leftrightarrow n = \log N$$

$$y \sim (\log N)^{8.5}$$

Which is polylogarithmical in  $N$



Time to solution scaling with  $n$ . Figure taken from original paper



Time to solution scaling with  $n$ . Measurements taken using our implementation of VQLS



# Description of the algorithm

## Input

- A matrix  $\mathbf{A}$  that, must be given as a linear combination of  $L$  unitaries  $A_1, A_2, \dots, A_L$  such that

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

where  $c_l$  is a complex number

- An operator  $U$  that prepares a quantum state  $|b\rangle$  that is proportional to the vector  $\mathbf{b}$ , such that

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

# Description of the algorithm

## Input

- A matrix  $\mathbf{A}$  that, must be given as a linear combination of  $L$  unitaries  $A_1, A_2, \dots, A_L$  such that

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

where  $c_l$  is a complex number

- An operator  $U$  that prepares a quantum state  $|b\rangle$  that is proportional to the vector  $\mathbf{b}$ , such that

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

## Goal

Determine a state  $|x\rangle$  such that  $A|x\rangle$  is proportional to  $|b\rangle$ , or equivalently:

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

# Description of the algorithm

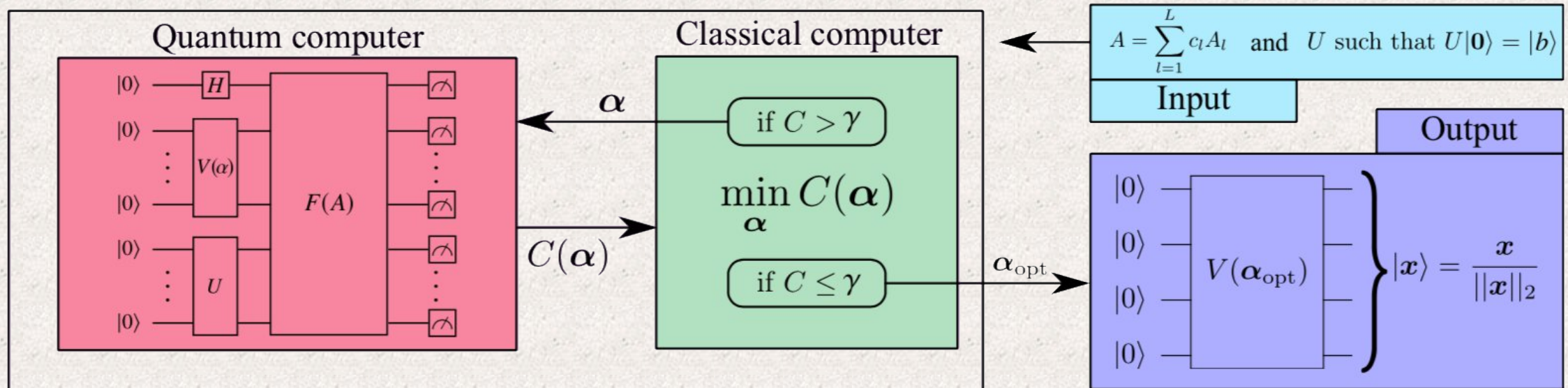
- $|x\rangle$  is approximated using a set of parameters  $\alpha = (\alpha_1, \alpha_2, \dots)$  such that
  - $|x(\alpha)\rangle = V(\alpha)|0\rangle$

The operator  $V(\alpha)$  is called **Ansatz** and is capable of generating any arbitrary state in Hilbert space

- The configuration of the parameters  $\alpha$  is happening through a classical optimizer that aims to minimize a **cost function**  $C(\alpha)$
- A quantum circuit evaluates the cost function  $C(\alpha)$

# Description of the algorithm

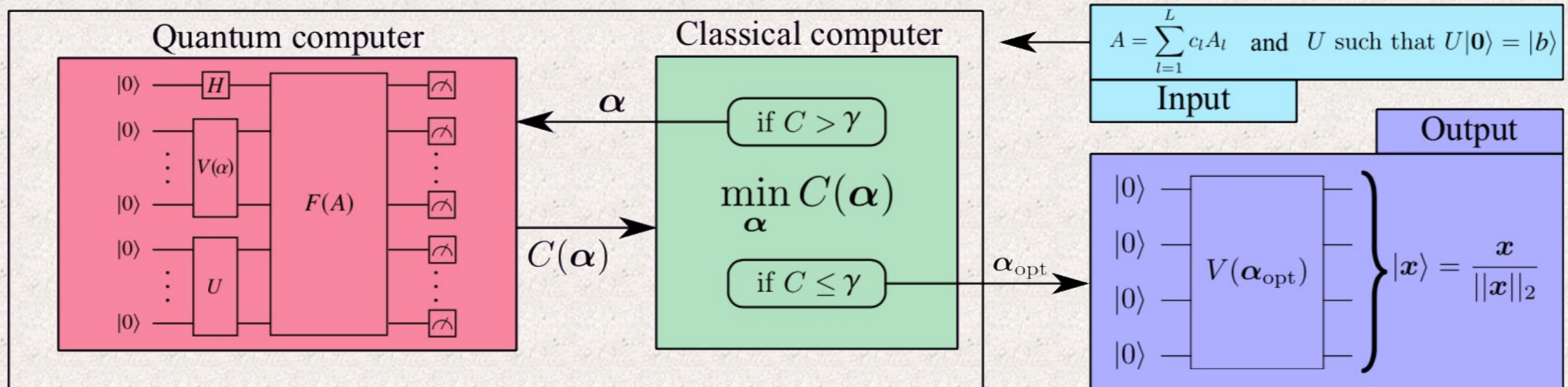
Interaction between the Quantum circuit and the classical optimizer



# Description of the algorithm

## Interaction between the Quantum circuit and the classical optimizer

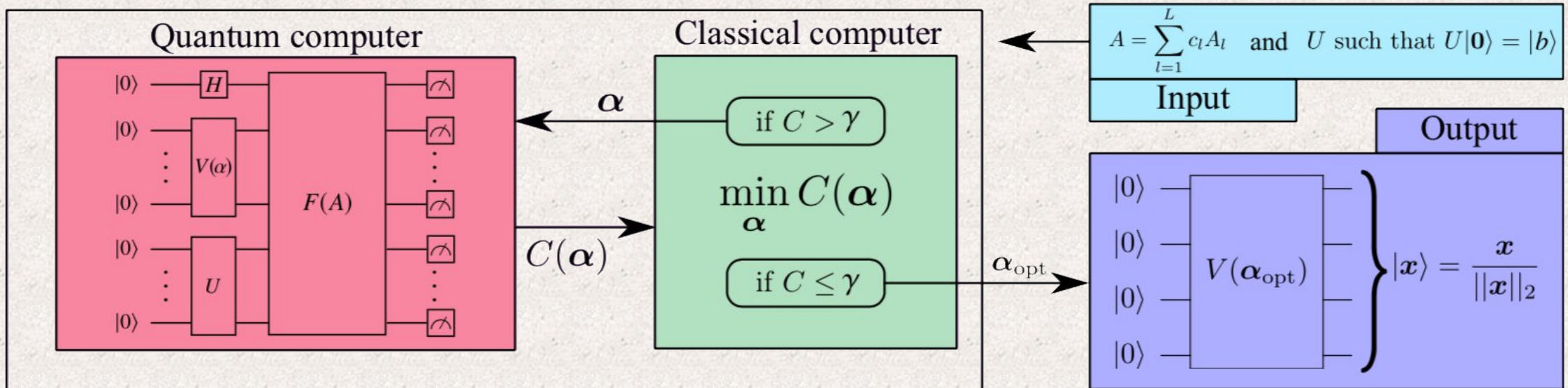
- The quantum circuit receives as input the parameters  $\alpha$  and evaluates the cost function on this point  $C(\alpha)$



# Description of the algorithm

## Interaction between the Quantum circuit and the classical optimizer

- The quantum circuit receives as input the parameters  $\alpha$  and evaluates the cost function on this point  $C(\alpha)$
- The classical optimizer, uses the output value  $C(\alpha)$ , to determine a new set of parameters  $\alpha'$  such that,  $C(\alpha') < C(\alpha)$

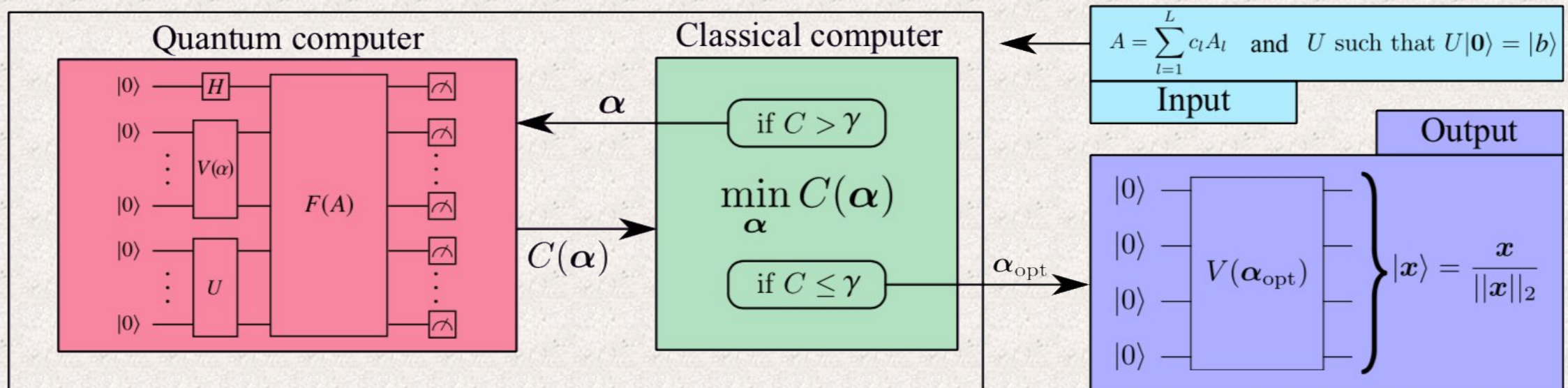




# Description of the algorithm

## Interaction between the Quantum circuit and the classical optimizer

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



# The Ansatz

Capable of generating any arbitrary state on Hilbert space, including the entangled ones



# The Ansatz

Capable of generating any arbitrary state on Hilbert space, including the entangled ones

Consists of :

- Rotation single-qubit gates  $R_X, R_Y, R_Z$  for “exploring” the space. Each one of the rotation gates should correspond to one of the parameters  $\alpha = (\alpha_1, \alpha_2, \dots)$
- Entanglement two-qubit gates  $CNOT, CZ$  to provide entanglement

# The Ansatz

Capable of generating any arbitrary state on Hilbert space, including the entangled ones

Consists of :

- Rotation single-qubit gates  $R_X, R_Y, R_Z$  for “exploring” the space. Each one of the rotation gates should correspond to one of the parameters  $\alpha = (\alpha_1, \alpha_2, \dots)$
- Entanglement two-qubit gates  $CNOT, CZ$  to provide entanglement

Distinguishing two types of Ansatzes:

- The Fixed-Structure Ansatz
- The Variable-Structure Ansatz

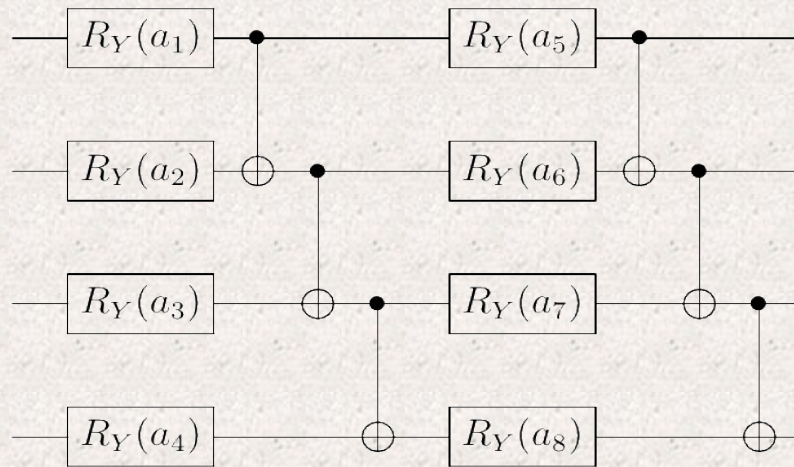
We are only concerned with Fixed-Structure Ansatz

# The Ansatz

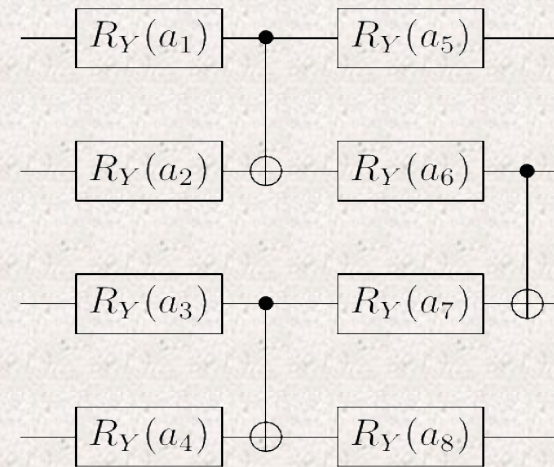
## Fixed-Structure Ansatz

Placement of the gates remains the same throughout the execution of the algorithm

Two examples are the following:



linear entanglement



alternating entanglement

Ansatz with alternating entanglement is more efficient in terms of effective parameters per two-qubit gate

# The Cost Function

Define  $|\psi\rangle = |\psi(\alpha)\rangle := \mathbf{A} |x(\alpha)\rangle$

We construct a cost function which takes its minimum value  $C=0$  when  $|\psi\rangle$  and  $|b\rangle$  have the maximum overlap

The general form of a cost function in VQA is  $C_G = \langle \psi | H_G | \psi \rangle$

# The Cost Function

Define  $|\psi\rangle = |\psi(\alpha)\rangle := \mathbf{A} |x(\alpha)\rangle$

We construct a cost function which takes its minimum value  $C=0$  when  $|\psi\rangle$  and  $|b\rangle$  have the maximum overlap

The general form of a cost function in VQA is  $C_G = \langle \psi | H_G | \psi \rangle$

A suitable Hamiltonian  $H_G$  to take advantage of the overlap is

$$H_G = \mathbb{I} - |b\rangle\langle b| = U (\mathbb{I} - |0\rangle\langle 0|) U^\dagger.$$

Hence the cost function is

$$\hat{C}_G = \langle \psi | (\mathbb{I} - |b\rangle\langle b|) | \psi \rangle = \langle \psi | \psi \rangle - |\langle b | \psi \rangle|^2 \quad \text{with } 0 \leq \hat{C}_G \leq \langle \psi | \psi \rangle$$

# The Cost Function

Define  $|\psi\rangle = |\psi(\alpha)\rangle := \mathbf{A} |x(\alpha)\rangle$

We construct a cost function which takes its minimum value  $C=0$  when  $|\psi\rangle$  and  $|b\rangle$  have the maximum overlap

The general form of a cost function in VQA is  $C_G = \langle \psi | H_G | \psi \rangle$

A suitable Hamiltonian  $H_G$  to take advantage of the overlap is

$$H_G = \mathbb{I} - |b\rangle\langle b| = U (\mathbb{I} - |0\rangle\langle 0|) U^\dagger.$$

Hence the cost function is

$$\hat{C}_G = \langle \psi | (\mathbb{I} - |b\rangle\langle b|) | \psi \rangle = \langle \psi | \psi \rangle - |\langle b | \psi \rangle|^2 \quad \text{with } 0 \leq \hat{C}_G \leq \langle \psi | \psi \rangle$$

A potential small  $\langle \psi | \psi \rangle$  would be problematic, so we use instead a normalized version of the cost function

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

# The Cost Function

Using that  $|b\rangle = U|0\rangle$  and  $|\psi\rangle = A|x(\alpha)\rangle = \sum_{l=1}^L c_l A_l V(\alpha)|0\rangle$

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

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



# The Cost Function

Using that  $|b\rangle = U|0\rangle$  and  $|\psi\rangle = A|x(\alpha)\rangle = \sum_{l=1}^L c_l A_l V(\alpha)|0\rangle$

$$\begin{aligned} C_G &= 1 - \frac{\langle\psi|b\rangle\langle b|\psi\rangle}{\langle\psi|\psi\rangle} = 1 - \frac{\left(\sum_{l=1}^L c_l^* \langle 0| V^\dagger(\alpha) A_l^\dagger U |0\rangle\right) \left(\sum_{l=1}^L c_l \langle 0| U^\dagger A_l V(\alpha) |0\rangle\right)}{\left(\sum_{l=1}^L c_l^* \langle 0| V^\dagger(\alpha) A_l^\dagger\right) \left(\sum_{l=1}^L c_l A_l V(\alpha) |0\rangle\right)} = \\ &= 1 - \frac{\sum_{l=1}^L \sum_{l'=1}^L c_{l'} c_l^* \langle 0| V^\dagger(\alpha) A_l^\dagger U |0\rangle \langle 0| U^\dagger A_{l'} V(\alpha) |0\rangle}{\sum_{l=1}^L \sum_{l'=1}^L c_{l'} c_l^* \langle 0| V^\dagger(\alpha) A_l^\dagger A_{l'} V(\alpha) |0\rangle} \end{aligned}$$

The above is called **Global Cost Function**

- Can exhibit barren plateaus for a large number of qubits



# The Cost Function

To tackle this problem we divide the problem up into multiple single-qubit terms by replacing the term  $|0\rangle\langle 0|$  with  $\frac{1}{n} \sum_{j=0}^{n-1} (|0_j\rangle\langle 0_j| \otimes \mathbb{I}_{\bar{j}})$ .

The produced cost is function is called **Local Cost Function**  $H_L = \mathbb{I} - \frac{1}{n} \sum_{j=1}^n |0_j\rangle\langle 0_j| \otimes \mathbb{I}_{\bar{j}}$

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

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

To prove that the local cost function suits our problem we use that

$$C_L \leq C_G \leq n C_L$$

Which implies that  $C_G \rightarrow 0 \Leftrightarrow C_L \rightarrow 0$

# The Cost Function

## Evaluation of the cost function using quantum circuit

- For Global Cost Function we have to evaluate the terms:

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

and

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

- For Local Cost Function we have to evaluate the terms:

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

and

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

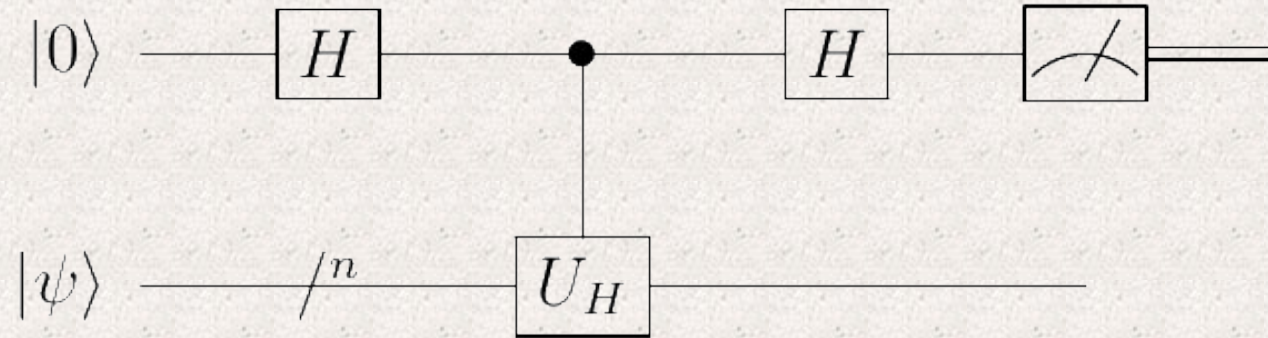
Using that  $|0_j\rangle \langle 0_j| = \frac{1}{2}(I_j + Z_j)$ , the last one can be rewritten as

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

# The Cost Function

How can we evaluate the previous terms?

We employ a quantum circuit called **The Hadamard Test**



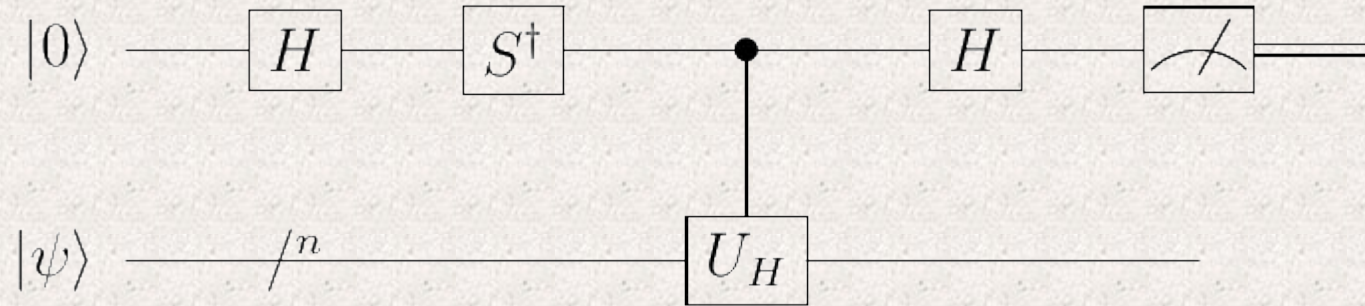
The circuit above can calculate the term  $Re [\langle \psi | U_H | \psi \rangle]$

- The probability of measuring the first qubit (**ancilla**) to be 0 is  $P(0) = \frac{1}{2}(1 + Re [\langle \psi | U_H | \psi \rangle])$
- The probability of measuring the ancilla qubit to be 1 is  $P(1) = \frac{1}{2}(1 - Re [\langle \psi | U_H | \psi \rangle])$

Hence  $P(0) - P(1) = Re [\langle \psi | U_H | \psi \rangle]$

# The Cost Function

For the Imaginary part we simple add a  $S^\dagger$  gate after the first Hadamard on the ancilla qubit



The circuit above can calculate the term  $Im [\langle \psi | U_H | \psi \rangle]$

- The probability of measuring the ancilla qubit to be 0 is  $P(0) = \frac{1}{2}(1 + Im [\langle \psi | U_H | \psi \rangle])$
- The probability of measuring the ancilla qubit to be 1 is  $P(1) = \frac{1}{2}(1 - Im [\langle \psi | U_H | \psi \rangle])$

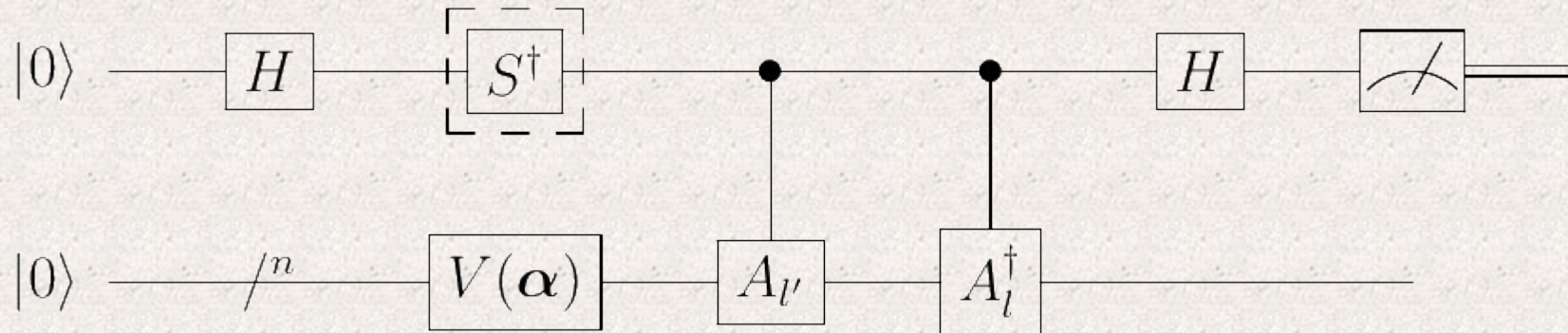
Hence  $P(0) - P(1) = Im [\langle \psi | U_H | \psi \rangle]$

# The Cost Function

How can we apply the hadamard test for our purpose?

To calculate the terms  $\beta_{ll'} = \langle 0 | V^\dagger(\alpha) A_l^\dagger A_{l'} V(\alpha) | 0 \rangle$  we set:

$|\psi\rangle = V(\alpha) |0\rangle$  and  $U_H = A_l^\dagger A_{l'}$

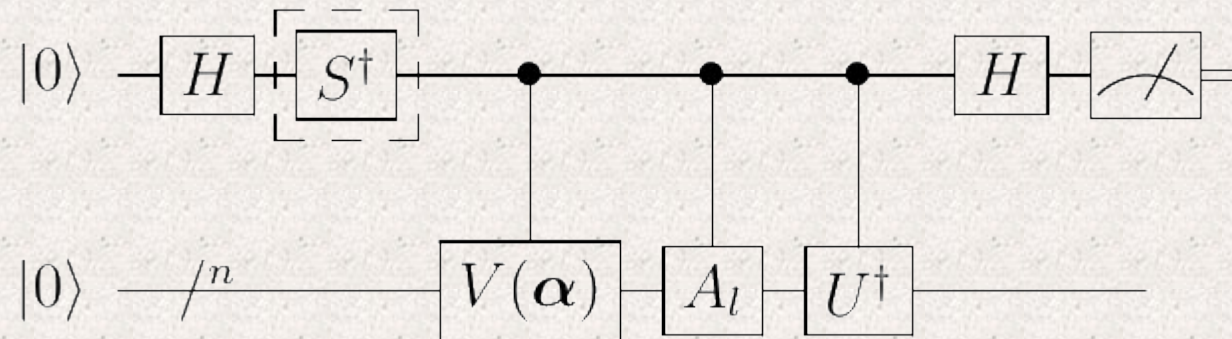


# The Cost Function

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

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

Thus by setting  $|\psi\rangle = |\mathbf{0}\rangle$  and  $U_H = U^\dagger A_l V(\boldsymbol{\alpha})$  we can calculate  $\gamma_{ll'}$  by using the following circuit twice



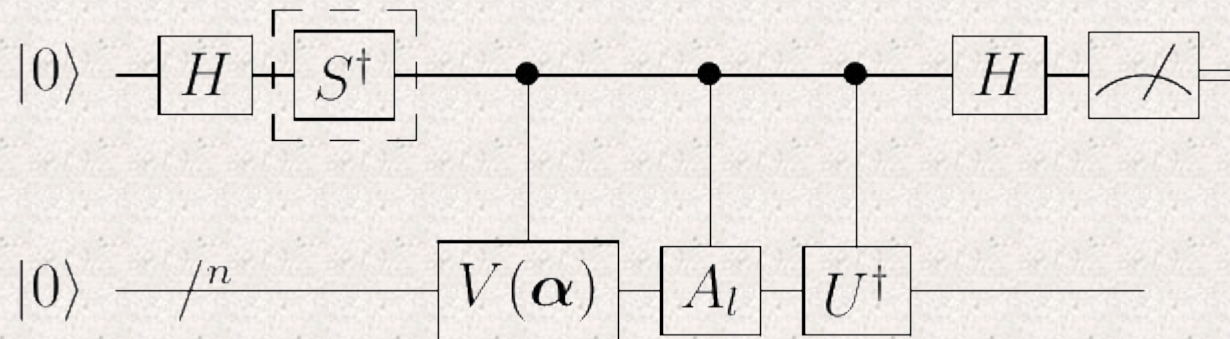


# The Cost Function

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

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

Thus by setting  $|\psi\rangle = |\mathbf{0}\rangle$  and  $U_H = U^\dagger A_l V(\boldsymbol{\alpha})$  we can calculate  $\gamma_{ll'}$  by using the following circuit twice

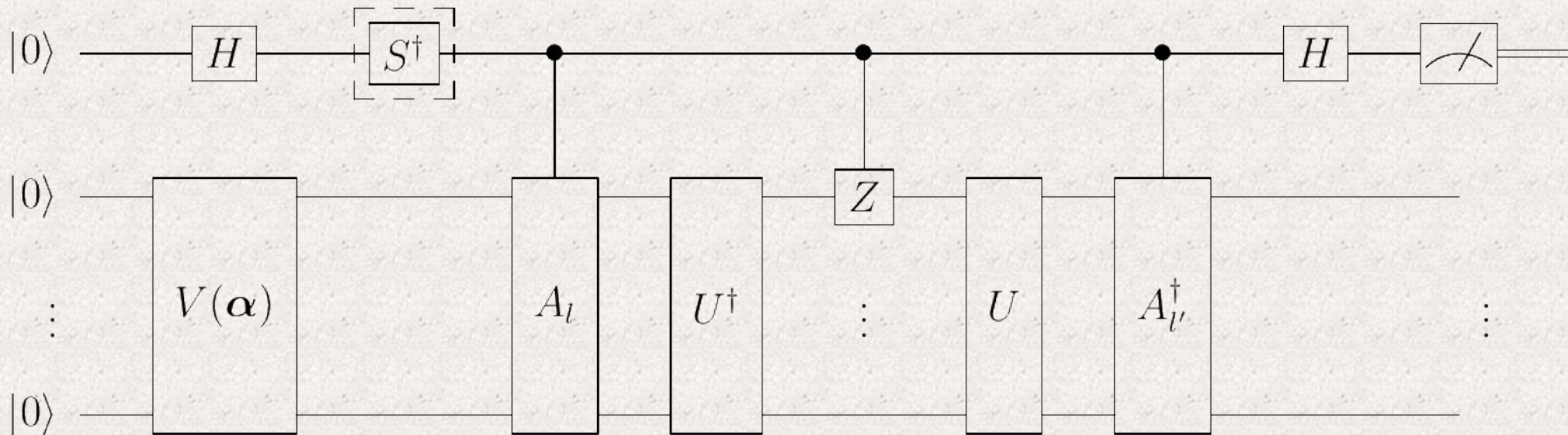


- An alternative way to calculate these term is by using an different subroutine called **The Hadamard-Overlap Test**

# The Cost Function

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

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





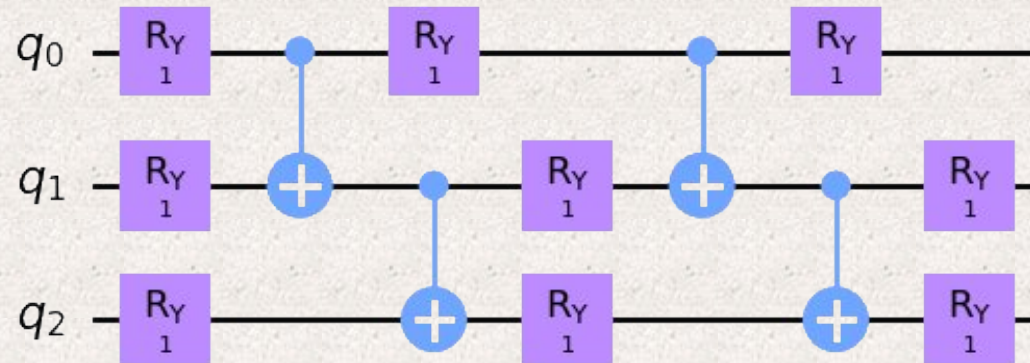
# Qiskit Implementation

Using Qiskit we attempt to implement the VQLS algorithm and solve an example of a 3-qubits linear equations system  $A\mathbf{x} = \mathbf{b}$  with

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

# Qiskit Implementation

Linear entanglement ansatz was used with 9 rotation gates  $R_Y$  (therefore 9 parameters) and 4 *CNOTs* gates

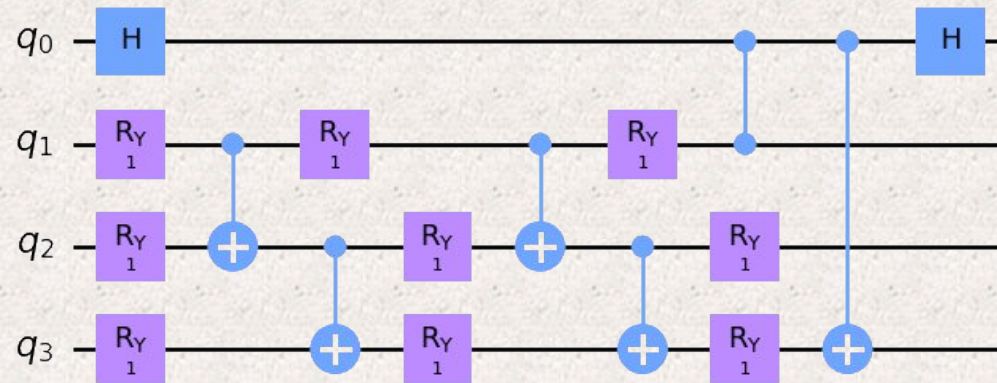


Since  $A$  and  $|b\rangle$  are real, only rotation around the  $y$ -axis is required

# Qiskit Implementation

Global cost function was used:

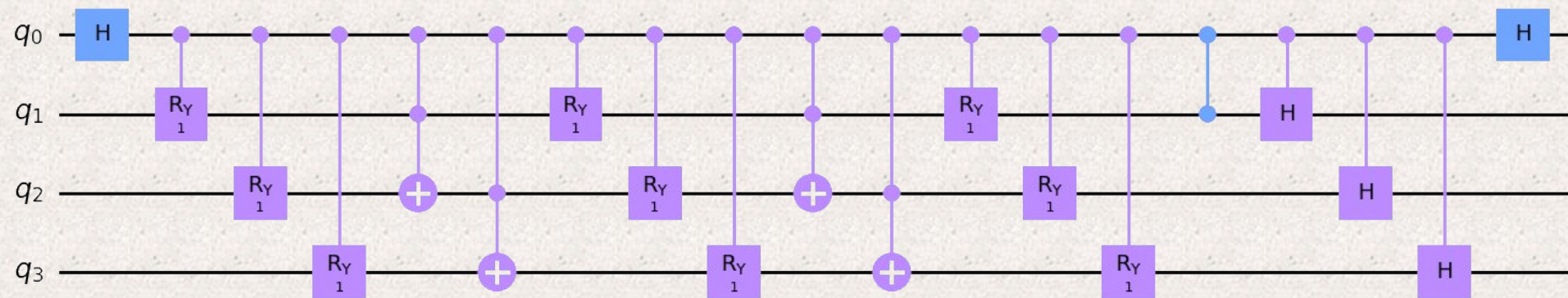
The Hadamard Test for evaluating the terms  $\beta_{ll'} = \langle \mathbf{0} | V^\dagger(\boldsymbol{\alpha}) A_l^\dagger A_{l'} V(\boldsymbol{\alpha}) | \mathbf{0} \rangle$



With  $A_l = Z_1$  and  $A_{l'} = X_3$

# Qiskit Implementation

... and for the terms  $\gamma_{ll'} = \langle 0 | V^\dagger(\alpha) A_l^\dagger U | 0 \rangle \langle 0 | U^\dagger A_{l'} V(\alpha) | 0 \rangle$



With  $A_l = Z_1$

# Qiskit Implementation

Our cost function is :

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

We minimize it using preferably a gradient free optimizer, like COBYLA, that is less vulnerable to barren plateaus

# Qiskit Implementation

The output of the algorithm is:

```
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 optimal point  $\alpha_{opt}$  which corresponds to the solution  $|x\rangle = V(\alpha_{opt})|0\rangle$

# Qiskit Implementation

To confirm that the obtained solution is correct we calculate the inner product

$$\langle xA|b\rangle$$

with the expectation of being close to 1. The output is:



# Qiskit Implementation

To confirm that the obtained solution is correct we calculate the inner product

$$\langle xA|b\rangle$$

with the expectation of being close to 1. The output is:

$$\langle xA|b\rangle = (0.9998736206641057-0j)$$

. . . It works !!!



# 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](https://pennylane.ai/qml/demos/tutorial_vqls.html)

## Anstaz

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

## Cost Function

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