The basics of quantum computing and the Variational Quantum Eigensolver algorithm - a numerical study of the Lipkin model Hamiltonian

Jonny Aarstad Igeh
Department of Physics, University of Oslo, Norway

March 26, 2024

Abstract

Modern technology will require immense computational power, and the current proccessor-technology is limited by physical contstraints, such as size, material and energy consumption. Quantum computing can potentially provide a solution to this computational demand by offering a new way of processing information. In this report, we have explored, and implemented, some of the basic concepts of quantum computing, and used the Variational Quantum Eigensolver (VQE) algorithm to compute the ground state energies of two simple Hamiltonians, and the Lipkin model.
We found that, by following the article in [2], with usage of the NumPy library, and simple matrix multiplications, can be used to simulate quantum circuits on a regular computer, and that the VQE algorithm can be used to good effect to find the ground states. Our implementation of the Lipkin Hamiltonian on our VQE successfully managed to capture the complex nature of entanglement in our system - and also found the avoided crossing in the energy spectrum. The conclusion is that these quantum computing algorithms have remarkable potential in solving such quantum systems, and that the future of quantum computing is brighter than ever.
The link to the GitHub repository containing all code can be found here: `https://github.com/Jonnyigeh/FYS5419-quantum-computing/tree/main/Project%201`

Contents

1 Introduction

The rapid development of modern technology has led to an increased demand for computational power. The classical computers we are familiar with today have certain limitations, and at some point in the (not-so) dinstant future we will reach a cap where the computational power achievable in a single (classical) processor unit will reach a limit. While the processing power may reach a limit, the processing needs of the world will not. This paves the way for a new era of computing, quantum computing.

Quantum computers are based on the principles of quantum mechanics, and have the potential to solve problems that are infeasible for classical computers, and also out-perform classical computers on certain metrics. Where a classical computer uses bits to represent information, a quantum computer uses quantum bits, or qubits, which have the possibility to contain more information per bit than a classical bit. This is due to the fact that a qubit can be in a superposition of states, and can also be entangled with other qubits. In this numerical report, we will explore the basics of quantum computing, and implement a simple quantum algorithm using our own code, the Variational Quantum Eigensolver, as well as comparing certain aspects of our code with the well-established QisKit library. We will be working with two different Hamiltonians. The first Hamiltonian is a simple one, which introduces the basics concepts for quantum computing and the VQE algorithm.
The second Hamiltonian is the Lipkin model.
Firstly, we will introduce necessary theory and concepts, and their numerical implementations, in the theory and method section. Then, we will present our results, and discuss these in the result section. Finally, we will conclude our findings and discuss the potential of quantum computing in the future.

2 Theory

2.1 Measurements

When we want to perform measurements on our quantum system, we need to measure the state of the qubits in a basis of choice. To properly mimic a quantum measurement, it is the *expectation value* that we should "measure". This means, that we need to make several measurements of the state, and then take the average of these measurements. A neat way to implement this will be presented in the method section.
An interesting, and important note, is that by clever usage of quantum gates - we can transform *any* Pauli measurement to a measurement in the computational basis (Z_0 measurement). This will be done by the transformations found in [3] pg. 251 - 252, table 6.1 and 6.2. This is a very useful property, as making measurements in the computational basis makes our life much simpler - and then we may find expectation values of any combination of Pauli gates, by doing Z_0 measurements on the transformed ansatz. For example, if we would like to perform an X_0 measurement, we would only need to apply the Hadamard gate to the first qubit, and then perform a Z_0 measurement.

2.2 Quantum Gates

Quantum gates are the building blocks of quantum circuits. They are the quantum analogues of classical logic gates, and are used to manipulate the state of a quantum system. In this section, we will discuss some of the most important quantum gates, and how they can be used to perform quantum computations.

2.2.1 The Pauli Gates

The Pauli gates are a group of three quantum gates, based on the Pauli matrices. They are the Pauli-X, Pauli-Y, and Pauli-Z gates, and are defined as follows:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The Pauli-X gate and the Pauli-Y gate are bit-flip gates, and the Pauli-Z gate is a phase-flip gate. They are all Hermitian and unitary, and are their own inverses. The Pauli-Y gate also introduce a phase shift on the qubit. These are arguably the most important gates, as these are the ones used for encoding the Hamiltonian matrices in the VQE algorithm.

2.2.2 The Hadamard Gate

The Hadamard gate is a single-qubit gate, and is defined as follows:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

The Hadamard gate is used to create superposition, and is also used to perform a change of basis.

2.2.3 Phase Gate

The phase gate is also a single-qubit gate, and is defined as follows:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

2.2.4 The CNOT Gate

The CNOT gate is a two-qubit gate, and is defined as follows:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

The CNOT gate is used to create entanglement, and is also used to perform conditional operations. The CNOT gate is also known as the controlled-X gate, and the need to create entanglement arises when we are working in a system where the ground state may be an entangled state. Then we need to be able to create entanglement in our trial wavefunction, otherwise we will not be able to properly converge towards the ground state.

2.3 Computational basis

The computational basis is the basis that we can use to represent the state of a quantum system. In a quantum computer, the computational basis is the set of all possible states of a qubit. The qubits in our system can be in a superposition of states, and we represent this superposition using a lineare combination of the computational basis states. For a single qubit system, the computational basis is as follows:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

This can easily be extended to a multi-qubit system by taking the tensor product of the single-qubit basis states. E.g for a 2 qubit system,

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$
$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

2.3.1 Bell basis

Another very useful basis is the Bell basis, which is a basis that is used to represent entangled states. The Bell basis is defined as follows:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\big(\,|00\rangle + |11\rangle\,\big), \quad |\Phi^-\rangle = \frac{1}{\sqrt{2}}\big(\,|00\rangle - |11\rangle\,\big),$$
$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}\big(\,|01\rangle + |10\rangle\,\big), \quad |\Psi^-\rangle = \frac{1}{\sqrt{2}}\big(\,|01\rangle - |10\rangle\,\big).$$

2.4 Variational Quantum Eigensolver

Quantum algorithms are algorithms that are specifically designed to be run on a quantum computer, and they are constructed by manipulating a quantum system using quantum gates. The algorithm that we are going to study in this report is the VQE (Variational Quantum Eigensolver) algorithm. This is an algorithm that is used to find an estimate to the ground state energy of a given Hamiltonian, and it is based on the variational principle.

2.4.1 The Variational Principle

The variational principle states that for any given Hamiltonian, the expectation value of the Hamiltonian in any state is always greater than or equal to the ground state energy of the Hamiltonian. This means that if we have a trial wavefunction, $|\Psi\rangle$, we can use this to estimate the ground state energy of the Hamiltonian.

$$E_0 \leq \frac{\langle\psi|\, H\, |\psi\rangle}{\langle\psi|\psi\rangle}.$$

As we can see, if we work to minimize the expectation value of the Hamiltonian, we will always find a "better" estimate to the ground state energy - while never going below it.

2.4.2 Pauli decomposition of the Hamiltonian

In order to use the VQE algorithm (or any quantum algorithm to be run on a quantum computer), we need to express the Hamiltonian in terms of quantum gates. This is done by expressing the Hamiltonian as a linear combination of Pauli operators, what we call Pauli strings.
To find this decomposition we can use the following formula (for a n-qubit system):

$$H^{2^n \times 2^n} = \sum_{(\xi_1,\xi_2,\ldots,\xi_n)\in\{I,X,Y,Z\}^n} \frac{1}{2^n}\mathrm{Tr}\big[(\bigotimes_{i=1}^{n}\sigma_{\xi_i}^i)\cdot M\big]\cdot\bigotimes_{i=1}^{n}\sigma_{\xi_i}^i \tag{1}$$

where the tensorproduct is all possible combinations of the pauli matrices, and the first term (the trace part) yields the coefficient infront of the matrix (latter part). With this, we can easily encode our Hamiltonian using the Pauli gates introduced earlier in the section.

2.4.3 The VQE algorithm

The VQE algorithm is a hybrid quantum-classical algorithm, and is as mentioned based on the variational principle. The algorithm is based on the following steps:

1. Choose a trial wavefunction, $|\Psi(\theta,\phi)\rangle$, that depends on some parameters, θ and ϕ, which are rotation angles for the qubit(s).
2. Calculate the expectation value of the Hamiltonian, H, in the trial wavefunction, $|\Psi(\theta,\phi)\rangle$.
3. Use a classical optimization algorithm to minimize the expectation value of the Hamiltonian, and find the optimal parameters, $\theta_{\text{opt}}, \phi_{\text{opt}}$.
4. Use the optimal parameters, θ_{opt}, ϕ_{opt}, to calculate the ground state energy of the Hamiltonian.

The performance of the VQE algorithm strongly depends on the choice of the initial trial wavefunction, if poorly initialized, one could get "stuck" in a local minima. Another important aspect is the choice of optimization algorithm, and choice of parameters of said algorithm.

2.4.4 Wavefunction ansatz

when we want to solve the various Hamiltonians, it is important that we make a good wavefunction ansatz. In this ansatz, it is of the essence that we manage to properly capture the entanglement (or other physical properties) in the system - otherwise, our VQE algorithm will not actually converge to a good ground state estimate. In the methodsection, we will present the ansatzses we've used for the different hamiltonians.

2.5 Entanglement

A very important topic for quantum computing is the concept of entanglement. This is a property of quantum systems that do not occur in classical systems, and is widely used when we perform quantum computing. Many quantum computing algorithms would not be possible without entanglement. In our VQE algorithm, we will study Hamiltonian systems with varying interaction strengths, which will lead to entanglement in our system.
The concept of entanglement (in a quantum computer) is the fact that two (or more) qubits become correlated in such a way that the state of one qubit is directly dependent on the state of the other qubit(s). This is a property we can use, since we then need only perform a measurement on one qubit to immediately know the state of the other qubit(s)!
We will later explain how we should capture the entanglement in the systems to be studied using our wavefunction ansatz. If we'd like to make a measurement of "how" entangled a system is, one can use the Von Neumann entanglement entropy, which can be used to quantify the entanglement in a quantum system.
The von Neumann entropy is defined as:

$$S(\rho) = -\mathrm{Tr}(\rho\log(\rho)), \tag{2}$$

where ρ is the density matrix of the system. The density matrix can be found as follows:

$$\rho = |\Psi\rangle\langle\Psi|, \tag{3}$$

where $|\Psi\rangle$ is the wavefunction of the system.
If the system is in a pure state, the von Neumann entropy is zero, and if the system is composed to two subsystems, A and B, the von Neumann entropy is maximized when the two subsystems are maximally entangled. The system can then be in a pure state, but the individual subsystems may be in a mixed state. Then we use the Von Neumann entropy of one of the subsystems as a measurement of the entanglement in our composite system as such

$$S(\rho_A) = -\mathrm{Tr}(\rho_A \log(\rho_A)), \tag{4}$$

where ρ_A is the reduced density matrix of subsystem A.

2.6 Optimization: Gradient Descent & Scipy.minimize

The optimization algorithms that we will be using in our numerical implementation of the VQE algorithm is the gradient descent algorithm, and the minimization module in the SciPy library[1]. Finding the optimal parameters, and thus the ground state energy, is a difficult task, and most gradient descent algorithms struggle with finding the global minimum for the VQE algorithm. The presence of "barren plateaus", regions in parameter space where the gradients are exponentially small, pose a big challenge for most gradient methods. However, this is not a major problem for the simple Hamiltonians we will encounter in this report - and thus we can make do with the simple gradient descent algorithm.
The gradient descent algorithm is as follows:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \nabla_\theta E(\theta_{\text{old}}), \tag{5}$$

I.e we calculate the gradient of the parameter(s) we are using to minimize the expectation value of the Hamiltonian, and then update the parameters by taking a step in the direction of the negative gradient.

2.7 Hamiltonians

As mentioned in the introduction, there are two different Hamiltonians that we will be working with in this report. The first basic Hamiltonian is a very simple Pauli string, which we will investigate for both one - and two qubits. The Hamiltonian is as follows:

$$\begin{aligned} H_0 &= \xi \mathbf{I} + \Omega \sigma_z \\ H_I &= c\mathbf{I} + \omega_z \sigma_z + \omega_x \sigma_x, \end{aligned}$$

$$H_{\text{one qubit}} = H_0 + \lambda H_I \tag{6}$$

where $\mathbf{I}$ is the identity matrix, and σ_z and σ_x are the Pauli-Z and Pauli-X matrices, respectively.
Similarly, for the two-qubit system

$$\begin{aligned} H_0 &= \epsilon_i \delta_{ijk} \\ H_I &= H_x \sigma_x \otimes \sigma_x + H_z \sigma_z \otimes \sigma_z, \end{aligned}$$

$$H_{\text{two qubit}} = H_0 + \lambda H_I \tag{7}$$

where δ_{ijk} is the Kronecker delta, ϵ_i are the corresponding diagonal elements of the non-interaction Hamiltonian.

2.8 Lipkin model

The final system, or Hamiltonian, that we are to study in this report is the Lipkin model. We will follow in particular the article in [2], where the authors study the Lipkin model using the VQE algorithm. The Lipkin model is a simplified quantum system that describes interaction between particles, in our case

[1] docs:`https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html#scipy.optimize.minimize`

fermions. Due to the interaction, the system shows entanglement and phase transitions, and is thus a good model to study using the VQE algorithm.
The Hamiltonian for the Lipkin model is as follows, written in the Pauli basis[2], using $W =$, i.e only one interaction term, shown for the $J = 1, N = 2$ case:

$$H_L^{(2)} = \frac{1}{2}(Z_0 + Z_1) - \frac{V}{2}(X_0 \otimes X_1 + Y_0 \otimes Y_1), \tag{8}$$

and for the $J = 2, N = 4$ case:

$$H_L^{(4)} = -(Z_0 + Z_1) - \frac{\sqrt{6V}}{2}(X_0 + X_1 + Z_1X_0 - X_1Z_0), \tag{9}$$

where the subscripts indicate on to which qubit the operator acts, and V is the interaction strength.

[2]For the re-write of the Hamiltonian, see the article [2]

3 Methods & Numerical implementation

3.1 Computational basis

In order to properly build our quantum ciruit, we need to initialize the computational basis as shown in the theory section. This is easily implemented using the arrays in the NumPy library, and we can build any n-qubit computational basis by using the kronecker product on a single qubit basis - much like we do when we write out such multi-qubit states analytically.
The following code snippet shows the computatinonal basis for a two-qubit system, as presented in the theory section:

```
    import  numpy as np
    def init_basis():
        q0 = np.array([1, 0])
        q1 = np.array([0, 1])

        return q0, q1

    def init_2basis():
        q0, q1 = init_basis()
        q00 = np.kron(q0, q0)
        q01 = np.kron(q0, q1)
        q10 = np.kron(q1, q0)
        q11 = np.kron(q1, q1)

        return q00, q01, q10, q11
```

This illustrates the basis concept of building the computational basis needed to carry out the quantum computations that we are to implement in the VQE algorithm.

3.2 Measurements

The way we implement the "randomness" of a quantum measurement, and how to properly find the expectation value of a quantum operator, is by use of the random module in NumPy. As an example, this is how one can implement a measurement of the expectation value of the Z operator on a single qubit (prepared in the state $|0\rangle$):

```
import numpy as np

wf = np.array([1, 0])
prob_wf = np.abs(wf)**2
n_measurements = 1000
expectation_value = 0

for i in range(n_measurements):
    measurement = np.random.choice([0, 1], p=prob_wf)
    if measurement == 0:
        expectation_value += 1
    else:
        expectation_value -= 1

expectation_value /= n_measurements
```

This is a simple example, but it shows how we can implement a measurement of the expectation value of the Pauli-Z operator on a single qubit. This would also work for a qubit prepared in a superposition state. How to implement this for more "advanced" measurements can be found in the source code available on the GitHub repository, but we will closely the procedure presented in the lecture notes in FYS5419, see [1].

3.3 Quantum Gates

When we want to apply the quantum gates to our qubits, we can do this by matrix multiplication. This is also easily done for multi-qubit systems, by use of the kronecker product in the NumPy library. As an example, this is how one can implement the Hadamard gate on a two-qubit system (prepared in the state $|00\rangle$) in Python:

```
import numpy as np

def Hadamard():
    H = 1/np.sqrt(2)*np.array([[1, 1], [1, -1]])
    return H

def apply_gate(gate, qubits):
    H = np.kron(Hadamard(), Hadamard())
    qbit = np.array([1,0])
    2qbit = np.kron(qbit, qbit)
    new_state = H @ 2qbit

    return new_state
```

Here we show how easily one can extend the single qubit operators to multi-qubit systems, by usage of the kronecker product (tensor product). And this generalizes to any number of qubits.

3.4 Hamiltonians

When we want to implement a Hamiltonian, we do this using the Pauli basis, as explained in the theory section. The following code snippet illustrates how one can implement the two-qubit Hamiltonian shown in the theory section:

```
import numpy as np

def Hamiltonian():
    epsilon_list = [eps1, eps2, eps3, eps4]
    Hx = 1.0; Hz = 1.0;
    X = pauli_x()
    Z = pauli_z()

    H0 = np.diag(epsilon_list)
    x_term = Hx * np.kron(X, X)
    z_term = Hz * np.kron(Z, Z)
    H = H0 + x_term + z_term

    return H
```

This is the basic concept we will use to implement all the Hamiltonians needed in our report. We will start by building our code for the simple one-qubit system, before going to the two-qubit system. The full implementation will then be extended to the Lipkin model Hamiltonian.

3.5 Variational Quantum Eigensolver

The full VQE implementations can be found in the source code available on the GitHub repository. Here we will present the algorithm as pseudo code, to give a brief overview of the numerical implementation of the VQE algorithm:

Algorithm 1 VQE algorithm

Initialize θ randomly
Initialize the quantum circuit
Initialize the Hamiltonian
while not converged **do**
    → Apply the quantum circuit to the initial state
    → Measure the expectation value of the Hamiltonian
    → Find the gradients of the paramters w.r.t. the expectation value
    → Update the parameters θ
end while

These are the algorithmic steps we've followed when developing our code for the VQE, as can readily be seen on GitHub.

3.6 Entanglement

As explained in the theory section, entanglement is a very important concept in our quantum system. When we are to make a measurement of the entanglement, we will make use of the Von Neumann entropy (see section (2.5)), and this can be done numerically by use of the NumPy library as shown in the following code snippet:

```
import numpy as np
H = some_hamiltonian()
N = 2
eigval, eigvecs = np.linalg.eigh(H)

def density_matrix(eigenvector):
    return np.outer(eigenvector.T.conj(), eigenvector.T)

den_matrix = density_matrix(eigvecs[0])
den_b = np.trace(den_matrix.reshape(N,N,N,N), axis1=0, axis2=2)
prob_b = np.linalg.eigvalsh(den_a)
von_neumann_entropy = -np.sum(prob_a * np.log2(prob_a))
```

This is a simple example of how one can calculate the Von Neumann entropy of subsystem B for a 2-qubit system given some Hamiltonian that can be diagonalized. The procedure can also be generalized to any number of qubits, but in this report, we will only consider the 2-qubit system. The reason we re-shape the matrix is so that we can properly "trace out" the correct axises that correspond to subsystem A (or B).

3.7 Wavefunction ansatz

As explained in the theory section, it is important that we properly construct "good" wavefunction ansatzses for our quantum circuits in the VQE algorithm. These ansatz should be properly entangled, so that they capture the physical properties of the system. The way one would construct a simple, entangled wavefunction ansatz with a variational parameter θ is shown in the following code snippet

```
def ansatz(theta):
    CNOT = np.array([[1,0,0,0], [0,1,0,0], [0,0,0,1], [0,0,1,0]])
    Y = np.array([[0, -1j], [1j, 0]])
    R_y = np.cos(theta/2) * np.eye(2) - 1j * np.sin(theta/2) * Y
    q00 = np.array([1,0,0,0])

    return CNOT @ R_y @ q00
```

Here we employ first the rotation about the Y-axis (on the bloch sphere), by usage of the Y-rotation gate, which is built from the pauli Y-gate σ_y and the identity, $R_y = \cos(\theta/2)\mathbf{1} - i\sin(\theta/2)\sigma_y$, before we add entanglement to our state by the controlled NOT gate (CNOT), with the first qubit being the control bit, second qubit the target bit.
This simple ansatz can be used in a system with some entanglement, but it is still a very simple ansatz - and more complex systems will require more complex ansatzses, as we shall see in later sections.

4 Results and discussion

4.0.1 Making measurements

Before we move on to construct the VQE algorithm, we firstly test our implementation of the computational basis as well as the gates we need (and some more). As we've previously mentioned, an important task for us is to make measurements of expectation values. By following the methodology in the theory section for making measurements and building a quantum circuit, we construct the ϕ_+ Bell state and make spin Z measurements of the two qubits.
Doing this for 1000 shots, yields the following histogram (of what states have been measured)

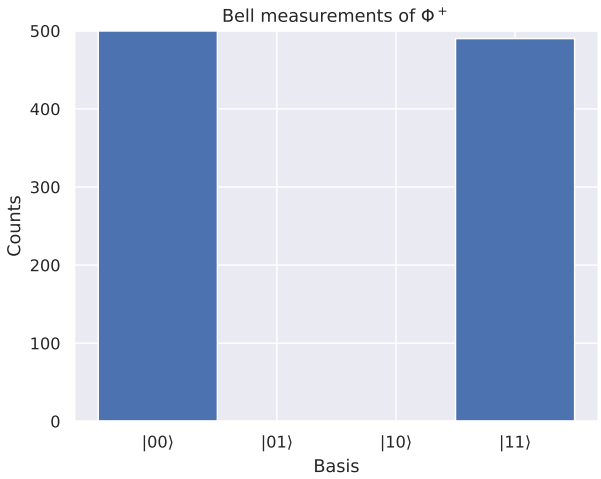

Figure 1: Histogram of the measured states of the ϕ_+ Bell state.

The histogram shows that the states $|00\rangle, |11\rangle$ are measured with (almost) equal occurance, which is expected since the ϕ_+ state is a superposition of $|00\rangle$ and $|11\rangle$. If we've done this for more shots, the histogram would have been more even. This is a good indication that our implementation of the computational basis and the gates are correct.
Now we can apply the Hadamard gate to the first qubit and make measurements of the two qubits. This will yield the following histogram in figure (2):

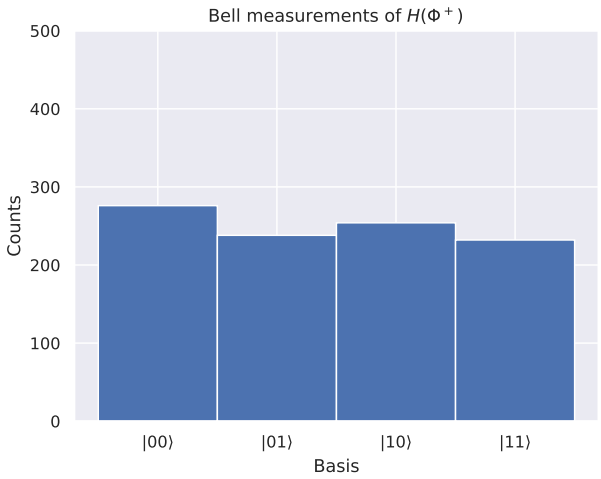

Figure 2: Histogram of the measured states of the ϕ_+ Bell state after applying the Hadamard gate to the first qubit.

Here we see that the state is an (almost) equal superposition of the basis states, which is to be expected when we apply the Hadamard gate. Again, with more shots, we'd achieve a more even distribution of the states - but this figure shows that our Hadamard gate implementation works as intended by mixing the states. It would be interesting to compare our own qubit measurement code with the framework provided by QisKit, but as of March 22nd, there seems to be some issues with the code we've previously written (as QisKit was recently updated), so this comparison will be done at a later time.

4.0.2 Simple one-qubit system

As stated in the previous sections, we build our code from the bottom by first looking at a simplified, one-qubit hamiltonian system. Implementing, and solving by diagonalization, the Hamiltonian for a single qubit system (6) gives us the following energy spectrum (3) for the ground state and the first excited state for increasing interaction strength λ:

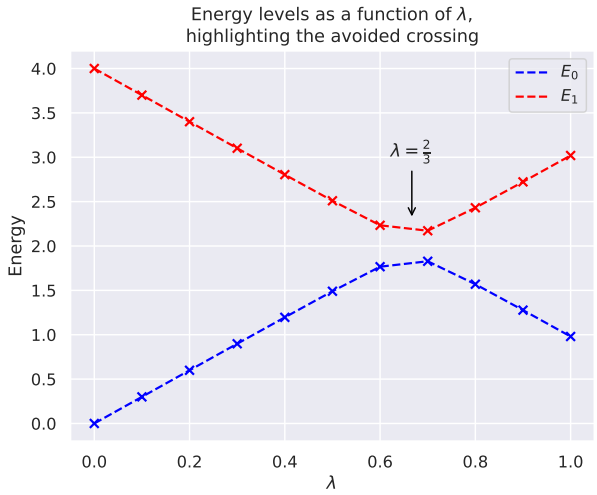

Figure 3: Energy spectrum for the ground state and the first excited state for increasing interaction strength λ.

Here we clearly see the avoided crossing occur for $\lambda = \frac{2}{3}$, where the ground state and the first excited state "repel" eachother due to the interaction term in the Hamiltonian.
This is an important feature in our system, which we will need to take into account when we now want to solve for the ground state using a VQE algorithm.

Implementing this one-qubit Hamiltonian in our VQE algorithm, using GD optimization, with the Pauli basis encoding presented in the theory and method sections, we can compute an estimate for the ground state energy. This energy estimate is shown in the following figure (3) for increasing interaction strength λ:

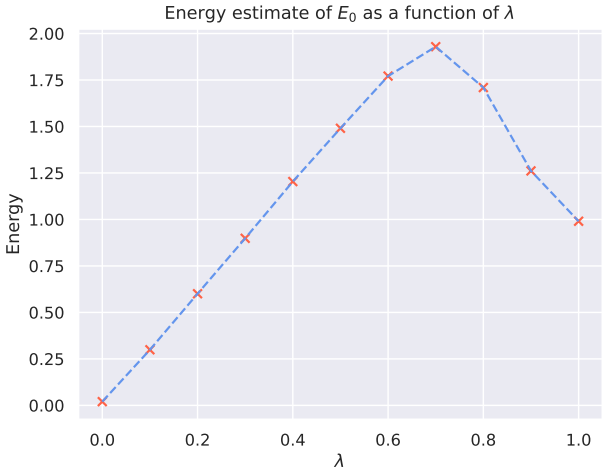

Figure 4: Energy estimate for the ground state for increasing interaction strength λ computed with the VQE algorithm.

In this plot we can clearly see again the avoided crossing, where the ground state energy estimate "turns around" and starts to decrease, after a steady increase up until $\lambda = \frac{2}{3}$, and our VQE algorithm seems to properly capture the entanglement in the system. This should indicate the we've made a good choice for our ansatz, and that our optimization algorithm is indeed working as intended.

4.0.3 Two-qubit system

Now we are ready to extend our program with another qubit, and implement the two-qubit Hamiltonian (7). Again we diagonalize the Hamiltonian to obtain the energy eigenvalues, presented in the following figure (5):

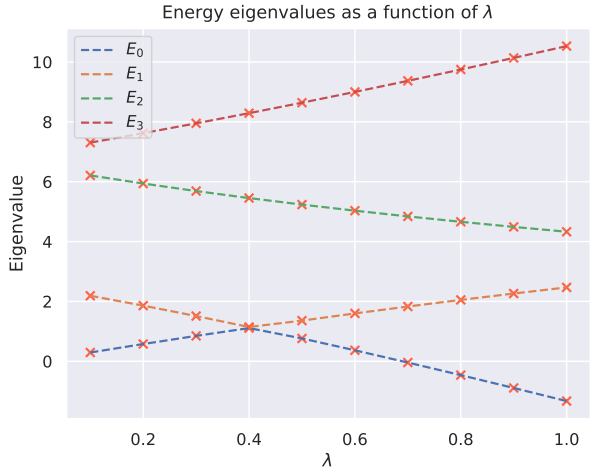

Figure 5: Energy spectrum for the 4 lowest energy states for increasing interaction strength λ in a two-qubit system.

Here we can again see an avoided crossing occur, but at $\lambda = 0.4$. This implies some form of entanglement in the energy eigenstates of our system when the interaction becomes non-neglible.
We can make a study of this entanglement by computing the Von Neumann entropy for one of the subsystems, and the results are presented in the following figure (6):
In figure (6) we can see a clear spike in entanglement entropy when we reach the avoided crossing, as expected.

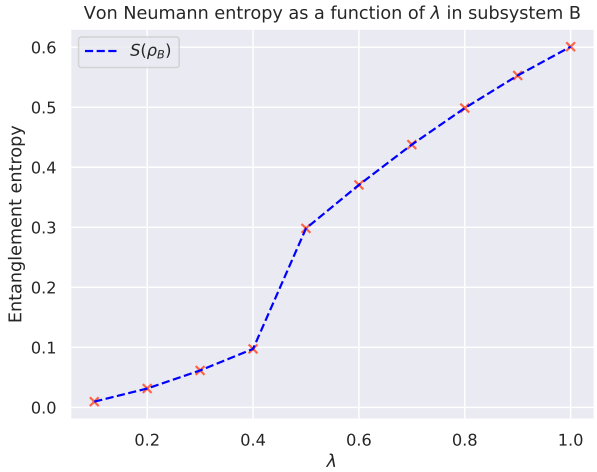

Figure 6: Von Neumann entropy for the two-qubit system for increasing interaction strength λ.

As before, we are also interested in computing the eigenvalues using the VQE algorithm - and the results are presented in the following figure (7):

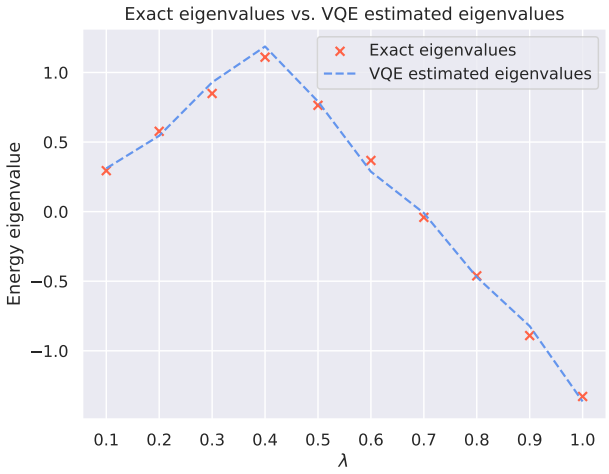

Figure 7: Energy estimate for the ground state for increasing interaction strength λ computed with the VQE algorithm against the exact values.

Here we've plotted the VQE estimates against the exact values found by diagonalization to illustrate an interesting point on the VQE algorithm. As it is a variational method, one would expect to never "undershoot" the ground state energy - however, due to the nature of the energy measurements (presented previously), the states are not measured exactly. This is a source of error in the VQE algorithm, and we can see that the VQE estimates are not always above the exact values.

4.0.4 Lipkin model

Finally, we are ready to implement the Lipkin model Hamiltonian. For the various encoding schemes we will reference multiple times to the article in [2] where the Lipkin model is studied in detail. We do this for $J = 1$ and $J = 2$, i.e 2 and 4 particle systems respectively.

We follow the methodology presented in the theory section and method section to again express the Lipkin hamiltonian (8) and (9) in the Pauli basis which we can easily implement numerically. The results from diagonalization, for rising interaction paramter V is presented in figure (8): We see here a

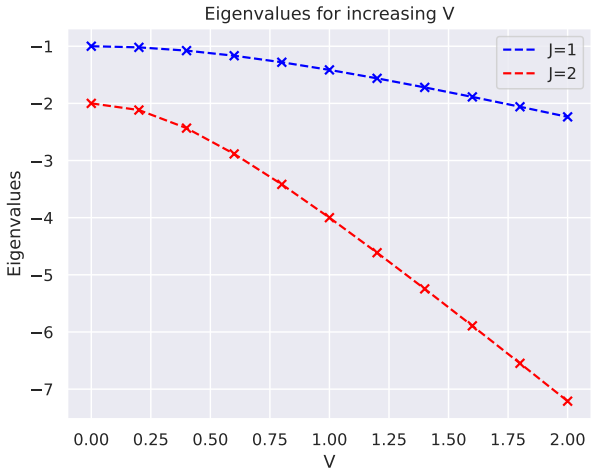

Figure 8: Ground state energy for increasing interaction strength V in the Lipkin model, for $J = 1$ and $J = 2$.

clear decrease in the energy for the ground state as the interaction strength increases, which is expected for the Lipkin model due to the nature of the Hamiltonian.

Next we will make an estimate of the ground state energy using the VQE algorithm, for both the $J = 1$ and $J = 2$ systems. We will first study the simple $J = 1$ system, where with clever manipulation (as seen in the article), we can, for $N = 2$ use a single-qubit basis system (when we are only interested in the ground state that is). As seen in the appendix (A4) the Hamiltonian becomes a 4x4 matrix (see eq. (46) in the article), and we have a very simple ansatz for the wavefunction in this case (see fig. 12 in appendix A in the article) - and the results are presented in the following figure

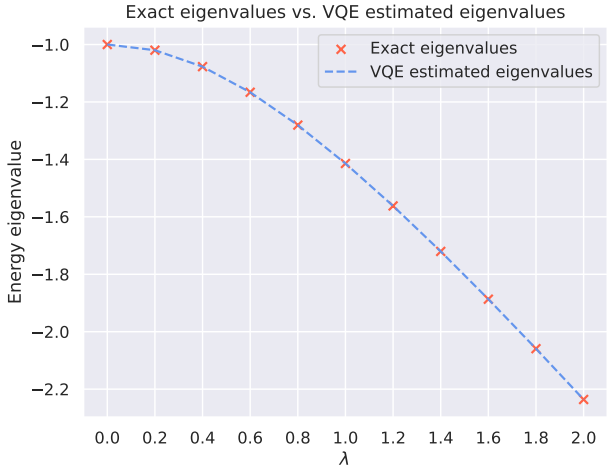

Figure 9: Ground state energy estimate for increasing interaction strength V in the Lipkin model, for $J = 1$ computed with the VQE algorithm.

We can see in that our VQE algorithm, with a single qubit, manages to capture the exact ground state energy extremely well - even for very high levels on interaction. This is a remarkable result, that with a clever encoding we (with good help from our friends in the Physics Review article) managed to reduce the $N = 2$ Lipkin model to a very simple Hamiltonian, and with a simple ansastz we find a (near) perfect estimate for the ground state energy.

Moving on, we'd like to to study the $J = 2$ system , with $N = 4$ particles. Here we will also use a clever encoding presented to us by our friends in the Physics Review, where the 4 particle system can be reduced to a 2-qubit system, with $W = 0$, since there will effectively only be 3 interacting states that contributes in the ground state. The pauli expansion of the reduced Hamiltonian can be seen in eq. (62) in the article, where the exact eigenstates are also presented.
The ansatz we will need to use for the VQE algorithm for the $J = 2$ system can be found in figure 1 in the article, and with this ansatz, and the methodology presented in the theory section, we can compute the ground state of the system for increasing interaction strength V. The results are presented in the following figure:

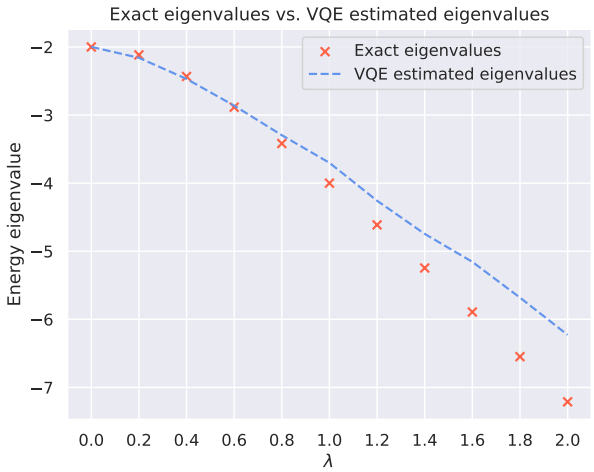

Figure 10: Ground state energy estimate for increasing interaction strength V in the Lipkin model, for $J = 2$ computed with the VQE algorithm.

In this plot we can clearly see that our VQE algorithm mimics the ground state energy, and makes good estimates for low levels of interaction. It seems to struggle more and more as we turn the interaction parameter up, and there may be many reasons for this. One major issue could lie in how we make the energy measurements, since we make a finite amount of shots, the expectation values used to compute the Hamiltonian will not be exact. Furthermore, as can be seen from the article in Physical Review C volume 106, the encoding of the $J = 2$ Hamiltonian with the ansatz we've used deviates from the true ground state energy for higher interaction strengths.
This means, that our (simplified) VQE algorithm captures much of the same properties in the system as the actual quantum computer that was used in the article, but it is not perfect, nor did we expect it to be.

5 Conclusion

In this report, we've explored the basic concepts behind quantum computing - starting from the simplest case of building a code that can make quantum measurements (i.e expectation values of various gates) and the implementation of the quantum gates and quantum qubits, which serves as the building blocks for a quantum computing algorithm. As seen in the results sections, we were succesful in performing Bell measurements (see figure 1) on our system, and also successfully performed the Hadamard gate on this Bellstate, resulting in the measurement presented in figure (2). With this in mind, we can conclude that our initial, simplest quantum implementation worked as intended.
We then moved our focus on implementing the Variational Quantum Eigensolver (VQE) algorithm, and used this to compute the ground state energies of firstly, two simple Hamiltonian, both for single- and two-qubit system. We saw, as presented in figure (4), that our VQE algorithm managed to capture the complex nature of the Hamiltonian, namely the entanglement, and gave us a good estimate of the ground state energy. Since it should "turn around" at the avoided crossing we see in the figure (3), we can be certain that our VQE implementation for the simplest system was a success.

This was also the case for the two-qubit system, where we saw that the VQE algorithm managed to capture the entanglement between the qubits, and gave us a good estimate of the ground state energy.

In the study of the Lipkin model, using the encoding schemes presented in the article, and the code we've developed through the report, the VQE algorithm for the single-qubit system was able to compute the ground state energies extremely well (see figure (8)). For the more complex two-qubit system we see that, even though the VQE algorithm captures somewhat the entanglement properties of the system, it does fail to accurately compute the ground state energy for the highest levels of interaction. This shows room for improvement in our otherwise successful VQE implementation, and is a good starting point for further work.

In conclusion, we have successfully implemented a simple quantum computing algorithm, the VQE algorithm, and used this to compute the ground state energies of two simple Hamiltonians, and the Lipkin model.

References

[1] Morten Hjorth-Jensen. *FYS5419 Lecture notes*. 2024. URL: https://github.com/CompPhysics/QuantumComputingMachineLearning/tree/gh-pages/doc/src/.

[2] Manqoba Q. Hlatshwayo et al. "Simulating excited states of the Lipkin model on a quantum computer". In: *Phys. Rev. C* 106 (2 Aug. 2022), p. 024319. DOI: 10.1103/PhysRevC.106.024319. URL: https://link.aps.org/doi/10.1103/PhysRevC.106.024319.

[3] Robert Hundt. *Quantum Computing for Programmers*. Cambridge University Press, 2022.