Superfluid-Mott Insulator quantum phase transition in the Bose-Hubbard model: Lanczos diagonalization and the mean-field approximation

M. Hantonne and R. Lyscar
Magistère de Physique Fondamentale d'Orsay
(Université Paris-Saclay)

Lattice models have become a central framework for studying many-body systems, thanks to their ability to capture complex interactions. In recent decades, both theoretical and numerical advancements have significantly deepened our understanding of these systems. Meanwhile, experimental techniques, especially the use of ultracold atoms in optical lattices, have made it possible to physically realize these models. That breakthrough enabled the experimental testing of models such as the Bose-Hubbard model, which were once explored mainly through simulations.
This report presents a numerical implementation in C++ of the Bose-Hubbard model, which is widely used to describe a system of interacting bosons on a lattice. The model is effective in capturing the evolution and dynamics of the system, as well as the quantum phase transition (i.e., at zero temperature) that occurs when the model parameters are changed.
This report begins by outlining the main objectives of the project (see I **Introduction**). Then, we develop the theoretical and numerical tools necessary for an efficient implementation (see **section** II and III) of the Bose-Hubbard model. Finally, the report will present the numerical results we obtained, as far as the evidence of the phase transition is concerned, and compare them to reference benchmarks for validation (**section** IV and **section** V). The C++ code developed for this study is publicly available on GitHub. *(Please note that the code is licensed under the GPL 3.0 License)*

I. INTRODUCTION

To begin with, let us boil down the main objectives of this project. In order to address the specific physical problem under study, we identified key concepts and skills to develop that are listed below :

- ◇ Develop an efficient implementation capable of handling large lattice systems, optimizing both memory usage and computation time.
- ◇ Implement effective diagonalization techniques for possibly large matrices, such as the Bose-Hubbard Hamiltonian.
- ◇ Characterize the quantum phase transition described by the Bose-Hubbard Hamiltonian, and highlight it with several and complementary numerical means.
- ◇ Compute a complete zero-temperature phase diagram within the mean-field approximation of the Bose-Hubbard model, in order to get some physical insights of the phase transition described by the model.

This report guides the reader through the concepts that enabled the achievement of these several objectives, providing a synthesis of relevant theoretical and numerical principles.

II. A SHORT INTRODUCTION TO THE BOSE-HUBBARD MODEL

Let us first present and discuss the Bose-Hubbard Hamiltonian $\hat{H}$, that describes a system of many interacting bosons on a lattice of dimension $d \in \mathbb{N}$. $\hat{H}$ reads:

$$\hat{H} = -J \sum_{<i,j>} \hat{a_i}^\dagger \hat{a_j} \; + \frac{U}{2} \sum_i \hat{n_i}(\hat{n_i} - 1) \; - \mu \sum_i \hat{n_i} \quad (1)$$

The first term is the so-called *hopping term* and J is the *hopping parameter*, since it characterizes the amplitude for a boson on a site j to jump to a neighboring site i due to the spatial expansion of the wave functions of the bosons up to the nearest neighbours. We note that J is chosen positive, which describes notably a situation where the potential at each site is attractive and the wave functions at each site are chosen real (which is always possible by making linear combinations of complex wave functions) and with the same sign when they recover each other. Such a situation is often reached in optical lattices. In the formalism of second quantization used in quantum field theory, $\hat{a_i}^\dagger$ and $\hat{a_i}$ are respectively the bosonic creation and annihilation operators at site i, such that $[\hat{a_i}, \hat{a_j}^\dagger] = \delta_{i,j}$.
The second term is the so-called *repulsion term*, which characterizes the on-site repulsion between bosons. We will only deal with repulsion, that is $U > 0$. The physical origin of such a repulsion for bosons will not be discussed here, but has a key role in the Superfluid-Mott insulator transition (see **section**V).
The third term allows boson number fluctuations in the lattice, with $\hat{\mu}$ the chemical potential and $\hat{n_i} = \hat{a_i}^\dagger \hat{a_i}$ and $\sum_i \hat{n_i} = \hat{N}$. Note that a derivation a the Bose-Hubbard Hamiltonian is shown in [1], starting from the most general many-body Hamiltonian describing interacting bosons in the formalism of second quantization.

The Bose-Hubbard Hamiltonian is effective in describing and predicting quantum phenomena observable

in optical lattices because of the interaction terms between bosons. However, solving this many-boson problem is challenging due to the large dimension of the Hilbert space. Indeed, for a lattice of M sites and N bosons, the dimension $\mathcal{D}$ of the Hilbert space $\mathcal{H}$ is :

$$\mathcal{D} = \frac{(N+M-1)!}{N!(M-1)!} \tag{2}$$

This can be shown easily because the problem is similar to a balls and bars combinatorics problem where the indistinguishable balls are replaced by indistinguishable bosons and distinguishable bins by distinguishable sites. Using this analogy, one can construct an orthonormal basis of the Hilbert space $\mathcal{B} = \{|n_1, n_2, ..., n_M\rangle\}_{(n_1,...,n_M)\in I}$ where $I = \{(n_1, ..., n_M) \in \mathbb{N}^M | \sum_{i=1}^{M} n_i = N\}$. This basis is the Fock state basis with a fixed number of bosons or occupation number basis.
The dimension of this basis growths exponentially with the number of bosons and sites. For example, if $M = N$, using the Stirling's approximation, we have that $\mathcal{D} \sim \frac{4^n}{\sqrt{\pi n}}$. You can find below the dimension $\mathcal{D}$ of the Hilbert space for low values of $M = N$:

N	3	5	7	9	11
$\mathcal{D}$	1.0×10^1	1.3×10^2	1.7×10^3	2.4×10^4	3.5×10^5

N	13	15	17	19	21
$\mathcal{D}$	5.2×10^6	7.6×10^7	1.1×10^9	1.6×10^{10}	2×10^{11}

The problem is solved exactly if one can fully diagonalize $\hat{H}$. Hence one is able to theoretically predict the unitary evolution of the system (assuming it is isolated from any kind of environment) starting from an initial state $|\Psi_0\rangle$ and using the Schrödinger equation. In view of the difficulty to numerically diagonalize huge matrices, the next section will present some methods to find the lowest eigenvalues of $\hat{H}$ and their eigenvectors. However, if the dimension of $\mathcal{H}$ becomes too large, alternative approaches, such as Quantum Monte Carlo, must be considered, but will not be discussed in this report.

III. NUMERICAL TOOLS FOR THE SIMULATION

We will now explore the numerical methods used to set numerically the Bose-Hubbard Hamiltonian $\hat{H}$ and efficiently diagonalize it, focusing primarily on the lowest energy states, with an emphasis on the ground state of the system.

A. Set up the Bose-Hubbard Hamiltonian

1. How to save computer memory?

When evaluating the resource cost of a program, both time and memory usage must be reckoned with. Modern computational methods typically calculate the action of the Hamiltonian $\hat{H}$ on a vector "on the fly," as matrix-vector multiplication is the primary operation needed to diagonalize $\hat{H}$ (see [2]). The following section presents an approach to fully construct the Bose-Hubbard Hamiltonian while limiting memory usage.

2. Sparse matrices

In some cases, the matrix to be saved in memory is such that most of the coefficients are equal to zero. Then it is not necessary to keep all those coefficients in memory, but only the information for the non-zero coefficients. An efficient numerical scheme for this purpose is to store the matrix as a *sparse matrix*. Instead of a matrix, usually three arrays are stored: one containing the values of the nonzero coefficients and called $vals$, one containing the rows of the non-zero coefficients and called $rows$, and one containing their columns and called $cols$. This is the most intuitive way to store a sparse matrix, and it is actually very efficient to implement matrix-vector multiplication (as presented in [3]). For instance, if the result of the matrix-vector multiplication of sparse matrix by a vector $\mathbf{x}$ is stored in a vector $\mathbf{y}$, an efficient implementation is:

for $i = 1$ `to` m **do**
  $y[rows[i]] \leftarrow y[rows[i]] + vals[i] * x[cols[i]]$
end for

where m is the number of nonzero elements in the sparse matrix. In fact, matrix vector multiplication can be achieved in $O(m)$ instead of $O(N^2)$ where N is the matrix size.

Even more efficient schemes have been developed, such as the *Compressed Column Storage* (CCS), which is composed of four arrays. The scheme used in this project, which is the method used in the C++ library *Eigen*, is detailed below:

- ◇ Values: stores the coefficient values of the non-zeros.
- ◇ InnerIndices: stores the row (resp. column) indices of the non-zeros.
- ◇ OuterStarts: stores for each column (resp. row) the index of the first non zero in the previous two arrays.
- ◇ InnerNNZs: stores the number of non-zeros of each column (resp. row). The word inner refers to an inner vector that is a column for a column-major matrix or a row for a row-major matrix. The word outer refers to the other direction.

In III A 4, we will see that the Hamiltonian will have, in 1D, 3 nonzero elements per row, and storing the Hamiltonian in a sparse matrix can help saving a lot of memory. For example, storing the Hamiltonian for 9 bosons on 9 sites in 1D on a sparse matrix rather than on a dense matrix reduces memory usage by more than 99% due to the low density of the Hamiltonian. The density Δ of a matrix M is defined as

$$\Delta(M) = \frac{\text{Number of nonzero elements in } M}{\text{Total number of elements in } M}$$

Below is the density of the Hamiltonian for N bosons on N sites in 1D :

N	3	4	5	6	7	8	9
Δ (%)	15.00	4.29	1.19	0.33	0.09	0.03	0.01

Eventually the gain in memory is more or less proportional to $100\% - \Delta$, so, we can see that storing the Hamiltonian in a sparse matrix paves the way for a dramatically more efficient implementation.

3. Generate the Fock states

This section and the next section are inspired by the implementation of [4]. The Fock states $|n_1, n_2, ..., n_M\rangle$ with a fixed number of bosons obviously verifies $\sum_{i=1}^{M} n_i = N$. A possible algorithm consists in generating the states using recursivity. This method has a complexity of $O(\mathcal{D} \times M)$. On top of that, the depth of the recursion tree is equal to M, which can entail stack overflow for huge lattices. Instead, we implemented the Fock states using a direct method of computation based on their *lexicographic order*, following the algorithm detailed below.

- ◇ **Step 1:** Start with the Fock state $|N, 0, ..., 0\rangle$, which is the biggest vector regarding lexicographic order.
- ◇ **Step 2:** If the previous computed state equals $|0, ..., 0, N\rangle$ we stop the process, because we reached the lowest state regarding lexicographic order. Else continue.
- ◇ **Step 3:** Assuming the previous computed state $|n_1, ..., n_M\rangle$ has for last non-zero coefficient n_k which means that $n_k \neq 0$ and $n_l = 0 \; \forall l \in [k+1, M]$. Then, the next vector in the lexicographic order is given by $n_k \leftarrow n_k - 1$ and $n_{k+1} \leftarrow N - \sum_{i=1}^{k} n_i$
- ◇ **Step 4:** Store the last computed Fock state as a column of a dense matrix and go to **Step 2**. Note that Fock States are stored in a $\mathcal{D}$ x M matrix that will be called $\mathcal{B}$.

The complexity of this algorithm is also $O(\mathcal{D} \times M)$, but it does not have a recursion tree, which can give a memory advantage. Using this method, we can generate the Fock states with a non-fixed number of bosons by concatenating the ones with fixed number with different total number of bosons.

4. Bose-Hubbard Hamiltonian in the occupation number basis

To determine the coefficient of the Bose-Hubbard Hamiltonian, we need to compute $\langle n'_1, n'_2, ..., n'_M | \hat{H} | n_1, n_2, ..., n_M \rangle \;\; \forall | n_1, n_2, ..., n_M \rangle \in \mathcal{B}$. The on-site interaction and chemical potential part of the Hamiltonian are diagonal in the Fock states basis and can be easily computed. The hopping part of the Hamiltonian is non-diagonal in the Fock states basis. For example,

$$\hat{a_i}^\dagger \hat{a_j} | n_1, ..., n_i, ..., n_j, ..., n_M \rangle = \sqrt{n_j(n_i + 1)} \; | n_1, ..., n_i + 1, ..., n_j - 1, ..., n_M \rangle \quad (3)$$

A naive approach to calculate the coefficients of the Hamiltonian would be to find for each $|n_1, ..., n_i, ..., n_j, ..., n_M\rangle$ the corresponding state $|n_1, ..., n_i + 1, ..., n_j - 1, ..., n_M\rangle$ after applying the annihilation and creation operators and to search in which position the latter is in $\mathcal{B}$. But searching for an element in a list is linear in the size of the list and that leads to a total complexity of $O(\mathcal{D}^3)$ to fill the Hamiltonian. We can reduce this by sorting the Fock state bases in another manner when generating them. We associate to each state $|n_1, ..., n_M\rangle$ a tag τ

$$\tau(|n_1, ..., n_M\rangle) = \sum_{i=1}^{M} log(p_i) n_i \quad (4)$$

where p_i is the i^{th} prime number. A proof similar to the one in [5] can show that this function is injective, so each tag corresponds to a unique state. When generating the Fock states, we can sort them directly in ascending order by using a binary insertion on their tag. This can be done in $O(\mathcal{D}^2 log_2(D) \times M)$. Then, instead of linearly searching the state after applying annihilation and creation operators, we can perform a binary search on the tag list to find the position of this state in $\mathcal{B}$. Doing so, we can fill the Hamiltonian in $O(\mathcal{D}^2 log_2(D))$ which is an improvement of the initial complexity in $O(\mathcal{D}^3)$. The algorithm in the 1D case is summarized below.

B. Lanczos algorithms

We would like to obtain the eigenvalues and eigenvectors of the Hamiltonian. A naive approach would be to calculate all the eigenvalues and eigenvectors using classical methods such as a QR decomposition but this method

Algorithm 1 Hopping Hamiltonian Filling

```
for k = 0 to 𝒟 do
    state_k = basis[:, k]
    for i = 0 to M do
        for j = 0 to M do
            if state_k[j] ≥ 1 then
                state_k[i] = state_k[i] + 1
                state_k[j] = state_k[j] − 1
                index = search_tag(state_k)   ▷ binary search
                H[k, index] = −J · √(state_k[i] · state_k[j] + 1)
                H[index, k] = H[k, index]
            end if
        end for
    end for
end for
```

has a complexity of $O\left(\mathcal{D}^3\right)$. Instead, we will only compute the lowest k eigenvalues with their eigenvectors using Lanczos algorithm. The algorithm, explained in [6] starts by choosing a given state $|v\rangle$:

- ◇ **Step 1:** Note $|v_0\rangle = \frac{|v\rangle}{||v||} = \frac{|v\rangle}{\sqrt{\langle v|v\rangle}}$ and let n = 1
- ◇ **Step 2:** Compute $|w_n\rangle = H\,|v_{n-1}\rangle$ with $n \geq 1$
- ◇ **Step 3:** Re-orthogonalize $|w_n\rangle$ by $|w_n\rangle = |w_n\rangle - \sum_{i=0}^{n-1} |v_i\rangle\,\langle v_i|w_n\rangle$ and at the same time set $a_{n-1} = \langle v_{n-1}|H|v_{n-1}\rangle$
- ◇ **Step 4:** Define the Lanczos coefficient $b_n = ||w_n||$
- ◇ **Step 5:** If $b_n = 0$ or we have done k iterations stop the process else set $|v_n\rangle = \frac{|w_n\rangle}{b_n}$ and restart from **Step 2** with n = n+1

At the end of the process, we set the tridiagonal matrix T with

$$(T_{i,j}) = \begin{cases} b_i & \text{if} \quad j = i+1 \\ b_j & \text{if} \quad i = j+1 \\ a_i & \text{if} \quad i = j \\ 0 & \text{else} \end{cases} \tag{5}$$

$$\text{and V} = (v_0|v_1|...|v_k)$$

This process is done in $O\left(\mathcal{D} \times k \times s\right)$ where s is the average number of nonzero elements per row in the Hamiltonian. This symmetric tridiagonal matrix can then be easily diagonalized in $O\left(k^2\right)$. But in fact, this algorithm is prone to numerical instability and its convergence really depends on the choice of the initial vector, which is why it is not implemented as above but in a more sophisticated algorithm called Implicitly Restarted Lanczos Method (IRLM) that is detailed in [7]. Using the IRLM, we can still find the lowest k eigenvalues with their eigenvectors in $O\left(\mathcal{D} \times k \times s\right)$ but with less stability issues.

C. Our implementation

The first step of the implementation of the Hamiltonian is to choose the geometry of the lattice. We set this geometry by determining the adjacency list in the form of the `std::vector`.
For the remainder of the project, we have chosen to use Eigen [8], a highly efficient C++ template library designed for linear algebra, which provides a wide range of tools for matrix operations, solving systems of linear equations, and eigenvalue problems. Additionally, we use Spectra [9], a C++ library that re-implements ARPACK [10] FORTRAN-based library for solving large-scale eigenvalue problems entirely in C++ and only with header files. This makes Spectra highly efficient and easy to integrate into C++ projects, particularly when working with large, sparse matrices.
The Fock state basis is generated following the algorithm of III A 3 and stored in ascending order of tag in a `Eigen::Matrix` while the sorted tags are stored in a `Eigen::Vector`. The Hamiltonian is then initialized as a `Eigen::SparseMatrix<double>` and filled following the procedure described in III A 4. The IRLM we used to find the lowest eigenvalues is the `Spectra::GenEigsSolver<SparseGenMatProd<double>` solver for sparse matrix implemented in Spectra and the eigenvalues and eigenvectors are respectively stored in a `Eigen::Vector` and `Eigen::Matrix`.
To optimize the loops and reduce calculation time, we use the API OpenMP that allows us to parallelize loops so blocks of code in loops can be executed concurrently by multiple threads.
A documentation of our C++ implementation of the Bose Hubbard can be generated in our GitHub using the documentation generator Doxygen [11]

IV. CHARACTERIZING THE MOTT INSULATOR AND SUPERFLUID PHASE

This section will present several characterizations of a phase transition well described by the Bose-Hubbard Model: the Superfluid-Mott Insulator (SF-MI) phase transition. It is a quantum phase transition, that is a transition that occurs at $T = 0\ K$. Then, the numerical schemes presented in the previous section will be used to perform Lanczos diagonalization of the Bose-Hubbard hamiltonian $\hat{H}$ and highlight the transition.

A. Spectral properties of the Bose-Hubbard Hamiltonian

The Bose-Hubbard model shows two different regimes depending on the value of the ratio $\frac{J}{U}$ (see **section V** for a more comprehensive discussion of the physics of the problem).
For $\frac{J}{U} << 1$, the system will be in the Mott insulator

(MI) phase and integrable. When increasing this ratio, from $\frac{J}{U} \sim 1$, the system will be in the superfluid (SF) phase and the system will not be integrable due to the presence of quantum chaos (the physics of the transition if further discussed in **section V**). The Bohigas-Giannoni-Schmit conjecture [12] asserts that the energy levels spacings statistics of integrable non-chaoctic and non-integrable chaotic systems follow two different distributions, a Poisson distribution for the first and a Wigner-Dyson distribution for the second, more precisely a Gaussian orthogonal ensemble (GOE) which is a particular Wigner-Dyson distribution with a Dyson index equal to 1.

To study those energy-level spacings, we introduce gap ratios, which are ratios of spacings between two consecutive energy levels, such as in [13] and [14]. Mathematically, we denote $\{\mathcal{E}_n\}_{n\in[1,\mathcal{D}]}$ the set of energies of the Hamiltonian sorted in ascending order. Then, the gap ratio r_n for $n \in [2, \mathcal{D}-1]$ is equal to :

$$r_n = min\left(\frac{\mathcal{E}_{n+1}-\mathcal{E}_n}{\mathcal{E}_n-\mathcal{E}_{n-1}}, \frac{\mathcal{E}_n-\mathcal{E}_{n-1}}{\mathcal{E}_{n+1}-\mathcal{E}_n}\right) \tag{6}$$

These gap ratios follow the same distribution than the energy levels spacings but are normalized and so do not require an unfolding procedure, because they are independent from the density of states. To analyze the distribution, one would calculate all the eigenvalues of the Hamiltonian (i.e. the energies) and do a regression analysis to find the fittest model between a Poisson or Wigner-Dyson distribution. But this is computationally heavy and we do not want to compute all the eigenvalues as explained in the previous section. Instead, we will focus on the mean of the gap ratios. For the Poisson distribution, it should take a value $\langle r\rangle = 2ln(2)-1 \approx 0.386$ while for the GOE it should take a value $\langle r\rangle \approx 0.53$. To calculate those mean values, we computed about the first twenty eigenvalues of the Hamiltonian and the associated gap ratios, for a 1D chain with $n = 8$ bosons and $m = 8$ sites. We obtained the map for different values of J and U:

We can observe that the mean gap ratio seems invariant along affine lines. An interpretation would be that the mean gap ratio depends only on the ratio $\frac{J}{U}$ but with a constant gap because these lines have a non-null y-intercept. A particular affine line seems to separate the map with a slope approximately equal to 1 which correspond to $\frac{J}{U} \approx 1$, between a plane where $\langle r\rangle \approx 0.53$ for $\frac{J}{U} > 1$ and a plane where $\langle r\rangle \approx 0.386$ for $\frac{J}{U} < 1$. This behavior is validated by the theory detailed in **section V**.

Another characteristic that differs in the two phases is the correlation between the energy-level spacings that inherit the gap ratios. In the MI phase, the gap ratios are highly correlated, while in the SF phase, the correlations become weak to null due to a more chaotic behavior. To reveal these correlations, one can perform a principal component analysis (PCA) on the gap ratios

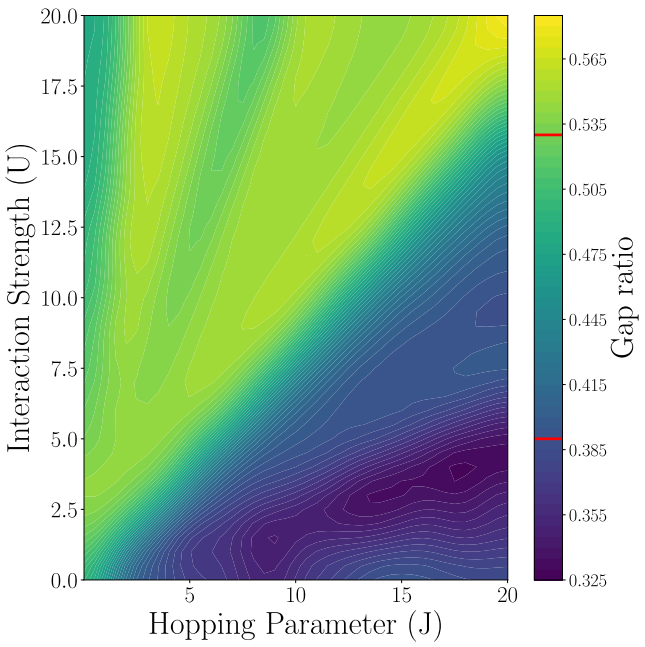


FIG. 1: Map of the mean gap ratio $\langle r\rangle$

between different sets of gap ratios in both MI and SF phases. The principle of the PCA is explained in [15]. On a gap ratio sets with $\frac{J}{U} > 1$, i.e. in the SF phase, we typically observe the projected points in the Principal Component Space (PCS) after applying a PCA below:

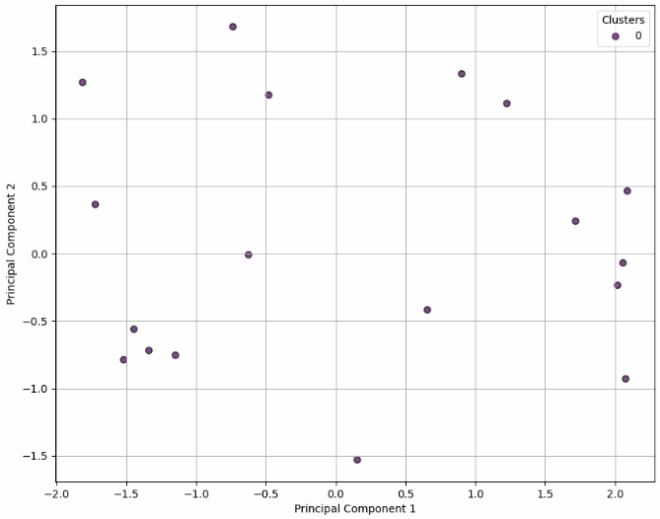


FIG. 2: PCA Projection of Gap Ratios into the PCS in the SF phase

The projected points seem randomly distributed in the space, which is typical of uncorrelated sets. When performing a k-means clustering on these points, no clusters seem to appear, which can corroborate that the points are randomly distributed. The dispersion of the projected points in the space equals 3.

On a gap ratio sets with $\frac{J}{U} < 1$, i.e. in the MI phase, we typically observe the projected points in the PCS after applying a PCA below:

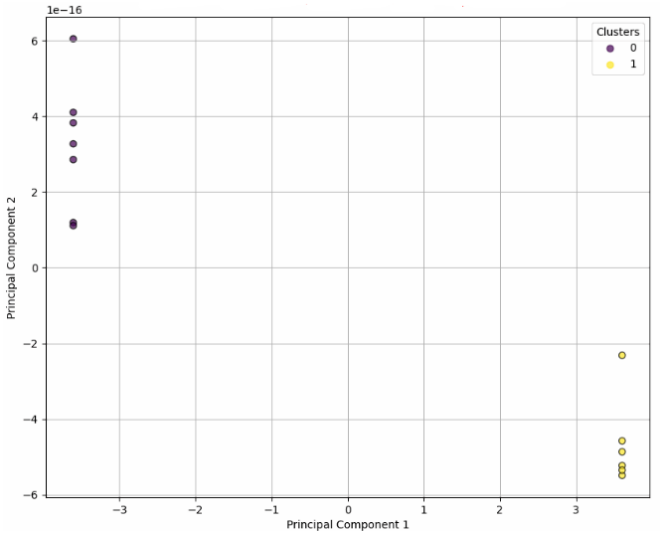


FIG. 3: PCA Projection of Gap Ratios into the PCS in the MI phase

The projected points are aligned along two lines, which is typical of highly correlated sets. When performing a k-means clustering on these points, two clusters seem to appear, which can corroborate that the points are correlated. The dispersion of the points is bigger and equals 14.
The two different behaviors of the gap ratios in the MI or SF phases can help to differentiate whether the system is in one phase or another.

V. QUANTUM PHASE TRANSITION IN THE MEAN-FIELD APPROXIMATION

This section deals with the Bose-Hubbard model using an approach very different from what has been described in **section III**. The main goal now is to familiarize with properties of the Bose-Hubbard Hamiltonian, and especially the physics of the Superfluid-Mott Insulator transition, that has been characterized in **section IV**. As we already mentioned, almost all the complexity (and the richness) of the problem is contained in the interaction terms that make the Hamiltonian non separable. Those interactions compel us to consider the whole occupation number basis $\mathcal{B} = \{|n_1, n_2, ..., n_M\rangle\}$ and make the problem impossible to solve by hand. But as usual when considering many-particle systems, it is possible to consider a mean-field approximation in order to make the many-particle Hamiltonian separable: each boson in the system interacts with all other bosons through an effective field which is the same for every boson (in particular finite size effects due to the lattice are not considered). As we will show, the mean-field approach is efficient in describing the Superfluid-Mott Insulator quantum phase transition of the Bose-Hubbard model.
The full derivation of the mean-field approximation and its consequences can be found in [1]. To begin with, one introduces the so-called superfluid order parameter $\Psi \equiv \langle \hat{a} \rangle$.
Then one can write the mean-field Hamiltonian :

$$\hat{H}^{MF} = -Jq \sum_i \{-\Psi(\hat{a}_i + \hat{a}_i{}^\dagger) + \Psi^2 - \frac{\mu}{qJ}\hat{n}_i + \frac{U}{qJ}\hat{n}_i(\hat{n}_i - 1)\} = \sum_i \hat{h}_i \quad (7)$$

We can now consider each boson separately, and forget the index i. The last step is to perform stationary perturbation up to the second order, to get the expression of the ground state energy that takes the form:

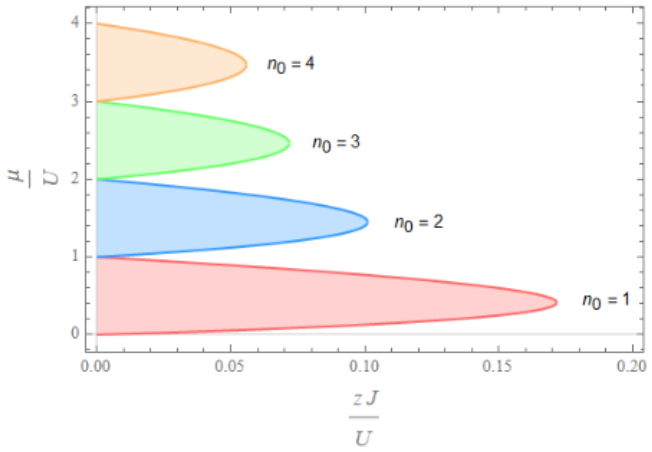
$$\epsilon = a_0 + a_2\Psi^2 + O(\Psi^4) \quad (8)$$

One recognizes a Landau expansion that is characteristic of a second order phase transition, the critical point corresponding to the condition $a_2 = 0$. Indeed, if $a_2 > 0$, $\Psi = 0$ is the solution of that self-consistent equation that leads to a minimization of the energy. Since Ψ is the Superfluid order parameter, $a_2 > 0$ corresponds to the Mott insulating phase. On the other hand, if $a_2 < 0$, then Ψ takes a finite value and it corresponds to the Superfluid phase. Solving equation $a_2 = 0$ leads to a relation between $\frac{\mu}{U}$ and $\frac{J}{U}$ that defines a contour in the parameter space (see **Figure** 4 below). Note that n_0 in the figure is the boson number at each site (when defined, *i.e.* in the Mott insulator phase), and $z = q$ is the number of neighbours of a site in the lattice.

The method to obtain such a phase diagram by numerical means is also explained in [1]. Indeed, the idea is to consider a self-consistent equation for the Superfluid parameter Ψ:

$$\Psi = \langle \Phi_0(\Psi) | \, \hat{a} \, | \Phi_0(\Psi) \rangle \quad (9)$$

where $|\Phi_0\rangle$ is the ground state of the system which depends on Ψ due to the mean-field approximation.



Source: [1]

FIG. 4: Zero temperature phase diagram in the mean-field approximation for the Bose-Hubbard model. The colored areas correspond to Mott insulator phase, with a fixed number of bosons at each site. The blank area corresponds to Superfluid phase, where bosons are delocalized over the whole lattice.

The numerical method that we used to compute the phase diagram shown in **Figure** 5 is described below. The main idea is the following: assume $U = 1$ for simplicity. For a couple (J, μ), we compute the value of Ψ that satisfies equation 9 up to a certain arbitrary precision (for instance we took a precision of 10^{-5}).

- ◇ **Step 1:** For a fixed couple (J, μ), initialize a random ansatz $\Psi_0 \in [0, 1]$ with uniform distribution for the Superfluid order parameter.
- ◇ **Step 2:** Starting with $n = 1$, compute the $(2n + 1, 2n + 1)$ real symmetric matrix $\hat{h}(\Psi, J, \mu)$ in the Fock basis. Then compute the lowest eigenvalue ϵ_0 and corresponding ground-state eigenvector $|\Phi_0\rangle$. Increase n until the convergence of ϵ_0 up to an arbitrary precision. No sparse matrices are required here, since n is never going to be too large.
- ◇ **Step 3:** Compute $\Psi = \langle\Phi_0|\,\hat{a}\,|\Phi_0\rangle$. Repeat **Step 2** until the convergence of Ψ up to an arbitrary precision.

Now by performing a double-loop on J and μ, it is possible to obtain a matrix containing the values for Ψ and to plot the phase diagram with Python. **Figure** 5 was obtained by computing 1 million of points (for 1000 values of J and μ). The whole computation took 20 minutes and 12 seconds.

The appearance of the phase diagram will not be fully discussed here, but a full discussion can be found, again, in [1]. Let us just discuss some limit cases in order to get some physical insights of the model. If one takes $J = 0$ in the Bose-Hubbard Hamiltonian $\hat{H}$, then there is no possibility for a boson to jump by tunnel effect from a site to another. The only interaction between bosons is on-site repulsive interaction, due to finite U. Hence the bosons are not delocalized and the system has to be in a Mott insulating phase, which is the case in our diagram. The other limit is $J \to \infty$. Here the delocalization over the whole lattice is necessarily observed, and the system has to be in the Superfluid phase, as shown in the figures.

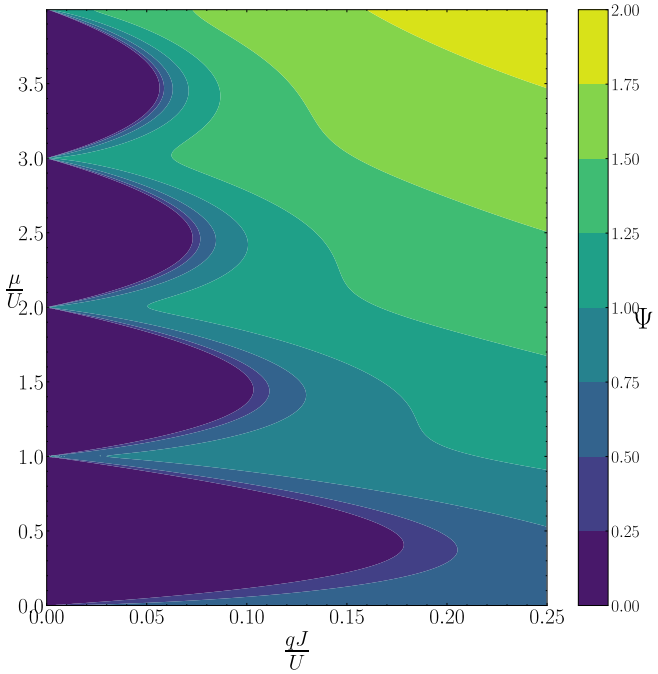


FIG. 5: Phase diagram in the mean-field approximation obtained numerically using the self-consistent method

So far the numerical result is only qualitative, and we would like to compare quantitatively to the analytic phase diagram of **Figure 4**. To do so, we compare the analytical level curves $\Psi = 0$ with the one obtained above. The theoretical formula for those level curves are, for some $n_0 \in \mathbb{N}$ and defining $\omega = \frac{qJ}{U}$:

$$\frac{\mu}{U} = n_0 - \frac{1}{2}(1+\omega) \pm \sqrt{1 - 2(1+2n_0)\omega + \omega^2} \qquad (10)$$

As shown in **Figure 6**, the deviation to the analytical predictions is visible, but not huge (especially for $n_0 = 4$). We computed the relative deviation regarding the maximum value of $\frac{qJ}{U}$ for the lobe $n_0 = 1$ for instance, which leads to an error $\epsilon = 2.1\%$.
Note that the result is still quite qualitative, but this is the very essence of mean-field to provide a good physical sense of the phase transition while not being quantitative. Indeed, more points and more precision do not make the deviation drop. Moreover, since the mean-field approximation does not contain size-effects of the system, the previous results are not prone to be

improved by better computer capacities.

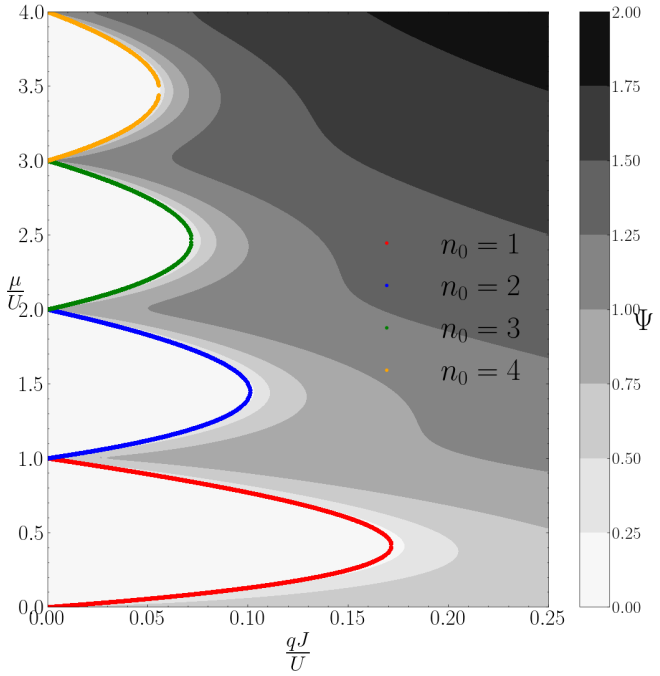


FIG. 6: Comparison between the phase diagram obtained by numerical means and the theoretical formula

VI. CONCLUSION AND PERSPECTIVES

This report discussed some of the theoretical and numerical tools capable of highlighting the Superfluid-Mott Insulator phase transition in the Bose-Hubbard model. In particular, the implementation in itself requires sparse matrices in order to save computer memory, and linear algebra libraries such as Eigen and Spectra to perform efficient Lanczos diagonalizations.

1. *Did we reach our goals ?*

As far as our initial goals are concerned, we have been able to fully implement the Bose-Hubbard Hamiltonian, including for large lattice systems, that is with a number of sites and bosons quite large. However, we struggled to perform diagonalization for such lattice systems. We stayed focus on the 1D chain during this project, and we reached the limits of our personal computer for $m = 8$ sites and $n = 8$ bosons. Still, for small systems, we were able to characterize the phase transition very qualitatively with the study of the mean gap ratio. In order to be way more quantitative, we tried more precise characterizations including the computation of the *Single Particle Density Matrix* (SPDM), whose largest eigenvalue gives an estimation of the superfluid order parameter (see the Penrose-Onsager criterion in [16]), but we couldn't reach relevant results, probably due the size effects of our small systems. Then the perspectives include the computation of more physical quantities of interests, especially through the SPDM, by working again on parallelization and computation, and possibly by using more powerful computers.
For the mean-field approximation, we definitely succeeded in reproducing the analytical phase diagram at $T = 0\ K$, which contains almost all of the physics of the superfluid-Mott Insulator phase transition and helped to understand the physics of the problem.

2. *What did we learn?*

To achieve this project, the development of new abilities was required. Most of them were explained in the report, but it does not include the programming skills such as the use of CMake to optimize the building process on our personal computers, or the loop parallelization using OpenMP. Moreover, this project was a great opportunity to learn many notions about many-body quantum systems that are currently being studied in many fields of Physics, and that we could not fully present in the report.

ACKNOWLEDGMENTS

We would like to thank Alexandre Ménard for his help with the code implementation and the valuable advices he gave us throughout the project.

CODE STRUCTURE

This appendix presents briefly the architecture of the program, which is fully available on GitHub.

The code is mainly composed of 3 namespaces (`Analysis`, `BH` and `Op`) and 1 class (`Neighbours`). The idea behind the structure is the following: an instance of a many-body bosons system is a combination of a lattice geometry (implemented in `Neighbouts` class) and a Bose-Hubbard Hamiltonian over that lattice. The reason why we actually do not have a class named `Hamiltonian` or `BH` is optimization. Indeed, in order to diagonalize efficiently sparse matrices regarding their properties (e.g hermitianity for a Hamiltonian), Spectra needs to know the Eigen type of the matrices. When adding a layer of abstraction over that specific type, for instance by embedding the Hamiltonian object in a class, the optimization cannot be done and performance

is really affected. Another concern is the memory cost. As a matter of fact, to obtain the diagrams presented in the article, it is necessary to make loops over the parameters of the Hamiltonian, and to set an Hamiltonian $\hat{H}(J, U, \mu)$ for each triplet. Even though sparse matrices are used, it is not relevant to store several instances of Hamiltonian throughout the execution. That's why we used a namespace `BH`, which sets each part of $\hat{H}$ separately once for all, so that it's possible to fully reconstruct $\hat{H}$ at each step by multiplying each term by J, U or μ.
Now, as shown in the UML diagram below, the namespace `Analysis` is the very core of the code. It contains all the functions that compute the quantities of interest. The main file only plays the role of an interface between the user and `Analysis`.

For instance, if one wants to plot the mean-gap ratio diagram, one has to execute the code by providing the so called *exact parameters* (see the *Readme* file associated with the project) which leads to a call of the function `Analysis::exact_parameters`. That function instantiates an object `neighbours` for a specific geometry, and it then possible to compare different geometries using different instances. The function then calls `BH` to set Hamiltonians needed in the loops over the model parameters, and perform Lanczos-like diagonalization using functions from `Op` (totally based on the Eigen and Spectra libraries). Several quantities can be computed and saved within files. Python scripts dedicated to the plots can then access those data files. Note that the project is also composed of header-only libraries, including `Eigen`, `Spectra` and `tqdm`. An other namespace `Resource` is used to handle time and memory cost of the code.

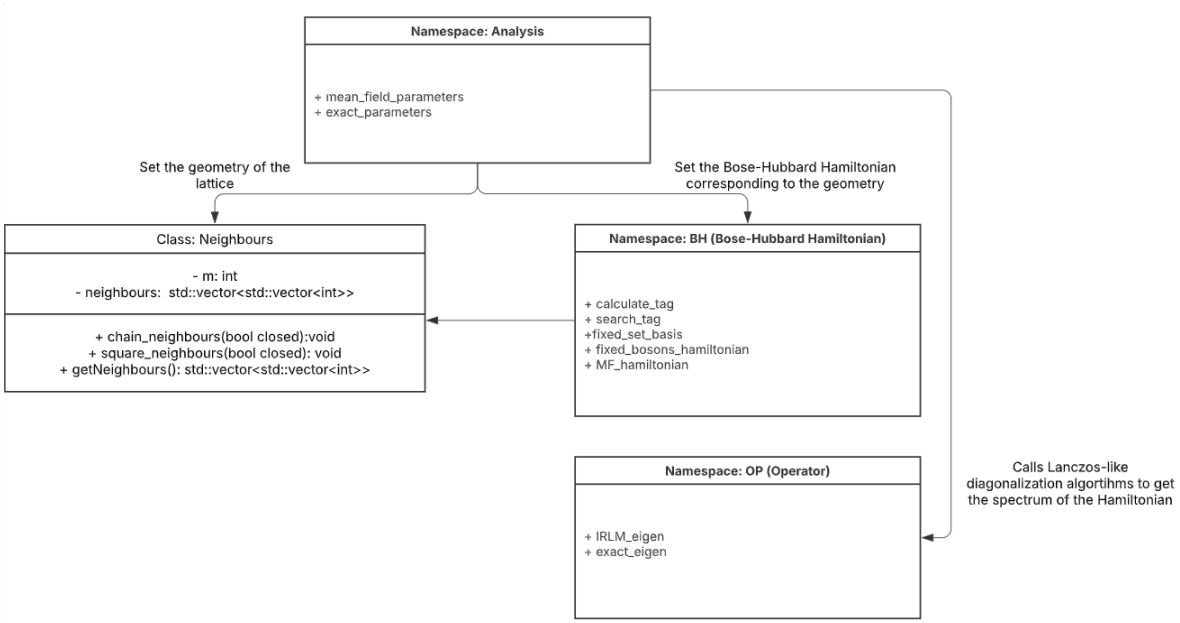


FIG. 7: Class and namespace diagram. Only the non-static functions are represented for namespaces

[1] N. De Ro, Superfluid-Mott Insulator Transition in the Bose-Hubbard Model, (2021).
[2] A. Weiße and H. Fehske, enExact Diagonalization Techniques, in en*Computational Many-Particle Physics*, edited by H. Fehske, R. Schneider, and A. Weiße (Springer, Berlin, Heidelberg, 2008) pp. 529–544.
[3] D. Sénéchal, *Méthodes numériques et simulations*.
[4] J. M. Zhang and R. X. Dong, Exact diagonalization: the Bose-Hubbard model as an example, European Journal of Physics **31**, 591 (2010), arXiv:1102.4006 [cond-mat].
[5] I. Boreico, My favorite problem: Linear independence of radicals, Harvard College Mathematics Review **2**, 87 (2008).
[6] A. Bhattacharyya, D. Ghosh, and P. Nandi, Operator growth and krylov complexity in bose-hubbard model, Journal of High Energy Physics **2023**, 10.1007/jhep12(2023)112 (2023).
[7] D. C. Sorensen, Implicitly restarted arnoldi/lanczos methods for large scale eigenvalue calculations, in *Parallel Numerical Algorithms*, edited by D. E. Keyes, A. Sameh, and V. Venkatakrishnan (Springer Netherlands, Dordrecht, 1997) pp. 119–165.
[8] G. Guennebaud, B. Jacob, *et al.*, Eigen: A c++ template library for linear algebra (2009).
[9] D. Tao, Z. Gan, *et al.*, Spectra: A c++ library for large-scale eigenvalue problems (2016).
[10] R. B. Lehoucq, D. C. Sorensen, and C. Yang, Arpack: A library for solving large scale eigenvalue problems (1998).

[11] D. van Heesch *et al.*, Doxygen: The documentation generator (1997-2024).
[12] D. Ullmo, Bohigas-Giannoni-Schmit conjecture, Scholarpedia **11**, 31721 (2016), revision #169195.
[13] J. Mateos, F. Sols, and C. Creffield, Spectral statistics of driven bose-hubbard models (2024), arXiv:2306.09785 [cond-mat.quant-gas].
[14] C. Kollath, G. Roux, G. Biroli, and A. M. Läuchli, Statistical properties of the spectrum of the extended bose–hubbard model, Journal of Statistical Mechanics: Theory and Experiment **2010**, P08011 (2010).
[15] J. Shlens, A tutorial on principal component analysis (2014), arXiv:1404.1100 [cs.LG].
[16] O. Penrose and L. Onsager, Bose-einstein condensation and liquid helium, Phys. Rev. **104**, 576 (1956).