Final Project: Quantum Monte Carlo techniques

Andrea Nicolai (1233407), Karan Kabbur Hanumanthappa Manjunatha (1236383) , Iriarte Delfina (1231682)

April 26, 2021

Abstract

In this work, the ground state energy of the one dimensional antiferromagnetic Heisenberg spin chain, for spin-1/2 particles, in presence of varying external field h is determined by means of the Variational Monte Carlo(VMC), which is a Quantum Monte Carlo method that applies the variational principle to approximate the ground state of a quantum system. In particular, the trial wavefunction used for such problem is the Marshall-Jastrow function. Furthermore, to cross-check the correctness of the results obtained from VMC, the ground state of the Hamiltonian is found also using the Lanczos procedure. Moreover, some analysis of algorithmic complexity using CPU-time as a unit measure is pursued.

1 Introduction

For many body problems, exact solutions are not available except for certain models, such as the two-dimensional classical Ising model or certain one-dimensional models. In general, even with exact projection methods, the problem of exponential complexity 2^N is not solved, therefore some workarounds have to be thought about. Over the last decade efficient algorithms for classical Monte Carlo simulations have been generalized to quantum systems, allowing them to simulate highly complex systems [Werner *et al.* 2006].

Quantum Monte Carlo methods encompass a large family of numerical approaches that are versatile and are effective tools to handle quantum many-body systems. In addition, it is a stochastic method, therefore results cannot be predicted precisely. However, one can improve precision and efficiency by changing some condition at the starting point. In our problem this will take the form of a proper choice of the so-called trial wave function, that should resemble as much as possible the function one would like to find. To maximize our algorithm efficiency a lot of users effort shall be invested, since it is crucial for obtaining accurate and low-computationally effort results.

One of the most important tasks in quantum mechanics is the determination of the ground state energies for many body systems. The Variational Monte Carlo (VMC)[Sorella and Becca 2016] method is one of the most promising methods for highly accurate calculations for general systems. In the VMC algorithm, a variational wave function is introduced along with variational parameters: one has to obtain an approximation for the ground-state by optimizing these parameters, according to the variational principle. Since the method is relatively insensitive to the size of the system, it can be applied to large systems where some other methods are computationally not feasible.

The eigenvalue problem is of central importance in many physics problem and much thought has been devoted to the designing of efficient methods of finding them. Exact diagonalization of such systems can be quite formidable, even on the fastest computers, due to the exponential increase in the dimensions of the Hamiltonian matrix with increasing N. In order to obtain them in only few iterations, and in an economical fashion, the Lanczos method is used. The latter was conceived as a method to obtain tri-diagonalizing Hermitian matrices, and is based on the power method. One of the main advantages is that it avoids the problem of calculating and saving huge matrices in the computer memory by constructing a basis that directly leads to a tridiagonal matrix. It is an example of a family of projection techniques known as Krylov subspace methods.

In this work, a Variational Monte Carlo algorithm is proposed to simulate a 1D $\frac{1}{2}$-spins chain in the presence of a external field for the Anti-ferromagnetic Heisenberg Hamiltonian. The ground state energy of the Hamiltonian is hence obtained using the variational principle, and finally is compared with the one obtained by the exact Lanczos method.

2 Theory

The Hamiltonian for the Anti-ferromagnetic Heisenberg model takes the following form:

$$\hat{\mathcal{H}} = h\sum_{i=1}^{N}\sigma_z^i + \sum_{i=1}^{N}\sigma_x^{i+1}\sigma_x^i + \sum_{i=1}^{N}\sigma_y^{i+1}\sigma_y^i + \sum_{i=1}^{N}\sigma_z^{i+1}\sigma_z^i \tag{1}$$

where the first term in the Hamiltonian accounts for the energy due to an external field with intensity h applied along the $\hat{z}$ direction, while others are interaction terms between nearest neighbors spins. Moreover, $\sigma_i, i = \{x, y, z\}$ are the Pauli matrices: $\sigma_x = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, $\sigma_y = \left(\begin{smallmatrix} 0 & -i \\ i & 0 \end{smallmatrix}\right)$ and $\sigma_z = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$. With coupling constant $J_x = J_y = J_z = J = 1$ being considered here and with $J > 0$ gives us isotropic anti-ferromagnetic XXX Heisenberg chain. The Hamiltonian with Periodic Boundary Conditions(PBC) $S_{N+j} = S_j$ can be rewritten using the operators defined as $S_i = \frac{1}{2}\sigma_i, i = \{x, y, z\}$, in such a way that:

$$\hat{\mathcal{H}} = \frac{h}{2}\sum_{i=1}^{N}S_z^i + \frac{1}{4}\left(\sum_{i=1}^{N}S_x^{i+1}S_x^i + \sum_{i=1}^{N}S_y^{i+1}S_y^i + \sum_{i=1}^{N}S_z^{i+1}S_z^i\right) \tag{2}$$

A further step in the analysis is the introduction of the spin-flip operators in order to make calculations more convenient to deal with. Namely,

$$S_j^+ = S_j^x + iS_j^y \qquad\qquad S_j^- = S_j^x - iS_j^y \tag{3}$$

whose action either raises or lowers spin states:

$$S^+|\downarrow\rangle = |\uparrow\rangle \qquad S^+|\uparrow\rangle = 0 \qquad S^-|\uparrow\rangle = |\downarrow\rangle \qquad S^-|\downarrow\rangle = 0 \tag{4}$$

Spin commutation relations read:

$$\left[S_j^z, S_{j'}^{\pm}\right] = \pm S_j^{\pm}\delta_{jj'} \qquad\qquad \left[S_j^+, S_{j'}^-\right] = \pm 2S_j^z\delta_{jj'} \tag{5}$$

And in particular interaction term can be rewritten as:

$$H = J\sum_{i=1}^{L}\vec{S}_i\cdot\vec{S}_{i+1} = J\sum_{i=1}^{L}\left\{S_i^zS_{i+1}^z + \frac{1}{2}\left(S_i^+S_{i+1}^- + S_i^-S_{i+1}^+\right)\right\} \tag{6}$$

In the absence of a magnetic field ($h = 0$), the ground state of the classical model is the **Neel state**, in which all spins are antiparallel with respect to one another. A crucial property of this model is that the total projection along z-axis of the spin $S_z = \sum_{i=1}^{L}s_z$ commutes with the Hamiltonian, due to the presence external field h which is assumed to be parallel of z-axis. Indeed:

$$[H, S_z] = 0 \tag{7}$$

Furthermore, since knowing the individual spin operator, the composite spin operator can be constructed by using the tensor product structure of the Hilbert space:

$$\sigma_z^i = \mathcal{I}^1\otimes\mathcal{I}^2\otimes\ldots\otimes\sigma_z^i\otimes\ldots\otimes\mathcal{I}^N \tag{8}$$

and the $x-$interaction term results in:

$$\sigma_x^i\sigma_x^{i+1} = \mathcal{I}^1\otimes\mathcal{I}^2\otimes\ldots\otimes\sigma_x^i\otimes\sigma_x^{i+1}\otimes\ldots\otimes\mathcal{I}^N \tag{9}$$

Equation 9 can be trivially extended to the case of the y and z interaction. For the Variational Monte Carlo, a 1D lattice where $J = 1$ is considered, as shown in Fig. 1, leading to the anti-ferromagnetic Hamiltonian.

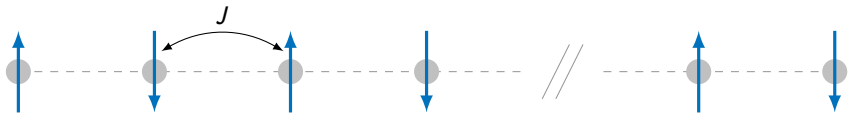


Figure 1. Lattice model for the Heisenberg hamiltonian

2.1 Variational Monte Carlo

The variational method is a powerful tool to estimate the ground state energy of a quantum system, and some of its low lying excited states, when an exact solution cannot be found. Variational Monte Carlo (VMC) relies on the variational principle to approximate the ground state of quantum many-body systems by optimizing the parameters of a variational wave function, which is used to sample quantum mechanical expectation values. The required Monte Carlo techniques for VMC are conceptually simple, but the practical application may turn out to be rather tedious and complex, relying on a good starting point for the variational wave functions. These wave functions should include as much as possible of the inherent physics to the problem, since they form the starting point for a variational calculation of the expectation value of the Hamiltonian $\hat{H}$.

In the variational Monte Carlo method, the ground state of a Hamiltonian $\hat{H}$ is approximated by some trial wave function $|\Psi_T\rangle$, whose form is chosen following a prior analysis of the physical system being investigated.

Theorem 2.1 (The variational principle)**.** *Given a Hamiltonian H and a trial wave function, the variational principle states that the expectation value of $< H >$, defined through:*

$$\langle H\rangle = \frac{\langle\Psi_\alpha|H|\Psi_\alpha\rangle}{\langle\Psi_\alpha|\Psi_\alpha\rangle} \tag{10}$$

is an upper bound to the exact ground state energy E_0 of Hamiltonian $\hat{H}$, i.e.:

$$E_0 \leq \langle H\rangle \tag{11}$$

Thus by varying Ψ until the expectation value of $\hat{H}$ is minimized, approximation can be obtained to the wave function and energy of the ground-state.

Proof. Suppose that Ψ_n and E_n are the true eigenstates and eigenvalues of $\hat{H}$, i.e.:

$$H\Psi_n = E_n\Psi_n \tag{12}$$

Furthermore, let $E_0 < E_1 < \dots$ so that Ψ_0 is
the ground state. The Ψ_n are assumed to be orthonormal:

$$\langle\Psi_n|\Psi_m\rangle = \delta_{n,m} \tag{13}$$

If our trial wavefunction Ψ is properly normalized, then it can be written:

$$\Psi = \sum_n c_n\Psi_n \qquad \sum_n |c_n|^2 = 1 \tag{14}$$

Now, the expectation value of $\hat{H}$ takes the form:

$$\langle\Psi|H|\Psi\rangle = \langle\sum_n c_n\Psi_n|H|\sum_m c_m\Psi_m\rangle = \sum_{n,m} c_n * c_m\langle\Psi_n|H|\Psi_m\rangle = \sum_n c_n * c_m E_m\langle\Psi_n|\Psi_m\rangle = \sum_n E_n|c_n|^2 \tag{15}$$

Therefore:

$$\langle\Psi|H|\Psi\rangle = E_0|c_0|^2 + \sum_{n>0} E_n|c_n|^2 \tag{16}$$

Notice that:

$$|c_0|^2 = 1 - \sum_{n>0} |c_n|^2 \tag{17}$$

Combining the previous equations:

$$\langle \Psi | H | \Psi \rangle = E_0 + \sum_{n>0} (E_n - E_0) |c_n|^2 \tag{18}$$

Now, the second term on the right-hand side of the above expression is positive definite, since $E_n - E_0 > 0$ for all $n > 0$. Hence, the desired result is obtained:

$$\langle \Psi | H | \Psi \rangle \geq E_0 \tag{19}$$

□

Therefore the problem reduces to finding the expectation value of $\hat{H}$ for some wave function with an adjustable (i.e. variational) parameter α, and the easiest way to evaluate it is by means of Monte Carlo techniques, in particular using the Metropolis algorithm.

Considering a trial wavefunction $|\Psi_\alpha>$ with a set of parameters α, the expectation value of the energy is calculated for the many-body Hamiltonian $\hat{H}$:

$$E(\alpha) = \frac{\langle \Psi_\alpha | H | \Psi_\alpha \rangle}{\langle \Psi_\alpha | \Psi_\alpha \rangle} = \frac{\sum_x e_L(x) \Psi_\alpha^2}{\sum_x \Psi_\alpha^2} \tag{20}$$

where e_L is the so-called **local energy:**

$$e_L(x) = \frac{\langle x | H | \Psi_\alpha \rangle}{\langle x | \Psi_\alpha \rangle} \tag{21}$$

The local energy can be obtained by noticing that Eq. 21 can be rewritten as:

$$e_L(x) = \frac{\langle x \mid H \mid \psi \rangle}{\langle x \mid \psi \rangle} = \langle x \mid H \mid x \rangle + \sum_{x' \neq x} \langle x \mid H \mid x' \rangle \frac{\langle x' \mid \psi \rangle}{\langle x \mid \psi \rangle} \tag{22}$$

Therefore, the calculation of E(g) can be recasted as the average of a random variable $e_L(x)$ over a probability distribution p_x:

$$p_x = \frac{\Psi_\alpha^2(x)}{\sum_x \Psi_\alpha(x)^2} \tag{23}$$

Finally, the evaluation of the expectation value of the energy can be thought as the mean of the random variable $e_L(x)$ of the visited configurations:

$$E(\alpha) = \frac{1}{N} \sum_{n=1}^{N} e_L(X_n) \tag{24}$$

The Model

The anti-ferromagnetic spin-1/2 Heisenberg model on a 1D lattice with L sites with periodic boundary condition is considered. In particular, the trial wavefunction in the basis of configuration $\{x\}$ with definite value of $s_i^z = \pm\frac{1}{2}$ is given by Jastrow-Marshall type:

$$\psi(x) = \text{Sign}_M(x) e^{\frac{\alpha}{2} \sum_{i \neq j} v_{i,j}^z (2S_i^z)(2S_j^z)} \tag{25}$$

where:

$$\text{Sign}_M(x) = (-1)^{\sum_{i=1}^{L/2}(S^z_{2i}+1/2)} \tag{26}$$

is the so called Marshall sign which is determined by the total number of up spins in the even sites, α is the variational parameter and then the form of spin Jastrow factor

$$v^z_{i,j} = In(d^2_{i,j}) \tag{27}$$

depends parametrically on the so called cord-distance $d_{i,j}$ between the two sites:

$$d_{i,j} = 2sin(\pi|i-j|/L) \tag{28}$$

and it correspond to the number of up spins in one sublattice contained in the spin configuration. This Jastrow factor is applied to electron gases, atoms and molecules and is found to give significant improvement at VMC levels. However, note that this is true only for **even number of L sites**, therefore our code was implemented accordingly.

2.2 Lanczos method

The Lanczos diagonalization algorithm comes handy when dealing with many-body quantum problems, which are generally described by very large matrices. It is an adaptation of power methods, that are used to find the m largest eigenvalues in module and their relative eigenvectors of a $n \times n$ Hermitian matrix, where usually $m \leq n$. This approach is particularly suited when dealing with large and sparse Hamiltonians.

The Algorithm

The algorithm takes as input the Hermitian matrix A, whose dimension is $n \times n$, the number of iterations m (set by default to be equal to n), and a random vector $\vec{v}_1$ whose euclidean norm is 1, and its elements are $v_i \in C^n$, $i = 1, ..., n$.

As a first step, the operator A is applied to the vector $\vec{v}_1$, thus obtaining $\vec{\omega'}_1 = A\vec{v}_1$. Then, the new state $\vec{\omega'}_1$ is decomposed in its normalized parallel and perpendicular components with respect to original state $\vec{v}_1$, respectively $\vec{\alpha}_1$ and $\vec{\omega}_1$. Successively, the norm of $\vec{\omega}_1$ is computed and the new vector $\vec{v}_2$ is defined, hence applying A and iterating the whole process. More formally, the algorithm looks like:

1. Initial iteration steps:
   (a) Let $\vec{\omega}'_1 = A\vec{v}_1$
   (b) Let $\vec{\alpha}_1 = \vec{\omega}' \cdot \vec{v}_1 = \vec{\omega}'^*_1 \vec{v}_1$
   (c) Let $\vec{\omega}_1 = \vec{\omega}'_1 - \vec{\alpha}_1 \vec{v}_1$
2. For $j = 2, ..., m$ do:
   (a) $\beta_j = ||\vec{\omega}_{j-1}||$ (euclidean norm)
   (b) If $\beta_j \neq 0$, then let $\vec{v}_j = \vec{\omega_{j-1}}/\beta_j$ else assign $\vec{v}_j$ as a random vector with euclidean norm 1 that is orthogonal to all the remaining $\vec{v}_1, ..., \vec{v}_{j-1}$
   (c) Let $\vec{\omega}'_j = A\vec{v}_j$
   (d) Let $\vec{\alpha}_j = \vec{\omega}'^*_j \vec{v}_j$
   (e) Let $\vec{\omega}_j = \vec{\omega}'_j - \vec{\alpha}_j \vec{v}_j - \beta_j \vec{v}_{j-1}$

Whereas the output of the algorithm the following is: given a matrix V with orthonormal columns $\vec{v}_1, ..., \vec{v}_m$, a tridiagonal real symmetric matrix $T = V^* A V$ of size $m \times m$ is obtained. T takes the form as below:

$$T = \begin{pmatrix} \alpha_1 & \beta_2 & & & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \alpha_3 & & \\ & \ddots & \ddots & \ddots & \\ & & & \beta_{m-1} & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ 0 & & & \beta_m & \alpha_m \end{pmatrix} \tag{29}$$

Finally, the only remaining task is to diagonalize the tridiagonal matrix, thus obtaining the energy eigenvalues the exact ground state energy of our Heisenberg model.

3 Code Development

3.1 Variational Monte Carlo

As explained in Section 2.1, the Variational Monte Carlo needs a few ingredients. To begin with, recall that the Marshall-Jastrow trial wavefunction given by equation 25, is needed to compute the local energy 22. More specifically, it is required that the quotient between $\Psi(x')$ and $\Psi(x)$, where x' is a configuration obtained from x by flipping spin k and spin $k+1$ and this is done in `wfRatio() function`, shown in Listing 1. The function takes as argument a local configuration `x'`, the index of the spin `k`, the variational parameter `alpha`, the Spin-Jastrow factor `v` given by Eq. 27 and Eq. 28, the spin matrix `sMatrix` whose entries are simply the products between spin i and spin j and a logical value `updateS` that update the matrix `sMatrix` if TRUE.

As an update rule, it has been decided to preserve the total spin projection S_z along z-axis. The only way to do it, was to propose a spin-flipping of two consecutive and antiparallel spins, in such way that the ratio of the Marshall sign given in Eq. 26 will always generate -1, thereafter `signM_ratio` is set to that value. Furthermore, the next spin to k is consider by calling the `function nextElementOf()`, which simply returns the next spin to k considering periodic boundary condition. Furthermore, the only contribution of the ratio of the wavefunction 25 will be given by these two spins. Therefore, in `expSum` the following contributions arise:

$$expSum = -(\sum_{i=0}^{L} v_{k,i} S_k S_i + \sum_{i=0}^{L} v_{k+1,i} S_{k+1} S_i + \sum_{i=0}^{L} v_{i,k} S_i S_k + \sum_{i=0}^{L} v_{i,k+1} S_i S_{k+1}) \quad \forall k \neq i \tag{30}$$

Notice that the elements $S_k S_i$ are nothing else but the $k-$th row and $i-$th column of `sMatrix`, and in particular `sMatrix` has the form of a lower triangular matrix. Consequently many of the terms of the previous equation are null, allowing to write:

$$expSum = -(\sum_{i=0}^{k} v_{k,i} S_k S_i + \sum_{i=0}^{k+1} v_{k+1,i} S_{k+1} S_i + \sum_{i=k}^{L} v_{i,k} S_i S_k + \sum_{i=k+1}^{L} v_{i,k+1} S_i S_{k+1}) \quad \forall k \neq i \tag{31}$$

If `updateS = True` then the spin matrix is updated. For the same reasoning as the `expSum`, again notice that there are only four terms being updated similarly as before. Finally, the `function wfRatio` returns the total ratio of the wave function.

```
def wfRatio( x, k, alpha, v, sMatrix, updateS = False):
    L = len(x)
    kNext = nextElementOf(k, L)
    #   Set the ratio of the marshal sign
    signM_ratio = -1
    #    Do the (half) sums in the ratio of the exponentials
    expSum = 0
    #     i = k
    for j in range(k):
        if j != kNext:
```

```
            expSum += -v[k,j]*sMatrix[k,j]
    #      i = kNext
    for j in range(kNext):
        if j != k:
            expSum += -v[kNext,j]*sMatrix[kNext,j]
    #     j = k
    for i in range(k+1, L):
        if i != kNext:
            expSum += -v[i,k]*sMatrix[i,k]
    #     j = kNext
    for i in range(kNext+1, L):
        if i != k:
             expSum += -v[i,kNext]*sMatrix[i,kNext]

    #  Update sMatrix if required
    if updateS == True:
        #  i = k
        for j in range(k):
            if j != kNext:
                sMatrix[k,j] *= -1
        # i = kNext
        for j in range(kNext):
            if j != k:
                sMatrix[kNext,j] *= -1
        # j = k
        for i in range( k+1, L):
            if i != kNext:
                sMatrix[i,k] *= -1
        # j = kNext
        for i in range(kNext+1, L):
            if i != k:
                sMatrix[i,kNext] *= -1
    return signM_ratio*np.exp(alpha*2.0*expSum), sMatrix
```

Listing 1. Wave function ratio of the flipped state and the original configuration state of spins

Apart from the wavefunction, another important quantity to compute when dealing with Variational Monte Carlo is the local energy given by Eq. 22. In Listing 2 the implementation of the `function localEnergy()` is shown. This function takes as input a local configuration `x`, the variational parameter `alpha`, the spin Jastrow factor `v` and the spin product matrix `sMatrix`.

The local energy can be computed into two chunks, one when $x' \neq x$ and the other when $x' = x$ as observed in Eq. 22. For the first, notice that the only meaningful contribution is given by $\sum_{i=1}^{L} S_i^z S_{i+1}^z$, in the code is given in line 43 of listing 2 which is simply the cumulative sum of product of successive neighbouring spins.

For the latter, notice that the contribution to the local energy for flipped state and the original state is given by the term:

$$\sum_{x' \neq x} \langle x \mid H \mid x' \rangle \frac{\langle x' \mid \psi \rangle}{\langle x \mid \psi \rangle} \tag{32}$$

where the sum runs through all of configuration x that are different than x'. Now consider a state $|x_{conf}\rangle$, notice that applying the Hamiltonian term $(S_i^+ S_{i+1}^- + S_i^- S_{i+1}^+)$ to any state will lead into flipping the i-th and the $i+1$th spin with opposite signs. Therefore, if the *two neighbours spins* have the same alignment, its overall contribution will be zero. However, the contribution coming from $S_i^z S_{i+1}^z$ will not, and the element $\langle x \mid H \mid x' \rangle$ will be non zero only if $x = x'$. If spins have opposite signs, the energy contribution given by $\langle x \mid H \mid x' \rangle \frac{\langle x' \mid \psi \rangle}{\langle x \mid \psi \rangle} = 1/2 \frac{\langle x' \mid \psi \rangle}{\langle x \mid \psi \rangle}$ since $S_i^z S_{i+1}^z$ term vanishes when $x \neq x'$. This term is computed in the `eLocal` variable. According to what has been just stated, the whole process involving dot products and vector multiplications can be shortened into a few easy and compact steps, as shown in the code.

```
# Add the contribution from x'!=x and calculate nPairs
for k in range(L):
    kNext = nextElementOf(k, L);
    if x[k] != x[kNext]:
        eBuffer, sMatrix = wfRatio( x, k, alpha, v, sMatrix, False)
        eLocal += eBuffer

eLocal = 0.5*eLocal

#adding contribution x'==x
for k in range(L):
        eLocal += 0.5*0.5*x[k]*x[nextElementOf(k, L)]
```

Listing 2. Main code of the `localEnergy(x, alpha, v, sMatrix)` function

It is known that the ground state of the anti-ferromagnetic Heisenberg model in absence of any external field is simply given the state known as Neel state, where all spins are alternating and antiparallel with respect to one another. However, when adding a non zero field h term, the spins will tend to align accordingly due to the presence of this field. Exploiting the fact that $\hat{H}(h = 0)$ and $\hat{H}(h \neq 0)$ have the same eigenstates, several simulations for different and fixed S_z can be run. To better explain how to fix in advance S_z, let us consider an example. Given a size (or number of spins) $nn = 4$, the only possible values for the spin total projection along z-axis are $S_z = \pm 4, \pm 2, 0$. Therefore, for each value of S_z the only possible configurations are:

- S_z = -4, Config: {[-1, -1, -1, -1]}
- S_z = -2, Config: {[1, -1, -1, -1], [-1, 1, -1, -1], [-1, -1, 1, -1], [-1, -1, -1, 1]}
- S_z = 0, Config: {[1, 1, -1, -1], [1, -1, 1, -1], [1, -1, -1, 1], [-1, 1, 1, -1], [-1, 1, -1, 1], [-1, -1, 1, 1]}
- S_z = 2, Config: {[1, 1, 1, -1], [1, 1, -1, 1], [1, -1, 1, 1], [-1, 1, 1, 1]}
- S_z = 4, Config: {[1, 1, 1, 1]}

For each value of S_z, given a list of configurations, a single configuration is randomly selected and that is set to be our initial configuration to start our Monte Carlo simulation with.

After initializing the configuration with a given total of S_z spins the Monte Carlo is performed. Two main loops are performed in it, the first one is over a range of variational parameter `alpha`, while the second one is through all the Monte Carlo steps, as shown in Listing 3.

```
    #Select a new move (i.e. flip k and k+1)
    k = round(random.random()*(L-1))
    kNext = nextElementOf(k, L)

    # Do Metropolis only if we flip different spins,
    # because of the constraint on the total spin
    if (x[kNext] != x[k]):
        sNew = deepcopy(sMatrix)
        mRatio, sNew = wfRatio(x, k, alpha, v, sNew,h,  True)
        mRatio = pow(mRatio, 2.0)

        r = np.random.rand()

        if ((r<mRatio and mRatio<1) or (mRatio>1)):
            acc +=1
            # Update x by ASSUMING we flip two different spins.
            # The assumption holds because of the "if" that skips
            # this section if we flip two equal spins.

            x[k] = -deepcopy(x[k])
            x[kNext] = -deepcopy(x[kNext])

            #Update the sMatrix
```

```
            sMatrix = deepcopy(sNew)

            # Calculate the local energy
            ELocal = localEnergy(x, alpha, v, sMatrix, h, prod)

    #Update the cumulant of the local energy
    ElocalCumulative += ELocal

    E.append(ElocalCumulative/(LenSim*L))
```

Listing 3. Main part of Variational Monte Carlo - Metropolis algorithm

The number of Monte Carlo steps is given by `LenSim`, which should be of the order of 10^5 or 10^6 to return results of 10^{-5} precision. At the start of the loop, a spin is selected randomly and its next neighbour is checked: only in case they are opposite the Metropolis algorithm is then proceed. The wave function ratio of the flipped state with respect to original state is computed and it is squared to give ratio of probability density representation. A random number $r \in [0, 1)$ is generated, and only if the condition where either both $mRatio < 1$ and $r < mRatio$ are simultaneously satisfied, or alternatively $mRatio > 1$, the proposed move is accepted: spins at the indices k and $k + 1$ are flipped, finally computing the local energy. The latter is cumulatively added at every iteration, thus saving memory. Indeed, after the end of all the Monte Carlo iterations, it will be divided by the total number of iterations to obtain the ground energy of the system, finally normalizing it to the total number of spins L, thus obtaining the ground state energy per site.

In order to compute the ground state energies of the Hamiltonian subject to an external magnetic field h, the following function is consider:

$$F(m) = E(m) - h \cdot m \tag{33}$$

where $m = 0.5 \cdot (\sum_{i=0}^{L} S_i^z)/L$ is the average magnetization, $E(m)$ is a collection of minimum energies found previously thanks to VMC for different values of fixed S_z, and h is the external field. This can be done since it is known from theory that Hamiltonian $\hat{H}$ commutes with S_z, and hence m. This implies that eigenvalues for H, i.e. energies $E(m)$, can be computed for fixed m and setting $h = 0$. Once having computed this set of energies $\{E(m)\}$, one can simply find the ground state energy by simply minimizing the Eq. 33 for different values if h. This is implemented in the `function get_energy_with_field()`. Moreover, the condition of keeping S_z fixed was taken care according to our choice of spin-flipping rule. In this way, it can be avoid to run simulations for every value that h, thus running only $L + 1$ simulations that is the number of possible values m can take.

Listing 4 instead presents the main implementation of the finding of the ground state by means of Variational Monte Carlo. For each number of particles `nn`, the optimal variational parameter and energy are found for all the different values of average magnetization `m`. Afterwards, the initial magnetization `m` with the lowest energy for a given `nn` and external field h is found by calling the already mentioned `function get_energy_with_field()`, which returns also the actual values of the energy.

```
for nn in [2,4,6,8,10,12]:
    for i in range(nn+1):
        #Computing best alpha
        E,alpha, xInit, t = main(L=nn,LenSim=100000, h=h_field, place_one=i)
        E_min = min(E)
        magn  = 0.5*np.mean(xInit)
        m_values.append(magn)
        E_vs_m.append(E_min)

        optE_alpha = alpha[np.argmin(E)]

    for h_val in np.arange(0, 3.05, 0.05):
        energies_ok = get_energy_with_field(E_vs_m, m_values, h_val)
        list_dic.append({"nn": nn, "h_field": h_val, "energy": min(energies_ok),
                         "optimal_alpha":alpha_list[np.argmin(energies_ok)],
```

```
                        "Init_config":str(xInit_list[np.argmin(energies_ok)]),"cpu_time(min)
    ":t_list[np.argmin(energies_ok)]})


def get_energy_with_field(energies, magnetizations, h_field):
    en_arr = np.array(energies)

    final_energies = np.array([energy - magn*h_field for energy, magn in zip(en_arr,
    magnetizations)])
    return final_energies
```

Listing 4. Search of ground states by Variational Monte Carlo

3.2 Lanczos Method

Creation of the Hamiltonian

The Hamiltonian in Eq.1 is built, by means of QUTIP [Johansson, Nation, and Nori 2012, 2013] which is a quantum library available for PYTHON. As seen in Eq 1, the Hamiltonian is composed by four terms: the interaction terms in x, y, z and the external field term h.

Recall that the structure of the interaction and field terms can be written by using the tensor products as given in Eq. 9 and Eq. 8. `Function Hx_int()` creates the interaction term along the $x-$ axis of the $1D$ Heisenberg model with no boundary condition. It takes as input the number `nn` of spins, the dimension `dim` of the Hilbert space, and the interaction strength `int_strength`. This function is presented in Listing 5. With no loss of generality, this can also be extended for the case of y and z interactions.

```
#sigma_x interactions
def Hx_int(dim, nn, int_strength = 1.):

    hx_int = create_emptyH_Qobj(dim, nn)
    #compute sigma x and interaction
    for index_ii in range(0, nn - 1):
        #create an operator for a single particle in 2-dim
        temp_obj = create_emptyH_Qobj(dim, 1)
        if index_ii == 0:
            temp_obj = sigmax()/2
        else:
            temp_obj = qeye(dim)
        for index_jj in range(1, nn):
            if ((index_ii == index_jj)|(index_jj == (index_ii+1))):
                temp_obj = tensor(temp_obj, sigmax()/2)
            else:
                temp_obj = tensor(temp_obj, qeye(dim))
        hx_int += int_strength*Qobj(temp_obj)
        #free RAM and delete temp_object, later it'll be recreated
        del temp_obj

    matrix_debug(DEBUG_FLAG,hx_int,dim,nn)

    return hx_int
```

Listing 5. Implementation of the interaction term along the $x-$ axis of the 1D Heisenberg model

Listing 6 shows instead the implementation of the contribution to the energy of the system of an external magnetic field h, aligned parallel to z-axis of the Heisenberg Hamiltonian. The function `H_ext()` takes as input the dimension, the number of particles and the interaction field `h_field` and it returns the field term by computing the tensor product as in equation 8.

```
def H_ext(dim, nn, h_field):
    h_ext = create_emptyH_Qobj(dim, nn)
    for index_ii in range(0,nn):
```

```
        if (index_ii == 0):
            temp_obj = sigmaz()/2
        else:
            temp_obj = qeye(dim)
        for index_jj in range(1,nn):
            if (index_jj == index_ii):
                temp_obj = tensor(temp_obj,sigmaz()/2)
            else:
                temp_obj = tensor(temp_obj,qeye(dim))
        h_ext += Qobj(temp_obj)
        #free RAM and delete temp_object, later it'll be recreated
        del temp_obj
    matrix_debug(DEBUG_FLAG,h_ext,dim,nn)
    return h_ext*h_field
```

Listing 6. Implementation of the field term of the 1D Heisenberg model

With Periodic Boundary Conditions(PBC):

```
def Hx_int_pbc(dim, nn, int_strength = 1.):
    hx_int = create_emptyH_Qobj(dim, nn)

    for index_ii in range(0, nn ):
        temp_obj = create_emptyH_Qobj(dim, 1)
        if index_ii == 0:
            temp_obj = sigmax()/2
        else:
            temp_obj = qeye(dim)
        for index_jj in range(1, nn):
            if ((index_ii == index_jj)|(index_jj == (index_ii+1))):
                temp_obj = tensor(temp_obj, sigmax()/2)
            else:
                temp_obj = tensor(temp_obj, qeye(dim))

        if index_ii==nn-1:
            temp_last_term = sigmax()/2
            for kk in range(1,nn-1):
                temp_last_term = tensor(temp_last_term,qeye(2))
            hx_int += int_strength*tensor(temp_last_term,sigmax()/2)
            del temp_last_term
        else:
            hx_int += int_strength*Qobj(temp_obj)
        del temp_obj

    matrix_debug(DEBUG_FLAG,hx_int,dim,nn)

    return hx_int
```

Listing 7. Implementation of the interaction term of the 1D Heisenberg model with PBC

Assuming for this example $nn = 3$, the interaction term along x-axis in the Hamiltonian with PBC takes the form:

$$\mathcal{H}_x^{int} = \sum_{i=1}^{3} \sigma_x^i \sigma_x^{i+1} = \sigma_x \otimes \sigma_x \otimes 1 + 1 \otimes \sigma_x \otimes \sigma_x + \sigma_x \otimes 1 \otimes \sigma_x \tag{34}$$

In the Eq. 34, which is taken as an example to explain how PBC is implemented, the first 2 terms are found in the same way as the one without PBC. The only difference noticeable refers to the last term of Eq. 34, and it is given by lines 20 - 27 in the above Listing 7.

3.3 Lanczos Implementation

Following the steps provided in section 2.2, a `function lanczos_full()` is created. The function performs the Lanczos algorithm to compute the largest in module eigenvalues of matrix A. It takes as input the matrix A, a starting random-valued vector `v0` and the number of Lanczos iterations `m`, which is in turn the size of the Krylov subspace. In particular, the diagonalization of the tri-diagonal matrix given by equation 29 is done by using the `function eigh_tridiagonal()` provided by SCIPY.

```
def lanczos_full(A,v0,m,eps=None):

    A.dot(Q[0,:],out=r)

    a[0] = np.vdot(Q[0,:],r).real
    _axpy(Q[0,:],r,-a[0])
    #compute the euclidean norm
    b[0] = np.linalg.norm(r)

    #iterate the real algorithm, where the loop breaks either when m iterations are reached
    #or euclidean norm of diagonal matrix elements is smaller than a certain threshold
    i = 0
    for i in range(1,m,1):
        v[:] = Q[i-1,:]
        np.divide(r,b[i-1],out=Q[i,:])

        A.dot(Q[i,:],out=r)

        _axpy(v,r,-b[i-1])
        a[i] = np.vdot(Q[i,:],r).real
        _axpy(Q[i,:],r,-a[i])

        #compute the euclidean norm
        b[i] = np.linalg.norm(r)

        #when euclidean norm of diagonal matrix elements is small, break the iteration
        if b[i] < eps:
            m = i
            break

    #finally return eigenvalues of the tridiagonal matrix
    E = eigh_tridiagonal(a[:m],b[:m-1],eigvals_only=True)

    return E
```

Listing 8. A piece of code implementing Lanczos algorithm right after initialization step. A is the matrix whose eigenvalues are to be found. Q is a n x m matrix to store elements of Lanczos basis, m is the number of iterations to be performed, v is the vector passed as input, eps is a threshold to eventually stop Lanczos iteration if euclidean norm of the vector is smaller than it.

One should note the presence of the function *_axpy*, which is defined by the means of @njit decorator (from *numba* [developers 2021] library), which allows to perform operations that are compiled to machine code "just-in-time" for execution at almost a at native machine code speed. The function, which exploits Numpy objects that are well suited for the aforementioned library, takes the form:

```
@njit
def _axpy(x,y,a):
    for i in range(x.size):
        y[i] += a*x[i]
```

Listing 9. Function was implemented exploiting njit decorator to allow machine to run the code almost at a native machine speed.

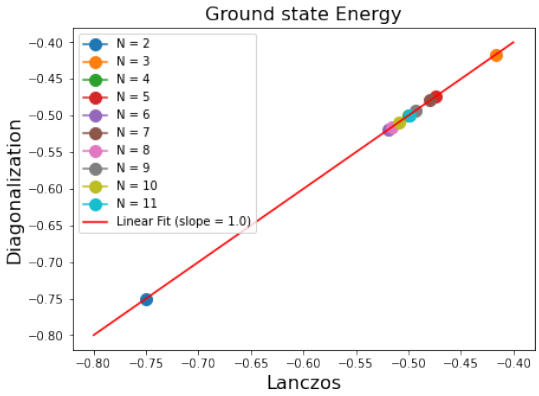
4 Results

4.1 Lanczos method

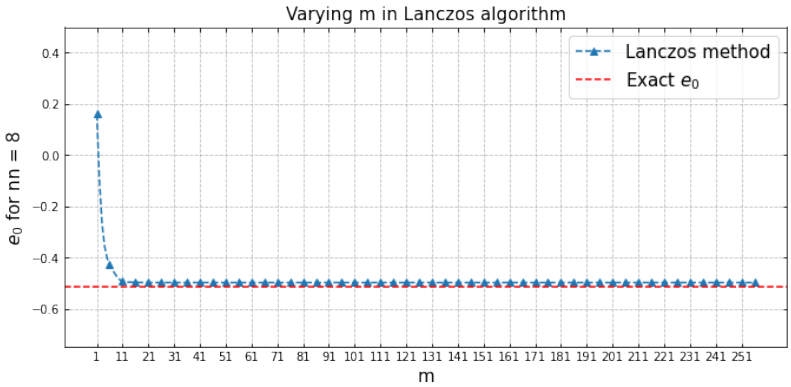
The ground state energy found for the Heisenberg model using Lanczos technique is presented in table 1, for different values of spins $\in [2, 11]$. Number of particles was chosen to be in such range due to memory errors that have arisen when dealing with very large matrices. The results obtained from Lanczos method are found to be exact with the one which has been found by direct diagonalization technique, moreover it has been cross checked to be perfectly correct from the function `H.eigenenergies()` of `Qutip` Python library. Thus, it is expected that E_0 per site values follow a linear trend as seen from the Fig. 2a.

nn	Direct diagonalization	Lanczos	Qutip
2	-0.750000	-0.750000	-0.750000
3	-0.416667	-0.416667	-0.416667
4	-0.500000	-0.500000	-0.500000
5	-0.473607	-0.473607	-0.473607
6	-0.519672	-0.519672	-0.519672
7	-0.479311	-0.479311	-0.479311
8	-0.516052	-0.516052	-0.516052
9	-0.492905	-0.492905	-0.492905
10	-0.509221	-0.509221	-0.509221
11	-0.499274	-0.499274	-0.499274

Table 1. E_0 values obtained for Heisenberg model and no external field (h = 0 case), using direct diagonalization, Qutip library and Lanczos method for different values of spins.



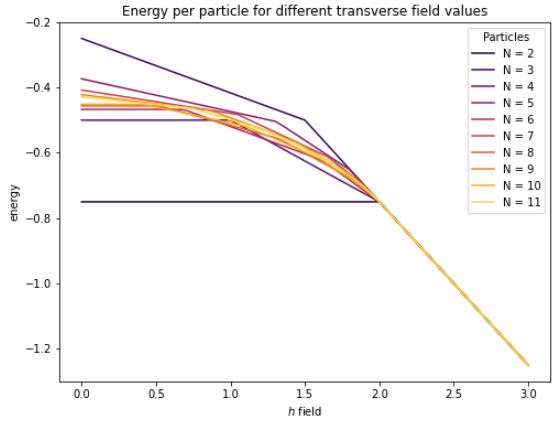
(a) Comparison of ground state E_0 values for Heisenberg XXX model using simple diagonalization technique with Lanczos method.



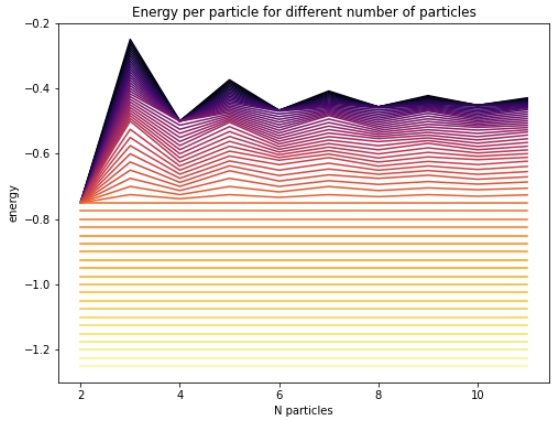
(b) Ground state energy E_0 with respect to the number of Lanczos iterations, m for N = 8 spins.

Figure 2. Results obtained for Lanczos iteration

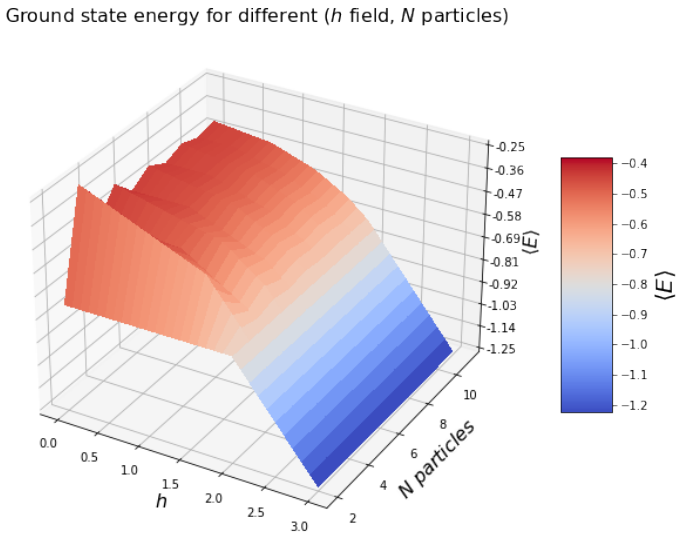
Another study has been done with regards to Lanczos iteration number m as shown in Fig. 2b. From theory it is known that m can take value at most n, where n is the size of the Hermitian matrix. However, it can be noticed that even with $m = 11$ iterations the algorithm converges quickly to the exact value of ground state energy, in the case of $N = 8$, which results in a matrix of dimension $2^N \times 2^N = 2^8 \times 2^8 = 256 \times 256$. So, for any values of N, it is not necessary to consider the largest number of Lanczos iterations $m = 2^N$, but just few handful iterations are enough for the convergence to the exact ground state.



(a) Variation of ground state energy E_0 with respect to external magnetic field h with each line in the plot representing given N spins in 1D lattice of Heisenberg model.



(b) Variation of ground state energy E_0 with respect to the external magnetic field h. Each line represents a different value for h in the range $[0, 3]$ with steps of $\Delta h = 0.05$: the darker the line, the lower the h. Energy per site is plotted as function of the number of particles N considered.



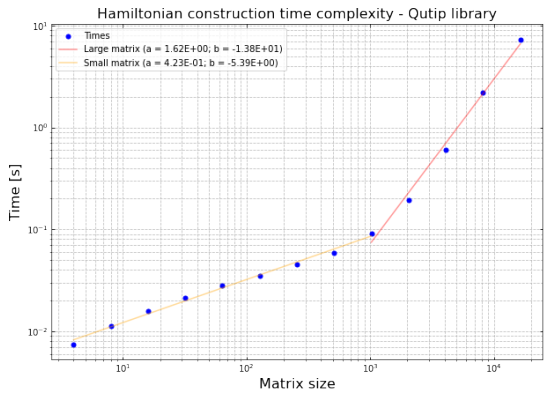
(c) 3 dimensional representation of variation of ground state energy E_0 (found by Lanczos method) with respect to both external field h and number of spins in 1D lattice of Heisenberg XXX model.

Figure 3. Results from Lanczos method.

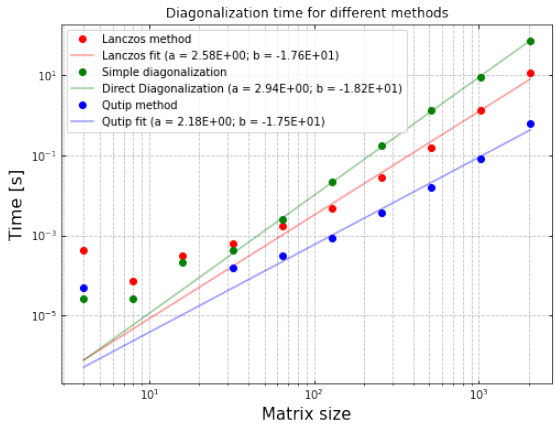
As seen from Fig. 3a, except for $N = 2$, all the ground state energy values when $h = 0$ are different and belong to the range $E_0 \in [-0.5, -0.2]$. Whereas when h field increases the energy in turn decreases. Starting from $h \sim 1.9$ all the curves begins to coincide. The Fig. 3b represents how ground state energy per particle changes for different number of particles, and different values of external field h are related to different lines. The lighter the color of the line, the larger h, with this parameter in the range $[0, 3]$. One can easily note as the instability of the ground state energy eigenvalues decreases with respect to h, in addition fluctuations decrease as N increases. Larger values of

h lead to an independence of the ground state energy with respect of the number of particles N, being the curve flat. This behavior can be noted in Fig. 3a, too, where ground state energy does depend only on the module of the external field. Finally, Fig 3c shows the $3D$ visualization of variational ground state energy per site, with respect to both number of particles N and external magnetic field h.

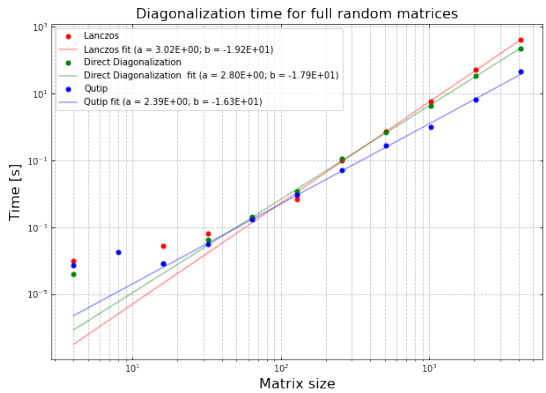
Figure 4 present how time complexity scales with respect to different matrix sizes. One has to recall that matrix sizes scale, in turn, as 2^N where N is the number of 1/2-spin particles considered. Moreover, when plotting in $log - log$ scale, one notices as times can be fitted by a straight line when matrices size is above ~ 30. Therefore a fit with a straight line of the kind $f(x) = a \cdot x + b$ is made, where the most important parameter is a, namely the exponential one in the original framework.



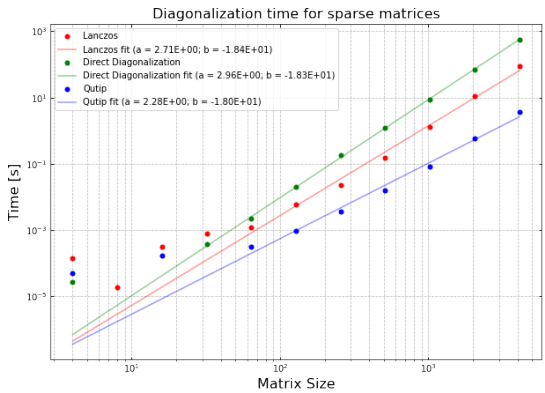
(a) Time elapsed building Heisenberg Hamiltonian using QuTip library, and related fits, for different matrix sizes (size of matrix goes as 2^N, where N is number of particles).



(b) CPU time taken for diagonalizing the **Heisenberg Hamiltonian** matrix of various sizes (size of matrix goes as 2^N, where N is number of particles)



(c) CPU time taken for diagonalizing a **full random matrix** of various sizes by various methods



(d) CPU time taken for diagonalizing a **sparse matrix** of various sizes by various methods

Figure 4. CPU time comparison

Fig. 4a consider how much time is needed for the Hamiltonian to be created. A singular behavior emerges: up to small matrices (~ 1000) times follow a certain exponential and $a \sim 1.6$, while above it scales as $a \sim 4.2$. Indeed it is a very large difference, and is actually difficult to explain: nonetheless it is curious and important to be mentioned. Figure 4b present the time needed in order to diagonalize matrices using different methods: the direct diagonalization

one, which is based on *Numpy linalg* library, our Lanczos implementation and finally the one given by *Qutip* library. It has been noted as Lanczos is faster than direct diagonalization specially for large matrices, whereas *Qutip* is the fastest. Finally, one of the main assumptions one makes to use Lanczos method has to be used when matrices are sparse. However, comparing Fig. 4c and Fig. 4d, one can catch as *Qutip* is the fastest one for both types of matrix. Nevertheless, as expected the Lanczos method is faster for sparse matrices, with regards to the function implemented by *Numpy.LinAlg* library. The best choice one can make is to use *Qutip* library, but in the case it were not available for any reason, to exploit *Lanczos* algorithm only when the matrix is sparse, being it faster than diagonalization computed by *Numpy.LinAlg*.

4.2 Variational Monte Carlo method

The number of iterations done for each Monte Carlo was 10^6 whereas the range of the variational parameter α considered was between $[0.2, 0.3]$ with steps of 0.01. Fig. 5 shows the ground states found for $h = 0$ in function of the variational parameter α. As it can be seen, different α leads to different results. In particular, the minimum values are found, which correspond to the exact ground state energy per site. Moreover, the optimal value obtained for α is 0.246 ± 0.0049.

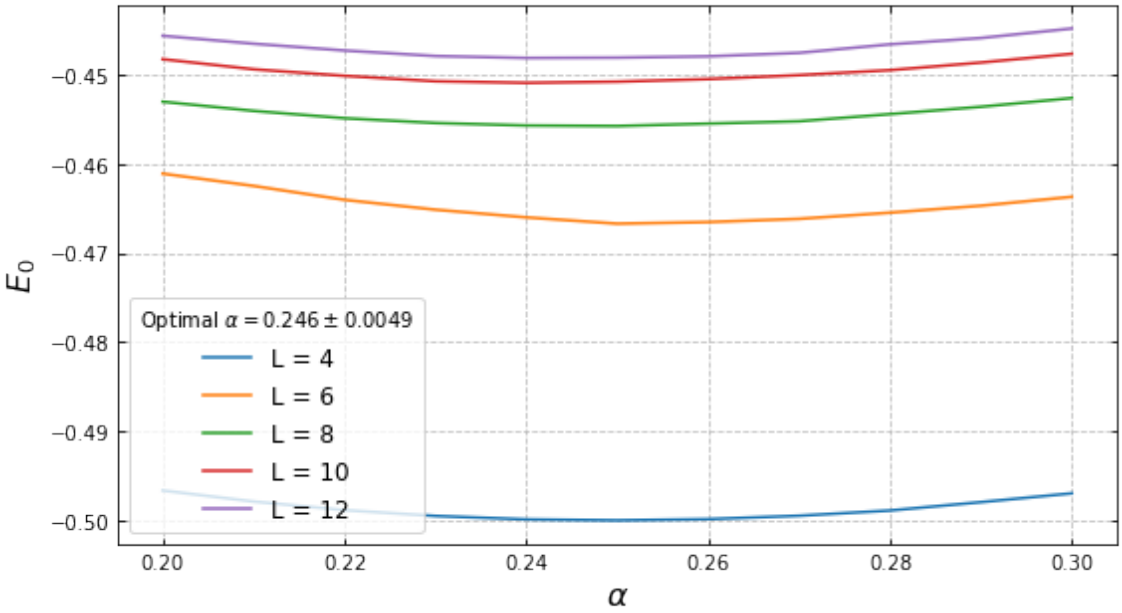
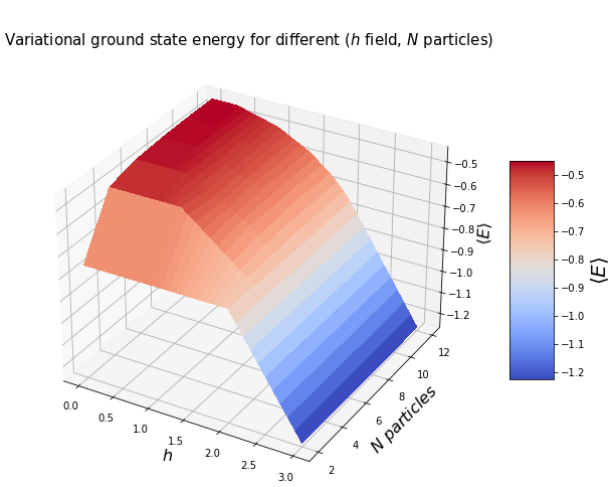
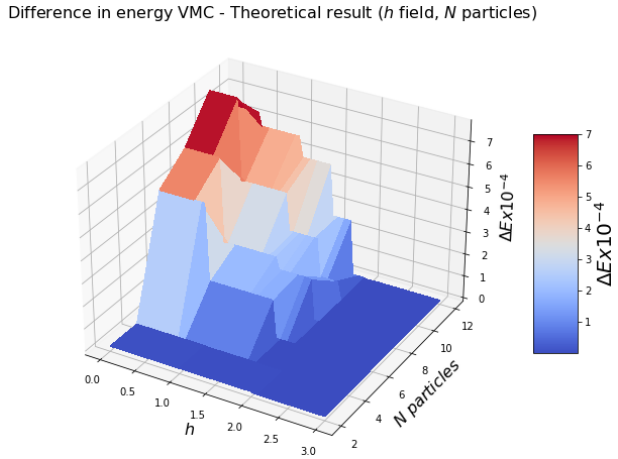


Figure 5. Variational parameter α against the ground state energies found for non external field h for even number of particles.

Fig. 6a shows the ground state energy for the several number of particles of the Heisenberg model subject to an external field $h = [0, 3]$ found by means of the Variational Monte Carlo. Fig. 6b depicts the difference between the exact values, theoretically computed via diagonalization, and the one obtained thanks to the aforementioned algorithm. As it can be seen, the order of magnitude for the error is relatively low: $O(-4)$ which allows to state that the ground state was successfully found, and with a good precision. On the other hand, it can be noticed that the largest error occurred when $h \sim [0, 2]$ and $N \sim [4, 12]$. A possible explanation for this is that there are many possible equivalent configurations with the same energy, since the external field is not as large to be able to align all the spins to a certain direction. Furthermore, the number of all possible alignments resulting in a low energy states increases with the size of the system, since there are more S_z that are allowed. For $h \sim 3$, however, energies of the ground state obtained via the two different methods match. Finally, but not least important, thanks to 6b it is found that variational energy is actually larger than the real ground state as one would expect from theory (cfr. 11).



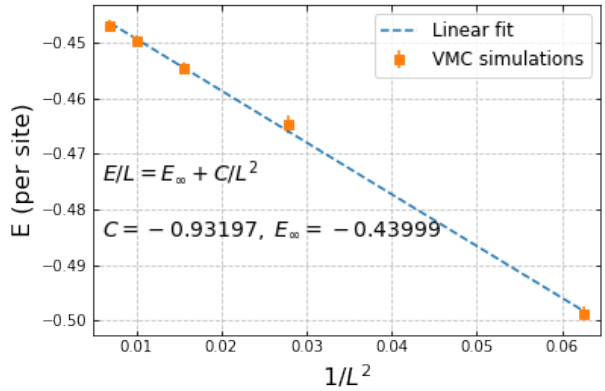
(a) Ground state energy found via Variational Monte Carlo.



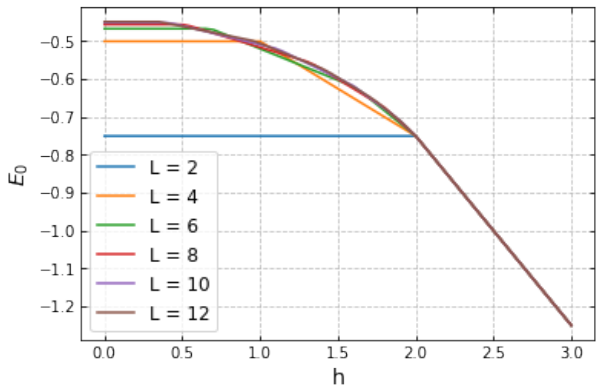
(b) Difference between exact ground state and ground state found by VMC.

Figure 6. Results obtained by Variational Monte Carlo, precision is around 10^{-4}.

Another interesting observation is seen from Fig. 7a where ground state energy E_0 per lattice site for $h = 0$, is inversely proportional and linearly related to reciprocal of total number of spins considered in the Heisenberg model. The estimated energy in the scaling with a formula $E/L = E_\infty + C/L^2$ [Kobayashi, Ohe, and Iguchi 1996], and from the fit $E_\infty = -0.43999$ is obtained which is quite close to the exact value by the Bethe ansatz, $E_{BA}/L = -(ln(2) - \frac{1}{4})J = -0.443147J$ [Bethe 1931]. The difference is only 0.71%. Therefore, trial wave function Eq. 25 well reproduces the true ground-state wave function of the 1D Heisenberg model.



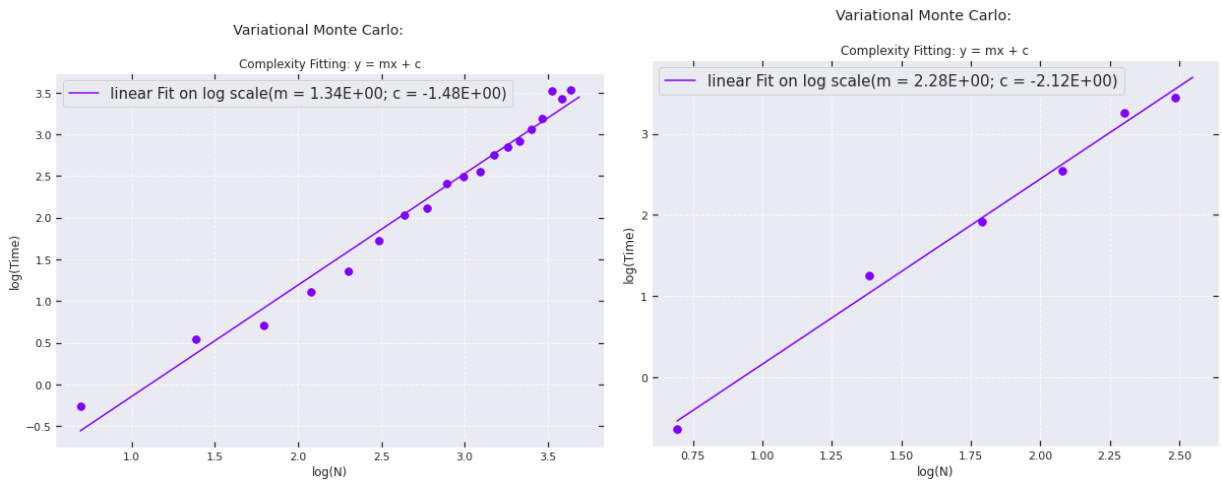
(a) Ground state energy per lattice against reciprocal of square of number of spins



(b) The variation of E_0 with respect to h when found by VMC is similar to the one found by Lanczos algorithm as seen in Fig. 33

Figure 7. Results obtained by Variational Monte Carlo for the Heisenberg model without the presence of magnetic term.

As already explained, one of the main drawbacks on computing the ground state when $h \neq 0$ comes to the fact that Eq. 33 need to be minimized for several m. On the other hand, when dealing with $h = 0$, the ground state wavefunction is given by the **Neel** State with $m = 0$, thereafter there is no need to loop over all magnetization m. In order to have further comparison between these two, Fig. 8a and Fig. 8b shows the time complexity of the Variational Monte Carlo according to increasing number of particles for the spin Heisenberg chain for the $h = 0$ and the $h = [0, 3]$ respectively. This was done for a fixed number of iterations (10^5). As it can be seen, the times follows a linear trend in a logarithmic scale. In particular, it can be seen that the slope for the $h = 0$ is lower (~ 1.3) than for the $h \neq 0$ case (~ 2.3), meaning that it is indeed efficient for larger systems. Furthermore for $h \neq 0$ it can be observed that it becomes unfeasible for already $N = 12$ particles since it almost needs one hour to run it.

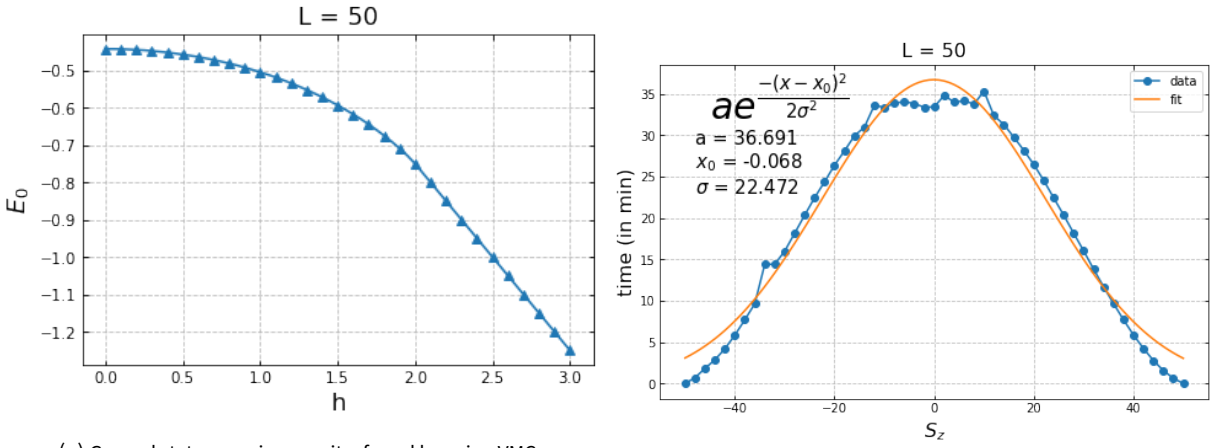


(a) Computing ground state for $h = 0$

(b) Computing ground state for $h = [0, 3]$

Figure 8. Time complexity found for the Variational Monte Carlo

One of the main advantages of computing the ground state energy by means of the Variational Monte Carlo Method for the spin chain is that it allows the computation of **really large** system size since all the resource allocation needed comes from a saving the $N-$vector. In fact, for the Lanczos case, memory issues of saving the Hamiltonian of size $2^N \times 2^N$ emerge. More specific for $N = 16$ particles no resources are available even for saving such model. Just as example, Fig. 9a presents the ground state energies per site obtained by means of VMC for number of spins 50. As it can be observed, the ground state energies follows a similar trend as for the other system sizes. In Fig. 9b, the S_z values ranging from -50 to +50 in steps of 2 are plotted against the time it takes to find the ground state energy E_0(as in $E(m)$ of the equation of $F(m)$ Eq. 33). The data points seems to follow a gaussian trend as observed from the fit. When either all the spins are completely up or completely down which corresponds to $S_z = +50$ or $S_z = -50$, the *"if" condition* as seen from the code in Listing 3, is never satisfied and so it never enters the Metropolis loop. Thus it ends up taking shorter time as compared to those configuration which have neighbouring spins alternate which corresponds to S_z values in the middle of the range [-50, 50].



(a) Ground state energies per site, found by using VMC, for the $N = 50$ spin chain Heisenberg model under the action of an external field $h = [0, 3]$

(b) CPU time taken for each S_z or configuration Metropolis iterations fixed to 10^5

Figure 9. VMC technique applied to large number of spins, N = 50

5 Conclusion

In conclusion, the ground state energies obtained by using Variational Monte Carlo and the Lanczos process were successfully found. In particular, the Lanczos method provides an efficient way to compute the eigenvalues specially with referral to sparse matrices. Indeed it behaves faster than Numpy library $linalg$ function under the hypothesis that matrix is sparse: on the contrary, if the matrix is full, Lanczos is worthless. However, it must be taken into account that $Qutip$ already-implemented function to compute Hamiltonian eigenenergies is actually the fastest method. As for this topic, it can be concluded that using already tested and implemented libraries is a better choice: it is pointless to reinvent the wheel apart from didactic purposes, as for this case.

Regarding the Variational Monte Carlo method, the trial wave function chosen provides accurate results for the even spin chain of the Heisenberg model by just varying one parameter, namely α. In particular, very large systems of spins can be studied by Quantum Monte Carlo methods, since no problems related to resource allocation have been faced. However, it should be also pointed out that the time consumption for such models is quite demanding specially when dealing with an external field $h \neq 0$, since it was needed to run simulations for any possible value of S_z, which was compatible with our system. Indeed in the $h = 0$ case it was enough to compute $S_z = 0$, since from theory it is well known that the ground state is Neel state. Nevertheless, it still provides a solution to the RAM memory issues arising from the Lanczos method, despite the time needed for the simulation. One last point worth to be mentioned is that the project was developed in Python for a matter of easier debug from our side, whereas for $C++$ or any other lower-level languages much better performances are expected. Finally, a possible limitations of our work is that choosing Jastrow Spin trial wave functions sets a condition for the number of spins in the problem. However, as N increases, the influence of it being either even or odd is less and less.

References

Bethe, H. 1931. "On the theory of metals. 1. Eigenvalues and eigenfunctions for the linear atomic chain." *Z. Phys.* 71:205–226. https://doi.org/10.1007/BF01341708.

developers, N. 2021. "Version 0.53.1 Numba release." https://numba.readthedocs.io/en/stable/index.html.

Johansson, J., P. Nation, and F. Nori. 2012. "QuTiP: An open-source Python framework for the dynamics of open quantum systems." *Computer Physics Communications* 183 (8): 1760–1772. ISSN: 0010-4655. https://doi.org/https://doi.org/10.1016/j.cpc.2012.02.021. https://www.sciencedirect.com/science/article/pii/S0010465512000835.

———. 2013. "QuTiP 2: A Python framework for the dynamics of open quantum systems." *Computer Physics Communications* 184 (4): 1234–1240. ISSN: 0010-4655. https://doi.org/https://doi.org/10.1016/j.cpc.2012.11.019. https://www.sciencedirect.com/science/article/pii/S0010465512003955.

Kobayashi, K., C. Ohe, and K. Iguchi. 1996. "Variational state based on the Bethe-ansatz solution and a correlated singlet liquid state in the one-dimensional t-J model." *Phys. Rev. B* 54 (18): 13129–13137. https://doi.org/10.1103/PhysRevB.54.13129. https://link.aps.org/doi/10.1103/PhysRevB.54.13129.

Sorella, S., and F. Becca. 2016. "SISSA - Lecture notes on Numerical methods for strongly correlated electrons. Academic year 2014-15." In *SISSA Lecture notes on Numerical methods for strongly correlated electrons.* Cambridge University Press. https://cm.sissa.it/phdsection/download.php?ID=11.

Werner, P., A. Comanac, L. de'Medici, M. Troyer, and A. J. Millis. 2006. "Continuous-Time Solver for Quantum Impurity Models." *Physical Review Letters* 97 (7). ISSN: 1079-7114. https://doi.org/10.1103/physrevlett.97.076405. http://dx.doi.org/10.1103/PhysRevLett.97.076405.