Computational Many-Body Physics - Sheet 6

Clemens Giesen (7322871), Tristan Kahl (7338950) & Felix Höddinghaus (7334955)

July 2nd, 2022

The full implementation of all exercises can be found under `https://github.com/Fhoeddinghaus/cmbp22-exercises/tree/main/sheet_6`.

1. Kitaev clusters

The Hamiltonian

$$H = -\sum_{i,j,\alpha} J_{ij^\alpha} S_i^\alpha S_j^\alpha,$$

$(i \neq j = 1, \ldots, N,\ \alpha = x, y, z)$ with each spin of the cluster connected to exaclty three different spins with different couplings (x, y, z).

a). Eigenenergies for $N = 6$

For $N = 6$, there are two different Kitaev clusters:

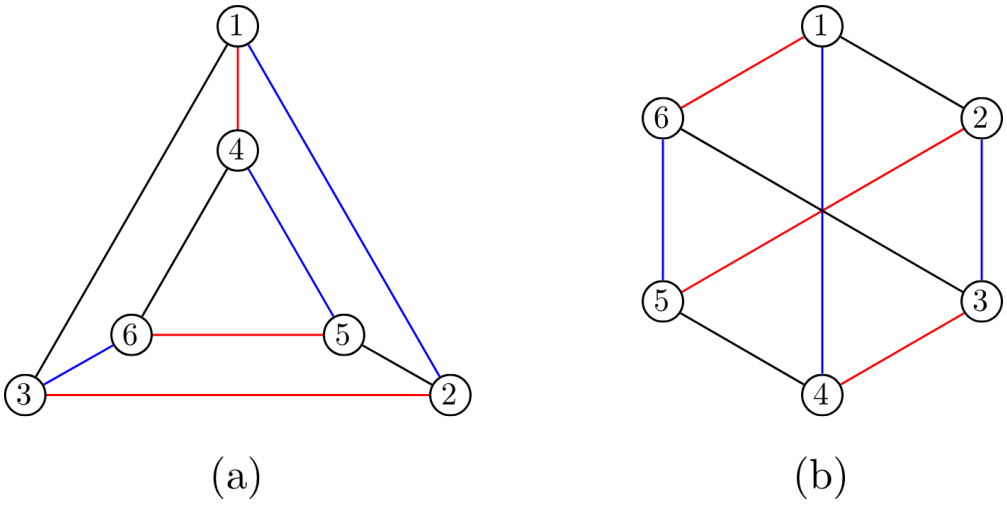

Figure 1.1: The two possible Kitaev clusters for $N = 6$, with x-links in red, y-links in blue and z-links in black.

We again use the syntax and functions from sheet 4[1] to calculate the Hamilton matrices.

[1] The functions are again implemented in `hamilton_spins.jl` in the repository.

a).1. Cluster (a)

At first we need to construct the couplings J_{ij}^{α} according to (Fig. 1.1), calculate the corresponding Hamilton matrix using `calculate_hamilton_matrix()` and then calculate the eigenenergies using `eigvals()` from `LinearAlgebra.jl`:

```
N = 6
J = 1

Js_N6_a = reset_Js(N)

# x links
Js_N6_a[1][1,4] = J
Js_N6_a[1][2,3] = J
Js_N6_a[1][5,6] = J
# y links
Js_N6_a[2][1,2] = J
Js_N6_a[2][3,6] = J
Js_N6_a[2][4,5] = J
# z links
Js_N6_a[3][1,3] = J
Js_N6_a[3][2,5] = J
Js_N6_a[3][4,6] = J

H_N6_a = calculate_hamilton_matrix(Js_N6_a, N)
ψ_N6_a = eigvecs(rationalize.(Matrix(H_N6_a)))
E_N6_a = unique(round.(eigvals(rationalize.(Matrix(H_N6_a))); digits=10))
```

This gives us the spectrum of the 11 degenerated eigenenergies:

$$E_1 = -1.157, \quad E_2 = -1.031, \quad E_3 = -0.75, \quad E_4 = -0.616, \quad E_5 = -0.25, \quad E_6 = -0.015,$$
$$E_7 = 0.25, \quad E_8 = 0.921, \quad E_9 = 1.031, \quad E_{10} = 1.116, \quad E_{11} = 1.25$$

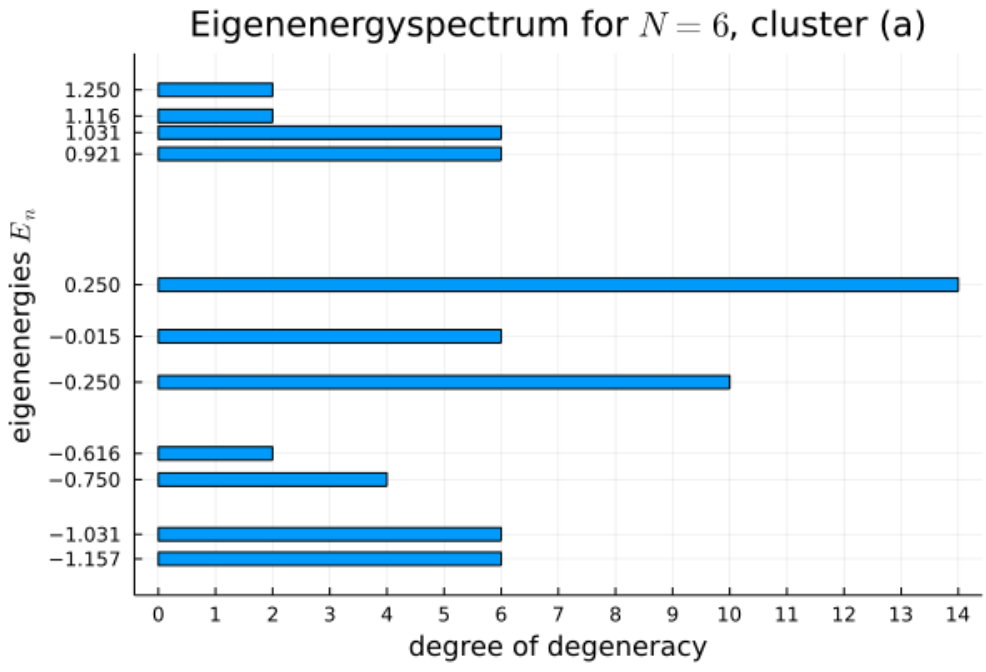

Figure 1.2

a).2. Cluster (b)

Like before, we construct the couplings according to (Fig. 1.1), calculate the Hamilton matrix and the eigenenergies:

```
N = 6
J = 1

Js_N6_b = reset_Js(N)

# x links
Js_N6_b[1][1,6] = J
Js_N6_b[1][2,5] = J
Js_N6_b[1][3,4] = J
# y links
Js_N6_b[2][1,4] = J
Js_N6_b[2][2,3] = J
Js_N6_b[2][5,6] = J
# z links
Js_N6_b[3][1,2] = J
Js_N6_b[3][3,6] = J
Js_N6_b[3][4,5] = J

H_N6_b = calculate_hamilton_matrix(Js_N6_b, N)
ψ_N6_b = eigvecs(rationalize.(Matrix(H_N6_b)))
E_N6_b = unique(round.(eigvals(rationalize.(Matrix(H_N6_b))); digits=7))
```

This gives us the spectrum of the 8 degenerated eigenenergies:

$$E_1 = -1.25, \quad E_2 = -1.0307764, \quad E_3 = -0.75, \quad E_4 = -0.25,$$
$$E_5 = 0.25, \quad E_6 = 0.75, \quad E_7 = 1.0307764, \quad E_8 = 1.25$$

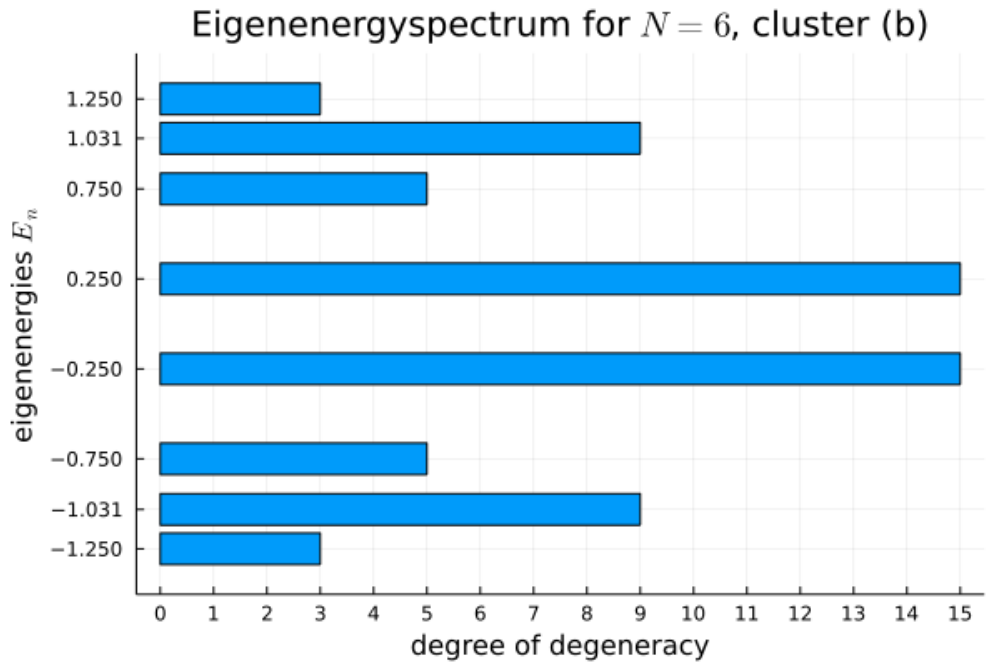

Figure 1.3

b). Spin-correlations for the Kitaev cube ($N = 8$)

A few of the Kitaev clusters for $N = 8$ can be visualized as cubes, e.g.

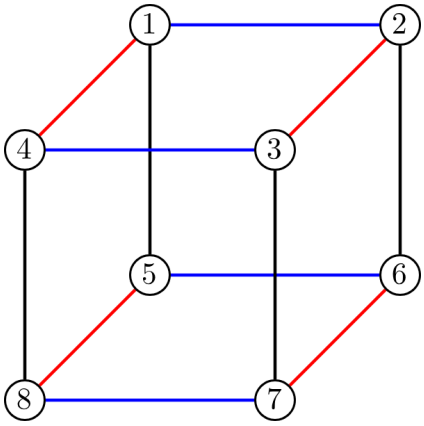

Figure 1.4: One possible Kitaev cube for $N = 8$

Like before, we construct the couplings according to (Fig. 1.4), calculate the Hamilton matrix and now calculate the eigenstates and extract the ground state ψ_g:

```
N = 8
J = 1

Js_N8 = reset_Js(N)

# x links
Js_N8[1][1,4] = J
Js_N8[1][2,3] = J
Js_N8[1][5,8] = J
Js_N8[1][6,7] = J
# y links
Js_N8[2][1,2] = J
Js_N8[2][3,4] = J
Js_N8[2][5,6] = J
Js_N8[2][7,8] = J
# z links
Js_N8[3][1,5] = J
Js_N8[3][2,6] = J
Js_N8[3][3,7] = J
Js_N8[3][4,8] = J

H_N8 = calculate_hamilton_matrix(Js_N8, N)
ψ_N8 = eigvecs(rationalize.(Matrix(H_N8)))
E_N8 = unique(round.(eigvals(rationalize.(Matrix(H_N8))); digits=7));

# ground state
E_N8_g = E_N8[1]
ψ_N8_g = ψ_N8[:,1]
```

We now use again the function from sheet 5[2] to calculate the spin correlation between sites n and m.

```
χs = zeros(8,8)
for n in 1:8
    for m in 1:8
        χs[n,m] = spin_correlation(N, ψ_N8_g, n,m)
    end
end
```

Listing 1: Calculating the spin correlations between all $N = 8$ sites.

To verify, that only the nearest neighbour sites have non-zero spin correlation, we use the helper function

[2]The function is again implemented in `spin_correlation.jl` in the repository.

```
function are_nearest_neighbours(n,m, Js, J)
    if Js[1][n,m] == J || Js[1][m,n] == J ||
        Js[2][n,m] == J || Js[2][m,n] == J ||
        Js[3][n,m] == J || Js[3][m,n] == J
        return true
    end
    return false
end
```

that returns `true` if the sites n and m are nearest neighbours according to the couplings `Js`, and `false` otherwise.

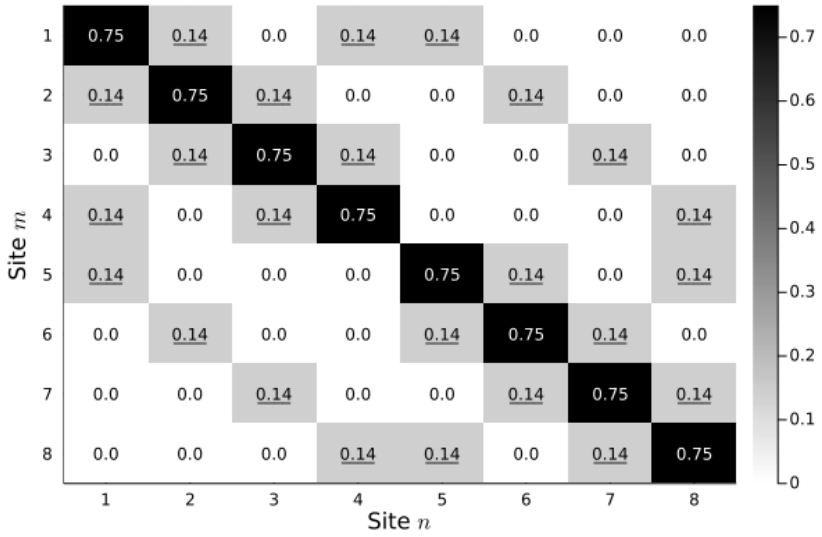

Figure 1.5: Spin correlations $\chi_{n,m}$ for the Kitaev cube in (Fig. 1.4). The values for the nearest neighbour sites are underlined.

As one can see, the underlined values (nearest neighbours) in (Fig. 1.5) correspond exactly to the non-zero spin correlations (except the trivially case $n = m$), and all other values for not-nearest neighbour sites are zero.

2. Entanglement entropy for the one-dimensional Heisenberg model

Consider the one-dimensional Heisenberg model with open boundary conditions:

$$H = -\sum_{i=1}^{N-1} J_i \vec{S}_i \cdot \vec{S}_{i+1}.$$

For an even number of sites N and $J_i < 0$ (anti-ferromagnetic case), the ground state is non-degenerate.

Now divide the system in two parts A and B with sites $1, \ldots, M_A$ and $M_A + 1, \ldots, N$.

a).

a).1. Entanglement entropy

For a state

$$|\psi\rangle = \sum_{i=1}^{\dim_A} \sum_{j=1}^{\dim_B} c_{ij} |i\rangle_A |j\rangle_B$$

the reduced density operator is (with density operator $\hat{\rho} = |\psi\rangle \langle\psi|$)

$$\hat{\rho}_A = \mathrm{Tr}_B(\underbrace{|\psi\rangle \langle\psi|}_{=\hat{\rho}}) = \sum_l \sum_{i,i'} c_{i'l} c^*_{il} |i'\rangle_{AA} \langle i|$$

$$\rightsquigarrow \rho_A = CC^\dagger$$

where C is the matrix $(c_{ij})_{i=1,\ldots,\dim_A; j=1,\ldots,\dim_B}$, because $(\rho_A)_{nm} = \sum_l \underbrace{(C)_{nl}}_{=c_{nl}} \underbrace{(C^\dagger)_{lm}}_{=c^*_{ml}}$.

Then the entanglement entropy is

$$S_e = -\mathrm{Tr}_A[\hat{\rho}_A \cdot \ln \hat{\rho}_A] = -\sum_\alpha w_\alpha \cdot \ln w_\alpha$$

with eigenvalues w_α of ρ_A.
To numerically figure out ρ_A to calculate S_e, we need to find C from a given $|\psi\rangle$.

a).2. Calculating C directly from $|\psi\rangle$

Consider $|\psi\rangle$ is given in the computational basis of the joint system $\{|i\rangle\}_{i=0,\ldots,2^N-1}$ as a vector of coefficients $\psi := \vec{\psi}$ with

$$|\psi\rangle = \sum_{i=0}^{2^N-1} \psi_i |i\rangle_N = \sum_{i=0}^{2^N-1} \psi_i |i_1, i_2, \ldots, i_N\rangle$$

If we now divide the system of N sites in M_A and $M_B = N - M_A$ sites, this can be seen as writing $|i_1, i_2, \ldots, i_N\rangle$ as a product state $|i_1, i_2, \ldots, i_{M_A}\rangle \otimes |i_{M_A+1}, \ldots, i_N\rangle$

$$\begin{aligned} |\psi\rangle &= \sum_{i=0}^{2^N-1} \psi_i \underbrace{|i_1, \ldots, i_{M_A}\rangle}_{=:|j\rangle} \otimes \underbrace{|i_{M_A+1}, \ldots, i_N\rangle}_{=:|l\rangle} \\ &= \sum_{j=0}^{2^{M_A}-1} \sum_{l=0}^{2^{M_B}-1} (C)_{jl} |j\rangle \otimes |l\rangle \end{aligned}$$

with now constructed C:

$$(C)_{jl} = \psi_{dec([j \circ l]_2)} = \psi_{dec([j_1,\ldots,j_{M_A},l_1,\ldots,l_{M_B}]_2)} = \psi_{i(j,l)}, \tag{2.1}$$

where $dec(x)$ is the decimal representation of binary x and $[j \circ l]_2 = [j]_2 \circ [l]_2$ means the concatenation of the bitstrings j and l.

a).3. Implementation of the Entanglement entropy

Before we can calculate the entanglement entropy, we need some helper functions:

```
dec2bin(x, pad=0) = join(digits(x, base=2, pad=pad))
bin2dec(x) = parse(Int, join(reverse(string.(x))), base=2)

function ψ2C(ψ, M_A, M_B)
    C = zeros(2^M_A,2^M_B)
    for j in 0:(2^M_A-1)
        for l in 0:(2^M_B-1)
            C[j+1,l+1] = ψ[bin2dec(dec2bin(j, M_A) * dec2bin(l, M_B)) + 1]
            #println("$j, $l: ", dec2bin(j,M_A),",", dec2bin(l, M_B), " → ", C[
                j+1,l+1])
        end
    end
    return C
end
```

Listing 2: Helper functions to calculate the entanglement entropy

- With `dec2bin()` we can convert a decimal number to a binary string,
- with `bin2dec()` we can convert a binary number string to a decimal number[3], and
- `ψ2C()` can convert a vector of coefficients in the computational basis of the whole system to the representation in the two subsystems A and B by calculating the matrix C according to (Eq. 2.1).

Last, we can calculate the entanglement entropy $S_e(\rho) = S_e(CC^\dagger)$ using

```
function S_e(ρ)
    return -sum(w_a lnw_a.(eigvals(ρ)))
end
```

Listing 3: Calculation of the entanglement entropy S_e of a reduced density operator ρ using the eigenvalues of ρ.

using the function `w_a lnw_a()`, which simply implements $w_\alpha \cdot \ln(w_\alpha)$ and handles the edge case of $w_\alpha \leq \epsilon$ for small $\epsilon > 0$: $\lim_{x \to 0(+\epsilon)} x \cdot \ln(x) = 0$

```
function w_a lnw_a(w_a)
    if w_a <= eps()
        return 0.0 # lim(x→0) x ln(x) = 0
    end
    return w_a * log(w_a)
end
```

[3]Note: We use again the convention of the reversed order of binary digits, therefore the binary number has to be reversed before conversion to a decimal integer.

a).4. One-dimensional Heisenberg model

Like before, we again use the functions from sheet 4 to calculate the ground state of the one-dimensional Heisenberg model for $N = 10$ and $J_i = -1$.
The setup looks again like this

```
J = -1
N = 10

Js = reset_Js(N)

# couplings between (i,i+1) for α=x (1)
for i in 1:(N-1)
    Js[1][i,i+1] = J
end

Js[2] = Js[1] # y
Js[3] = Js[1] # z

H = calculate_hamilton_matrix(Js, N)
ψ_g = eigvecs(rationalize.(Matrix(H)))[:,1];
```

Listing 4: Setup for the 1d Heisenberg model and calculation of the non-degenerate ground state ψ_g

We can now calculate the entanglement entropy as a function of M_A:

```
function S_e_M_A(ψ_g, M_A, N)
    M_B = N-M_A
    # calculate C
    C = ψ2C(ψ_g, M_A, M_B)
    # calculate S_e
    return S_e(C * C')
end
```

This gives for $1 \leq M_A < N = 10$:

$S_e(M_A = 1) = 0.693$, $S_e(M_A = 2) = 0.408$, $S_e(M_A = 3) = 0.726$, $S_e(M_A = 4) = 0.492$, $S_e(M_A = 5) = 0.738$,
$S_e(M_A = 6) = 0.492$, $S_e(M_A = 7) = 0.726$, $S_e(M_A = 8) = 0.408$, $S_e(M_A = 9) = 0.693$

b).

The maximal possible entanglement entropy is given by the dimensions of the subsystems as

$$S_e^{max}(M_A) = \log(\min(\dim_A, \dim_B)) = \log(\min(2^{M_A}, 2^{M_B}))$$

with $M_B = N - M_A$.

We can now plot $S_e(M_A)$ and $S_e^{max}(M_A)$[4] together:

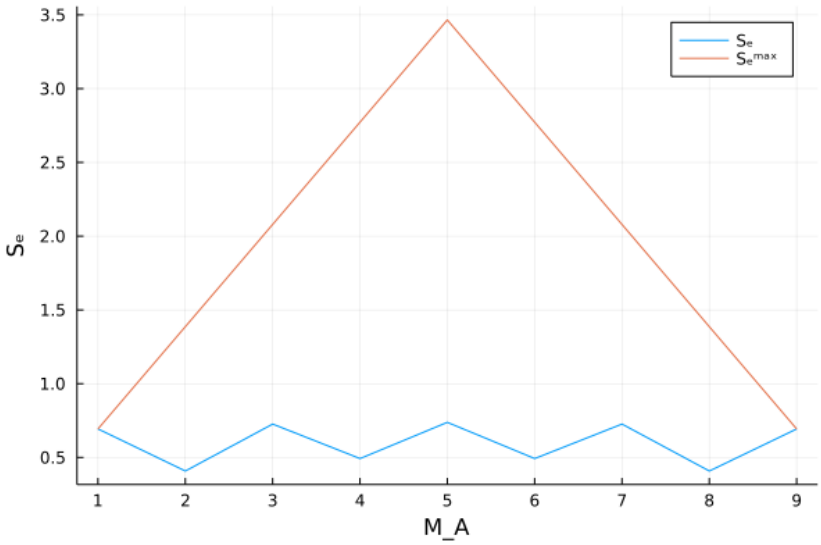

Figure 2.1

c).

For $J_i = -1, N = 6$, and $M_A = 3$ the entanglement entropy $S_e = 0.711\ldots$ is much lower than the maximal value $S_e^{max} = 3\ln(2) = 2.079\ldots$.
We now want to find a set of values for $-1 \leq J_i < 0$, such that $S_e > 1.5$.

One approach could be to use random values, another would be a brute-force search of 'all' possible values for J_i, both running until some combination is found s.t. $S_e > 1.5$.

Both approaches have the disadvantage of having a huge expected computation time, especially the brute-force approach needs to check $\left(\frac{1}{\Delta J}\right)^{N-1}$ possible combinations of values $\{-1, -1+\Delta J, -1+2\Delta J, \ldots, 0\}$ and therefore is not a good approach as the stepsize, ΔJ, needs to be quite small.

Instead we can combine the brute-force method with a clever manual choice of parameters and replace the 'matching' condition $S_e > 1.5$ with the search for the maximum value of S_e for the given set of possible parameters.

1. Choose K (K not too big) reasonable possible different values for J_i: $\mathcal{J} = \{a_1, a_2, \ldots, a_K\}$, $-1 \leq a_i < 0$
2. Calculate all combinations of the possible values: $\mathcal{J}^{N-1} = \{a_1, a_2, \ldots, a_K\}^{N-1}$
3. Keep track of the maximal value of S_e and the corresponding set of J_i
4. For each set of $(J_1, \ldots, J_{N-1}) \in \mathcal{J}^{N-1}$:
   (a) calculate the Hamilton matrix H
   (b) calculate the ground state ψ_g and C for given M_A
   (c) calculate the entanglement entropy S_e
   $\rightsquigarrow$ if $S_e > S_{max}$: use as new maximum

[4] `Semax(M_A, M_B) = log(minimum([2^M_A, 2^M_B]))`

Afterwards, the maximal possible value of S_e (and the corresponding set of J-values) for the given choosen parameters $\mathcal{J}$ is found. It can be manually analysed for patterns to make a reasonable choice of new parameters. With this approach, we only have to check K^{N-1} possible values before narrowing down the reasonable values for the next iteration.

For a given set of possible J-values `J_possible`, we can calculate all possible combinations using

```
Jis = collect(Iterators.product([J_possible for i in 1:5]...))
```

After that, we find the maximum according to the algorithm described above:

```
N6 = 6
N=N6

Smax = 0
jmax = 0
@showprogress for j in 1:length(Jis)
    Ji = Jis[j]

    # set up the Hamilton matrix and calculate the ground state
    Js_N6 = reset_Js(N6)
    # couplings between (i,i+1) for α=x (1)
    for i in 1:(N6-1)
        Js_N6[1][i,i+1] = Ji[i]
    end
    Js_N6[2] = Js_N6[1] # y
    Js_N6[3] = Js_N6[1] # z
    H_N6 = calculate_hamilton_matrix(Js_N6, N6)
    ψ_N6_g = eigvecs(rationalize.(Matrix(H_N6)))[:,1]

    # calculate C and S_e
    C_N6 = ψ2C(ψ_N6_g, 3,3)
    S = S_e(C_N6 * C_N6')

    # keep the maximum
    if S > Smax
        Smax = S
        jmax = j
    end
end
println("Maximum S_e: $Smax @ $(Jis[jmax])")
```

A snippet from the manual search in detail:

#	J_possible	combinations	Smax	$(J_1, J_2, J_3, J_4, J_5)$
1	$[-1, -.75, -.5, -.25, 0]$	3125	1.242	$(0.0, -0.25, -1.0, -0.25, 0.0)$
2	$[-1, -.25, -.125, 0]$	1024	1.3558	$(0.0, -0.125, -1.0, -0.125, 0.0)$
3	$[-1, -.25, -.125, -.0625, 0]$	3125	1.3796	$(0.0, -0.0625, -1.0, -0.0625, 0.0)$
4	$[-1, -.25, -.125 : .025 : 0...]$	32768	1.385	$(0.0, -0.025, -1.0, -0.025, 0.0)$
5	$[-1, -.25, -.025 : .0125 : 0...]$	3125	**1.7575**	$(-0.0125, -0.25, -1.0, -0.25, -0.0125)$
6	$[-1, -.25, -.0125 : .00625 : 0...]$	3125	**1.8979**	$(-0.00625, -0.25, -1.0, -0.25, -0.00625)$
7*	[5]	(441)	**1.935**	$(-0.0005, -0.25, -1.0, -0.25, -0.0005)$
8*	[6]	(3969)	**2.0489**	$(-5e{-}5, -0.125, -1.0, -0.125, -5e{-}5)$
9*	[7]	(1521)	**2.073**	$\underline{(-5e{-}5, -0.05, -1.0, -0.05, -5e{-}5)}$

All J-values after iteration 5 have an entanglement entropy $S_e > 1.5$. In the last three iterations, some J_i values were fixed and only some were varied. The final value of $S_e = 2.073$ is very close to the maximum value of $S_e^{max} = 3\ln(2) = 2.079\ldots$.[8]

[5] Fixed values for J_2, J_3, J_4, values for $J_{1,5} = [-.01 : .0005 : 0...]$

[6] Fixed value for $J_3 = -1$, values for $J_{1,5} = [-.001 : .00005 : 0...]$, for $J_{2,4} = [-.5, -.25, -.125]$

[7] Fixed value for $J_3 = -1$, values for $J_{1,5} = [-.0001 : .00005 : 0...]$, for $J_{2,4} = [-.125 : 0.0125 : 0.025...]$

[8] This set of J-values was found with this method in only 52223 possible combinations, which is less than needed in the pure brute-force approach with precision $\Delta J = 0.125$ (9 possible values per coupling), therefore our approach is better (faster and more precise).

Plotting $S_e(M_A)$ for the last 5 iterations, we get:

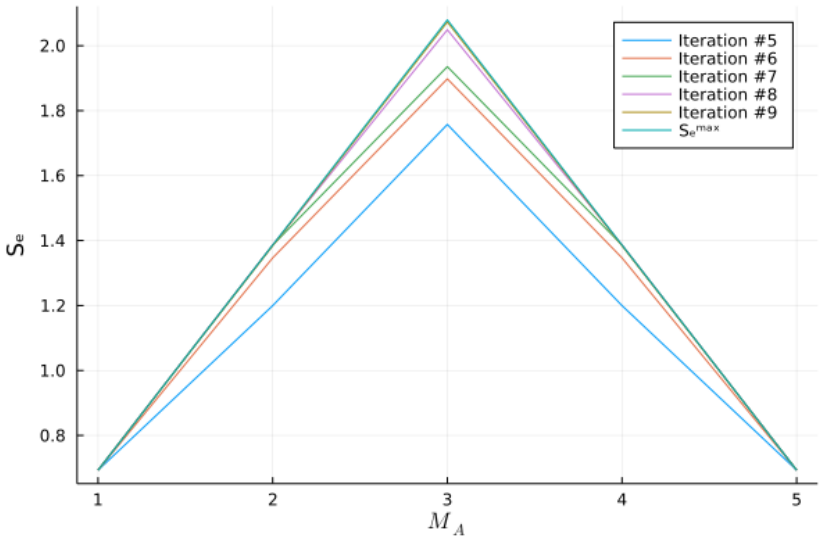

Figure 2.2