SSE for the dimerization transition on the square lattice

Dr. Maxime Debertolis, Kriti Baweja and Prof. Dr. David Luitz

December 21, 2023

For the successful completion of this project it is required to hand in a link to the `git` repository which contains the full development history of the LaTeX source and PDF of the project report and all source codes of all computer programs required to reproduce the results. It is not sufficient to only hand in the final version! It is preferable to use `gitlab.uni-bonn.de` and private sharing with `mdeberto` and `dluitz` is sufficient. The submission of this link and a pdf copy of the project report should be done via ecampus.

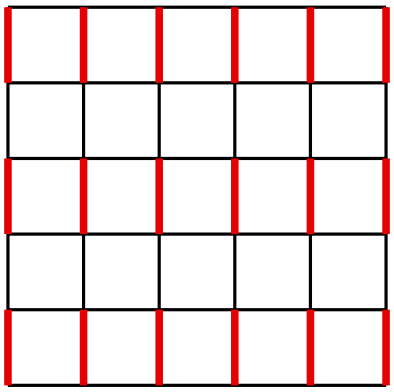
Requirement This project should be done in a compiled language, like C, C++, Rust. Julia might work, too but is not preferred.

Model We study the Heisenberg Hamiltonian on a 2d square lattice defined as:

$$\hat{H} = J_1 \sum_{\langle i,j \rangle_1} \vec{S}_i \cdot \vec{S}_j + J_2 \sum_{\langle i,j \rangle_2} \vec{S}_i \cdot \vec{S}_j, , \tag{1}$$

where the sum $\langle i, j \rangle_x$ runs over nearest neighbors of type x. We have two types of bonds $x = 1, 2$. We choose to work with periodic boundary conditions on both directions, such that the upper and lower (as well as left and right) parts of the lattice are connected, i.e. we have a torus topology.

The bond types are illustrated in the following figure. Red bonds are of type 1, black bonds are of type 2, (periodic boundaries are not shown).



This model has a quantum phase transition in the ground state (limit $\beta \to \infty$) as a function of J_1/J_2. When the J_1 bonds dominate, we are in a dimer phase, for $J_1 = J_2$ and in the vicinity we are in the antiferromagnetic phase. The transition can be observed in terms of the staggered magnetization (Néel order parameter)>

For an implementation of SSE with such boundary conditions, both lattice dimensions should be even to avoid undesired sign problems. We first set the coupling strengh to $J_2 = 1$.

1. The Hamiltonian can be decomposed as a sum over two-site bond operators. To make computations efficient, rewrite these bond operators as a diagonal and off-diagonal parts, in which the matrix elements are positive.

2. Simplify the partition function $\mathcal{Z}$ to make it implementable by SSE (Taylor expansion and convenient forms of the Hamiltonian written in the previous question). How is the obtained final expression more suitable for computational implementation compared to the bare definition $\mathcal{Z} = \mathrm{Tr}[\exp\left(-\beta\hat{H}\right)]$?

We now turn to the actual implementation of the algorithm. More details can be found in the review article by Anders W. Sandvik *Computational Studies of Quantum Spin Systems*, and in arXiv:1909.10591.

3. Start by creating the lattice. It's best to store the lattice as a graph of vertices (sites) and edges (bonds) so that you will no longer have to think about geometry moving forward.

4. Having calculated the non-zero matrix elements, write a function which takes as an input a given initial state and updates diagonal operators for allowed sequences of operators (i.e. the final propagated state is the same as the initial state). The goal of this function is to sweep over the entire operator string and to randomly replace each diagonal operator with the identity operator (and vice-versa) according to the corresponding acceptance probability (see lecture notes).

   Note: To iterate over all the operator strings it is useful to introduce an object having several fields which stores the following informations:

   - The *type* of the operator (which can be *diagonal*, labelled by **D**, *off-diagonal*, **OD**, or *identity*, **Id**).
   - Sites on which the operator acts.
   - States of the sites on which the operator acts (before and after, only relevant for OD operators).
   - The propagated state before and after the action of the operator.
   - Each operator leg stores the index and the leg of the operator it is connected to (links).

   Since identity operators are not part of the *directed loop* update, it is useful to assign illegal field values for this operator so that you only consider diagonal and off-diagonal operators.

5. Write a function which creates/removes *links* between operators whenever an operator is inserted or removed in the operator string. This function will be called only during diagonal updates.

6. Write a function testing if the links created by the previously built function are *correct* by checking that there are links in both forward and backward direction. In other words, it checks if a leg of a given operator is linked to the leg of another operator, and that the latter leg is also connected to the former one (consistency of links in the sequence).

7. For off-diagonal updates, directed loop update is the most efficient one for the Heisenberg model. For a randomly picked non-identity operator **and** leg, the type of the operator is changed, and the states of the sites associated to the chosen leg and its exit leg are flipped (such that the input/output states are consistent with the operator type). Then, this procedure is repeated until the loop is closed. If the loop is traced over the periodic boundary in the *imaginary time* then the initial state is also changed. Write a function implementing the directed loop update.

8. Write the main section of the code, using all the previously defined functions. During the thermalization process, the expansion order n_{ord} (defined as the number of operators of type D and OD) and n_{op} (total number of operators) will increase, and saturate at some point. In practice, we change n_{op} if, after a diagonal update, $n_{\text{ord}} \geq c\, n_{\text{op}}$ with c chosen in practice to be around 0.8 or 0.9. n_{op} is then increased by a factor $1-c$. We iterate until thermalization is reached.

   Note: When performing the off-diagonal update, some specific situations may yield loops that are not tracing a reasonable amount of operators. Thus, we count the number n_{tr} of operators traced in the directed loop update (it can count twice the same operator), and perform the process again until $n_{\text{tr}} \geq n_{\text{ord}}$.

In the following, every observable should be plotted with their respective error bars computed using the bootstrap method.

8. Plot the expansion order n and the length of operator list n_{op} as a function of Monte Carlo steps.

9. Plot the energy as a function of Monte Carlo steps for different values of β and benchmark it against Exact Diagonalization (at the same β) for small system sizes.

10. Calculate the staggered magnetization (Néel order parameter) for different system sizes $L \times L$ as a function of the parameter J_1/J_2 in the limit $\beta \to \infty$. You do this by calculating the observable for different β and using β large enough for convergence. Something like $\beta = 4L, 6L$ should be enough. You should observe a crossing of these curves close to the critical point J_c.

    Bonus question We have formulated SSE using the expansion of the trace in terms of the computational basis (these are the $|\alpha\rangle$ states). Then, all appearing matrix elements of diagonal and offdiagonal operators are of course also expressed in this language. In principle, we are free to choose any

basis we like. Can you formulate SSE using *translation invariant momentum states* $|\psi_{k=0}, \vec{n}\rangle$ which we have used for exact diagonalization? There, we have discussed how to get the matrix elements of the Hamiltonian in terms of representative states. The reason for restricting to $k = 0$ is that we know that at low temperature this sector dominates. Also, this sector is the only one which doesn't yield complicated phase factors, which might generate a sign problem. The advantage of such a formulation would be faster sampling (of all configurations related by translations at the same time) and faster convergence in β due to the restriction to $k = 0$. This is a completely open research problem. One possible outcome is that you can show why this doesn't work.