RANK   TEAM   **CHALLENGES**   SUBMISSIONS   SUPPORT

🚀 **CHALLENGE COMPLETED**   **View successful submissions**

∨ Jump to code   — Collapse text

Ising Uprising

500 points

Backstory

Zenda and Reece model Sqynet as a spin chain, and they come up with a strategy. What if, in addition to using plasma bombs and missiles to increase the temperature of the device, they use a strong magnetic field? After all, magnetic fields might pass through Sqynet's outer shell more easily. The scientists proceed to simulate the effect of a magnetic field on a closed spin chain to quantify the effects.

Ground state of an Ising spin chain

A simple way to model Sqynet is by considering it as a closed spin chain of length N. A spin chain contains particles of spin 1/2 in each of its N sites. The spins may be pointing in the positive or negative z direction, and we consider that there may be an external magnetic field acting on the system.
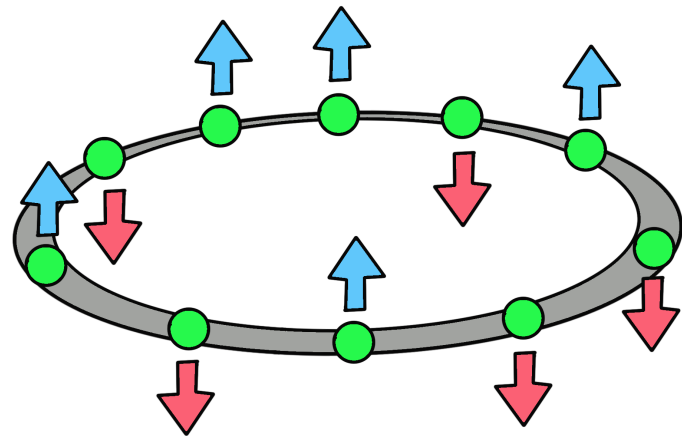
Closed spin chain with N≈10 sites

Such a quantum system is described by the *Transverse Ising Hamiltonian*. For closed spin chain with a transverse magnetic field of intensity h, the Transverse Ising Hamiltonian reads

$$H = -\sum_{i=1}^{N} Z_i \otimes Z_{i+1} - h\sum_{i}^{N} X_i.$$

The subindices i indicate the spin site where the operators act. In a closed spin chain, we identify site $N+1$ with the first site.

A possible plan for Zenda and Reece is to use a strong magnetic field that changes the ground energy of Sqynet, causing it to malfunction.

Your task is to help Zenda and Reece calculate the effect of external magnetic forces on the ground energy. Using the Variational Quantum Eigensolver (VQE) algorithm, you will compute the ground energy of a closed spin chain of length $N = 4$.

▼ **Epilogue**
Zenda and Reece fire their powerful magnetic field, missiles, and bombs into the tangle of spins that is Sqynet. As its wavefunction starts to collapse,

reality around them shimmers and shifts, and they are enveloped in the purple smoke characteristic of the Oracle World Transform.

The smoke clears. The Bloch clock is ticking; they are late for their weekly brainstorming session at Trine's Designs. Trine is fussing excitedly with the new coffee machine. The network is patchy. Zenda and Reece smile at each other. There are far worse things, they realize, than business as usual.

Challenge code

In this challenge you will be given the following functions:

- `create_Hamiltonian`: In which you build the Transverse Ising Hamiltonian for $N = 4$ and a magnetic field intensity `h`. **You must complete this function.**

- `model`: This QNode builds a general enough ansatz for the ground state. This circuit must depend on some parameters `params`, which you will later optimize. It returns the expectation value of the Hamiltonian for the output state of the circuit. **You must complete this function.**

- `train`: This function returns the parameters that minimize the output of `model`. **You must complete this function.**

Input

As input to this problem, you are given:

- `h` (`float`): Magnetic field intensity applied to the spin chain.

Output

This code will output a `float` corresponding to the energy of the ground state.

If your solution matches the correct one within the given tolerance specified in `check` (in this case it's an relative tolerance of `0.1`), the output will be `"Correct!"` Otherwise, you will receive a `"Wrong answer"` prompt.

Good luck!

Code

❓ Help

```python
1   import json
2   import pennylane as qml
3   import pennylane.numpy as np

4 v def create_Hamiltonian(h):
5       """
6       Function in charge of generating the Hamiltonian of the stat
7
8       Args:
9           h (float): magnetic field strength
10
11      Returns:
12          (qml.Hamiltonian): Hamiltonian of the statement associat
13      """
14
```

```python
15      I = np.array([[1, 0], [0, 1]])
16      X = qml.PauliX.matrix
17      Y = qml.PauliY.matrix
18      Z = qml.PauliZ.matrix
19  # Define Pauli matrices
20      I = np.array([[1, 0], [0, 1]])
21      X = np.array([[0, 1], [1, 0]])
22      Y = np.array([[0, -1j], [1j, 0]])
23      Z = np.array([[1, 0], [0, -1]])
24
25      # Define Hamiltonian terms
26      term1 = qml.Hamiltonian([0.5], [qml.Identity(0) @ qml.Identi
27      term2 = qml.Hamiltonian([0.5], [qml.Identity(2) @ qml.Identi
28      term3 = qml.Hamiltonian([-0.5], [qml.Identity(0) @ qml.Ident
29      term4 = qml.Hamiltonian([h], [qml.PauliZ(0)])
30      term5 = qml.Hamiltonian([h], [qml.PauliZ(1)])
31      term6 = qml.Hamiltonian([h], [qml.PauliZ(2)])
32      term7 = qml.Hamiltonian([h], [qml.PauliZ(3)])
33
34      # Combine Hamiltonian terms
35      H = term1 + term2 + term3 + term4 + term5 + term6 + term7
36
37      return H
38
39
40
```

```python
41  dev = qml.device("default.qubit", wires=4)
42
43  @qml.qnode(dev)
44 ∨ def model(params, H):
45      """
46      To implement VQE you need an ansatz for the candidate ground
47      Define here the VQE ansatz in terms of some parameters (para
48      create the candidate ground state. These parameters will
49      be optimized later.
50
51      Args:
52          params (numpy.array): parameters to be used in the varia
53          H (qml.Hamiltonian): Hamiltonian used to calculate the e
54
55      Returns:
56          (float): Expected value with respect to the Hamiltonian
57      """
58

59      # Put your code here #
60

61 ∨ def train(h):
62      """
63      In this function you must design a subroutine that returns t
64      parameters that best approximate the ground state.
65
66      Args:
67          h (float): magnetic field strength
68
69      Returns:
70          (numpy.array): parameters that best approximate the grou
71      """
72

73      # Put your code here #
74
```

```python
75   # These functions are responsible for testing the solution.
76 ∨ def run(test_case_input: str) -> str:
77       ins = json.loads(test_case_input)
78       params = train(ins)
79       return str(model(params, create_Hamiltonian(ins)))
80
81
82 ∨ def check(solution_output: str, expected_output: str) -> None:
83       solution_output = json.loads(solution_output)
84       expected_output = json.loads(expected_output)
85       assert np.allclose(
86           solution_output, expected_output, rtol=1e-1
87       ), "The expected value is not correct."
88


89   test_cases = [['1.0', '-5.226251859505506'], ['2.3', '-9.6638🔒


90 ∨ for i, (input_, expected_output) in enumerate(test_cases):
91       print(f"Running test case {i} with input '{input_}'...")
92
93 ∨   try:
94           output = run(input_)
95
96 ∨   except Exception as exc:
97           print(f"Runtime Error. {exc}")
98
99 ∨   else:
100 ∨      if message := check(output, expected_output):
101              print(f"Wrong Answer. Have: '{output}'. Want: '{expe
102
103 ∨      else:
104              print("Correct!")
```

Copy all

Submit

Open Notebook ↗

Reset