February 26-March, 2024: Quantum Computing, Quantum Machine Learning and Quantum Information Theories

Morten Hjorth-Jensen[1,2]

Department of Physics, University of Oslo[1]

Department of Physics and Astronomy and Facility for Rare Isotope Beams, Michigan State University[2]

February 28

Plans for the week of February 26-March 1

1. Reminder on basics of the VQE method and how to perform measurements for the simpler one- and two-qubit Hamiltonians
2. Simulating efficiently Hamiltonians on quantum computers with the VQE method and gradient descent to optimize the state function ansatz
3. Introducing the Lipkin model
4. Reading suggestion, VQE review article

Reminder on technicalities

The most important operators are the Pauli matrices, often referred to the X, Y and Z gates

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

Through the tensor product we can compose operators acting on multiple qubits.

Rotations

Another important set of gates are the *rotation operators* R_x, R_y and R_z. By application to a qbit, we can reach any point on the Bloch sphere by usage of all three once. They are expressed as

$$R_x(\theta) = \exp -iX\theta/2 = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix},$$

$$R_y(\theta) = \exp -iY\theta/2 = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ -\sin(\theta/2) & \cos(\theta/2) \end{bmatrix},$$

$$R_z(\theta) = \exp -iZ\theta/2 = \begin{bmatrix} \exp -i\theta/2 & 0 \\ 0 & \exp i\theta/2 \end{bmatrix}$$

with all having a period of 4π.

Rayleigh-Ritz variational principle

The Rayleigh-Ritz variational principle states that for a given Hamiltonian H, the expectation value of a trial state or just ansatz $|x\rangle$ puts a lower bound on the ground state energy E_0.

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

The ansatz

The ansatz is typically chosen to be a parameterized superposition of basis states that can be varied to improve the energy estimate, $|\psi\rangle \mathit{equiv} |\mathit{psi}(\boldsymbol{\theta})\rangle$ where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$ are the M optimization parameters.

Rotations again

To have any flexibility in the ansatz $|\psi\rangle$, we need to allow for parametrization. The most common approach is the so-called R_y ansatz, where we apply chained operations of rotating around the y-axis by $\boldsymbol{\theta} = (\theta_1, \dots, \theta_Q)$ of the Bloch sphere and CNOT operations.
Applications of y rotations specifically ensures that our coefficients always remain real, which often is satisfactory when dealing with many-body systems.

Measurements and more

After the ansatz has been constructed, the Hamiltonian must be applied. As discussed, the Hamiltonian must be written in terms of Pauli strings.
To obtain the expectation value of the ground state energy, one can measure the expectation value of each Pauli string,

$$E(\boldsymbol{\theta}) = \sum_i w_i \langle \psi(\boldsymbol{\theta}) | P_i | \psi(\boldsymbol{\theta}) \rangle \equiv \sum_i w_i f_i,$$

where f_i is the expectation value of the Pauli string i.

Collecting data

This is estimated statistically by considering measurements in the appropriate basis of the operator in the Pauli string.
With N_0 and N_1 as the number of 0 and 1 measurements respectively, we can estimate f_i since

$$f_i = \lim_{N \to \infty} \frac{N_0 - N_1}{N},$$

where N as the number of shots (measurements).
Each Pauli string requires it own circuit, where multiple measurements of each string is required. Adding the results together with the corresponding weights, the ground state energy can be estimated. To optimize with respect to $\boldsymbol{\theta}$, a classical optimizer is often applied.

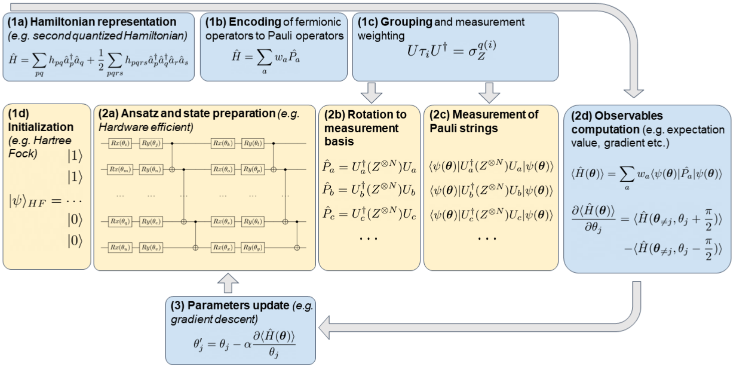
The VQE algoritm

The VQE algorithm consists of several steps, partially done on a classical computer:

1. A parameterized ansatz for the quantum state is implemented on a quantum computer.
2. The ansatz is measured in a given measurement basis.
3. Postprocessing on a classical computer converts the measurement outcomes to an expectation value.
4. Classical minimization algorithms are used to up- date the variational parameters.

The updated variational parameters are then sent back to the quantum computer, and the process is repeated until the optimal variational parameters are found.

VQE overview


(1a) Hamiltonian representation (e.g. second quantized Hamiltonian)
$\hat{H} = \sum_{pq} h_{pq} \hat{a}_p^\dagger \hat{a}_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_r \hat{a}_s$
(1b) Encoding of fermionic operators to Pauli operators
$\hat{H} = \sum_a w_a \hat{P}_a$
(1c) Grouping and measurement weighting
$U \tau_i U^\dagger = \sigma_Z^{q(i)}$
(1d) Initialization (e.g. Hartree Fock)
$|\psi\rangle_{HF} = |1\rangle |1\rangle \cdots |0\rangle |0\rangle$
(2a) Ansatz and state preparation (e.g. Hardware efficient)
$Rx(\theta_i)$ $Ry(\theta_j)$ $Rx(\theta_k)$ $Ry(\theta_l)$
$Rx(\theta_m)$ $Ry(\theta_n)$ $Rx(\theta_o)$ $Ry(\theta_p)$
$Rx(\theta_q)$ $Ry(\theta_c)$ $Rx(\theta_s)$ $Ry(\theta_t)$
$Rx(\theta_u)$ $Ry(\theta_v)$ $Rx(\theta_x)$ $Ry(\theta_y)$
(2b) Rotation to measurement basis
$\hat{P}_a = U_a^\dagger (Z^{\otimes N}) U_a$
$\hat{P}_b = U_b^\dagger (Z^{\otimes N}) U_b$
$\hat{P}_c = U_c^\dagger (Z^{\otimes N}) U_c$
$\cdots$
(2c) Measurement of Pauli strings
$\langle \psi(\boldsymbol{\theta}) | U_a^\dagger (Z^{\otimes N}) U_a | \psi(\boldsymbol{\theta}) \rangle$
$\langle \psi(\boldsymbol{\theta}) | U_b^\dagger (Z^{\otimes N}) U_b | \psi(\boldsymbol{\theta}) \rangle$
$\langle \psi(\boldsymbol{\theta}) | U_c^\dagger (Z^{\otimes N}) U_c | \psi(\boldsymbol{\theta}) \rangle$
$\cdots$
(2d) Observables computation (e.g. expectation value, gradient etc.)
$\langle \hat{H}(\boldsymbol{\theta}) \rangle = \sum_a w_a \langle \psi(\boldsymbol{\theta}) | \hat{P}_a | \psi(\boldsymbol{\theta}) \rangle$
$\frac{\partial \langle \hat{H}(\boldsymbol{\theta}) \rangle}{\partial \theta_j} = \langle \hat{H}(\boldsymbol{\theta}_{\neq j}, \theta_j + \frac{\pi}{2}) \rangle - \langle \hat{H}(\boldsymbol{\theta}_{\neq j}, \theta_j - \frac{\pi}{2}) \rangle$
(3) Parameters update (e.g. gradient descent)
$\theta'_j = \theta_j - \alpha \frac{\partial \langle \hat{H}(\boldsymbol{\theta}) \rangle}{\theta_j}$

🟨 Quantum computing subroutines

🟦 Traditional computing subroutines

VQE and efficient computations of gradients

We start with a reminder on the VQE method with applications to the one-qubit system discussed last week.
Here we revisit the one-qubit system and develop a VQE code for studying this system using gradient descent as a method to optimize the variational ansatz.
We start with a simple 2×2 Hamiltonian matrix expressed in terms of Pauli X and Z matrices, as discussed in the project text.

Symmetric matrix

We define a symmetric matrix $H \in \mathbb{R}^{2\times 2}$

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix},$$

We let $H = H_0 + H_I$, where

$$H_0 = \begin{bmatrix} E_1 & 0 \\ 0 & E_2 \end{bmatrix},$$

is a diagonal matrix. Similarly,

$$H_I = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix},$$

where V_{ij} represent various interaction matrix elements.

Non-interacting solution

We can view H_0 as the non-interacting solution

$$H_0|0\rangle = E_1|0\rangle, \tag{2}$$

and

$$H_0|1\rangle = E_2|1\rangle, \tag{3}$$

where we have defined the orthogonal computational one-qubit basis states $|0\rangle$ and $|1\rangle$.

Rewriting with Pauli matrices

We rewrite H (and H_0 and H_I) via Pauli matrices

$$H_0 = \mathcal{E} I + \Omega \sigma_z, \quad \mathcal{E} = \frac{E_1 + E_2}{2}, \; \Omega = \frac{E_1 - E_2}{2},$$

and

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

with $c = (V_{11} + V_{22})/2$, $\omega_z = (V_{11} - V_{22})/2$ and $\omega_x = V_{12} = V_{21}$. We let our Hamiltonian depend linearly on a strength parameter λ

$$H = H_0 + \lambda H_{\mathrm{I}},$$

with $\lambda \in [0, 1]$, where the limits $\lambda = 0$ and $\lambda = 1$ represent the non-interacting (or unperturbed) and fully interacting system, respectively.

Selecting parameters

The model is an eigenvalue problem with only two available states. Here we set the parameters $E_1 = 0$, $E_2 = 4$, $V_{11} = -V_{22} = 3$ and $V_{12} = V_{21} = 0.2$.
The non-interacting solutions represent our computational basis. Pertinent to our choice of parameters, is that at $\lambda \geq 2/3$, the lowest eigenstate is dominated by $|1\rangle$ while the upper is $|0\rangle$. At $\lambda = 1$ the $|0\rangle$ mixing of the lowest eigenvalue is 1% while for $\lambda \leq 2/3$ we have a $|0\rangle$ component of more than 90%. The character of the eigenvectors has therefore been interchanged when passing $z = 2/3$. The value of the parameter V_{12} represents the strength of the coupling between the two states.

Setting up the matrix

This part is best seen using the jupyter-notebook

```
from  matplotlib import pyplot as plt
import numpy as np
dim = 2
Hamiltonian = np.zeros((dim,dim))
e0 = 0.0
e1 = 4.0
Xnondiag = 0.20
Xdiag = 3.0
Eigenvalue = np.zeros(dim)
# setting up the Hamiltonian
Hamiltonian[0,0] = Xdiag+e0
Hamiltonian[0,1] = Xnondiag
Hamiltonian[1,0] = Hamiltonian[0,1]
Hamiltonian[1,1] = e1-Xdiag
# diagonalize and obtain eigenvalues, not necessarily sorted
EigValues, EigVectors = np.linalg.eig(Hamiltonian)
permute = EigValues.argsort()
EigValues = EigValues[permute]
# print only the lowest eigenvalue
print(EigValues[0])
```

Now rewrite it in terms of the identity matrix and the Pauli matrix X and Z

```
# Now rewrite it in terms of the identity matrix and the Pauli matrix
X = np.array([[0,1],[1,0]])
Y = np.array([[0,-1j],[1j,0]])
```

Implementing the VQE

For a one-qubit system we can reach every point on the Bloch sphere (as discussed earlier) with a rotation about the x-axis and the y-axis.
We can express this mathematically through the following operations (see whiteboard for the drawing), giving us a new state $|\psi\rangle$

$$|\psi\rangle = R_y(\phi)R_x(\theta)|0\rangle.$$

Multiple ansatzes

We can produce multiple ansatzes for the new state in terms of the angles θ and ϕ. With these ansatzes we can in turn calculate the expectation value of the above Hamiltonian, now rewritten in terms of various Pauli matrices (and thereby gates), that is compute

$$\langle\psi|(c+\mathcal{E})\boldsymbol{I}+(\Omega+\omega_z)\boldsymbol{\sigma}_z+\omega_x\boldsymbol{\sigma}_x|\psi\rangle.$$

Rotations again

We can now set up a series of ansatzes for $|\psi\rangle$ as function of the angles θ and ϕ and find thereafter the variational minimum using for example a gradient descent method.
To do so, we need to remind ourselves about the mathematical expressions for the rotational matrices/operators.

$$R_x(\theta) = \cos\frac{\theta}{2}\boldsymbol{I} - \imath\sin\frac{\theta}{2}\boldsymbol{\sigma}_x,$$

and

$$R_y(\phi) = \cos\frac{\phi}{2}\boldsymbol{I} - \imath\sin\frac{\phi}{2}\boldsymbol{\sigma}_y.$$

Simple code

```
# define the rotation matrices
# Define angles theta and phi
theta = 0.5*np.pi; phi = 0.2*np.pi
Rx = np.cos(theta*0.5)*I-1j*np.sin(theta*0.5)*X
Ry = np.cos(phi*0.5)*I-1j*np.sin(phi*0.5)*Y
#define basis states
basis0 = np.array([1,0])
basis1 = np.array([0,1])

NewBasis = Ry @ Rx @ basis0
print(NewBasis)
# Compute the expectation value
#Note hermitian conjugation
Energy = NewBasis.conj().T @ Hamiltonian @ NewBasis
print(Energy)
```

Not an impressive results. We set up now a loop over many angles θ and ϕ and compute the energies

```
# define a number of angles
n = 20
angle = np.arange(0,180,10)
n = np.size(angle)
ExpectationValues = np.zeros((n,n))
for i in range (n):
    theta = np.pi*angle[i]/180.0
    Rx = np.cos(theta*0.5)*I-1j*np.sin(theta*0.5)*X
    for j in range (n):
```

Gradient descent and calculations of gradients

In order to optimize the VQE ansatz, we need to compute derivatives with respect to the variational parameters. Here we develop first a simpler approach tailored to the one-qubit case. For this particular case, we have defined an ansatz in terms of the Pauli rotation matrices.

Setting up gradients

These define an arbitrary one-qubit state on the Bloch sphere through the expression

$$|\psi\rangle = |\psi(\theta,\phi)\rangle = R_y(\phi)R_x(\theta)|0\rangle.$$

Each of these rotation matrices can be written in a more general form as

$$R_i(\gamma) = \exp -(\imath\frac{\gamma}{2}\sigma_i) = \cos(\frac{\gamma}{2})\boldsymbol{I} - \imath\sin(\frac{\gamma}{2})\boldsymbol{\sigma}_i,$$

where σ_i is one of the Pauli matrices $\sigma_{x,y,z}$.

Derivatives

It is easy to see that the derivative with respect to γ is

$$\frac{\partial R_i(\gamma)}{\partial \gamma} = -\frac{\gamma}{2}\boldsymbol{\sigma}_i R_i(\gamma).$$

Derivatives of the expectation value of the Hamiltonian

We can now calculate the derivative of the expectation value of the Hamiltonian in terms of the angles θ and ϕ. We have two derivatives

$$\frac{\partial}{\partial\theta}\left[\langle\psi(\theta,\phi)|\boldsymbol{H}|\psi(\theta,\phi)\rangle\right] = \frac{\partial}{\partial\theta}\left[\langle\boldsymbol{H}(\theta,\phi)\rangle\right] = \langle\psi(\theta,\phi)|\boldsymbol{H}(-\frac{\imath}{2}\boldsymbol{\sigma}_x|\psi(\theta,\phi)\rangle+$$

and

$$\frac{\partial}{\partial\phi}\left[\langle\psi(\theta,\phi)|\boldsymbol{H}|\psi(\theta,\phi)\rangle\right] = \frac{\partial}{\partial\phi}\left[\langle\boldsymbol{H}(\theta,\phi)\rangle\right] = \langle\psi(\theta,\phi)|\boldsymbol{H}(-\frac{\imath}{2}\boldsymbol{\sigma}_y|\psi(\theta,\phi)\rangle-$$

Two addtional expectation values

This means that we have to calculate two additional expectation values in addition to the expectation value of the Hamiltonian itself. If we stay with an ansatz for the single qubit states given by the above rotation operators, we can, following for example the article by Maria Schuld et al, show that the derivative of the expectation value of the Hamiltonian can be written as (we focus only on a given angle ϕ)

$$\frac{\partial}{\partial\phi}\left[\langle \boldsymbol{H}(\phi)\rangle\right] = \frac{1}{2}\left[\langle \boldsymbol{H}(\phi+\frac{\pi}{2})\rangle - \langle \boldsymbol{H}(\phi-\frac{\pi}{2})\rangle\right].$$

Rotations again and again

To see this, consider again the definition of the rotation operators. We can write these operators as

$$R_i(\phi) = \exp -\imath(\phi\boldsymbol{\sigma}_i),$$

with ***sigma***$_i$, with $\boldsymbol{\sigma}_i$ being any of the Pauli matrices X, Y and Z. The latter can be generalized to other unitary matrices as well. The derivative with respect to ϕ gives

$$\frac{\partial R_i(\phi)}{\partial\phi} = -\frac{\imath}{2}\boldsymbol{\sigma}_i \exp -\imath(\phi\boldsymbol{\sigma}_i) = -\frac{\imath}{2}\boldsymbol{\sigma} R_i(\phi).$$

Bloch sphere math

Our ansatz for a general one-qubit state on the Bloch sphere contains the product of a rotation around the x-axis and the y-axis. In the derivation here we focus only on one angle however. Our ansatz is then given by

$$|\psi\rangle = R_i(\phi)|0\rangle,$$

and the expectation value of our Hamiltonian is

$$\langle\psi|\hat{H}|\psi\rangle = \langle 0|R_i(\phi)^\dagger \hat{H} R_i(\phi)|0\rangle.$$

Derivatives

Our derivative with respect to the angle ϕ has a similar structure, that is

$$\frac{\partial}{\partial\phi}\left[\langle\psi(\theta,\phi)|\boldsymbol{H}|\psi(\theta,\phi)\rangle\right] = \langle\psi(\theta,\phi)|\boldsymbol{H}(-\frac{\imath}{2}\boldsymbol{\sigma}_y|\psi(\theta,\phi)\rangle + \text{ h.c.}$$

Rewriting

In order to rewrite the equation of the derivative, the following relation is useful

$$\langle\psi|\hat{A}^\dagger\hat{B}\hat{C}|\psi\rangle = \frac{1}{2}\Big[\langle\psi|(\hat{A}+\hat{C})^\dagger\hat{B}(\boldsymbol{A}+\hat{C})|\psi\rangle - \langle\psi|(\hat{A}-\hat{C})^\dagger\hat{B}(\boldsymbol{A}-\hat{C})|\psi\rangle$$

where $\hat{A}$, $\hat{B}$ and $\hat{C}$ are arbitrary hermitian operators.

Final manipulations

If we identify these operators as $\hat{A} = \boldsymbol{I}$, with $\boldsymbol{I}$ being the unit operator, $\hat{B} = \hat{H}$ our Hamiltonian, and $\hat{C} = -\imath\boldsymbol{\sigma}_i/2$, we obtain the following expression for the expectation value of the derivative (excluding the hermitian conjugate)

$$\langle\psi|\boldsymbol{I}^\dagger\hat{H}(-\frac{\imath}{2}\boldsymbol{\sigma}_i|\psi\rangle = \frac{1}{2}\Big[\langle\psi|(\boldsymbol{I} - \frac{\imath}{2}\boldsymbol{\sigma}_i)^\dagger\hat{H}(\boldsymbol{I} - \frac{\imath}{2}\boldsymbol{\sigma}_i)|\psi\rangle - \langle\psi|(\boldsymbol{I} + \frac{\imath}{2}\boldsymbol{\sigma}_i)^\dagger\hat{H}($$

The expressions to implement

If we then use that the rotation matrices can be rewritten as

$$R_i(\phi) = \exp -(\imath\frac{\phi}{2}\sigma_i) = \cos(\frac{\phi}{2})\boldsymbol{I} - \imath\sin(\frac{\phi}{2})\boldsymbol{\sigma}_i,$$

we see that if we set the angle to $\phi = \pi/2$, we have

$$R_i(\frac{\pi}{2}) = \cos(\frac{\pi}{4})\boldsymbol{I} - \imath\sin(\frac{\pi}{4})\boldsymbol{\sigma}_i = \frac{1}{\sqrt{2}}\left(\boldsymbol{I} - \frac{\imath}{2}\boldsymbol{\sigma}_i\right).$$

Final expression

This means that we can write

$$\langle\psi|\boldsymbol{I}^\dagger\hat{H}(-\frac{\imath}{2}\boldsymbol{\sigma}_i|\psi\rangle = \frac{1}{2}\Big[\langle\psi|R_i(\frac{\pi}{2})^\dagger\hat{H}R_i(\frac{\pi}{2})|\psi\rangle - \langle\psi|R_i(-\frac{\pi}{2})^\dagger\hat{H}R_i(-\frac{\pi}{2})^\dagger|\psi$$

Basics of gradient descent and stochastic gradient descent

In order to implement the above equations, we need to remind the reader about basic elements of various optimization approaches. Our main focus here will be various gradient descent approaches and quasi-Newton methods like Broyden's algorithm and variations thereof.
This material is covered by the lectures from FYS4411 on gradient optimization

Computing quantum gradients

Let us implement efficient implementations of gradient methods to the derivatives of the Hamiltonian expectation values.

```python
from  matplotlib import pyplot as plt
import numpy as np
from scipy.optimize import minimize
dim = 2
Hamiltonian = np.zeros((dim,dim))
e0 = 0.0
e1 = 4.0
Xnondiag = 0.20
Xdiag = 3.0
Eigenvalue = np.zeros(dim)
# setting up the Hamiltonian
Hamiltonian[0,0] = Xdiag+e0
Hamiltonian[0,1] = Xnondiag
Hamiltonian[1,0] = Hamiltonian[0,1]
Hamiltonian[1,1] = e1-Xdiag
# diagonalize and obtain eigenvalues, not necessarily sorted
EigValues, EigVectors = np.linalg.eig(Hamiltonian)
permute = EigValues.argsort()
EigValues = EigValues[permute]
# print only the lowest eigenvalue
print(EigValues[0])

# Now rewrite it in terms of the identity matrix and the Pauli matrix
X = np.array([[0,1],[1,0]])
Y = np.array([[0,-1j],[1j,0]])
```

A smarter way of doing this

The above approach means that we are setting up several matrix-matrix and matrix-vector multiplications. Although straight forward it is not the most efficient way of doing this, in particular in case the matrices become large (and sparse). But there are some more important issues.
In a physical realization of these systems we cannot just multiply the state with the Hamiltonian. When performing a measurement we can only measure in one particular direction. For the computational basis states which we have, $|0\rangle$ and $|1\rangle$, we have to measure along the bases of the Pauli matrices and reconstruct the eigenvalues from these measurements.

Using properties of Pauli matrices

From our earlier discussions we know that the Pauli Z matrix has the above basis states as eigen states through

$$Z|0\rangle = \boldsymbol{Z}|0\rangle = +1|0\rangle,$$

and

$$Z|1\rangle = \boldsymbol{Z}|1\rangle = -1|1\rangle,$$

with eigenvalue -1.

Pauli X reminder

For the Pauli X matrix on the other hand we have

$$\boldsymbol{\sigma}_x|0\rangle = \boldsymbol{X}|0\rangle = +1|1\rangle,$$

and

$$\boldsymbol{\sigma}_x|1\rangle = \boldsymbol{X}|1\rangle = -1|0\rangle,$$

with eigenvalues 1 in both cases. The latter two equations tell us that the computational basis we have chosen, and in which we will prepare our states, is not an eigenbasis of the σ_x matrix.

Rewriting the Pauli X matrix

We rewrite the Pauli X matrix in terms of a Pauli Z matrixcusing the Hadamard matrix twice, that is

$$\boldsymbol{X} = \boldsymbol{\sigma}_x = \boldsymbol{HZH}.$$

The Pauli Y matrix

The Pauli Y matrix can be written as

$$\boldsymbol{Y} = \boldsymbol{\sigma}_y = \boldsymbol{HS}^\dagger \boldsymbol{ZHS},$$

where S is the phase matrix

$$S = \begin{bmatrix} 1 & 0 \\ 0 & \imath \end{bmatrix}.$$

The code for the one qubit case (code developed by August Gude, 2023)

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set_theme(font_scale=1.5)
from tqdm import tqdm

sigma_x = np.array([[0, 1], [1, 0]])
sigma_y = np.array([[0, -1j], [1j, 0]])
sigma_z = np.array([[1, 0], [0, -1]])
I = np.eye(2)

def Hamiltonian(lmb):
    E1 = 0
    E2 = 4
    V11 = 3
    V22 = -3
    V12 = 0.2
    V21 = 0.2

    eps = (E1 + E2) / 2
    omega = (E1 - E2) / 2
    c = (V11 + V22) / 2
    omega_z = (V11 - V22) / 2
    omega_x = V12

    H0 = eps * I + omega * sigma_z
    H1 = c * I + omega_z * sigma_z + omega_x * sigma_x
```