Analysis of the Lipkin Model with a Variational Quantum Eigensolver Algorithm

Carl Peter Kirkebo[1], Markus Fuhrmann [1]
[1] Department of Physics, University of Oslo, Norway

The topic of this report is the calculation of the eigenvalues of a Lipkin Hamiltonian with a variational quantum eigensolver (VQE). We first revisit basic concepts of quantum mechanics and eigenvalue problems such as state preparation, measurements and entanglement. Introducing the VQE algorithm for a two dimensional Hamiltonian, we compare it to standard eigenvalue solvers provided by NumPy/SymPy. After implementing the VQE algorithm on a two qubit basis we introduce the Lipkin model and different encoding schemes to simulate the Lipkin Hamiltonian on a VQE. The report shows, that the VQE is indeed a viable method of finding eigenvalues through simulation of quantum experiments. Its accuracy however does not seem overwhelming at first glance and offers a lot of optimization potential in the hyperparameter search of the classical optimization problem as well as in the simulation of quantum measurements.

***Index Terms*—VQE, Lipkin model, Quantum Computing.**

I. Introduction

THE basis for this report is a project in a course on quantum computing and quantum machine learning (FYS5419) at the University of Oslo (UiO) spring 2024.

Quantum computing is a field where there is great practical promise but there still remain significant obstacles in making a physical implementation work except for in quite simple cases. The potential of quantum computing has attracted investors with several companies reaching market caps in the billions of USD [1].

The current status of quantum computing means that there is still a strong emphasis on the theoretical aspects and approximation techniques. The hope is that this can lead among others to better understanding and ultimately development of commercially viable quantum computers.

In the article [2] the Lipkin model is presented. This is a model that has been proposed as a toy model which can be used to test approximation techniques for solving the quantum many-body problem. The project follows closely the lastly mentioned article.

The Lipkin model is rich enough to offer problems that are interesting to study, e.g. nontrivial many-body interactions. At the same time it has several symmetries which in many cases can reduce the dimensionality of the problem whereby one can avoid the common "dimensionality curse" which usually is the case in quantum many-body problems.

The Lipkin model will be studied by solving for the eigenvalues of the Hamiltonian by applying a variational quantum eigensolver (VQE) algorithm. We will first get familiar with the topic by looking at a the simplest case with a system with a one-qubit basis and the effect of applying various gates on these states. Thereafter we will perform measurements both using our own code and comparing this with Qiskit. Continuing on the simplest case, we will study a Hamiltonian $H = H_0 + H_I$ given by a 2×2 matrix. Here H_0 is a diagonal matrix and H_I describes the interactions. We will find an exact solution of this eigenvalue problem and compare the solutions with an implementation of a VQE.

Building on the above initial investigations, we will extend to a two-qubit system. This increases the dimensionality and hence the complexity although the underlying techniques will be similar: By setting up the the circuits needed, the VQE method can be applied to find the eigenvalue corresponding to the lowest state.

Developing on the experience from the above, we will finally introduce the Lipkin Hamiltonian. Following closely the approach in [2], we will solve the eigenvalue problem for $j = 1, 2$ on a VQE.

In addition to [2], this paper will also draw heavily on the lecture notes of the course FYS5419 and the books [3] and [4]. If not indicated otherwise, code has been written in Python. In general only excerpts or descriptions of the code are given in this text. For the full code we refer to the Github repository [5].

II. Preliminary considerations

This part will serve to get familiar with the basic concepts of the theory and also the rudimentaries of the Python package Qiskit [6], an open-source software development kit (SDK) for working with quantum computers at the level of circuits, pulses, and algorithms [7].

We will among others study the effect of applying Pauli matrices and various gates to different states among them one of the Bell states of our choosing. The chosen state was

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B\right) = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & 0 & \frac{\sqrt{2}}{2} \end{bmatrix}^T .$$

Applying a Hadamard gate and thereafter a CNOT on Φ^+ gives $\begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}^T$.

The effect of implementing a Hadamard gate and thereafter a CNOT followed by a measurement has been determined both with and without using Qiskit. Using 10000 shots in the former case, the obtained count was $\begin{bmatrix} 2547 & 2529 & 2511 & 2413 \end{bmatrix}$. This shows (numerically) that the probability of measuring the first qubit in state $|0\rangle$ is $1/2$ and in state $|1\rangle$ is also $1/2$ and that the final state is an equal superposition of all basis states. An exercpt of the code is shown in listing 1 in the appendix.

We have also implemented the same example in Qiskit. To be precise, the effect of implementing a Hadamard gate and thereafter a CNOT on all four Bell states has been calculated. The results are shown below in figure 1.

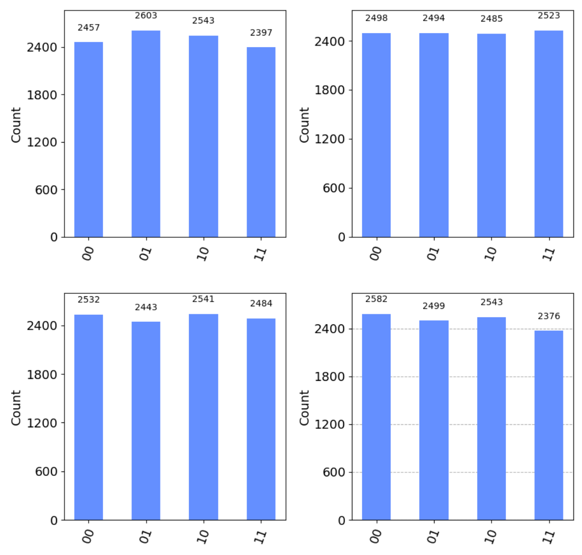

Figure 1: Hadamard gate and thereafter a CNOT on all Bell states using Qiskit

Overall the same results are being produced. In theory one should for both examples perform a statistical hypothesis test to see if the observed results are consistent with each outcome being equally likely (probality $(1/2)^2 = 1/4$), but we have settled for a visual confirmation.

The attention will now turn to studying the eigenvalues of a symmetric 2×2 Hamiltonian $H = H_0 + \lambda H_I$, where $H_0 = \begin{bmatrix} E_1 & 0 \\ 0 & E_2 \end{bmatrix}$ represents the non-interacting solution and $H_I = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}$. The strength parameter λ will vary in $[0, 1]$.

To make the situation simpler, the study will specialize to $E_1 = 0$, $E_2 = 4$, $V_{11} = 3$, $V_{22} = -3$ and $V_{12} = V_{21} = 0.2$. In this case the eigenvalue problem reduces to finding the eigenvalues of

$$H = \begin{bmatrix} 3\lambda & \frac{\lambda}{5} \\ \frac{\lambda}{5} & 4 - 3\lambda \end{bmatrix} \tag{1}$$

Thus, it remains to solve

$$\det(H - xI)$$

for x. This gives the two eigenvalues

$$x_1 = -\frac{\sqrt{2}\sqrt{113\lambda^2 - 150\lambda + 50}}{5} + 2$$
$$x_2 = \frac{\sqrt{2}\sqrt{113\lambda^2 - 150\lambda + 50}}{5} + 2$$

In figure 2 the eigenvalues are plotted as a function of λ. Here $D = \sqrt{226\lambda^2 - 300\lambda + 100}$.

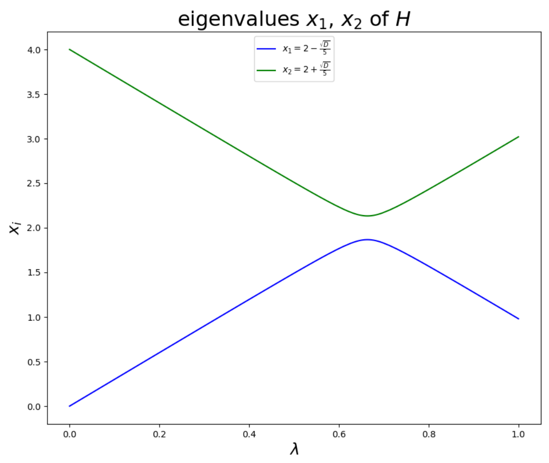

Figure 2: Eigenvalues of the Hamiltonian H as a function of the strength parameter λ

For $\lambda = 1$ the exact eigenvalues

$$x_1 = 2 - \frac{\sqrt{26}}{5} \, (= 0.98020) \tag{2}$$
$$x_2 = \frac{\sqrt{26}}{5} + 2 \, (= 3.01980) \tag{3}$$

are obtained.

To perform exact calculations, SymPy has been used which is a Python library for symbolic mathematics. In listing 2 in the appendix is given some exercepts of the code.

Below are the eigenvectors shown where v_1, v_2 respectively belong to the eigenvalues x_1, x_2 and both eigenvectors are normalized so that the last component is 1.

$$v_1 = \left[\frac{15\lambda - 20}{\lambda} + \frac{5\left(-\frac{\sqrt{2}\sqrt{113\lambda^2-150\lambda+50}}{5} + 2\right)}{\lambda},\ 1\right]$$

$$v_2 = \left[\frac{15\lambda - 20}{\lambda} + \frac{5\left(\frac{\sqrt{2}\sqrt{113\lambda^2-150\lambda+50}}{5} + 2\right)}{\lambda},\ 1\right]$$

Figure 3 shows the behavior of the eigenstates as functions of the interaction strength λ. Note that the domination by one of the basis states changes around the avoided level crossing. The lower eigenstate is dominated by the $|0\rangle$ component for $\lambda < 2/3$ whereas for $\lambda > 2/3$ the dominant basis state is $|1\rangle$. For the upper eigenstate it's the other way around.

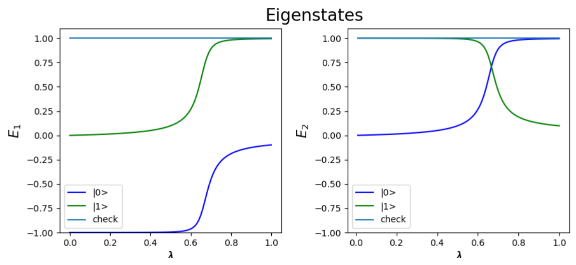

Figure 3: Behavior of the eigenstates as functions of the interaction strength λ

III. Enter Variatinoal Quantum Eigensolver

The variational quantum eigensolver (VQE) is an algorithm for finding the eigenvalues of a Hamiltonian through variational measurement of quantum experiments. The core idea consists of three steps:

1) Define quantum circuits for measuring operators in default basis
2) Prepare random ansatz by rotating the basis state
3) Calculate the energy for that ansatz
4) Iterate and move the rotation angles in the direction of decreasing energy

The code snippets for the following section can be found in ”PartC.ipynb”. Writing H_I as a linear combination of Pauli-matrices,

$$H_I = c\boldsymbol{I} + \omega_z\sigma_z + \omega_x\sigma_x, \tag{4}$$

it then follows from the definitions of H_0 and H_I that $c = (V_{11} + V_{22})/2$, $\omega_z = (V_{11} - V_{22})/2$ and $\omega_x = V_{12} = V_{21}$. The setup of a one-qubit ansatz is illustrated in figure 4.

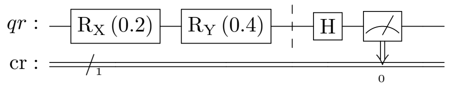

Figure 4: Quantum circuit for measuring the X operator in the z-basis with the VQE.

To try to find the lowest eigenvalue of the Hamiltonian, the chosen first line of attack was using a simple gradient descent method. An exercpt of the code is shown in listing 3 in the appendix.

Figure 5 shows the convergence towards the eigenvalue $x_1 = 0.98020$ given in (2).

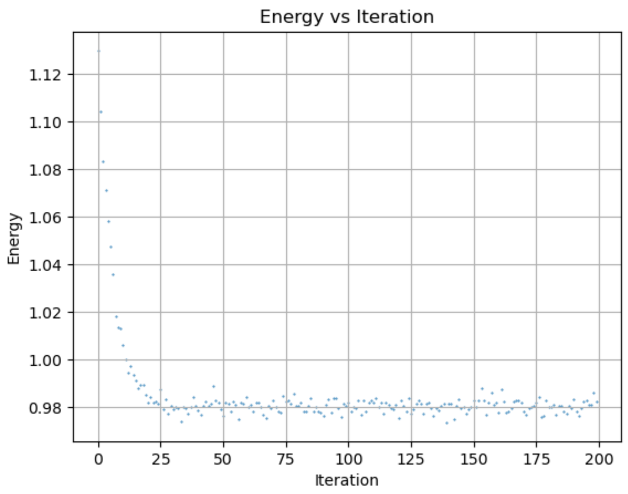

Figure 5: Convergence of VQE using a learning parameter

Regarding that this value differs from that of a NumPy or SymPy eigenvalue solver, the second chosen line of attack utilizes SciPy's built in optimizers. A description of these can be found in [8]. Table I shows the results of trying a selected number of these opimizers.

method	Powell	BFGS	CG	COBYLA	SLSQP
result	0.97508	1.6319	1.62288	0.98104	2.82752
success	True	False	False	True	True

Table I: VQE using predefined optimizer from SciPy

Table I indicate that there are several issues concerning convergence and the usage of the predefined optimizers. Below some comments will be attached to the columns indicated with red.

- BFGS: Broyden-Fletcher-Goldfarb-Shanno algorithm.
  $\Rightarrow$ Table I indicates that the BFGS method in this case does not seem to converge. This is consistent with the output message ”Desired error not necessarily achieved due to precision loss.”
- CG: Minimization of scalar function of one or more variables using the conjugate gradient algorithm.
  $\Rightarrow$ Same issue as for the BFGS method.
- SLSQP: Minimize a scalar function of one or more variables using Sequential Least Squares Programming.
  $\Rightarrow$ The output leads the user to believe that one has found the solution which in fact is not true.

In listing 4 in the appendix is shown an excerpt from the code.

As the code indicates, there are parameters one can adjust. However, the general impression is that one has to show caution and using these opimizers is not a sure fire approach. In particular the ”false” solution from the the SLSQP method causes worry.

For the method Powell the VQE converged to a value lower than (2). This might either be due to weaknesses inherit in the method or the fact that the energy is estimated using a number of ”shots” (in this case n=10000). Thus the energy is actually not calculated exactly but rather estimated as an average. Typically this might converge slowly with an error that decreases with $1/\sqrt{n}$. We actually observed something similar in figure 5 where several of the iterations where below 0.98.

To further understand the challenges connected to the VQE method, the energy surface as a function of θ and ϕ is shown in figure 6.

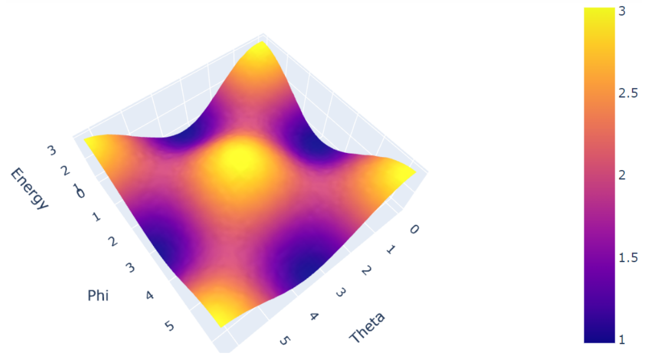

Figure 6: Energy surface as a function of θ and ϕ

Figure 6 shows a symmetrical surface with four parts descending to a minimum near 1. This indicates that it should actually be a relatively easy problem to find a global minimum. One could imagine a surface less symmetrical and with many local minima. In that case it would be much harder to find a global minimum and a search algorithm that can distinguish between local and global minima should be used, for example a grid based search algorithm. This however poses the new problem of exploding computational effort, since the time would increase with l^d where l is the number of values for a single parameter in the grid and d the number of parameters. Seeing that this problem is very complex and would lead us away from the Lipkin model, we therefore waived any further efforts on getting better approximations. A comparison of the eigenvalues for different interaction strengths between the NumPy solver (line) and our VQE algorithm (dots) is in figure 7.

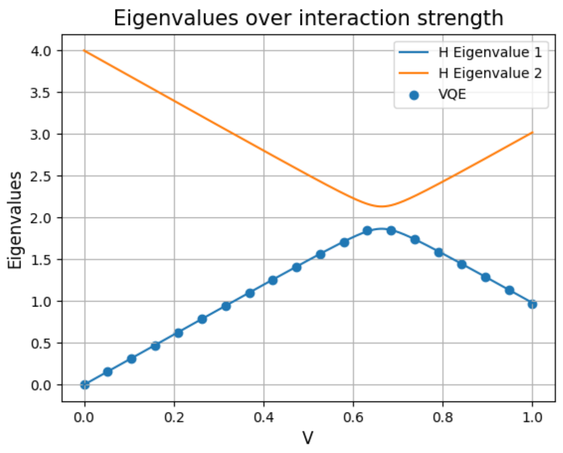

Figure 7: Eigenvalues with the numpy solver (line) and VQE (dots)

IV. Expanding to a Two-Qubit System

After discussing two-dimensional Hamiltonians that can be simulated with a single qubit, the next step is to deal with two qubit systems and study their behaviour and new phenomena arising from the interaction between them. Consider two subsystems A and B with a respective computational basis

$$|0\rangle_{\text{A,B}} = \begin{bmatrix}1 & 0\end{bmatrix}^T \qquad |1\rangle_{\text{A,B}} = \begin{bmatrix}0 & 1\end{bmatrix}^T .$$

The basis states of the combined system are calculated via tensor products and are then written as

$$|00\rangle = |0\rangle_{\text{A}} \otimes |0\rangle_{\text{B}} = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T ,$$

and

$$|01\rangle = |0\rangle_{\text{A}} \otimes |1\rangle_{\text{B}} = \begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}^T ,$$

and

$$|10\rangle = |1\rangle_{\text{A}} \otimes |0\rangle_{\text{B}} = \begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}^T ,$$

and

$$|11\rangle = |1\rangle_{\text{A}} \otimes |1\rangle_{\text{B}} = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T ,$$

Now consider the Hamiltonian $H = H_0 + H_{\text{I}}$, where the interaction Hamiltonian is given by $H_{\text{I}} = H_{\text{x}}\sigma_{\text{x}} \otimes \sigma_{\text{x}} + H_{\text{z}}\sigma_{\text{z}} \otimes \sigma_{\text{z}}$ and the non-interacting hamiltonian is given as $H_0 = \text{diag}(\epsilon_{00}, \epsilon_{01}, \epsilon_{10}, \epsilon_{11})$. The matrix representation of the total Hamiltonian in the new basis reads

$$H = \begin{bmatrix} \epsilon_{00} + H_{\text{z}} & 0 & 0 & H - \text{x} \\ 0 & \epsilon_{01} - H_{\text{z}} & H_{\mathbf{x}} & 0 \\ 0 & H_{\text{x}} & \epsilon_{10} - H_{\text{z}} & 0 \\ H_{\text{x}} & 0 & 0 & \epsilon_{11} + H_{\text{z}} \end{bmatrix} \tag{5}$$

Calculating the eigenvalues of the Hamiltonian in (5) in dependence of the interaction strength λ for $H_{\text{x}} = 2$, $H_{\text{z}} = 3$ and $H_0 = \text{diag}(0, 2.5, 6.5, 7)$ yields the plot in figure 8. The script used for this calculation is part of the notebook ”PartD.ipynb”.

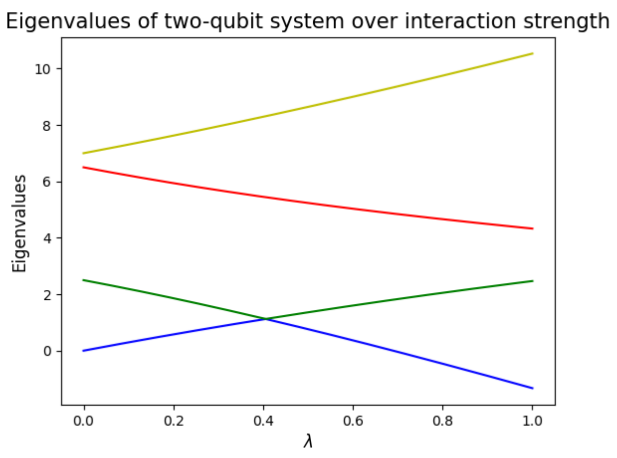

Figure 8: Eigenvalues of two-qubit Hamiltonian over interaction strength λ.

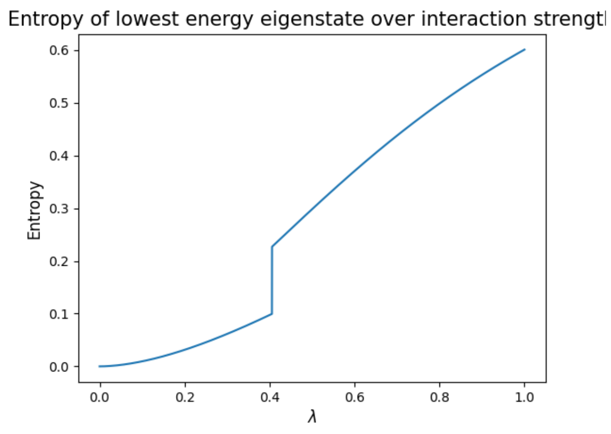

Figure 9: Plot of von Neumann entropy for the lowest energy eigenstate

V. Entanglement and von Neumann Entropy

An interesting phenomenon arises from studying two interacting subsystems. Looking at one of the Bell states

$$\psi_+ = \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right),$$

we see that neither system A nor system B is in a defined state, but they are rather in a superposition of their own possible states. This is nothing new, however the consequences of this specific superposition are drastic. Imagine that one measures system A first and finds that system A is in the state $|0\rangle$. This indicates, that the system collapes to the state $|01\rangle$, since this is the only state in the superposition, where system A is in the state $|0\rangle$. A measurement of system B will, afterwards, always result in B being in state $|1\rangle$. Even if the two systems A and B are spatially divided, the measurement of A changes the total system immediately and therefore also the state in which system B is. The states are said to be quantum mechanically entangled[1]. A measure of how much the two systems are entangled is the von Neumann entropy. It is defined as

$$S_{\mathrm{A,B}} = -\mathrm{Tr}(\rho_{\mathrm{A,B}} \log(\rho_{\mathrm{A,B}})), \tag{6}$$

where the density matrix is given by $\rho_{\mathrm{A,B}} = \rho_{\mathrm{A}} + \rho_{\mathrm{B}} = \mathrm{Tr}_B(\rho_0) + \mathrm{Tr}_A(\rho_0)$ with the density matrix for the lowest energy eigenstate $\rho_0 = \sum_{\mathrm{i}=0}^{3} c_{\mathrm{i}} |i\rangle \langle i|$. A plot of (6) is shown in figure 9. Note that the entropy contains a jump around the interaction strengths where the two lowest levels cross. What has been the lowest energy eigenstate in the left half has become the second lowest in the right half. The general result is, that the entanglement increases with the interaction strength. The jump shows that it is dependent on the Hamiltonian and the non-interacting energies as well.

[1]This phenomenon is sometimes subject of teleportation or above-lightspeed-information transport. Note however, that the measurement of A is random, meaning also the output of the entangled system is random. A proposed loophole for this uses the statistic distribution of measuring system A and multiple cloned states, in order to pass information from A to B. This however is not possible due to the Non-Cloning-Theorem.

VI. Two-Qubit System on a VQE

Now after getting more familiar with two-qubit systems, the VQE algorithm first described in III needs to be changed towards a two qubit basis. Before even thinking about implementing the VQE, an expression of the Hamiltonian in terms of Pauli-matrices and their Kronecker-products needs to be found. A code snippet in "PartE.ipynb" does exactly that. It uses the fact that each hermitian matrix H can be written as

$$H = \frac{1}{N}\sum_{\mathrm{j}} \mathrm{Tr}(\sigma_{\mathrm{J}} H)\sigma_{\mathrm{J}}$$

with

$$\sigma_{\mathrm{J}} = \prod_{\mathrm{k}=1}^{\mathrm{n}} \sigma_{\mathrm{J_k}}^{(\mathrm{k})}$$

where $\sigma_{\mathrm{j}}^{(\mathrm{k})}$ is the Pauli-matrix σ_{j} applied on the k-th qubit. To get the composition for each σ_{J} the product between the matrix H and the Pauli-product σ_{j} can be calculated, in analogy to projecting out specific states from a superposition. The Pauli-decomposition of the matrix (5) becomes[2]

$$H = 4 - 0.75 \cdot Z_1 + 2 \cdot X_1 X_2 - 2.75 \cdot Z_1 + 2.5 \cdot Z_1 Z_2 \tag{7}$$

Now the VQE algorithm can be changed to a two qubit basis. The full algorithm is in "PartE.ipynb". Main changes to III include the setup of a two-qubit ansatz, which includes the additional application of the CNOT-gate to create entangled states as well as the implementation of the respective circuits for measuring Pauli-products in the default base. An example of the quantum circuit for measuring the $X \otimes X$ operator is shown in figure 10. The needed basis transformations can be looked up in [4].

[2]Note that Z_1 means $Z_1 \otimes \mathbb{1}_2$

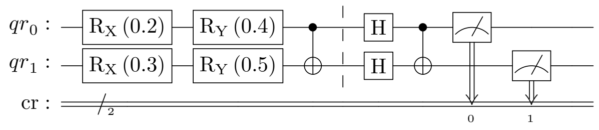

Figure 10: Quantum circuit for measuring the $X \otimes X$ operator in z-basis with the VQE. Contains set up of ansatz, basis change and measurement.

The gradient descent algorithm changes to include two additional angles for rotating the second qubit. This leads to a four dimensional parameter space of the energy, which cannot be visualized in a 3D plot anymore. The approximated energy fits the one calculated with NumPy more or less well; $E_{\text{np}} = -1.3284$, $E_{\text{VQE}} = -1.35$. This offers a lot of space for optimization and different minimum search algorithms. Appart from the learning rate, the number of iterations and the gradient descent method, another important parameter to keep in mind is the number of shots in the simulation of quantum measurements. As mentioned earlier this will not be the focus of this report. More important should be the demonstration of the VQE for two qubit systems. This can be done by comparing the VQE to the NumPy solver for different interaction strengths. A plot of this result can be seen in 11. There are two problems with the VQE for this example. Problem 1 is a constant offset of around 0.5. We don't know where it comes from. The other problem is, that the VQE doesn't take the level crossing into account. This problem was fixed however, since the second part of the VQE, the creation of an ansatz, was still out of the loop after calculating it for different interaction strengths. This meant, that the VQE was stuck in the local minimum of the eigenstate, even after the level crossing. The fixed plot is shown in figure 12. The offset still remains and is interestingly in the opposite direction for the other eigenstate. Also the gradient descent still gets stuck in some local minima, but works better than before.

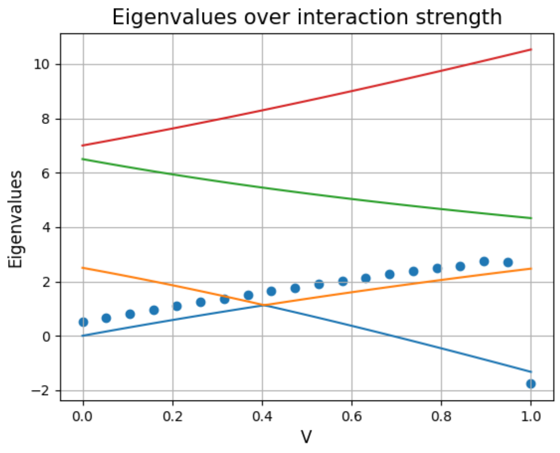

Figure 11: Eigenvalues of the Hamiltonian over interactions strength with a NumPy eigenvalue solver and the VQE. Problematic ansatz creation made the VQE stick to a specific minima.

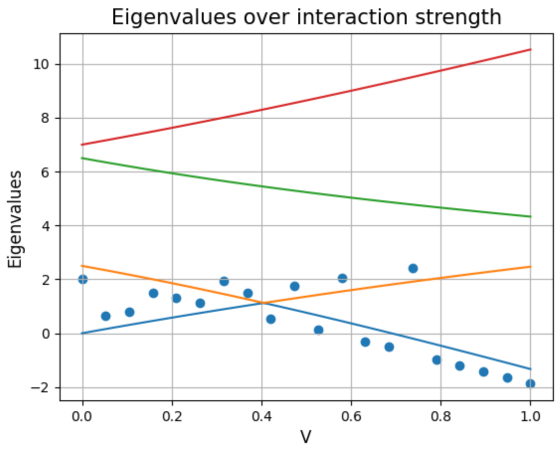

Figure 12: Eigenvalues of the Hamiltonian over interactions strength with a NumPy eigenvalue solver and the VQE. Fixed the ansatz creation.

VII. Enter Lipkin Model

The Lipkin model [9] first proposed by H. J. Lipkin in 1964 is an approach to describe many-body systems. More precisely, it offers a way to find eigenvalues, in some cases analytically, more easily of N fermion systems that interact by monopole-monopole forces. The model Hamiltonian can be expressed through quasi-spin operators, which in turn can be expressed as Pauli-products and therefore can be fed directly into the VQE. In this section we will consider the Lipkin model for 2 and 4 spin1/2 particles and how it can be expressed with Pauli-matrices in order to feed it into the VQE in the next section. Let's start with the Lipkin Hamiltonian in second quantization which is given by

$$H = H_0 + H_1 + H_2 \tag{8}$$

where the the different terms can be rewritten with the quasispin operators as

$$\begin{aligned} H_0 &= \epsilon J_z \\ H_1 &= \frac{1}{2} V (J_+^2 + J_-^2) \\ H_2 &= \frac{1}{2} W (-N + J_+ J_- + J_- J_+) \,. \end{aligned}$$

With the eigenvalue equations for the angular momentum operators $J_z \left| j, m \right\rangle = \hbar m \left| j, m \right\rangle$, $J_+ \left| j, m \right\rangle = \hbar \sqrt{(j-m)(j+m+1)} \left| j, m+1 \right\rangle$ and $J_- \left| j, m \right\rangle = \sqrt{(j+m)(j-m+1)} \left| j, m-1 \right\rangle$, the matrix representation of (8) in the quasi-spin basis can be calculated for the case of $j = 1$ (for two spin $1/2$ particles) by calculating the matrix elements $\left\langle j, m \right| H \left| j, m \right\rangle$ for $m = -1, 0, 1$. For $W = 0$ the first matrix element becomes

$$\left\langle 1, -1 \right| (\epsilon J_z + \frac{1}{2} V (J_+^2 + J_-^2)) \left| 1, -1 \right\rangle = -\hbar \epsilon \,.$$

Here the fact was used that the first matrix element is on the main diagonal, meaning terms with an unequal number of ascending and descending ladder operators lead to orthogonal states. This method of finding exception rules before trying to calculate the matrix elements becomes especially handy for higher dimensional problems. Repeating this for the other matrix elements, the Lipkin-hamiltonian for $j = 1, W = 0$ becomes

$$H_{\mathrm{j=1}} = \begin{pmatrix} -\epsilon & 0 & -V \\ 0 & 0 & 0 \\ -V & 0 & \epsilon \end{pmatrix} \tag{9}$$

In order to simulate this hamiltonian on a quantum computer and calculate the eigenvalues with the VQE, it needs to be expressed with Pauli-matrices. Since the Lipkin-hamiltonian is defined via the total angular momentum, we should first have a short revision of angular momentum addition. The total angular momentum in the z-direction for a 2 particle-system can be written as

$$J_{\mathrm{z}} = J_{\mathrm{z,1}} \otimes \mathbb{1}_2 + \mathbb{1}_1 \otimes J_{\mathrm{z,2}} \,.$$

For the N particle-system this becomes

$$J_{\mathrm{z}} = \sum_{\mathrm{i=1}}^{\mathrm{N}} \bigotimes_{\mathrm{j=1}}^{\mathrm{i-1}} \mathbb{1}_{\mathrm{j}} \otimes J_{\mathrm{z,i}} \otimes \bigotimes_{\mathrm{k=i+1}}^{\mathrm{N}} \mathbb{1}_{\mathrm{k}} \tag{10}$$

[2] shows how to exploit the symmetries of the $W = 0$ case for an eficient encoding scheme. In natural units the quasi-spin operators of each particle become $1/2$ the Pauli-matrices

$$J_{\mathrm{k,i}} = \frac{1}{2}\sigma_{\mathrm{k,i}}$$

with $k \in 1, 2, 3, 4$ for the Pauli-matrices. This means an expression for the Lipkin Hamiltonian via Pauli-matrices can be obtained directly by plugging (10) into the Hamiltonian. The $j = 1$ Hamiltonian then becomes

$$H_{\mathrm{j=1}} = \frac{\epsilon}{2}\left(Z_1 + Z_2\right) - \frac{W+V}{2} X_1 X_2 - \frac{W-V}{2} Y_1 Y_2, \tag{11}$$

where the tensorproducts have been omitted for increased readability. The $N = 4$ Hamiltonian becomes

$$\begin{aligned} H_{\mathrm{j=2}} =& \frac{\epsilon}{2}\left(Z_1 + Z_2 + Z_3 + Z_4\right) + \frac{W-V}{2}\Big(X_1X_2 + \\ & X_1X_3 + X_1X_4 + X_2X_3 + X_3X_4 + X_3X_4\Big) \\ & + \frac{W+V}{2}\Big(Y_1Y_2 + Y_1Y_3 + Y_1Y_4 + Y_2Y_3 \\ & + Y_3Y_4 + Y_3Y_4\Big) \,. \end{aligned}$$

These expressions can be used to calculate the lowest eigenenergy in a two or four-qubit based VQE. However, reconsidering the Hamiltonian in (8) another symmetry can be noticed for $W = 0$. This comes from the fact that the H_1 part only contains ascending and descending operators in the power of two, meaning there is an exception rule that turns all matrix elements in the quasispin basis into 0, if the difference in angular momentum is an uneven integer. The basis can therefore be reduced to every second state. Consider a $N = 2k$ with $k \in \mathbb{N}$ particle system which therefore has a reduced basis dimensionality of $d = j + 1$. The encoding scheme for mapping the quasi-spin states to qubits then becomes:

$$\begin{aligned} |j, -j\rangle &\to |\mathrm{bin}(0)\rangle \\ |j, -j+2\rangle &\to |\mathrm{bin}(1)\rangle \\ &\vdots \\ |j, j-2\rangle &\to |\mathrm{bin}(d-1)\rangle \\ |j, j\rangle &\to |\mathrm{bin}(d)\rangle \end{aligned}$$

where $|\mathrm{bin}(k)\rangle \equiv |q_1, q_2, \ldots, q_{\mathrm{n}}\rangle, k = \sum_{\mathrm{i=1}}^{\mathrm{n}} q_{\mathrm{i}} 2^{\mathrm{n-i}}$ with $q_{\mathrm{i}} \in \{0, 1\}$ according to [2]. This is called the standard binary (SB) encoding. An encoding that is more suitable for calculations on quantum computers is the Gray code (GC), which encodes the states in an order that two adjacent states only differ by a single bit. For example a 4-qubit system would have following encoding schemes

$$\text{SB:} \begin{cases} |2,-2\rangle & \to |00\rangle \\ |2,0\rangle & \to |01\rangle \\ |2,2\rangle & \to |10\rangle \end{cases} \quad \text{GC:} \begin{cases} |2,-2\rangle & \to |00\rangle \\ |2,0\rangle & \to |01\rangle \\ |2,2\rangle & \to |11\rangle \end{cases}$$

The Hamiltonian can then in turn be written as a linear combination of the outer products of the qubit product states according to [2]. The coefficients for the diagonal elements are

$$a_{\mathrm{k}} = \epsilon(2k - j)$$

and for the off-diagonal elements

$$b_{\mathrm{k}} = -\frac{V}{2} \cdot F_+(m = 2k - j)$$

with $F_+ = \sqrt{((j(j+1) - m(m+1)) \cdot (j(j+1) - m(m+1)))}$. Let's apply this encoding scheme to the Hamiltonian for $j = 1$ in (9). The mapping of the quasi state $|1, -1\rangle$ is $|0\rangle$ whereas for $|1, 1\rangle$ it is $|1\rangle$. The coefficients read $a_0 = -\epsilon$, $a_1 = \epsilon$ and $b_0 = b_1 = -V$. The reduced Hamiltonian therefore reads

$$H_{\mathrm{j=1}} = \begin{pmatrix} -\epsilon & -V \\ -V & \epsilon \end{pmatrix} . \tag{12}$$

For the $j = 2$ case the calculational efforts increases slightly and with the Gray encoding the Hamiltonian becomes

$$H_{\mathrm{j=2}} = \begin{pmatrix} -2\epsilon & -\sqrt{6}V & 0 & 0 \\ -\sqrt{6}V & 0 & 0 & -\sqrt{6}V \\ 0 & 0 & 0 & 0 \\ 0 & -\sqrt{6}V & 0 & -2\epsilon \end{pmatrix}^3 \tag{13}$$

Setting $\epsilon = 1$ and feeding (12) and (13) into the Pauli-decomposition, we arrive at the final expressions for the VQE:

$$H_{\mathrm{j=1}} = -V \cdot X - Z \tag{14}$$

and

$$\begin{aligned} H_{\mathrm{j=2}} = -1 - \frac{\sqrt{6}}{2} V (X_2 + X_1 - X_1 Z_2 \\ + Z_1 X_2) - Z_1 Z_2 \,. \end{aligned} \tag{15}$$

VIII. Lipkin model on VQE

Implementing the two Hamiltonians (14) and (15) into the VQE ("PartG.ipynb"), and calculating the lowest eigenvalue for different interaction strengths, we arrive at figures 13 and 14. As expected the increasing interaction strength splits up the degenerate levels.

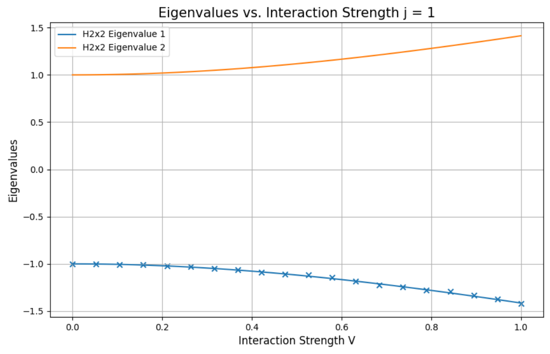

Figure 13: Eigenvalues of the Hamiltonian over interactions strength with a NumPy eigenvalue solver and the VQE for the Lipkin Hamiltonian for $j = 1$.

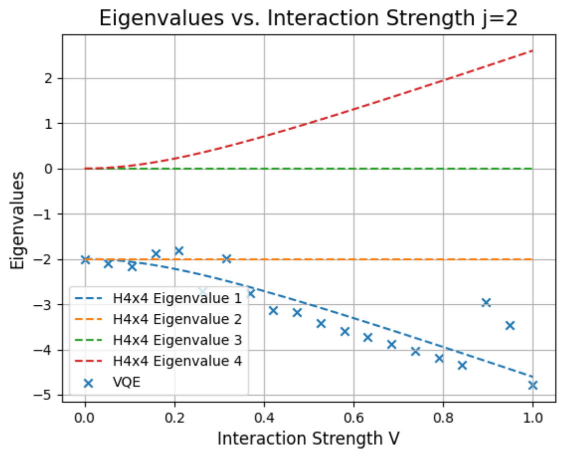

Figure 14: Eigenvalues of the Hamiltonian over interaction strength with a NumPy eigenvalue solver and the VQE for the Lipkin Hamiltonian for $j = 2$.

IX. Conclusion and Outlook

The results from comparing standard eigenvalue solvers like those provided by NumPy to the VQE ((11), (13), (14)) show promising evidence that the VQE algorithm is working properly most of the time and gives a good estimation of the eigenvalues to some extent. A still remaining problem is that the gradient dsecent sometimes gets stuck in local minima and can't find the global minimum. Appart from fixing that, the next step would be to quanitify this comparison by finding analytical solutions to the eigenvalue problem and fitting the VQE values to this function. From this statistically important quantities like standard derivative could be calculated to give a more rigorous explanation of how good the VQE approximates the eigenvalues.
Additionally, the report shows the vast region of optimization that still needs to be done. This should be divided into two parts. First, the classical computational optimization problem of finding the local energy minima of the energy-parameter space, which should also incorporate a grid based search algorithm to work around possible local minima. Second, the optimization of the VQE algorithm and its parameters like the number of shots for each quantum experiment. With both of these areas it is important to reduce computational efforts, as computation time explodes for higher dimensional problems.
Furthermore we would like to add, that the code snippets so far are tailored to the specific problems. Each algorithm, i.e. the Pauli decomposition of a matrix, the implementation of a Hamiltonian into a VQE, the optimization or the plotting could be, in future, combined to a master-VQE, that solves the eigenvalue problem for arbitrary Hamiltonians.

Appendix

Below are code excerpts referred to in the main text.

Listing 1: code exercpt measuring with "brute force"

```
state2 = CNOT @ np.kron(H, I) @ phiPlus
flattened_array = np.array(state2.flatten())
# We set the seed so that we can reproduce the results.
# Set the seed
np.random.seed(42)
# Function measure from the lecture notes:
measure(flattened_array[0], n_shots=10000)
```

Listing 2: Using SymPy

```
import sympy as sp
import numpy as np

H11, H12, H21, H22 = sp.Symbol('H11'), sp.Symbol('H12'), ↩
     sp.Symbol('H21'), sp.Symbol('H22')
E1, E2 = sp.Symbol('E1'), sp.Symbol('E2')
V11, V12, V21, V22 = sp.symbols('V11, V12, V21, V22')

H0 = sp.Matrix([[E1, 0], [0, E2]])
HI = sp.Matrix([[V11, V12], [V21, V22]])

H = H0 + lamda*HI

values={"E1":0, "E2":4, "V11":3, "V22":-3, "V12":sp.↩
     Rational(1, 5), "V21":sp.Rational(1, 5)}
Hnew = H.subs(values)
display(Math(f'H = {sp.latex(Hnew)}'))
evals = Hnew.eigenvals()
evects = Hnew.eigenvects()

evals_values = list(evals.keys())
x1, x2 = evals_values[0], evals_values[1]
display(evals_values[0])
display(evals_values[1])

v1, v2 = [list(tup[2][0]) for tup in evects]
display(Math(f'v_1 = {sp.latex(v1)}'))
display(Math(f'v_2 = {sp.latex(v2)}'))
```

Listing 3: VQE-method with a learning parameter

```
# Gradient descent parameters
eta = 0.05
Niterations = 200
energies = []
```

```
# Random angles using uniform distribution
theta = 2 * np.pi * np.random.rand()
phi = 2 * np.pi * np.random.rand()
pi2 = 0.3 * np.pi
# Perform gradient descent
for iter in range(Niterations):

    thetagradient = 1 / (pi2) * (vqe_step(theta + pi2, ↩
         phi, False) - vqe_step(theta - pi2, phi, False))
    phigradient = 1 / (pi2) * (vqe_step(theta, phi + pi2,↩
          False) - vqe_step(theta, phi - pi2, False))
    theta -= eta * thetagradient
    phi -= eta * phigradient
    energy = vqe_step(theta, phi, False)
    energies.append(energy)
    #print("Iteration:", iter, "Energy:", energy)

print("Minimum energy: ", vqe_step(theta, phi, False))
```

Listing 4: VQE-method using predefined opimizer from SciPy

```
def vqe_step2(angles, verbose = False):
    return vqe_step(angles[0], angles[1], verbose = False↩
         )

np.random.seed(42)
angles_start = np.random.uniform(low = 0, high = np.pi, ↩
     size = 2)
print(f'angles_start = {angles_start}')

#['Powell','Nelder-Mead', 'BFGS', 'CG', 'COBYLA', 'SLSQP↩
     ']
for met in ['Powell']:
    res = minimize(vqe_step2, angles_start,args = (None),↩
          method = met, options = {'maxiter': 10000}, tol ↩
         = 1e-10)
    print(f'method = {met}, res = {res}')
```

ACKNOWLEDGMENT

The authors would like to thank the lecturers Morten Hjorth-Jensen, Ruben Guevara and Keran Chen for their generous and enriching guidance. We would also like to thank the likes of Niels Bohr, Paul Dirac et al for their pioneering contributions to unveiling the secrets of this beautiful topic of physics, eg. quantum physics.

REFERENCES

[1] N. Rossolillo, “Investing in quantum computing stocks,” updated Nov 13, 2023. [Online]. Available: https://www.fool.com/investing/stock-market/market-sectors/information-technology/ai-stocks/quantum-computing-stocks/
[2] M. Q. Hlatshwayo, Y. Zhang, H. Wibowo, R. LaRose, D. Lacroix, and E. Litvinova1, “Simulating excited states of the lipkin model on a quantum computer,” *Physical Review C*, vol. 106, no. 024319, pp. 024 319–1 – 024 319–19, 2022.
[3] W. Scherer, *Mathematics of Quantum Computing*. Kingston, UK: Springer, 2019.
[4] R. Hundt, *Quantum Computing for Programmers*. Cambridge University Press, 2022.
[5] C. P. Kirkebo and M. Fuhrmann, updated April 1, 2024. [Online]. Available: https://github.com/markfuhr23/QuantumComputing
[6] I. Research, open-source software development kit (SDK) for working with quantum computers. [Online]. Available: https://docs.quantum.ibm.com/
[7] Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Qiskit
[8] SciPy. [Online]. Available: https://docs.scipy.org/doc/scipy/tutorial/optimize.html
[9] H. Lipkin, N. Meshkov, and A. Glick, “Validity of many-body approximation methods for a solvable model: (i). exact solutions and perturbation theory,” *Nuclear Physics*, vol. 62, no. 2, pp. 188–198, 1965. [Online]. Available: https://www.sciencedirect.com/science/article/pii/002955826590862X