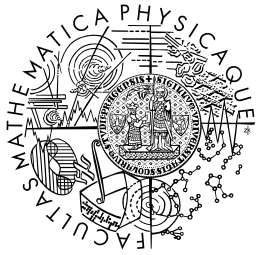

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Igor Bajo

Measures of quantum entanglement

Institute of Particle and Nuclear Physics

Supervisor of the bachelor thesis: prof. RNDr. Pavel Cejnar, Dr., DSc.
Study programme: Physics
Study branch: General Physics

Prague 2022

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank prof. RNDr. Pavel Cejnar, Dr., DSc. for constructive critique and usefull ideas.

Also I would like to thank my girlfriend for help with a grammar.

Title: Measures of quantum entanglement

Author: Igor Bajo

Department: Institute of Particle and Nuclear Physics

Supervisor: prof. RNDr. Pavel Cejnar, Dr., DSc., Institute of Particle and Nuclear Physics

Abstract: We present measures of bipartite entanglement in the multiparticle spin-1/2 systems. We also touch on the phenomenon of Quantum Phase Transitions and spin coupling. In the end we create a Python code for graphical visualisation of entanglement in relation to ground state energy behaviour.

Keywords: Von Neumann Entropy, Entanglement of Formation, Quantum Phase Transition, Lipkin Hamiltonian, collective spin models

Contents

Introduction

The aim of this work is to introduce the topic of quantum entanglement measures in multiparticle systems. Instead of focusing on the historical background we attempt at creating a code which will allow us to explore the entanglement behaviour under different conditions.

We will necessarily go through the description of quantum statistical ensembles as well as through the formalism needed to capture quantum phenomena. We will focus on a two-level system consisting of spin 1/2 particles also known as *qubits*. This kind of research is becoming more and more relevant due to its importance in quantum computing.

As our "playground" we opt for the so-called *Lipkin model*. Mostly because of its mathematical elegance and simplicity. The model elaborates only on a spinspin interaction, so we will take a deeper dive into the spin coupling and the properties of spin operators.

Another phenomenon relevant to our line of research is that of *Quantum Phase Transition*. Unlike 'classical' phase transitions, which occur due to changes in thermal parameters, Quantum phase transitions are changes in the quantum ground state energy behaviour, taking place at zero temperature. We will explore in depth only two kinds of bipartite entanglements — *Von Neumann Entropy* and *Entanglement of Formation*. Both characterise ensemble entanglement but differ in the conditions under which those two measures can be utilised.

The goal of the present work is to graphically examine the entanglement behaviour as a function of external parameters. The system's entanglement peak should match the ground state Quantum Phase Transition visible in Hamiltonian's spectra.

Let us also note, that we will solve all the problems at hand numerically since at this level of complexity dealing with the analytical solution is out of our reach (except for a few special cases). The code is added as an attachment to this work.

1. System description

1.1 Lipkin-Meshkov-Glick Hamiltonian

Spin represents the internal angular momentum of particles. As all quantum observables, the components of spin 1/2 particles (qubits) are represented by matrices. These are 2x2 Hermitian matrices, denoted as $\hat{\sigma}_k$ (Pauli matrices). The spin operator of a single qubit is then a vector:

$$\hat{\mathbf{S}} = \frac{1}{2}(\hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z) \tag{1.1}$$

We can consider spin to be a form of an angular momentum since its components are satisfying commuting relations of angular momentum [10].

$$[\hat{S}_i, \hat{S}_j] = i\epsilon_{ijk}\hat{S}_k \tag{1.2}$$

$$[\hat{S}^2, \hat{S}_z] = 0 \tag{1.3}$$

In systems consisting of multiple particles, we can speak of their collective spin (isospin). In such a case, we have a single spin operator $\hat{\mathbf{J}}$, which is obtained by summing over all spin operators of a single particles $\hat{\mathbf{S}}_\mathbf{n}$:

$$\hat{\mathbf{J}} = \sum_n \hat{\mathbf{S}}_\mathbf{n} \tag{1.4}$$

Lipkin-Meshkov-Glick Hamiltonian or just Lipkin Hamiltonian describes multi-particle systems via the total spin operator's components $\hat{J}_n$. Such systems are called fully connected because we consider spin interaction between all qubits mutually. These interaction is of an infinite range (in oppose to *Ising model* for example [1]). Lipkin models are rather a class of Hamiltonians. Using up to second power of $\hat{J}_n$ operators, we can express Lipkin models as follows:

$$\hat{H} = A\hat{J}_z + \sum_{i,j} \frac{\theta_{ij}}{2N}(\hat{J}_i\hat{J}_j + \hat{J}_j\hat{J}_i) \tag{1.5}$$

In this form $\hat{H}$ conserves the total squared spin $\hat{\mathbf{J}}^2$. Big advantage of such models is that using them we can reduce the number of *degrees of freedom* from N (each qubit contributes by its spin) to 1 (collective spin). This singled degree of freedom is thus encoded in the collective spin of the system. Lipkin Hamiltonian is also fully symmetric, meaning the system is invariant under exchange of any pair of qubits. Throughout this study we will work with a Hamiltonian in the following form:

$$\hat{H} = \hat{J}_z - \frac{\lambda}{N}[\hat{J}_x + \chi(\hat{J}_z + \frac{N}{2}\cdot\mathbb{I})]^2 \tag{1.6}$$

This will provide the desired behaviour of the Quantum Phase Transition, which will be examined in depth in following chapters. Parameter λ represents the

interaction parameter. It is scaled by the particle number N, in order to prevent the dominance of the interaction term in the total energy for higher number of qubits.

Let us now focus on the behaviour of the $\hat{J}_n$ operators. Consider having only one qubit. States of the particle $\{|jm\rangle\}$ will make basis vectors of the Hilbert space belonging to the system. Here the j represents eigenvalue of $\hat{J}^2$ operator whereas m is the eigenvalue of $\hat{J}_z$:

$$\begin{aligned} \hat{J}^2 &= \hat{J}_x^2 + \hat{J}_y^2 + \hat{J}_z^2 \\ \hat{J}^2 \left|jm\right\rangle &= j(j+1) \left|jm\right\rangle \\ \hat{J}_z \left|jm\right\rangle &= m \left|jm\right\rangle \end{aligned} \tag{1.7}$$

We can think of j as the size of the spin and of m as the projection of the spin onto the z-axis. It should be unsurprising then, that in our one spin-1/2 particle system, the only possible value of j is $j = 1/2$. For m we obtain $m \in \{-1/2, 1/2\}$. Thus we end up with two possible states, often called *up* and *down*.

Finally , we would like to find the matrix forms of the $\hat{J}_n$ operators. Easiest to create is the $\hat{J}_z$ matrix. Because we are in the $\{|jm\rangle\}$ basis. $\hat{J}_z$ is going to be a diagonal matrix consisting of all possible m values. To find the rest let us introduce the $\hat{J}_\pm$ operators:

$$\begin{aligned} \hat{J}_\pm \left|jm\right\rangle &= \alpha_m^\pm \left|jm \pm 1\right\rangle \\ \alpha_m^\pm &= \sqrt{j(j+1) - m(m \pm 1)} \\ \hat{J}_x &= \frac{\hat{J}_+ + \hat{J}_-}{2} \\ \hat{J}_y &= \frac{\hat{J}_+ - \hat{J}_-}{2i} \end{aligned} \tag{1.8}$$

We see that $\hat{J}_\pm$ applied on $|jm\rangle$ state is returning a new state $|jm \pm 1\rangle$. This is going to become important in later chapters where we will focus more on the practical part of Hamiltonian creation in a multiparticle system.

1.2 Spin Coupling

Above we introduced the meaning and some properties of spin operators $\hat{\mathbf{J}}$ Now, in the case of two systems (A and B), the resulting system will have Hilbert space as follows:

$$\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B \tag{1.9}$$

There exist two ways describing the system - meaning we have to find the *complete set of commuting operators* and the basis of $\mathcal{H}$. The first way of doing this, is in *separable basis*. The complete set of operators is then:

$$\{\hat{J}^{(A)2}, \hat{J}_z{}^{(A)}, \hat{J}^{(B)2}, \hat{J}_z{}^{(B)}\} \tag{1.10}$$

with the corresponding basis:

$$\Delta \equiv \{|j_A, m_A\rangle, |j_B, m_B\rangle\} \tag{1.11}$$

Basis Δ consist of state vectors of qubits - A and B. Values of $j_{A,B}$ are eigenvalues of the spin operator $\hat{J}^{(A,B)2}$ and $m_{A,B}$ are eigenvalues of $\hat{J}_z{}^{(A,B)}$ operators. The second option is *coupled basis*:

$$\{\hat{J}^{(A)2}, \hat{J}^{(B)2}, \hat{J}_z, \hat{J}^2\} \tag{1.12}$$

$$\Gamma \equiv \{|j, j_A, m_A, m\rangle\} \tag{1.13}$$

Now we have the last two operators in the set without an index. Those belong to the whole coupled system, as well as j and m in the basis ket-vectors. The obvious question would then be, what are all the possible values of j and m? A definite requirement is for Γ and Δ to have the same dimension. The general rule for coupling is ([10]) is as follows:

$$\begin{aligned} &|j_A - j_B| \leq j \leq |j_A + j_B| \\ &m \in \{j, j-1, ..., -j\} \end{aligned} \tag{1.14}$$

Thus, we end up with two different basis of the same dimension describing the same system. The last thing we need to do is find the unitary transformation from Γ to Δ. Or, in the other words, we have to express each state vector from coupled basis as a linear combination of vectors from separable basis. The task can be performed by using *Clebsch-Gordan coefficients*, or just CG-coefficients $C^{j,m}_{j_A,m_A,j_B,m_B}$:

$$|j, j_A, j_B, m\rangle = \sum_{m_A=-j_A}^{j_A} \sum_{m_B=-j_B}^{j_B} C^{j,m}_{j_A,m_A,j_B,m_B} |j_A, m_A\rangle |j_B, m_B\rangle \tag{1.15}$$

If the conditions (1.14) or $m \neq m_A + m_B$ are not being satisfied, then the belonging CG-coefficient is equal to zero. Larger portion of the technicalities behind this calculation is kept for further discussion in section 3.1, where we will obtain CG-coefficient iteratively.

1.3 Quantum Phase Transition

Phase transitions, as conventionally understood, are changes in the physical properties of matter induced by changes in thermodynamic parameters. *Quantum Phase Transition* also occurs due to changes in external parameters, but *theoretically* at a temperature of absolute zero. Which means this parameters can not

be thermal ones. In the system described by Hamiltonian (1.6) this non-thermal parameter is captured by λ which represents the strength of spin-spin interaction.

Transition of phases in this context means the system's transition between quantum states (extensively covered in [14]). Generally, we say that each quantum state corresponds with the energy of the system (more accurately: the eigenvector of the Hamiltonian represents the given state and its eigenvalue represents the energy). When the temperature is zero, the ensemble will be necessarily in the state with the lowest energy, called the *ground state*. We define the Quantum Phase Transition as a nonanalyticity in the ground states's energy. This can occur only in the thermodynamic limit, where the number of particles tends toward infinity. Practically, we will instead observe QPT of the *first order* or *second order*. The former means that the energy has a singularity in its first derivative, whereas in the latter the singularity appears only in a second derivative of energy (seen as a function of λ).

Quantum phase transitions are related to Hamiltonian's *parity*. In the model, defined by (1.6), we expect to see the second order QPT for $\chi = 0$ [12]. In such a case, the Hamiltonian *conserves* its parity as $\hat{\Pi} = (-1)^{\hat{J}_z}$. If we set the parameter to be $\chi \neq 0$, we expect to observe first order QPT, due to the parity breaking. The transition should occur for a critical value of the interaction parameter λ_c :

$$\lambda_c = \frac{1}{1+\chi^2} \tag{1.16}$$

The Quantum Phase Transition also corresponds with the maximal entanglement of the system. We are going to choose the ensemble parameters N, χ and we will observe how the entanglement measure E change in relation to the interaction parameter λ. The energy spectrum of the system will be plotted as a function of λ. The phase transition and the maximum of the entanglement should occur for the same value of λ. Although, as in [11], the correspondence between the Quantum Phase Transition and entanglement is not one to one.

It is worth noting that energies of our Lipkin model Hamiltonian usually do not cross (but for example energies with different parity can cross for $\chi = 0$). Graphically it will appear as if energies belonging to different states do in fact cross, in reality, however, we will instead observe the so called *avoided crossing*. At the point of crossing exist a tiny energy gap, which is too small for this work's numerical observational capacity. What looks like sharp crossing is then a first order QPT, where the ground state energy instantaneously decreases in its value, creating a singularity in its first derivative.

2. Entanglement Measures

Here, we will focus only on the so- called bipartite entanglement. As the name suggests, it describes a 'connection' between two parts of the ensemble. There exists a rather high number of possible measures of entanglement but we will explore only two of them.

The first is the *Von Neumann Entropy*. Application wise we will have to work in a space consisting of quite the small number of particles N. This is due to the high numerical precision required of our technique throughout the computation. Using Von Neumann Entropy we will be able to calculate entanglement between two subsystems (denoted simply as A and B) consisting of N - M and M particles. Let us also note that the system partition does not represent a real physical barrier or separation, but rather our choice of mathematical description (as is usually the practice in all of physics).

The second is *Entanglement of Formation*. We will be limited to the observation of a qubit pair's entanglement in the larger system only. Main difference is that our qubit pair will be in the so-called *mixed state*. Vaguely speaking, the entanglement of the pair is going to be diluted in the otherwise bigger, tangled ensemble. In comparison, VNE works with the pair of subsystems (A and B) occupying the *whole* system, meaning, that the pair is in a *pure state*.

Let us step up form intuition to strict definitions. But first we have to introduce the required mathematical apparatus.

2.1 Quantum Statistical Physics

The multiparticle system is well described as quantum statistical ensemble. The statistical ensemble consists of copies of the system, each in a different state. Because we are dealing with statistics and quantum objects, it should be no surprise, that the keyword here is *probability*.

We have a set of states $\{|\psi_n\rangle\}$. $\{p_n\}$ is then a set of probabilities belonging to the set of states. The probability distribution in the entire Hilbert space is captured by the *density operator* as follows:

$$\hat{\rho} = \sum_n p_n \left|\psi_n\right\rangle \left\langle\psi_n\right| \tag{2.1}$$

It is worth mentioning here that we are *not* generally assuming orthogonality of the states. What is very important is the distinction between *pure* and *mixed* states of the system. If we can express the density operator as:

$$\hat{\rho} = \left|\psi\right\rangle \left\langle\psi\right| \tag{2.2}$$

we can say that the system is in a *pure* state. On the other hand, if we have to use (2.1) than we speak about a *mixed* state. There exist also another option,

how to distinguish a mixed state from a pure one. Suppose that the matrix $\hat{\rho}$ is expressed in a basis, where it is diagonal. In such a case we can say, the system is in a mixed state, if:

$$\operatorname{Tr} \hat{\rho}^2 < 1 \tag{2.3}$$

and that the system is in a pure state if:

$$\operatorname{Tr} \hat{\rho}^2 = 1 \tag{2.4}$$

Since we are interested in bipartite entanglement let us have a Hilbert space consisting of two subsystems:

$$\begin{aligned} \mathcal{H} &= \mathcal{H}_1 \otimes \mathcal{H}_2 \\ \mathcal{H}_1 &= span\{|\psi_{1n}\rangle\} \\ \mathcal{H}_2 &= span\{|\psi_{2k}\rangle\} \end{aligned} \tag{2.5}$$

Generally, then, we can then write any state of $\mathcal{H}$ as follows:

$$|\Theta\rangle = \sum_{ij} \gamma_{ij} |\psi_{1i}\rangle |\psi_{2i}\rangle \tag{2.6}$$

and the states of its subsystems as the linear combination of the belonging basis vectors :

$$\begin{aligned} |\eta_1\rangle &= \sum_i \alpha_i |\psi_{1i}\rangle \\ |\eta_2\rangle &= \sum_j \beta_j |\psi_{2j}\rangle \end{aligned} \tag{2.7}$$

If we are able to find coefficients α_i and β_j which satisfies $\gamma_{ij} = \alpha_i \beta_j$, we say that this state is *separable*. But for most of the states from $\mathcal{H}$, it is going to be impossible to satisfy the identity above, which means that $\gamma_{ij} \neq \alpha_i \beta_j$. In this case we call the state *entangled* - we really can not say that any of the systems is in a *pure* state. As such entanglement represents a quantum correlation between different parts of the ensemble.

Information about the subsystems is provided by the *reduced density operator*, that can be calculated by tracing the density operator $\hat{\rho}$ of the entire Hilbert space $\mathcal{H}$ over all subsystem basis vectors:

$$\hat{\rho}_{1,2} = \sum_k \langle \psi_{2,1k}| \hat{\rho} |\psi_{2,1k}\rangle = \operatorname{Tr}_{2,1}(\hat{\rho}) \tag{2.8}$$

Here, the indices are marking the subsystem to which the given reduced density operator belongs. The last identity will play a major role in later computations of bipartite entanglement.

2.2 Von Neumann Entropy

In information theory exists the concept of *Shannon information entropy*. Let us have a list of n discrete mutually exclusive events $\{a, b, ..., z\}$ with a list of their probabilities $\{p_a, p_b, ..., p_z\}$. Shannon entropy is then:

$$S = -\sum_{1}^{n} p_i \ln p_i \tag{2.9}$$

The meaning is thus quite intuitive if we think of S as of the measure of 'surprise' - if there is only one outcome a with $p_a = 1$, then $S = 0$ and we 'cannot be surprised'. On the other hand, if there is n possible events with evenly distributed probabilities $p_i = 1/n$, then entropy is reaching its maximum at $S = \ln n$ and the outcome reaches a maximum randomness. To be able to apply such concept to our quantum ensemble, we will need to work a little bit with a density operator.

Suppose we have bipartite system, with Hilbert space $\mathcal{H}$, in the state $|\Theta\rangle$ and we know the basis of its subsystems as in (2.5), (2.6), (2.7). After obtaining reduced density operators $\hat{\rho}_1$ and $\hat{\rho}_2$, we need to express them in the new basis. In this new basis both operators will be diagonal, and they will share the same eigenvalues. This is known as *Schmidt decomposition*. Let us then denote the new basis as:

$$\begin{aligned} span\{|\chi_{1n}\rangle\} &= \mathcal{H}_1 \\ span\{|\chi_{2k}\rangle\} &= \mathcal{H}_2 \end{aligned} \tag{2.10}$$

and $\{\rho_i\}$ is set of shared eigenvalues of $\hat{\rho}_1$, $\hat{\rho}_2$. Hence, we can write:

$$|\Theta\rangle = \sum_i \sqrt{\rho_n}\, |\chi_{1i}\rangle\, |\chi_{2i}\rangle \tag{2.11}$$

Finally we can define the Von Neumann Entropy:

$$E = S(\hat{\rho}_{1,2}) = -\sum_i \rho_i \ln \rho_i = -\operatorname{Tr}\Big\{\hat{\rho}_{1,2} \ln \hat{\rho}_{1,2}\Big\} \tag{2.12}$$

Because of the their shared spectrum $\{\rho_i\}$, we are free to choose between a reduced density operator $\hat{\rho}_1$ or $\hat{\rho}_2$ for the calculation of E. Intuitively it makes only sense, because E is telling us about the mutual correlation between parts of the composite system.

2.3 Entanglement of Formation

The previous method shows us how to calculate entanglement measure for a system in the pure state. In comparison, the Entanglement of Formation defines entanglement between a pair of quibits in the generally mixed state. As noted in [7], the Entanglement of Formation is the amount of pure state entanglement

needed to create a single copy of the mixed state. Formally we can derive EoF's definition by expanding the idea behind Von Neumann Entropy.

Let us have a mixed state described by $\hat{\rho}$ in the form of (2.1). Now $\hat{\rho}$ is being expressed as a sum of pure states $|\Phi_k\rangle$, with density operators $\hat{\rho}_k = |\Phi_k\rangle \langle\Phi_k|$. Entanglement of Formation is then defined as the minimal average possible entropy of the pure state decomposition:

$$\mathsf{E}_f(\hat{\rho}) = \min \sum_k p_k S(\hat{\rho_k}) \tag{2.13}$$

Since we did not need the states to be orthogonal in (2.1), we are now dealing with infinity many possible pure state decompositions of $\hat{\rho}$! Examples are to be found in [2].

We will than chose approach using a quantity called *concurrence* as introduced in [8]. But first lets define operation called *spin flip*:

$$\hat{\rho}^+ = (\hat{\sigma_y} \otimes \hat{\sigma_y})\hat{\rho}^*(\hat{\sigma_y} \otimes \hat{\sigma_y}) \tag{2.14}$$

where $\hat{\sigma_y}$ are Pauli matrices and the symbol * over the density operator denotes complex conjugation. Concurrence is then:

$$C = \max\{0, \lambda_1 - \lambda_2 - \lambda_3 - \lambda_4\} \tag{2.15}$$

where $\{\lambda_i\}$ stands for eigenvalues of $\rho\rho^+$ operator in decreasing order. Once we calculate C we can finally obtain Entanglement of Formation:

$$\mathsf{E}_f(\hat{\rho}) = h\left(\frac{1 + \sqrt{1 - C(\hat{\rho})^2}}{2}\right) \tag{2.16}$$

where $h(x) = -x \ln x - (1-x)\ln(1-x)$. Observe this function has a domain $D_h = (0, 1)$, although it is easy to see that $\lim_{x\to 0} h(x) = \lim_{x\to 1} h(x) = 0$, meaning that $C = 0$ is not an issue.

Because we choose Lipkin Hamitlonian with N particles, we will use *rescaled concurrence* (as in [5]):

$$C_R = (N-1)C \tag{2.17}$$

Note that this section is a mere introduction to the theoretical background of entanglement measures. The goal of this study is to incorporate concepts above into the Lipkin model. Thus we will have to figure out, how to work with reduced density operator of a qubit pair in a mixed state.

So in the following chapters we are going to explore the used methodology as well as the challenges that might be met when solving problem programmatically.

3. Methodology

3.1 Von Neumann Entropy Methodology

Entanglement between up to three particles was the outcome of the work [2]. The present thesis is trying take on and expand the concepts presented there. Our goal is to explore the behaviour of Von Neumann Entropy in ensembles of up to a few dozen qubit.

One of the most challenging tasks we can perform on a computer is matrix diagonalization. The computational complexity is about $O(k^3)$, that is to say the time we need to finish the task grows as a cube of the size k, where k is the rank of our matrix. We also need to take in account, that the dimension of Hilbert space in N-qubit ensemble is growing exponentially as 2^N. Meaning, that if we would reach 30 particles, we would need to diagonalize a matrix with the size of around $10^9 \times 10^9$.

It is known, that the *ground state* will be a linear combination of the $|j, m\rangle$ states with a maximum of j ([5]). Since we are interested only in a bipartite entanglement the subsystems A and B also need to be in the maximum j_A and j_B states. This can help us significantly.

We can perform spin coupling iteratively - in a from of *tree diagram* [13]. The *leaves* of such a diagram represent the individual qubits. The nodes closest to the leaves would represent the qubit pairs. The further nodes would represent the coupled pairs. The nodes closest to the root represent the subsystems A and B and the root node stands for the entire system. The full Hilbert space $\mathcal{H}$ is then spanned by $\{|j, j_A, ..., j_Z, m\rangle\}$ (coupled basis) and its dimension is 2^N. The terms j_X in the basis ket-vectors are j values of the node (subsystem) X.

Now we will focus only on the maximal j subspace $\mathcal{H}_J$ and in the same time, we will completely dismiss, what are the values of $j_A, j_B...j_Z$. Obeying the rule (1.14) for possible values of m, we will end up with only *N+1* states. Those are known in the literature as *Dicke states*, and are denoted as $|J, M\rangle$. We are perhaps adding now into the confusion between j , J , $\hat{J}$ and $\hat{\mathbf{J}}$, but because $J = N/2$ always, the states herein will receive the abbreviation $|M\rangle$.

The dimension of full Hilbert space is $dim(\mathcal{H}_J) = 2J + 1$, the dimension of its subsystems is $dim(\mathcal{H}_{J_{B,A}}) = 2J_{B,A} + 1$. It is important to stress out that $dim(\mathcal{H}_J) \neq dim(\mathcal{H}_{J_{B,A}}) + dim(\mathcal{H}_{J_{B,A}})$! The inequality would turn into an equality only if the full Hilbert space would be the direct tensor product of its subspaces. Not considering $j_A, j_B...j_Z$ terms in $\{|j, j_A, ..., j_Z, m\rangle\}$ leads to a loss of information about states, which would be otherwise distinguishable (as an exercise, dear reader can try to figure out all 32 states $\{|j, j_A, j_B, m\rangle\}$ in 2/2 partition).

After we generate the basis states of coupled basis $\{|M\rangle\}$, it holds out that:

$$\hat{J}_- |M\rangle = \alpha_m^- |M - 1\rangle \tag{3.1}$$

Of course we still need to arrive at a matrix form of the Hamiltonian. So we have to perform a 'vectorization' of the very symbolic $|M\rangle$ states. Because those are creating an orthogonal basis of $\mathcal{H}_J$, we can simply attribute each state to a basis vector. So $|M\rangle$ becomes for us $(1, 0, 0...)$, $|M - 1\rangle$ is $(0, 1, 0...)$ and so on until

we reach $|M - N/2\rangle$ represented as $(0, 0, ..., 1)$. If we take the resulting vector from (3.1) for each of our Dicke states, we obtain the columns of $\hat{J}_-$ operator in a matrix form! Then if we perform the Hermitian transposition of $\hat{J}_-$, we end up with $\hat{J}_+$.

Those two matrices are sufficient enough to create a matrix form of $\hat{J}_x$ and $\hat{J}_y$. $\hat{J}_z$ is simply a diagonal matrix created out of all M values. Meaning that we can put all the pieces together and finally obtain Hamiltonian as defined by (1.6) (nevertheless the interaction and the anisotropy parameters have to be chosen).

Now we are ready to find the ground state. Please note, once more, that the states $|M\rangle$ are *not* the eigenvectors of $\hat{H}$! They are only the basis vectors of our Hilbert space and the eigenvectors are going to be their linear combination.

As the prefix 'eigen' suggests, we are going to diagonalize the matrix $\hat{H}$, which means we have to solve the eigenproblem and as a result obtain a set of eigenvalues and eigenvectors. The former represents the system's energy, the latter all the possible states of the system.

By picking the one with the lowest energy we arrive at the desired ground state denoted here as $|gs\rangle$. The corresponding density operator will be:

$$\hat{\rho}_{gs} = |gs\rangle \langle gs| \tag{3.2}$$

Hence to be able to calculate $\hat{\rho}_{A,B}$, we need to be able to express ground state in separated basis, as in (1.15). The Clebsh-Gordan coefficionts can be obtained recursively as follows [10]:

$$\begin{aligned}
\Lambda &= \sqrt{\frac{j_A(j_A+1) - m_A(m_A-1)}{j(j+1) - m(m-1)}} \\
\Xi &= \sqrt{\frac{j_B(j_B+1) - m_B(m_B-1)}{j(j+1) - m(m-1)}} \\
C^{j,m}_{j_A,m_A,j_B,m_B} &= \Lambda \cdot C^{j,m-1}_{j_A,m_A-1,j_B,m_B} + \Xi \cdot C^{j,m-1}_{j_A,m_A,j_B,m_B-1} \\
C^{j,j}_{j_A,j_A,j_B,j_B} &= 1
\end{aligned} \tag{3.3}$$

Then we can express ground state as:

$$|gs\rangle = \sum_k \mu_k |M_k\rangle = \sum_k \mu_k \left(\sum_{nm} C_{nm} |M^n_A\rangle |M^m_B\rangle) \right) = \sum_{ij} \alpha_{ij} \left|M^i_A\right\rangle \left|M^j_B\right\rangle \tag{3.4}$$

Here we used abbreviation $C_{nm} = C^{J,M_k}_{J_A,M^n_A,J_B,M^m_B}$. The reduced density operator of subsystem A, for example, is then going to be:

$$\hat{\rho}_A = \sum_k \left\langle M^k_B \right| \hat{\rho}_{gs} \left|M^k_B\right\rangle = \sum_{ii'} \left(\sum_j \alpha_{ij}\alpha_{i'j}) \right) \left|M^i_A\right\rangle \left\langle M^{i'}_A\right| \tag{3.5}$$

Because $\hat{\rho}_A$ is a real symmetric matrix, the *Schmid decomposition* is equivalent to its diagonalization. By that we yield the eigenvalues $\{\rho_i\}$. Finally using (2.2) we calculate the *Von Neumann Entropy*.

Note that it really does not matter, if we decide to use matrix $\hat{\rho}_A$ or $\hat{\rho}_A$. Nonzero eigenvalues are going to be the same.

Algorithmically (code is in the attachments) we can describe the whole procedure as:

- choose N, γ, λ and a partition
- generate *coupled* basis of $\mathcal{H}_J$
- generate *separated* basis $\mathcal{H}_{J_A}$ and $\mathcal{H}_{J_B}$
- 'vectorize' the $|j, m\rangle$ states
- find out all relevant CG coefficients
- create Hamiltoninan $\hat{H}$ as in (1.6) using $\hat{J}_{\pm}$ operators on the coupled states
- diagonalize $\hat{H}$, choose the state with the lowest energy $|gs\rangle$
- express $|gs\rangle$ in the *separated* basis
- create the density operator $\hat{\rho}_{gs}$
- create the reduced density operator $\hat{\rho}_{A,B}$
- diagonalize $\hat{\rho}_{A,B}$ and obtain the eigenvalues $\{\rho_i\}$
- calculate *Von Neumann Entropy* as in (2.2)
- repeat for a different initial parameter values
- plot S as a function of λ

3.2 Entanglement of Formation Methodology

Let us remind ourselves, that the Entanglement of Formation calculation as stated in (2.13) - (2.3) works for a single *pair* of qubits. The input here is its density operator $\hat{\rho}$. The intricacy lies in finding the *reduced* density operator for an arbitrarily selected qubit pair. [4] have presented a technique which we are going to deploy in order to obtain the Entanglement of Formation for higher number of particles. The rationale behind it is as follows.

We will work in the same Hilbert space, the one with a maximum j, as previously. Its generation and the finding of the lowest energy state works in much the same way as above. The whole system is now in the pure state and the corresponding density operator will be again $\hat{\rho}_{gs} = |gs\rangle \langle gs|$. The work[4] suggests that we can find the reduced density operator $\hat{\rho}_{pair}$ of the qubit pair followingly:

$$\hat{\rho}_{pair} = \begin{pmatrix} v_+ & x_+^* & x_+^* & u^* \\ x_+ & w & y & x_-^* \\ x_+ & y & w & x_-^* \\ u & x_- & x_- & v_- \end{pmatrix} \tag{3.6}$$

where the meaning of the matrix elements is:

$$v_\pm = \frac{N^2 - 2N + 4\langle \hat{J}_z^2 \rangle \pm 4\langle \hat{J}_z \rangle (N-1)}{4N(N-1)} \tag{3.7}$$

$$x_\pm = \frac{(N-1)\langle \hat{J}_+ \rangle \pm 4\langle [\hat{J}_+, \hat{J}_z]_+ \rangle}{2N(N-1)} \tag{3.8}$$

$$w = y = \frac{N^2 - 4\langle \hat{J}_z^2 \rangle}{4N(N-1)} \tag{3.9}$$

$$u = \frac{\langle \hat{J}_+ \rangle}{N(N-1)} \tag{3.10}$$

where $[\hat{A}, \hat{B}]_+ = \hat{A}\hat{B} + \hat{B}\hat{A}$ denotes anti-commutation relation. Now, using the notation $\langle \hat{X} \rangle$ stand for 'mean value of operator $\hat{X}$ in the state $|gs\rangle$', or more precisely:

$$\langle \hat{X} \rangle = \langle \hat{X} \rangle_{gs} = \langle gs | \hat{X} | gs \rangle \tag{3.11}$$

(the subscript was relinquished for the sake of simplicity). After we constructed $\hat{\rho}_{pair}$, we can easily figure out Entanglement of Formation using the equations (2.14) to (2.3). The code through which the whole procedure was accomplished is added separately in the attachment, but let us here summarise the algorithm step by step. The cookbook recipe for quibit pair entanglement is:

- choose parameters λ, γ and N
- generate N+1 states $\{|M\rangle\}$
- generate Hamiltonian and find its ground state $|gs\rangle$
- create density operator $\hat{\rho}_{gs}$
- calculate all needed mean values of the operators $\langle \hat{X} \rangle$
- using (3.6) and (3.7) obtain the reduced density operator $\hat{\rho}_{pait}$
- *spin flip* $\hat{\rho}_{pair}$ matrix
- by getting its eigenvalues, determine *concurrency*
- with help of equations (2.3) and (2.3) we finally get *Entanglement of Formation* of a qubit pair in a N-particle system
- repeat for other values of λ
- plot E_f as a function of λ for constant value of γ and N

4. Results

4.1 Entanglement of Formation up to $N = 20$

We present here a graphical results of our work. Because we are also interested in a behaviour of ground state energy, entanglement will always be plotted together with the belonging spectrum.

We opt for four different χ values and for tree different amounts of qubits N.

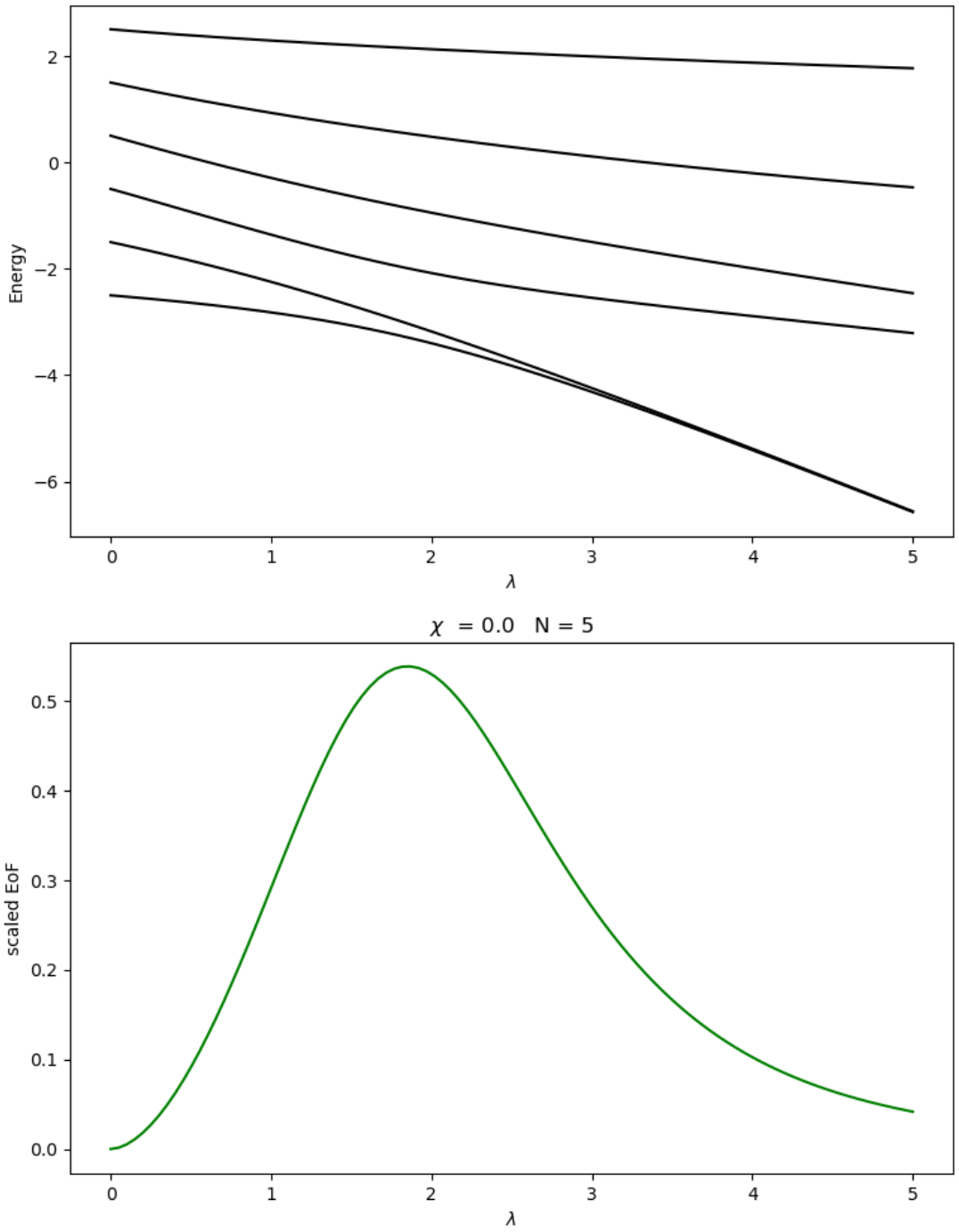

Figure 4.1: EoF for 5-qubit system with $\chi = 0$.

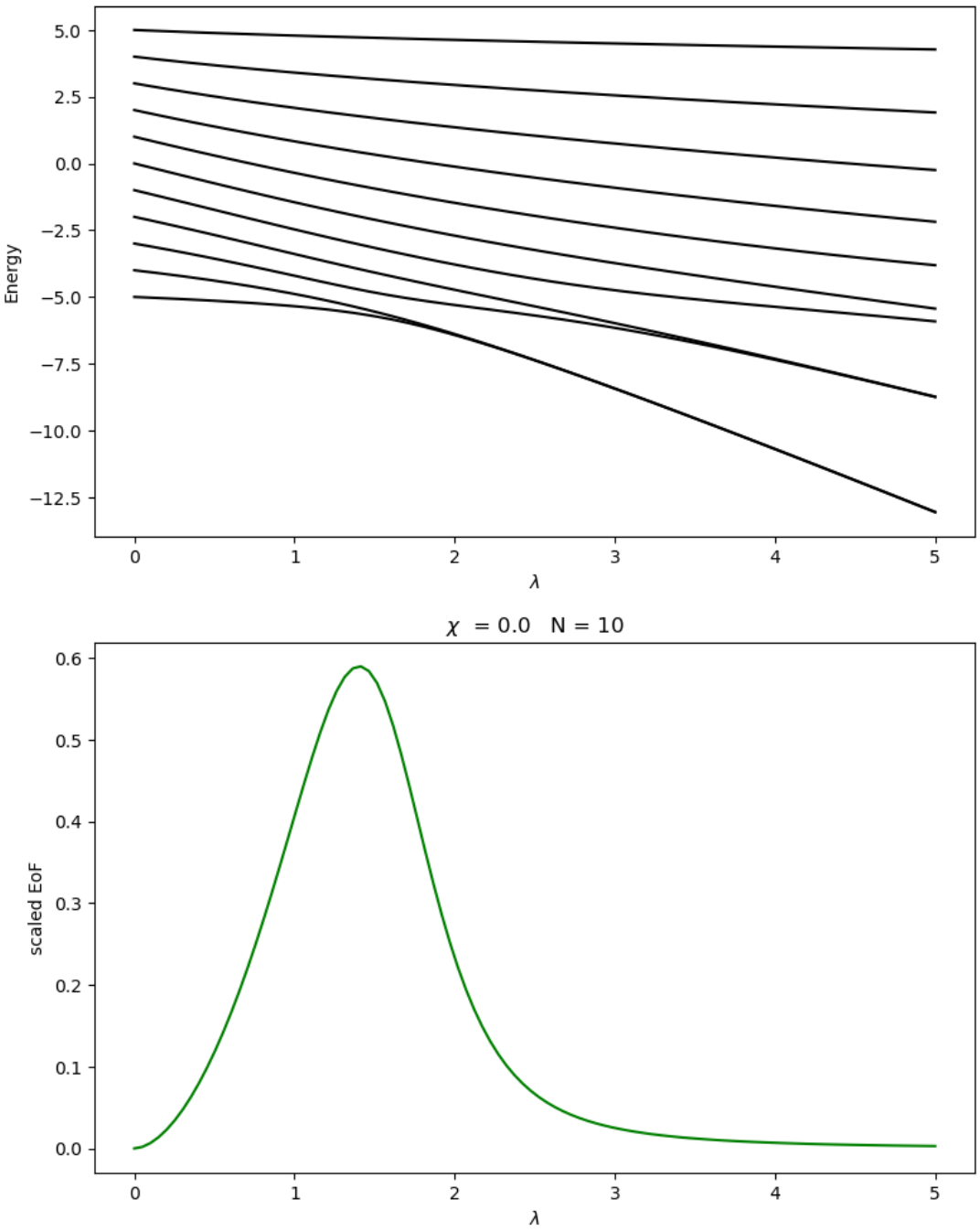

Figure 4.2: EoF for 10-qubit system with $\chi = 0$.

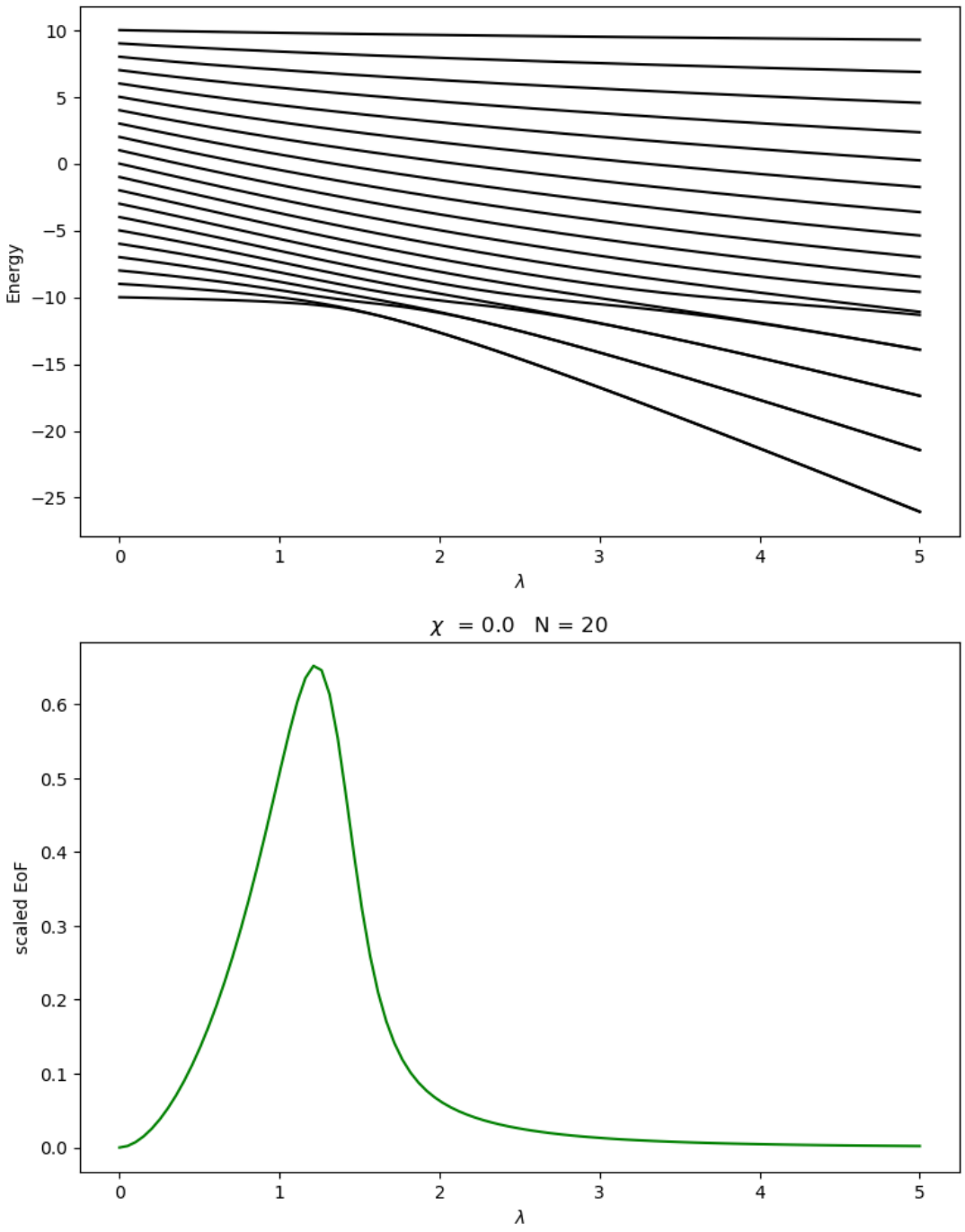

Figure 4.3: EoF for 20-qubit system with $\chi = 0$.

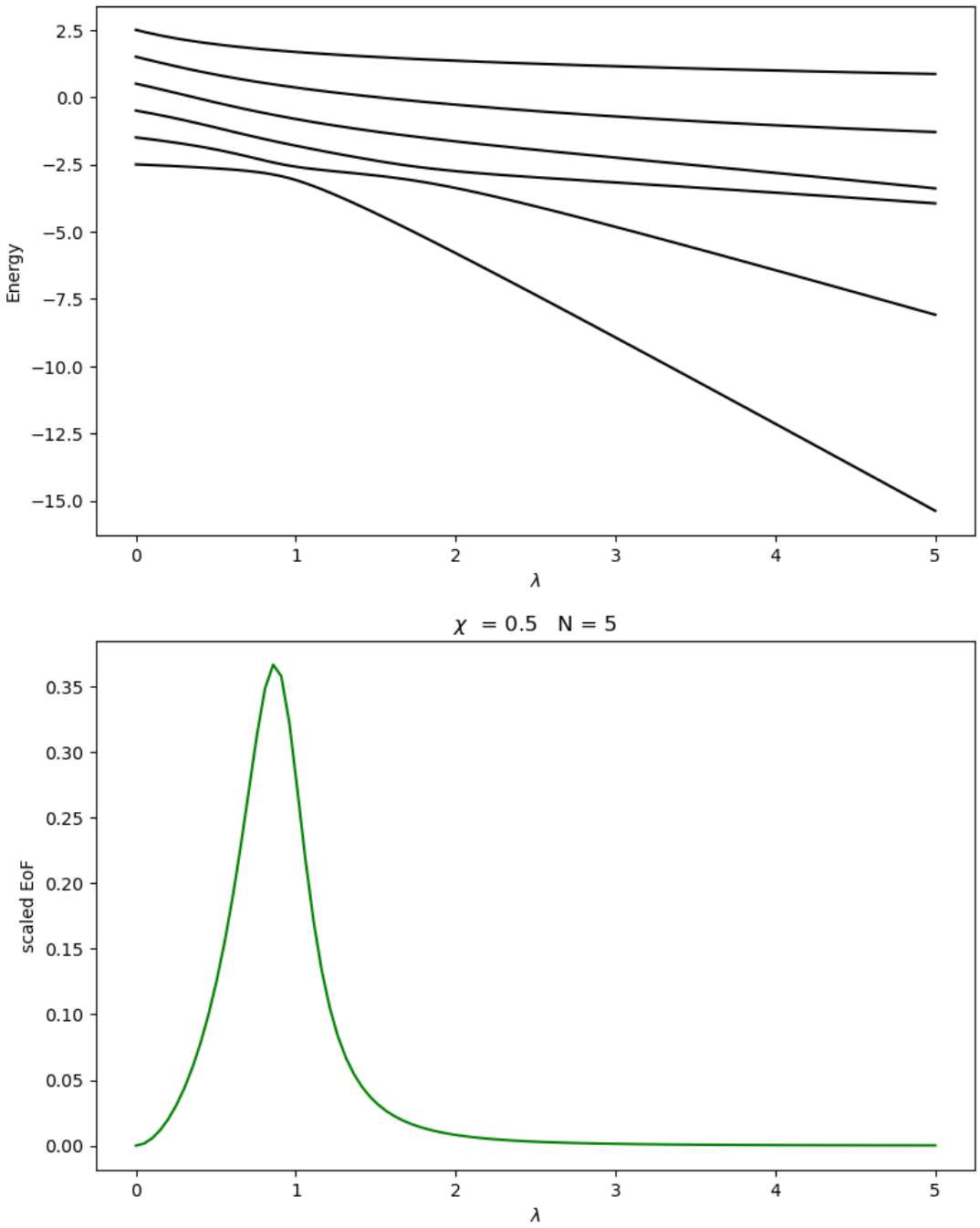

Figure 4.4: EoF for 5-qubit system with $\chi = 0.5$.

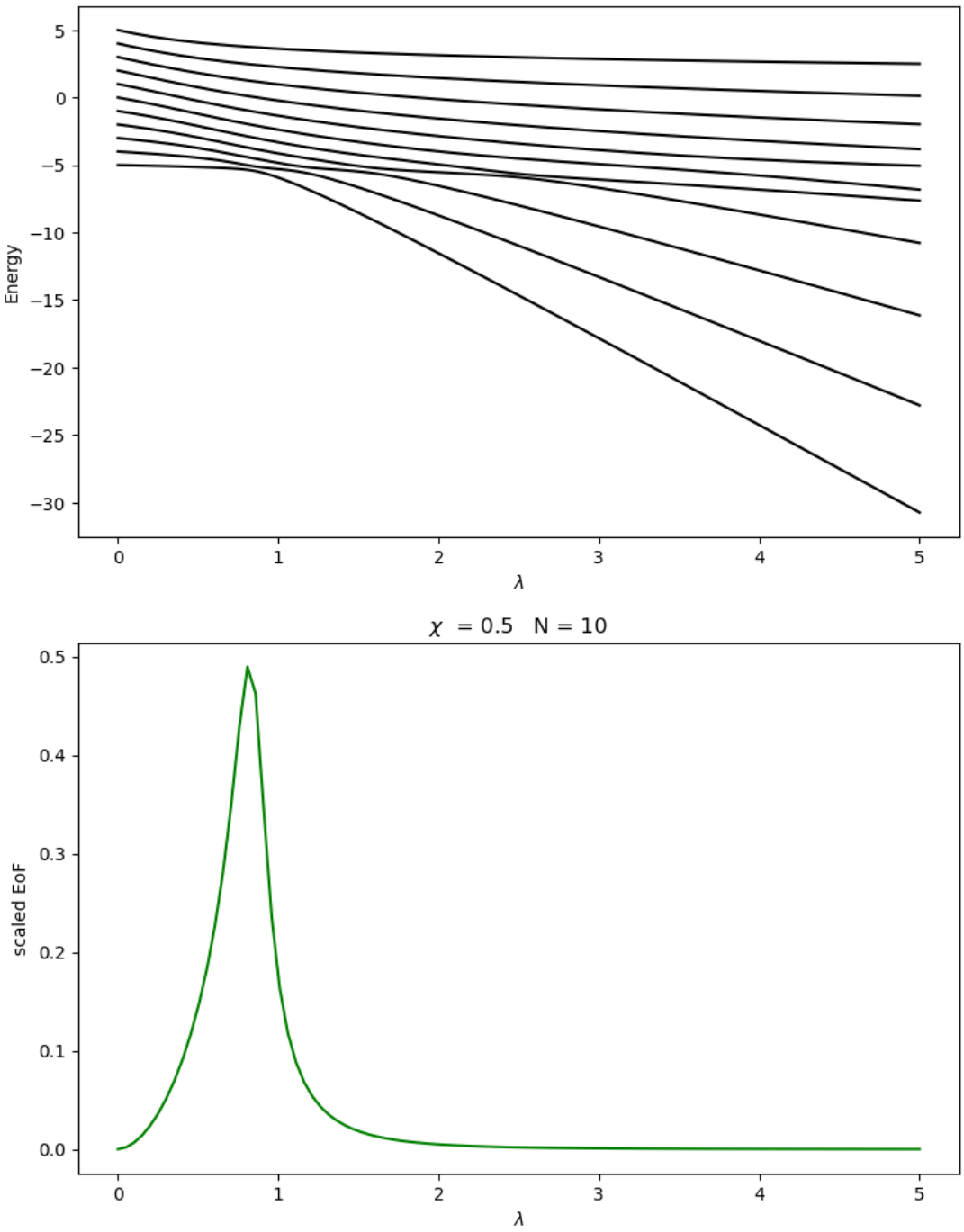

Figure 4.5: EoF for 10-qubit system with $\chi = 0.5$.

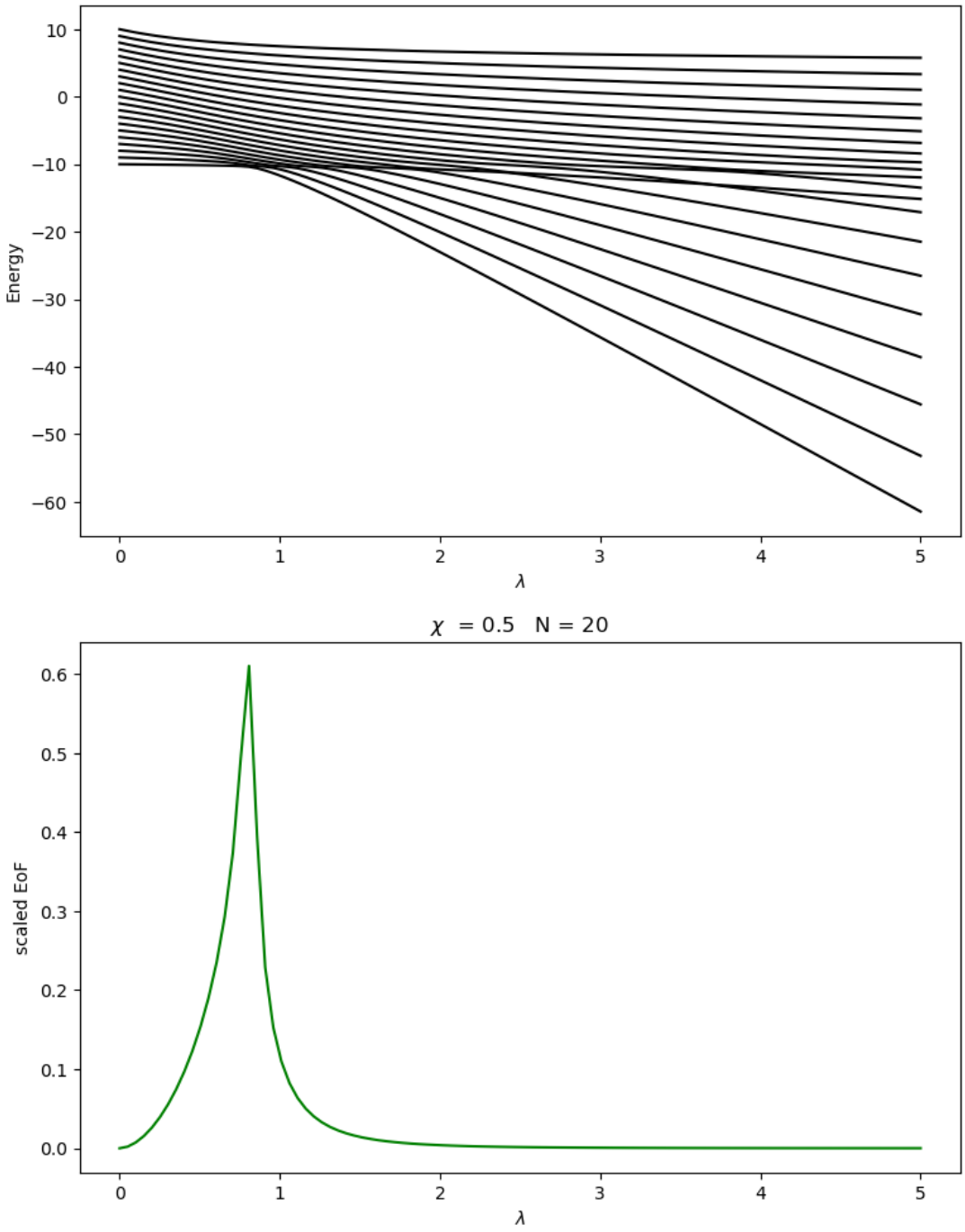

Figure 4.6: EoF for 20-qubit system with $\chi = 0.5$.

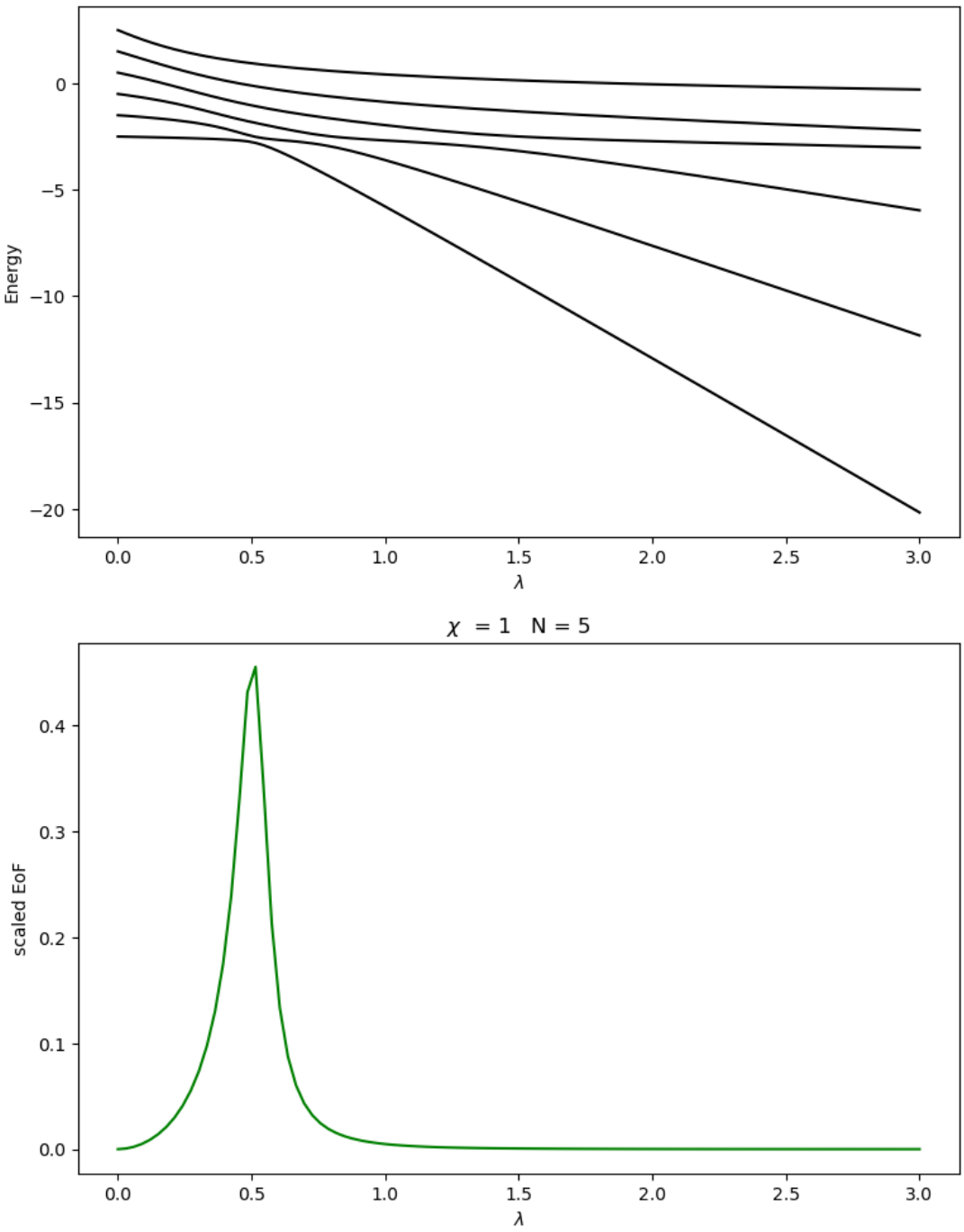

Figure 4.7: EoF for 5-qubit system with $\chi = 1$.

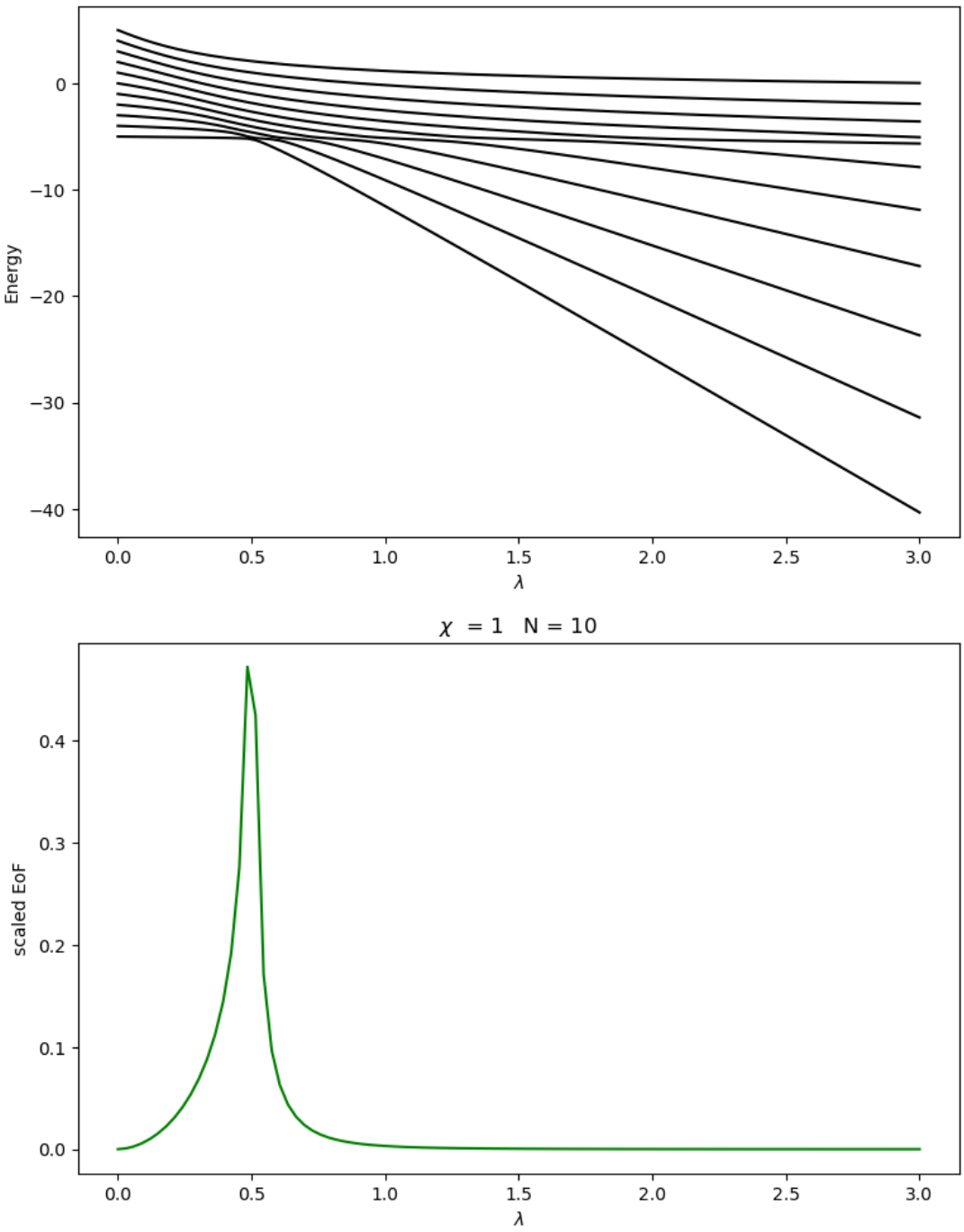

Figure 4.8: EoF for 10-qubit system with $\chi = 1$.

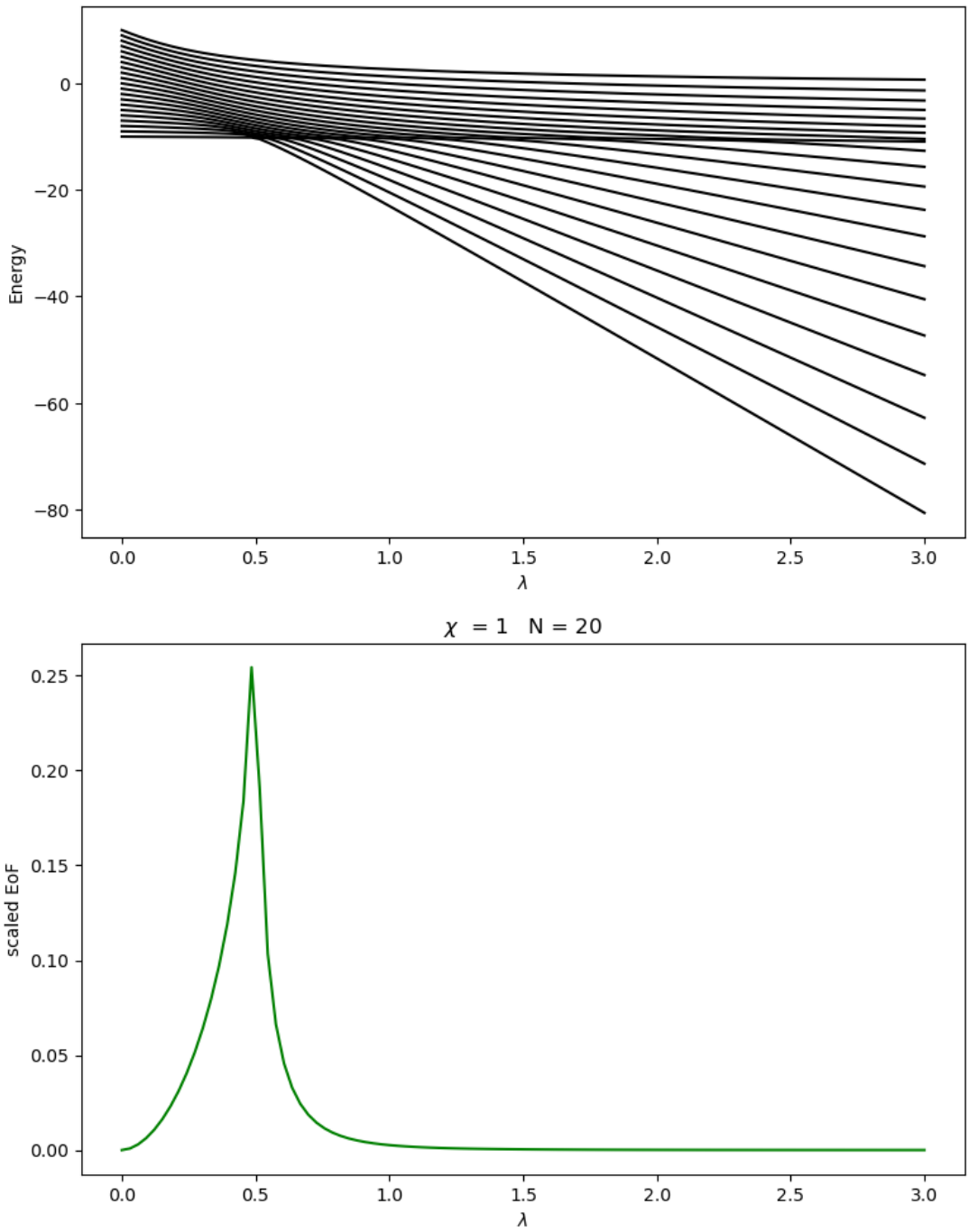

Figure 4.9: EoF for 20-qubit system with $\chi = 1$.

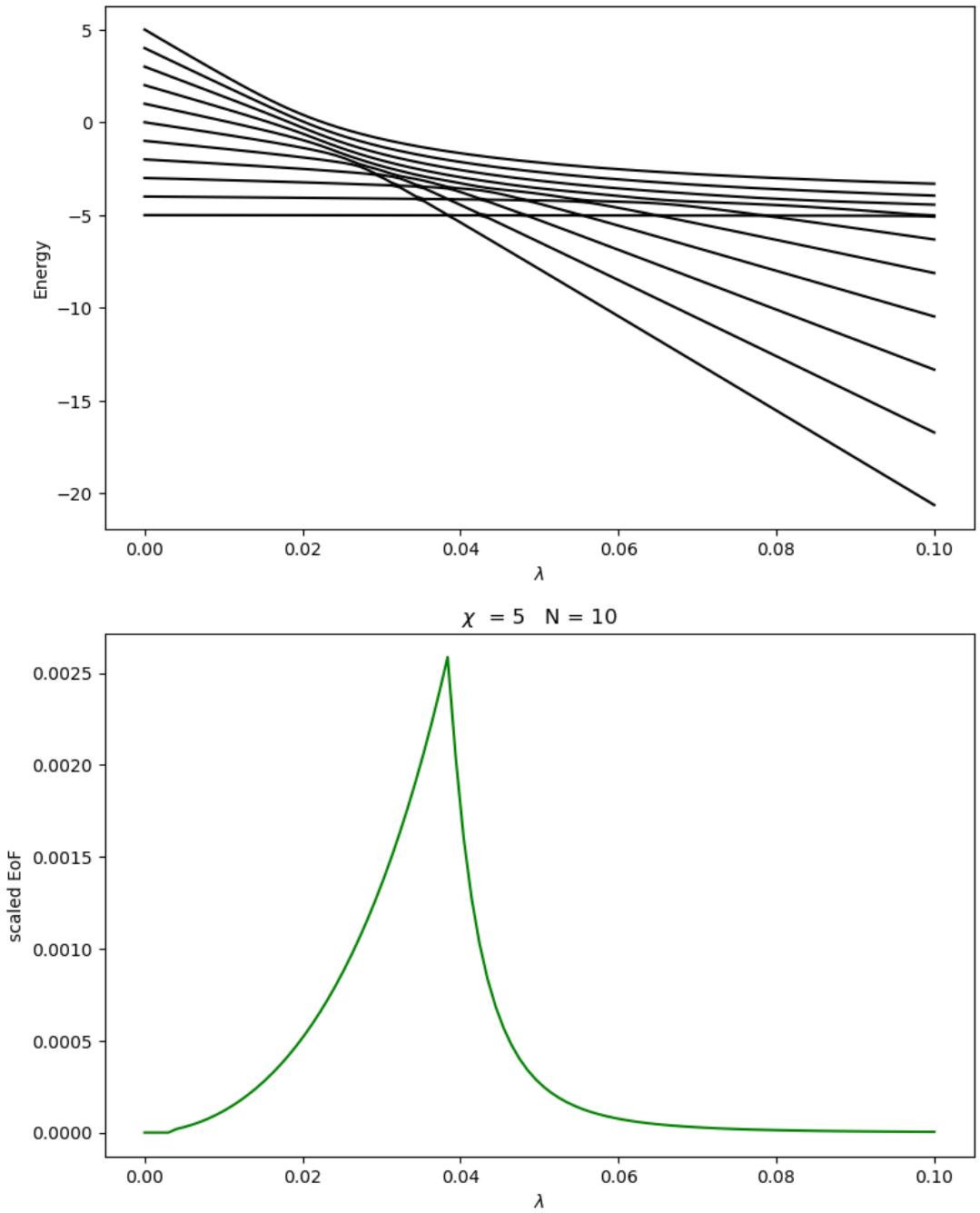

Figure 4.10: EoF for 5-qubit system with $\chi = 5$, detail on spectra.

4.1.1 Entanglement Of Formation - Heat Map

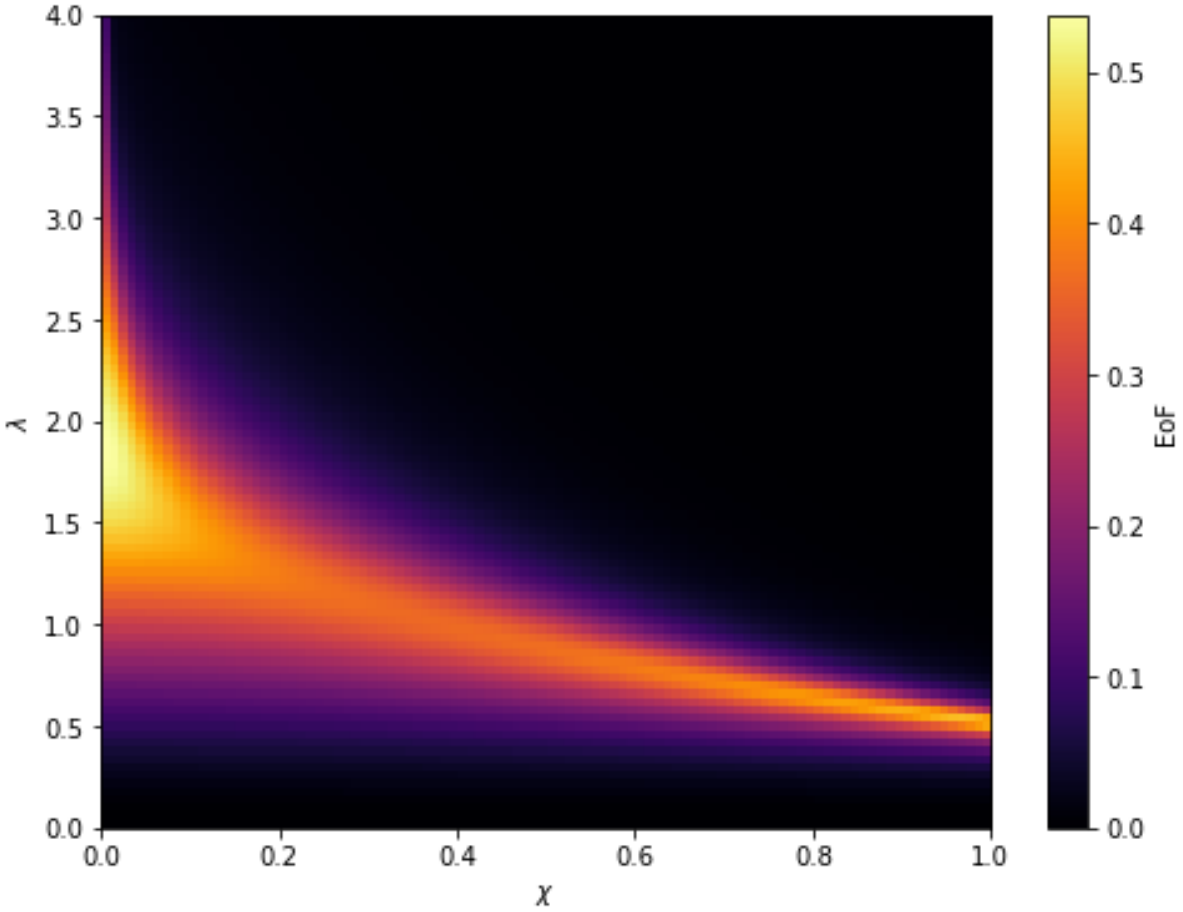

Figure 4.11: EoF for 5-qubit system.

As we can see, peak in EoF matches the value of λ for witch ground state energy changes the most. We can conclude that for $\chi = 0$ this transition is of a second order. Clear example of QPT of the first order is visible on figure 4.1 for $\chi = 0$.

Generally, we can say that the greater the parameter χ is, the closer to zero the entanglement peak will occur. We can clearly observe this behaviour in 'heat map' above where we see entanglement as a function of both the χ and λ parameters.

4.2 Von Neumann Entropy for $N = 12$

Here we deploy the same visualization strategy as above. The only difference being that now we are examining systems under different separations. Each plot will then involve five different curves, each representing the entropy of a system with a specific partition. Partitions are denoted as (x, y) and their values speak of the qubit number in each subsystem.

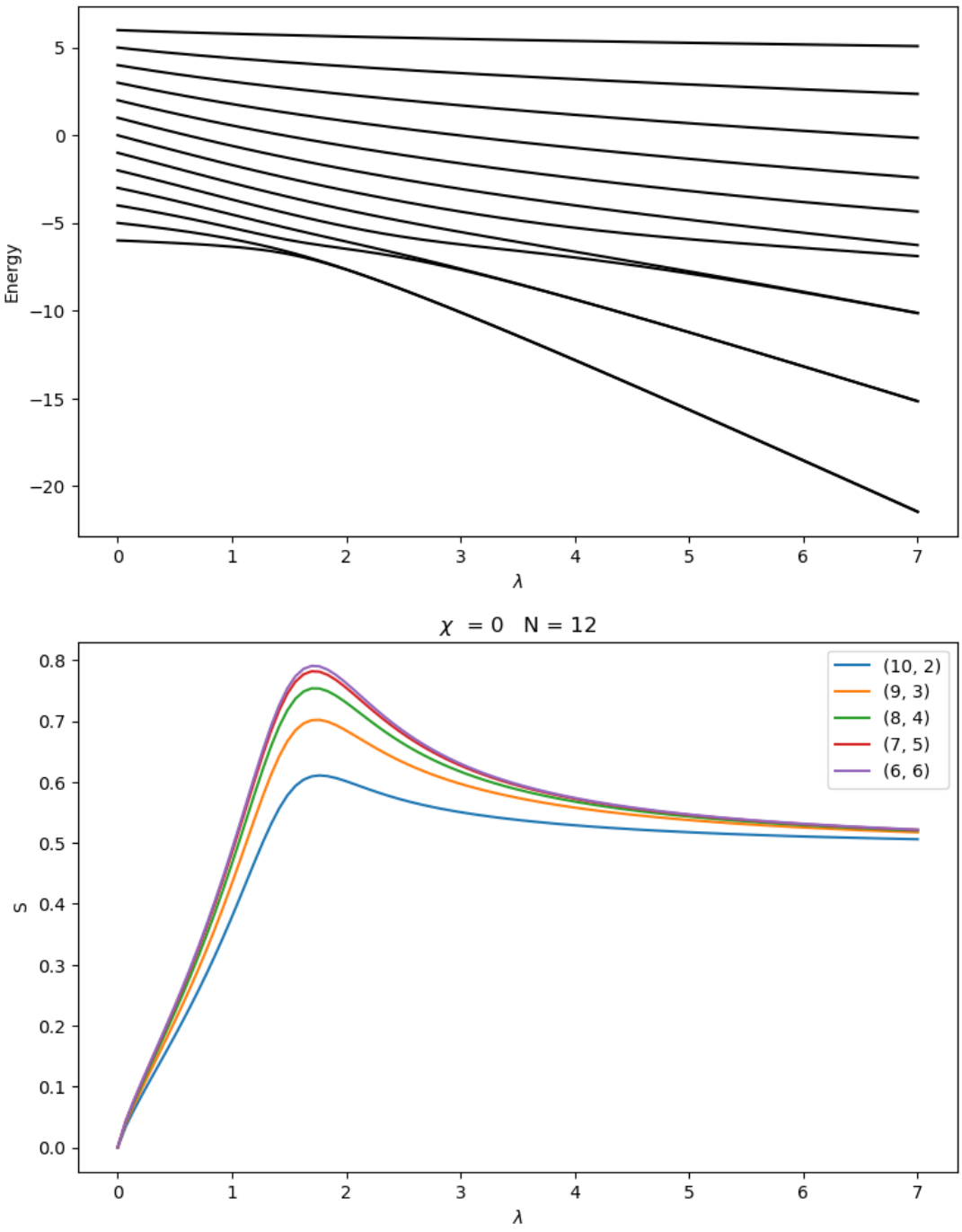

Figure 4.12: VNE for 12-qubit system with $\chi = 0$ in different partitions.

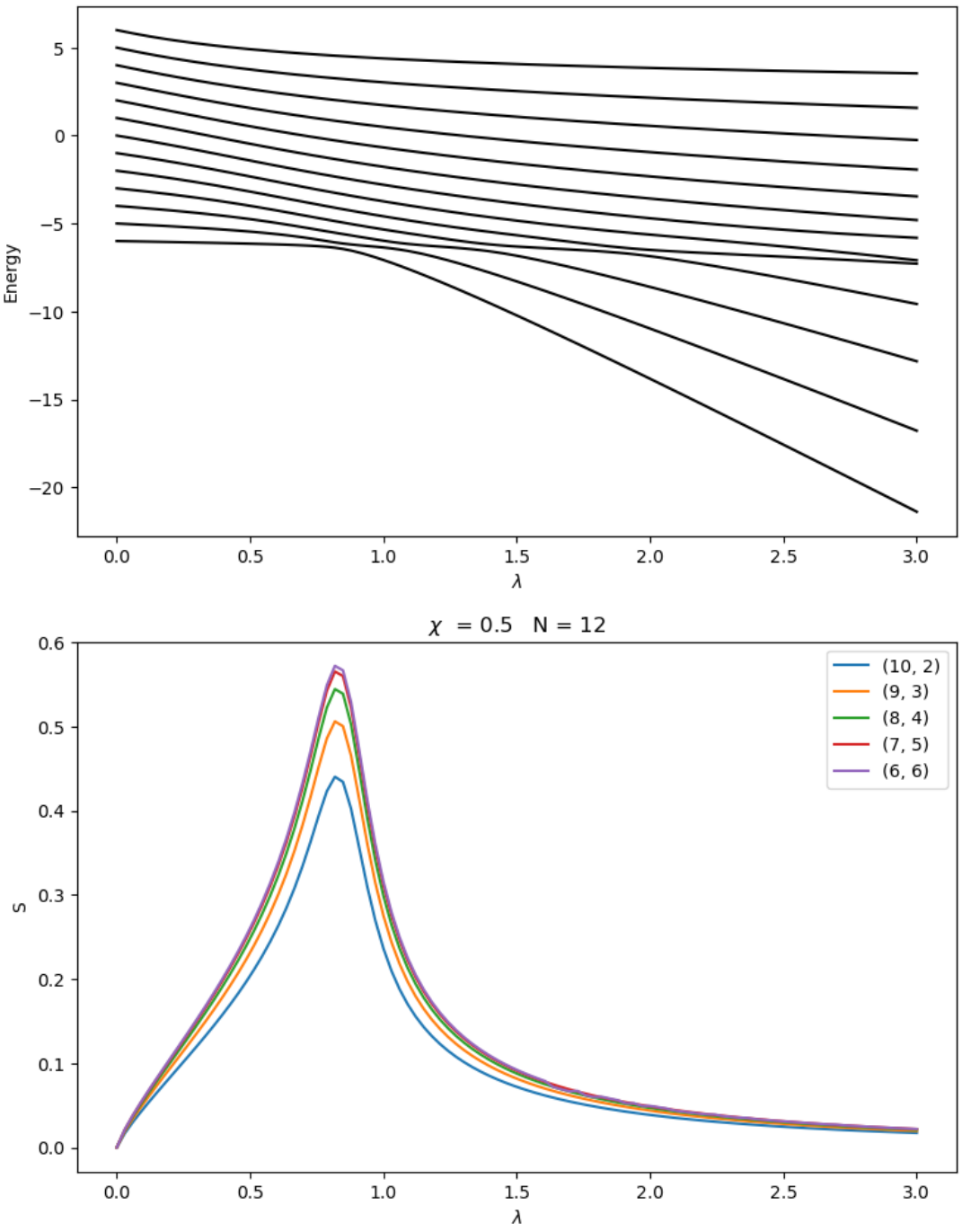

Figure 4.13: VNE for 12-qubit system with $\chi = 0.5$ in different partitions.

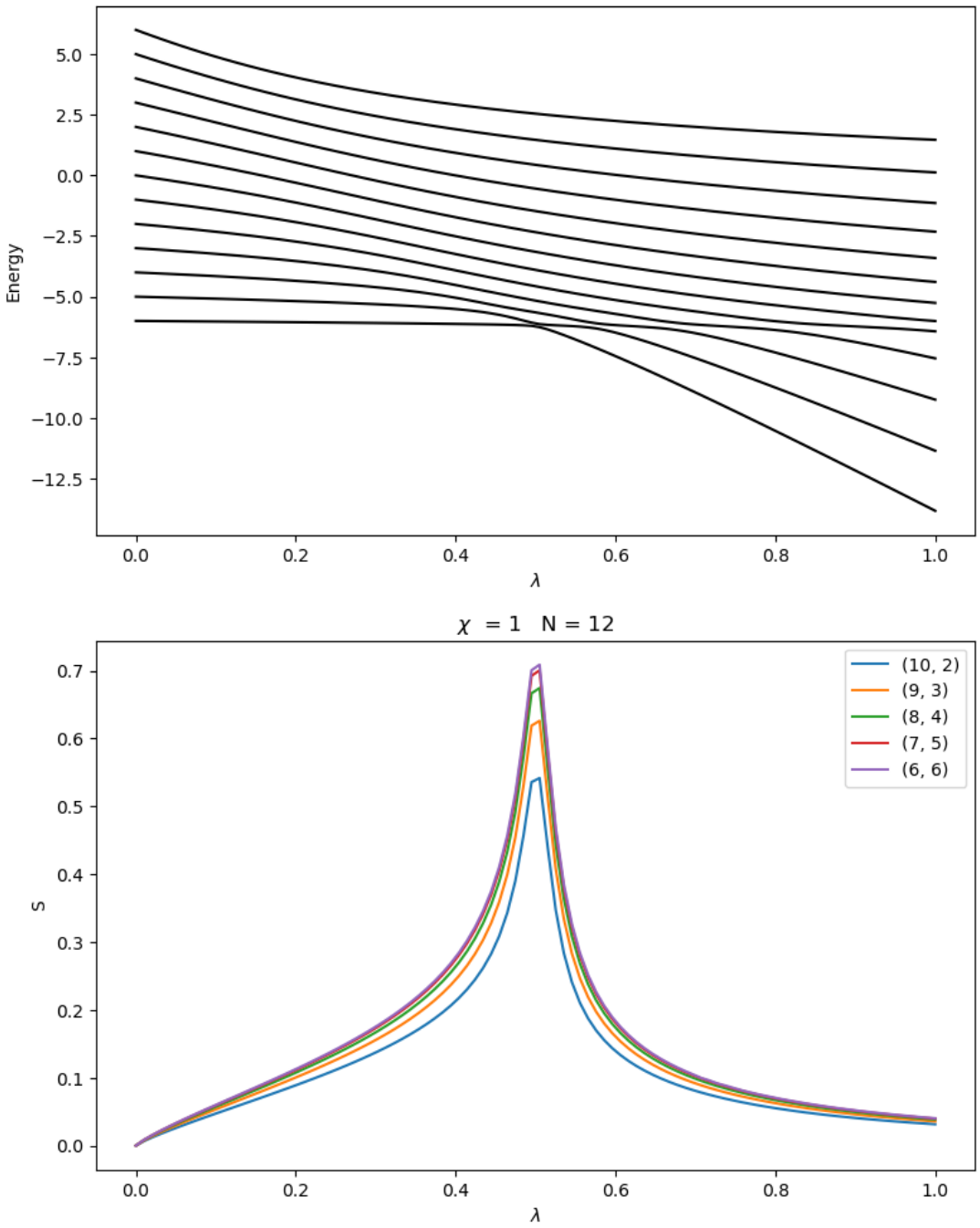

Figure 4.14: VNE for 12-qubit system with $\chi = 1$ in different partitions.

4.2.1 Von Neumann Entropy - Heat Map

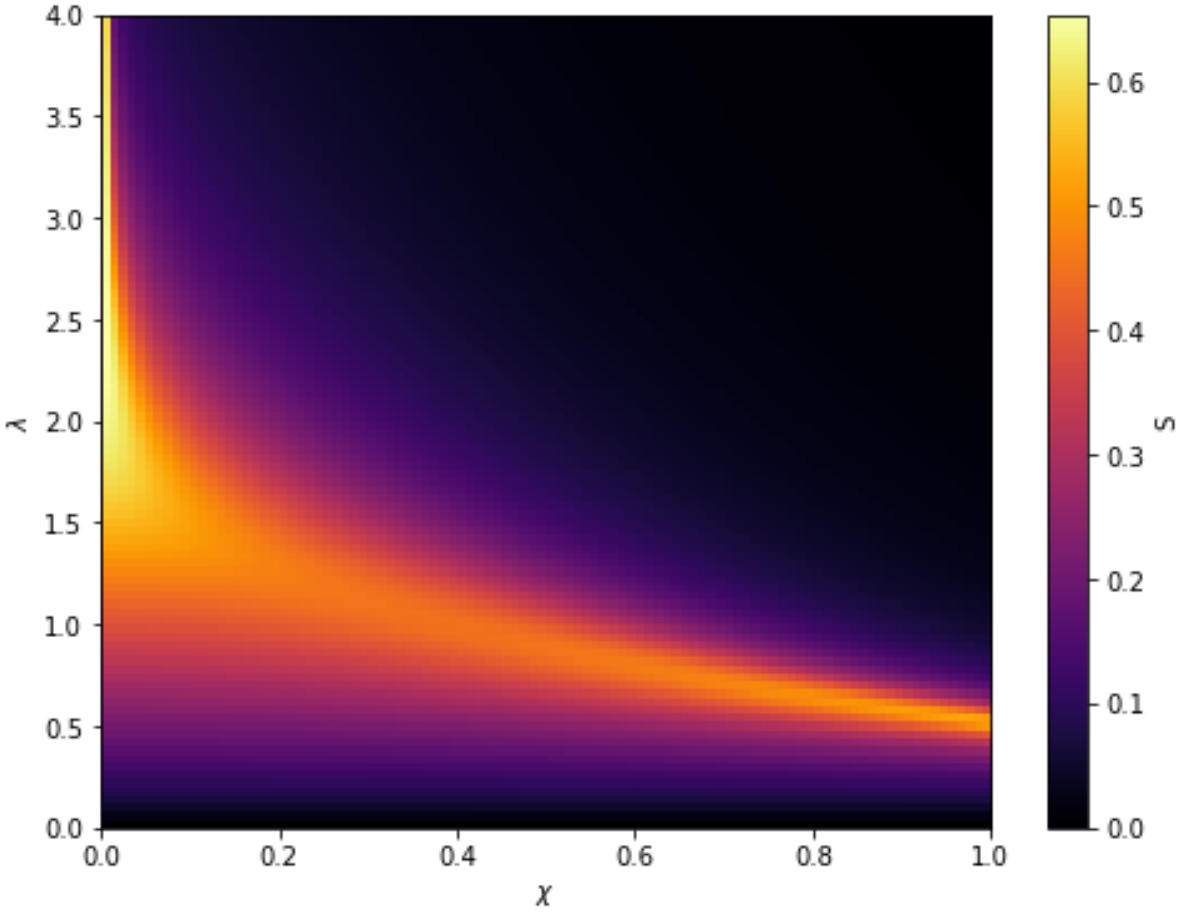

Figure 4.15: VNE for symmetric partition of 6-qubit system.

For nonzero χ the behaviour is similar to the that of the EoF – the peak is much closer to the origin with increasing value of χ. This parameter also seems to impact the peak's kurtosis. We can conclude, that the VNE is larger for partitions which are more even. On all plots we see the same order of VNE values - from $(6, 6)$ to $(10, 2)$. This should not come as a surprise. Recall that the VNE is equivalent to Shannon entropy where the maximal value can be $\ln(n)$, and where n is the number of all possible outcomes. Also, it does not matter if we work with the eigenvalues of the reduced density operator of subsystem A or B. Thanks to the SVD (which in our case is the same as diagonalization) the spectra of both reduced operators are going to be the same.

For example, we can obtain 2 eigenvalues if we trace over the subsystem A and 6 eigenvalues if we trace over the subsystem B. In the latter case, at least 4 values are going to be zero, so that the spectra are technically the same. This is limiting the maximal value of VNE to $\ln(y)$ for uneven partitions (x, y), where $y < x$. On the other hand, symmetric partitions (x, x) allow for the VNE to be $\ln(x)$. This also makes sense at an intuitive level. The greater the number of qubits in each subsystem – the higher the amount of opportunities for particle exchanges without any change in the system behaviour, meaning greater entropy.

5. Conclusion

We introduced a decsribtion of multiparticle spin systems using Lipkin Hamiltonian.We also explored spin coupling. We presented the theories behind measures of entanglement — Von Neumann Entropy, and Entanglement of Formation.

We also touched upon the problematics of Qunatum Phase Transitions of the first and second order. After the completion of a code, we plotted the entanglement as a function of the λ parameter, later on also as a function of λ and χ and compared those results with the energy spectra. Both kinds of QPT were clearly visible in the Hamiltonian's spectra.

We observed that both entanglement measures are peaking where the derivative of ground state energy reach its maximum. The result is the expected — maximum entanglement matching the QPT locations. QPT of the second order appeared on all figures involving $\chi = 0$.

We can also perceive the differences in the behaviours of VNE and EoF seen as a function of both λ and χ parameters. VNE appears to vanish much slower (in relation to λ). For zero value of χ it even seem as if the entropy converges to some nonzero value. Focusing on the VNE we can conclude that entanglement is higher for more evenly partitioned system. This result is as expected, as we discussed above. One anomaly we encounter is the shape of the VNE's peaks, which should be either smooth or should be sharp spike. Instead we see some form of irregular edges. Those are probably present due to the numerical inaccuracy, involved when we are performing matrix diagonalizations, close to critical value of λ.

Briefly, let us note the work that has been done, but left unpublished here, due to the scope of this work. First, we created a code which was able to do separation of the system in full Hilbert space (not just in the subspace with maximal j). This allowed us to distinguish between $|jm\rangle$ states which would be undistinguishable otherwise. It also allowed us to measure the entanglement for excited states. The task involved spin coupling in a tree diagrams which is an overarching task involving a fair amount of combinatorics, and touches on some very interesting aspects of contemporary physics. For example spin networks (utilised in loop quantum gravity) (see [13]).

Second, we tried to examine the behaviour of different forms of Lipkin Hamiltonian (as in [5]), which are showing different spectrum and QPT behaviour. Unfortunately, there is no studies to compare the results with, so we opted for the better established form of Hamiltonian.

Both alternative Lipkin models, as well as non-ground state QPT and entanglement can be an inspiration for later works.

Bibliography

[1] Ising E., *Beitrag zur Theorie des Ferromagnetismus*, Z. Physik 31, 253–258, 1925

[2] Matulík M., *Kvantová provázanost*, Bachelor thesis, Faculty of Mathematics and Physics, Charles University, Prague, 2021,

[3] Unanyan R. G., Ionescu C., Fleischhauer M., *Bi-partite and global entanglement in many-particle system with collective spin coupling*, doi: 10.48550/ARXIV.QUANT-PH/0412164, 2004

[4] Wang X., Mølmer K., *Pairwise entanglement in symmetric multi-qubit systems*, The European Physical Journal D, 2001

[5] Vidal J., Palacios G., Mosseri R., *Entanglement in a second-order quantum phase transition*, Physical Review A 69, 022107, 2004

[6] Kitagawa M., Ueda M., *Squeezed spin states*, Physical Review A. 47(6) P.5138-P.5143, 1993

[7] Amico L., Fazio R, Osterloh A., Vedral V., *Entanglement in many-body systems*, Reviews of Modern Physics, 80, April–June, 2008

[8] Wootters W. K., *Entanglement of Formation of an Arbitrary State of Two Qubits*, Physical Review Letters, 80, num. 10, 2245–2248,1998

[9] Kloc M., Stránský P., Cejnar P., *Quantum phases and entanglement properties of an extended Dicke model*, Annals of Physics 382 (2017) 85–111, 2016

[10] Cejnar P., *A Condensed Course of Quantum Mechanics*, Karolinum Press, Charles University, 2013, ISBN: 978-80-246-2349-8

[11] Yang M., *Reexamination of entanglement and the quantum phase transition*, American Physical Society (APS, 71, 3, 2005

[12] Cejnar P., Stranský P., Macek M., Kloc M., *Excited-state quantum phase transitions*, Journal of Physics A: Mathematical and Theoretical, 54, 13, 133001, 2021

[13] Robenilson F. Santos, Carla A., Bitencourt P., Ragni M., F. V. Prudente, C. Coletti, A. Marzuoli, V. Aquilanti, *Couplings and recouplings of four angular momenta: Alternative 9j symbols and spin addition diagrams*, Journal of Molecular Modeling volume 23, 147, 2017

[14] Sachdev S., *Quantum Phase Transitions*, Cambridge University Press, 2011, ISBN: 978-0521514682

List of Figures

List of Abbreviations

EoF - Entanglement of Formation
VNE - Von Neumann Entropy
LMG - Lipkin-Meshkov-Glick
QPT - Quantum Phase Transition
SVD - Singular Value Decomposition

A. Attachments

A.1 Script for Enatanglement of Formation

```
import numpy as np
import matplotlib.pyplot as plt
import math
import qutip as q
from mpl_toolkits import mplot3d
import matplotlib.cm as cm

def subsystem(N):
    j = N/2
    m = j
    baze = []
    while m >= -j:
        baze.append([j,m])
        m -=1
    return baze

def alfa_m(j, m):
    return math.sqrt(j*(j+1) - m*(m-1))

def Hmaker(N,g,l):
    J_plus = q.jmat(N/2,'+')
    J_minus = q.jmat(N/2,'-')
    J_y = q.jmat(N/2,'y')
    J_x = q.jmat(N/2,'x')
    J_z = q.jmat(N/2,'z')
    id_op = q.qeye(N+1)
    second_arg = (J_x + g*(J_z + (N/2)*id_op))**2
    H = J_z - (1/N)*second_arg
    return {'Jx':J_x,'Jz':J_z,'Jy':J_y,'H':H,'Jm':J_minus,'Jp':J_plus}


def vp(N,evJz2,evJz):
    nom = N**2 - 2*N +4*evJz2 + 4*evJz*(N-1)
    denom = 4*N*(N-1)
    return nom/denom

def vm(N,evJz2,evJz):
    nom = N**2 - 2*N +4*evJz2 - 4*evJz*(N-1)
    denom = 4*N*(N-1)
    return nom/denom

def w(N,evJz2):
    nom = N**2 - 4*evJz2
    denom = 4*N*(N-1)
    return nom/denom

def xp(N,evJp,evJpJzantik):
    nom = (N-1)*evJp + evJpJzantik
    denom =2*N*(N-1)
    return nom/denom

def xm(N,evJp,evJpJzantik):
    nom = (N-1)*evJp - evJpJzantik
    denom =2*N*(N-1)
    return nom/denom

def u(N,evJp2):
    nom = evJp2
    denom = N*(N-1)
    return nom/denom


def reduced_density_op(N,evJz,evJz2,evJp,evJp2,evJpJzantik):
    vp0 = vp(N,evJz2,evJz)
    vm0 = vm(N,evJz2,evJz)
    w0 = w(N,evJz2)
    xp0 = xp(N,evJp,evJpJzantik)
    xm0 = xm(N,evJp,evJpJzantik)
    u0 = u(N,evJp2)
    rho = q.Qobj([[vp0,xp0,xp0,u0],
                  [xp0,w0,w0,xm0],
                  [xp0,w0,w0,xm0],
                  [u0,xm0,xm0,vm0]])
    return rho

def spin_flip(rho):
    sig_y_sig_y = q.Qobj([[0,0,0,-1],
                          [0,0,1,0],
                          [0,1,0,0],
                          [-1,0,0,0]])

    fliped =  sig_y_sig_y*rho.conj()*sig_y_sig_y
    rr = rho*fliped
    vals = rr.eigenenergies()
    return vals

def Concurrence(g,l,N):
    mats = Hmaker(N,g,l)
    H = mats['H']
```

```
    operators = {'Jx':mats['Jx'],'Jy':mats['Jy'],'Jz':mats['Jz'],'Jp':mats['Jp'],
                 'Jz^2':mats['Jz']**2,'Jp^2':mats['Jp']**2,
                 'JpJzantik':mats['Jp']*mats['Jz'] +mats['Jz']*mats['Jp']}
    v = H.eigenstates()[1][0]
    energies = H.eigenenergies()
    ops_exp_val = {k:[] for k in operators}
    for k in operators:
        op = operators[k]
        exp_val = q.expect(oper = op, state = v)
        ops_exp_val[k] = exp_val
    v = np.array(v)
    evJz = ops_exp_val['Jz']
    evJz2 = ops_exp_val['Jz^2']
    evJp = ops_exp_val['Jp']
    evJp2 = ops_exp_val['Jp^2']
    evJpJzantik = ops_exp_val['JpJzantik']
    rho = reduced_density_op(N,evJz,evJz2,evJp,evJp2,evJpJzantik)
    cv = sorted(spin_flip(rho))[::-1]
    cs = [float(np.sqrt(c)) for c in cv]
    cs = [np.sqrt(c) for c in cv]
    cf = cs[0] - cs[1] - cs[2] -cs[3]
    if cf <= 0:
        C = 0
    else:
        C = cf
    return [C,energies,v]

def rescaled_C(g,l,N):
    return (N-1)*Concurrence(g,l,N)

def EoF(C):
    x = (1 + np.sqrt(1-C**2))/2
    if (x == 0) or (x==1):
        E = 0
    else:
        E = -x*math.log(x,2)-(1-x)*math.log(1-x,2)
    return E

def revlist(lst):
    new = []
    for i in range(len(lst[0])):
        row = [lst[j][i] for j in range(len(lst))]
        new.append(row)
    return new


#%%
# single 2D plot of Eof as function of chi and

Ns= [20] # qubit number
vse = [] # list containing numerical results
Gs = [5] # chi value
lams = np.linspace(0,0.1,1000) # lambda values
for N in Ns:

        Cons = {g:0 for g in Gs}
        Ens = {g:0 for g in Gs}
        gstates = {g:0 for g in Gs}

        for g in Gs:
            cons_and_energies = [Concurrence(g,l,N) for l in lams]
            cons = [CE[0] for CE in cons_and_energies]
            ens = [CE[1] for CE in cons_and_energies]
            Cons[g] = cons
            Ens[g] = ens

        Ensor = {g:[sorted(E) for E in Ens[g]] for g in Gs}
        Ensr = {g:revlist(Ensor[g]) for g in Gs}

        Eofs = {g:0 for g in Gs}
        for g in Gs:
            con = Cons[g]
            eof = [EoF(c) for c in con]
            Eofs[g] = eof

        Eofscaled = {g:0 for g in Gs}
        for g in Gs:
            con = Cons[g]
            eof = [EoF(c*(N-1)) for c in con]
            Eofscaled[g] = eof

        for g in Gs:
            ens = Ensr[g]
            eofs = Eofscaled[g]
        vse.append({'ene':Ensr,'eofs':Eofscaled,'N':N,'gstates':gstates})

for v in vse:
    for g in Gs:
        plt.style.use('default')
        ens = v['ene'][g]
        eofs = v['eofs'][g]
        N = v['N']
        fig, ax = plt.subplots(nrows = 2,figsize = (9,12))
        plt.title('$\chi$  = '+str(g) +'    ' + 'N = ' + str(N))
        for e in ens:
            ax[0].plot(lams,e,color = 'black')
        ax[0].set(xlabel = r"$\lambda$")
        ax[0].set(ylabel = 'Energy')

```

```
189
190         ax[1].plot(lams,np.array(eofs),label = r"$\gamma$ = "+str(g), color = 'green')
191         ax[1].set(xlabel = r"$\lambda$")
192         ax[1].set(ylabel = 'scaled EoF')
193
194 #%%
195 N = 5
196 Gs = np.linspace(0,1,100)
197 lams = np.linspace(0,4,100)
198
199 Cons = {g:0 for g in Gs}
200
201 for g in Gs:
202     cons_and_energies = [Concurrence(g,l,N) for l in lams]
203     cons = [CE[0] for CE in cons_and_energies]
204     ens = [CE[1] for CE in cons_and_energies]
205     Cons[g] = cons
206     print(g)
207
208 Eofs = {g:0 for g in Gs}
209 for g in Gs:
210     con = Cons[g]
211     eof = [EoF(c) for c in con]
212     Eofs[g] = eof
213
214 Eofscaled = {g:0 for g in Gs} # ready for 3D plot!
215 for g in Gs:
216     con = Cons[g]
217     eof = [EoF(c*(N-1)) for c in con]
218     Eofscaled[g] = eof
219
220
221 #%%
222 Y = Gs
223 X = lams
224 Y,X = np.meshgrid(Y,X)
225 Z = np.array([[float(Eofscaled[g][i]) for i in range(len(lams))] for g in Gs])
226 #%%
227 # heat map of EoF
228
229 plt.figure(figsize=(8,6))
230 im = plt.imshow(Z.transpose(),
231                 origin='lower',
232                 cmap = cm.inferno,
233                 extent =[Y.min(), Y.max(), X.min(), X.max()],
234                 aspect = 'auto'
235                 )
236 CBI = plt.colorbar(im,
237                    orientation='vertical',
238                    shrink=1,
239                    label = 'EoF'
240                    )
241 CS = plt.contour(Y,X,Z.transpose(),
242                  levels = 0,
243                  origin='lower',
244                  linewidths=1,
245                  )
246 plt.xlabel(r"$\chi$")
247 plt.ylabel(r"$\lambda$")
248 #%%
249 # visualize 3D plot od EoF from many angles
250
251 fi = np.linspace(-180,180,18)
252 for f in fi:
253     my_cmap = plt.get_cmap('cool')
254     fig = plt.figure()
255     ax = plt.axes(projection='3d')
256     #surf = ax.plot_surface(X, Y, Z, cmap = my_cmap, edgecolor ='none')
257     ax.contour3D(X, Y, Z, 20, cmap= my_cmap)
258     ax.view_init(20, -f)
259     ax.set_xlabel(r"$\chi$", labelpad=20)
260     ax.set_ylabel(r"$\lambda$", labelpad=20)
261     ax.set_zlabel('EoF', labelpad=20)
262     ax.grid(False)
263
264 #%%
265
266 # single 3D wirefeame / surface plot
267
268 my_cmap = plt.get_cmap('cool')
269 fig = plt.figure()
270 ax = plt.axes(projection='3d')
271 #surf = ax.plot_surface(X, Y, Z, cmap = my_cmap, edgecolor ='none')
272 surf = ax.plot_wireframe(X, Y, Z, cmap = my_cmap,color = 'black')
273 ax.view_init(80, 0) # elevation
274 ax.set_xlabel(r"$\chi$")
275 ax.set_ylabel(r"$\lambda$")
276 ax.grid(False)
277
278 #%%
```

A.2 Script for Von Neumann Entropy

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import identity
from sympy import symbols, Matrix, Symbol
import qutip as q
import matplotlib.cm as cm


def alfa_p(j, m):
    return (np.sqrt(j*(j+1) - m*(m+1)))

def alfa_m(j, m):
    return (np.sqrt(j*(j+1) - m*(m-1)))

def vectors(n):
    vektory = identity(n).toarray()
    return vektory


def subsystem(N):
    j = N/2
    m = j
    baze = []
    while m >= -j:
        baze.append([j,m])
        m -=1
    return baze

def matrix_from_vectors(v,h):
    vertical = np.array(v)
    horizontal = np.array(h)
    mat = []
    for i in range(len(vertical)):
        mat.append(vertical[i]*horizontal)
    M = np.array(mat)
    return M



def CG(j,m,jA,mA,jB,mB):
    return q.clebsch(jA, jB, j, mA, mB, m)

def coupled_to_separated(baze,Abaze,Bbaze):
    dct = {str(b):[] for b in baze}
    for state in baze:
        for a_state in Abaze:
            for b_state in Bbaze:
                j = state[0]
                m = state[-1]
                jA = a_state[0]
                mA = a_state[-1]
                jB = b_state[0]
                mB = b_state[-1]
                cg = CG(j,m,jA,mA,jB,mB)
                if cg != 0:
                    dct[str(state)].append([cg,a_state,b_state])
    return dct


def Hamiltonian(N,g,l):
    J_x = q.jmat(N/2,'x')
    J_z = q.jmat(N/2,'z')
    id_op = q.qeye(N+1)
    second_arg = (J_x + g*(J_z + (N/2)*id_op))**2
    H = J_z - (1/N)*second_arg
    return H

def min_energy_vector(H):
    gs = H.groundstate()[1]
    energies = H.eigenenergies()
    return {'groundstate':gs,'energies':energies}

def rho(v):
    return q.ket2dm(v)

def min_state_to_bazove(gs,baze):
    idx = []
    for i in range(len(gs)):
        if gs[i] != 0.0:
            idx.append(i)
    to_bazove = [(gs[i],baze[i]) for i in idx]
    return to_bazove
#%%
def min_state_to_separovane(to_bazove, coupled_to_separated):
    to_separovane = []
    for el in to_bazove:
        c_state = el[1]
        const = el[0]
        s_state = coupled_to_separated[str(c_state)]
        for element in s_state:
            new_const = const*element[0]
            to_separovane.append([new_const,element[1],element[2]])
    return to_separovane

def separovane_to_numericke(to_separovane,Abaze,Bbaze):
    Avecs = vectors(len(Abaze))
    Bvecs = vectors(len(Bbaze))
    Adict = {str([Abaze[i][0],Abaze[i][-1]]):Avecs[i] for i in range(len(Avecs))}
    Bdict = {str([Bbaze[i][0],Bbaze[i][-1]]):Bvecs[i] for i in range(len(Bvecs))}
```

```
    vectorized = []
    for el in to_separovane:
        c = el[0]
        v1 = str(el[1])
        v2 = str(el[2])
        vectorized.append((c,Adict[v1],Bdict[v2]))
    return vectorized

def numeric_to_reduced(sep_to_num,Abaze,Bbaze):
    alfa_matrix = [[0 for i in range(len(Abaze))] for j in range(len(Bbaze))]
    Bs = vectors(len(Bbaze))
    As = vectors(len(Abaze))
    for el in sep_to_num:
        j = np.where(el[1] == 1)[0][0]
        i = np.where(el[2] == 1)[0][0]
        alfa = el[0]
        alfa_matrix[i][j] = alfa
    reduced = []
    for i in range(len(Bs)):
        for i_prime in range(len(Bs)):
            const = 0
            for j in range(len(As)):
                c = alfa_matrix[i][j]*alfa_matrix[i_prime][j]
                const +=c
            submatrix = const*matrix_from_vectors(Bs[i],Bs[i_prime])
            reduced.append(submatrix)
    rho = reduced[0]
    for i in range(1,len(reduced)):
        rho = rho + reduced[i]
    #print('Trace of reduced rho: ', sum([rho[i][i] for i in range(len(rho))]))
    return rho


def matrix_to_eigen(rho):
    rho = Matrix(rho)
    evec = rho.eigenvects()
    vects = [evec[i][-1][0] for i in range(len(evec))] # ready for operations...
    vals = [evec[i][0] for i in range(len(evec))] # ready for operations...
    #print(len(vals))
    #print('sum rho_i: = ',sum(vals))
    #print('sum squered rho_i: = ',sum([v*v for v in vals]))
    return {'vals':vals,'vects':vects}


def eigen_to_VNE(vals):
    res = []
    vals = [float(v) for v in vals]
    for c in vals:
        if c < 0:
            print(vals)
            break
        if c != 0:
            c = np.sqrt(c)
            res.append(c*np.log(c))
    return -sum(res)

def params_to_VNE(NA,NB,g,l):
    print('-------------------------------------------')
    N = NA + NB
    Abaze = subsystem(NA)
    Bbaze = subsystem(NB)
    ABbaze = subsystem(NA+NB)
    coup_to_sep = coupled_to_separated(ABbaze,Abaze,Bbaze)
    H = Hamiltonian(N,g,l)
    mev = min_energy_vector(H)
    gs = np.array(mev['groundstate'])
    print(gs)
    energies = mev['energies']
    to_bazove = min_state_to_bazove(gs,ABbaze)
    to_separovane = min_state_to_separovane(to_bazove,coup_to_sep)
    sep_to_num = separovane_to_numericke(to_separovane,Abaze,Bbaze)
    rho = numeric_to_reduced(sep_to_num,Abaze,Bbaze)
    vals = matrix_to_eigen(rho)['vals']
    res = eigen_to_VNE(vals)
    print('-------------------------------------------')
    return {'vne':res,'energies':energies}

def vne(lambdas,NA,NB,g):
    VNEs= []
    energie = []
    for l in lambdas:
        ptvne = params_to_VNE(NA,NB,g,l)
        entropy = ptvne['vne']
        e = ptvne['energies']
        print('lambda : ',l, 'gamma :',g)
        VNEs.append(entropy)
        energie.append(e)
    return {'VNEs':VNEs,'energie':energie}

def plot_it(vnes,ens,title,lams): # vne + spektrum
     plt.style.use('default')
     fig, ax = plt.subplots(nrows = 2,figsize = (9,12))
     plt.title('$\chi$  = '+str(g) +'    ')
     ens = [sorted(E) for E in ens]
     ens = [[ens[i][j] for i in range(len(ens))] for j in range(len(ens[0]))]
     for e in ens:
       ax[0].plot(lams,e,color = 'black')
     ax[0].set(xlabel = r"$\lambda$")
     ax[0].set(ylabel = 'Energy')
```

```
199       ax[1].plot(lams,np.array(vnes),label = r"$\gamma$ = "+str(g), color = 'red')
200       ax[1].set(xlabel = r"$\lambda$")
201       ax[1].set(ylabel = 'Entropy')
202
203   def plot_it_coups(coups,ens,title,lams): # spectrum + vne
204       plt.style.use('default')
205       fig, ax = plt.subplots(nrows = 2,figsize = (9,12))
206       plt.title('$\chi$  = '+str(g) +'   ' + 'N = ' + str(N))
207       ens = [sorted(E) for E in ens]
208       ens = [[ens[i][j] for i in range(len(ens))] for j in range(len(ens[0]))]
209       for e in ens:
210         ax[0].plot(lams,e,color = 'black')
211       ax[0].set(xlabel = r"$\lambda$")
212       ax[0].set(ylabel = 'Energy')
213       for c in coups:
214           vnes = coups[c]['ans']
215           ax[1].plot(lams,np.array(vnes),label = c)
216       plt.legend()
217       ax[1].set(xlabel = r"$\lambda$")
218       ax[1].set(ylabel = 'S')
219
220   def plot_en(ens,title,lams): # spektrum only
221       plt.style.use('default')
222       plt.figure()
223       plt.title('$\chi$  = '+str(g) +'   ')
224       ens = [sorted(E) for E in ens]
225       ens = [[ens[i][j] for i in range(len(ens))] for j in range(len(ens[0]))]
226       for e in ens:
227         plt.plot(lams,e,color = 'black')
228       plt.xlabel(r"$\lambda$")
229       plt.ylabel('Energy')
230   #%%
231   g = 0
232
233   coups = {(10,2):{'ans':0,'ens':0},
234            (9,3):{'ans':0,'ens':0},
235            (8,4):{'ans':0,'ens':0},
236            (7,5):{'ans':0,'ens':0},
237            (6,6):{'ans':0,'ens':0},}
238
239   N = 12
240   lams = np.linspace(0,7,100)
241   for c in coups:
242       ans = []
243       ens = []
244       for l in lams:
245           ptvne = params_to_VNE(c[0],c[1],g,l)
246           vnes = ptvne['vne']
247           e = ptvne['energies']
248           print(g,l)
249           ans.append(vnes)
250           ens.append(e)
251       coups[c]['ans'] = ans
252       coups[c]['ens'] = ens
253
254   plot_it_coups(coups,coups[(10,2)]['ens'],g,lams)
255   #%%
256
257   lams = np.linspace(0,4,100)
258   Gs = np.linspace(0,1,100)
259
260   Y = Gs
261   X = lams
262   X,Y = np.meshgrid(X,Y)
263
264   res = {g:[] for g in Gs}
265
266   for g in Gs:
267       for l in lams:
268           res[g].append(params_to_VNE(3,3,g,l)['vne'])
269
270   Z = np.array([[float(res[g][i]) for i in range(len(lams))] for g in Gs])
271
272   #%%
273   plt.figure(figsize=(8,6))
274   im = plt.imshow(Z.transpose(),
275                   origin='lower',
276                   cmap = cm.inferno,
277                   extent =[Y.min(), Y.max(), X.min(), X.max()],
278                   aspect = 'auto'
279                   )
280   CBI = plt.colorbar(im,
281                      orientation='vertical',
282                      shrink=1,
283                      label = 'S'
284                      )
285   CS = plt.contour(Y,X,Z.transpose(),
286                    levels = 0,
287                    origin='lower',
288                    linewidths=1,
289                    )
290   plt.xlabel(r"$\chi$")
291   plt.ylabel(r"$\lambda$")
292
293   #%%
294   # 3D plot from different angles
295
296   fi = np.linspace(-180,180,120)
297   for f in fi:
```

```
    my_cmap = plt.get_cmap('cool')
    fig = plt.figure()
    ax = plt.axes(projection='3d')
    #surf = ax.plot_surface(X, Y, Z, cmap = my_cmap, edgecolor ='none')
    #ax.plot_wireframe(X, Y, Z, color = 'red')
    ax.contour3D(X, Y, Z, 20, cmap= my_cmap)
    ax.view_init(30, -f)
    ax.set_xlabel(r"$\chi$", labelpad=20)
    ax.set_ylabel(r"$\lambda$", labelpad=20)
    ax.set_zlabel('S', labelpad=20)
    ax.grid(False)

#%%
my_cmap = plt.get_cmap('cool')
fig = plt.figure()
ax = plt.axes(projection='3d')
#surf = ax.plot_surface(X, Y, Z, cmap = my_cmap, edgecolor ='none')
ax.contour3D(X, Y, Z, 20, cmap= my_cmap)
ax.view_init(50, 5)
ax.set_xlabel(r"$\chi$", labelpad=20)
ax.set_ylabel(r"$\lambda$", labelpad=20)
ax.set_zlabel('S', labelpad=20)
ax.grid(False)
#%%
```