Transverse Field Ising Model – Numerical Study

Tomas Gradowski, Clement Rauch

November 9, 2025

1 Model Definition

We consider the transverse field Ising model for n spin-$\frac{1}{2}$ particles:

$$\hat{H} = \sum_{k=1}^{n-1} \hat{\sigma}_{z,k}\hat{\sigma}_{z,k+1} - \sum_{k=1}^{n} \hat{\sigma}_{x,k},$$

where the Pauli matrices are

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

2 $n = 2$ case

the case for n = 2 is easy to compute by hand:

$$\begin{aligned} \hat{H} &= \hat{\sigma}_{z,1}\hat{\sigma}_{z,2} - \hat{\sigma}_{x,1} - \hat{\sigma}_{x,2} \\ &= \hat{\sigma}_z \otimes \hat{\sigma}_z - (\hat{\sigma}_x \otimes \mathbb{I} + \mathbb{I} \otimes \hat{\sigma}_x) \end{aligned}$$

In the z-basis $\{|++\rangle, |+-\rangle, |-+\rangle, |--\rangle\}$,

$$\hat{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

therefore:

$$\boxed{\hat{H} = \begin{pmatrix} 1 & -1 & -1 & 0 \\ -1 & -1 & 0 & -1 \\ -1 & 0 & -1 & -1 \\ 0 & -1 & -1 & 1 \end{pmatrix}}$$

$$\boxed{\mathrm{Sp}(\hat{H}) = \{-\sqrt{5},\, -1,\, 1,\, \sqrt{5}\}}$$

hence $E_0(n{=}2) = -\sqrt{5} \approx -2.236$.

3 Python Implementation

The code below constructs H for arbitrary n and diagonalizes it. It also computes the ground energy and the required memory.It uses the matrix product (`@`) once and the tensor product np.kron.

Listing 1: Ising Hamiltonian Construction and Diagonalization

```
import numpy as np

Sx = np.array([[0, 1],[1, 0]], dtype = complex)
Sz = np.array([[1, 0],[0, -1]], dtype = complex)
Id = np.array([[1, 0],[0, 1]], dtype = complex)

def kron_list(lst):
    a = lst[0]
    for i in lst[1:]:
        a = np.kron(a, i)
    return a
def sx(k, n):
    a = [Id]*n
    a[k] = Sx
    return kron_list(a)
def sz(k, n):
    b = [Id]*n
    b[k] = Sz
    return kron_list(b)
def ising(n):
    H = np.zeros((2**n, 2**n), dtype=complex)
    for k in range(n-1):
        H += sz(k, n) @ sz(k+1, n)
    for k in range(n):
        H -= sx(k, n)
    return H

def printH(n):
```

```
    H = ising(n)
    print("The hamiltonian is:")
    print(np.real_if_close(H))

def gndstate(n):
    H = ising(n)
    E0 = np.min((np.linalg.eigh(H))[0])
    print("its grounstate energy is:")
    print(E0)

def memory(n):
    print("the required space in terabytes is:")
    a = (2**(2*n)*(np.zeros((1,1), dtype=complex).nbytes))
       /10**12
    print(a)
```

4 Results

4.1 Case $n = 2$

The Hamiltonian matrix is

$$H_{n=2} = \begin{pmatrix} 1 & -1 & -1 & 0 \\ -1 & -1 & 0 & -1 \\ -1 & 0 & -1 & -1 \\ 0 & -1 & -1 & 1 \end{pmatrix}.$$

and the ground-state energy:

$$E_0(n = 2) = -\sqrt{5} \approx -2.236.$$

4.2 Case $n = 10$

Running the code for $n = 10$ give for the Hamiltonian matrix:

$$H_{n=10} = \begin{pmatrix} 9 & -1 & -1 & \dots & 0 & 0 & 0 \\ -1 & 7 & 0 & \dots & 0 & 0 & 0 \\ -1 & 0 & 5 & \dots & 0 & 0 & 0 \\ & & & \dots & & & \\ 0 & 0 & 0 & \dots & 5 & 0 & -1 \\ 0 & 0 & 0 & \dots & 0 & 7 & -1 \\ 0 & 0 & 0 & \dots & -1 & -1 & 9 \end{pmatrix}.$$

$$E_0(n = 10) \approx -12.38.$$

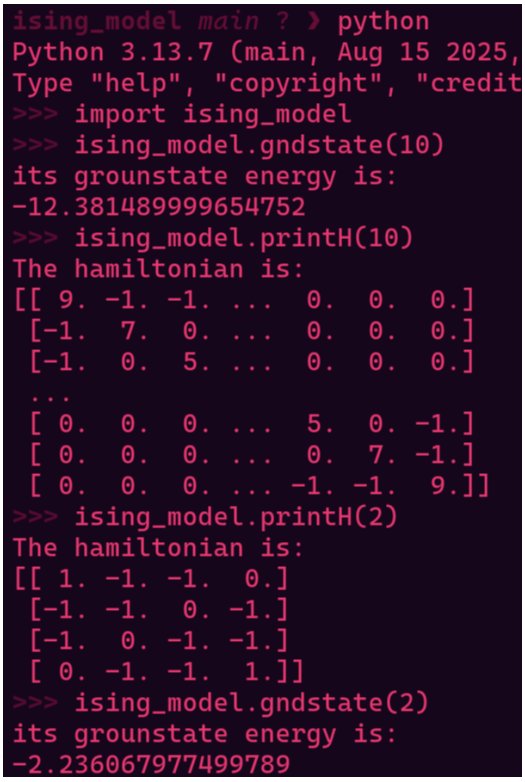

Figure 1: results

The energy per site thus converges to

$$\frac{E_0}{n} \approx -1.238,$$

4.3 Why $n = 30$ is Impractical

For n spins, the Hamiltonian has dimension $2^n \times 2^n = 2^{2n}$. and since its entry of the matrix is a float, of size 4bits we can approximate the memory

required is:

$$\begin{aligned}\text{memory}(n) &= \frac{\dim(H_n) \times 4\text{bites}}{10^9 \frac{\text{bites}}{Gb}} \\ &= \frac{2^{2n} \times 2^2}{10^9}\text{Gb} \\ &\approx \frac{2^{2n+2}}{2^{30}}\text{Gb} \\ &= 2^{(2n-26)}\text{Gb}\end{aligned}$$

$$\begin{aligned}n = 10 &: 2^{20} \times 16 \approx 1.68 \times 10^7 \text{ bytes } \approx \text{ 16 MiB}, \\ n = 18 &: 2^{36} \times 16 \approx 1.1 \times 10^{12} \text{ bytes } \approx \text{ 1.0 TiB}, \\ n = 20 &: 2^{40} \times 16 \approx 1.76 \times 10^{13} \text{ bytes } \approx \text{ 16 TiB}, \\ n = 30 &: 2^{60} \times 16 \approx 1.84 \times 10^{19} \text{ bytes } \approx \text{ 18 EB } (\sim 18 \times 10^6 \text{ TB}).\end{aligned}$$

The code contains the function memory(n) to perform that computation. Using the formula or the function, we can verify that the required memory grows exponentially and quickly requires huge amount of RAM memory for apparently small numbers of n. for n=10, the required memory is only 4 mega bytes, while for n = 18, the required memory is already of 1Tb far beyond any RAM capacity, for n = 20 that value reaches 16 Terabytes, and for n = 30, the surprising value of 17 million terabytes Hence, exact diagonalization becomes computationally impossible for $n \gtrsim 19$.

```
>>> ising_model.memory(2)
the required space in terabytes is:
2.56e-10
>>> ising_model.memory(10)
the required space in terabytes is:
1.6777216e-05
>>> ising_model.memory(15)
the required space in terabytes is:
0.017179869184
>>> ising_model.memory(18)
the required space in terabytes is:
1.099511627776
>>> ising_model.memory(20)
the required space in terabytes is:
17.592186044416
>>> ising_model.memory(30)
the required space in terabytes is:
18446744.07370955
```

Figure 2: memory requirements

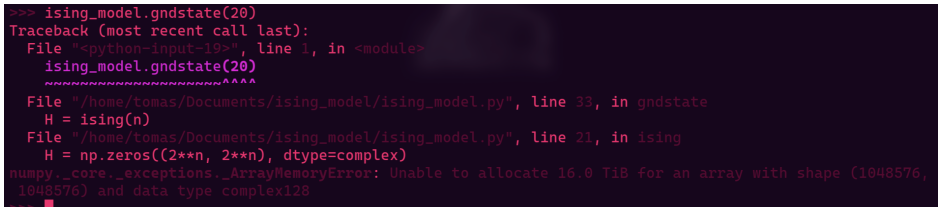

Figure 3: case of n = 20

5 Repository

The full source code is available at:

`https://github.com/Tomas-Gradowski/ising-model/`

https://tomas-gradowski.github.io/portifolio/