LUND UNIVERSITY
Faculty of Science

Optimizing the functionalities of a code for real time dynamics of finite systems

Yasser Mahfoud

Thesis submitted for the degree of Bachelor of Science
Project duration: 2 months

Supervisor: Assoc. Prof. Claudio Verdozzi
Co-Supervisor: Dr. Miroslav Hopjan

LUND
UNIVERSITY

Department of Physics
Division of Mathematical Physics
May 2018

Abstract

The many-body problem in quantum mechanics always presents new challenges and ways to discover new properties of existing materials. The difficulty (if not the impossibility) to solve the many-body problem analytically makes the numerical methods an appealing approach. Here we consider a Python code, which is under development to become an open-source tool to perform studies of finite lattice systems in- and out-of-equilibrium. The code already includes many features, such as studying the dynamical properties of a system using Lanczos adapted time evolution method, temperature-dependent expectation values and groundstate calculations and Optimal control. To expand this code's applicability to a wider range of problems, we added two new functions in the linear sector, they are responsible for computing the density-density response function in the frequency and the time domain. The two new functions were used to study three different clusters with a specific perturbation. We have also created a booklet enriched with worked out examples and details to guide the user through the installation and running process.

Acknowledgments

I would like to express my deep gratitude to my supervisor Professor Claudio Verdozzi for his support and guidance during my Bachelor thesis. I greatly appreciate the time you spent on helping me and the advise you provided throughout my work. Thank you for your patience in the times where I was lacking and your sense of humor that made this a fun experience. I would like to also thank Dr. Miroslav Hopjan for all his help and his constructive feedback during the writing of the thesis.

CONTENTS

List of Abbreviations

BO	Born-Oppenheimer
DMRG	Density-Matrix Renormalization Group
LRT	Linear Response Theory
ExCITeD	Exact Configuration Interaction Temporal Dynamics
QuTiP	Quantum Toolbox in Python
HH	Hubbard-Holstein

CHAPTER 1

INTRODUCTION

In this day in age, modern technology is an integral part of our life and is extremely embedded in our society. The quick pace at which devices are upgraded has become a norm, and the demand for better devices and performance is never ending. Industries are striving after shrinking the components in order to produce more compact devices with higher performance, but these attempts go hand in hand with quantum effects becoming more present at that scale. Those effects in solids can no longer be treated as being negligible, and one must go back to the fundamentals, and study the physics of solids, particularly their electronic structures.

The Schrödinger equation in principle describes solids, and solving it explains different materials and unravels ways to design new and more superior ones. The famous non-relativistic Schrödinger equation is given by

$$i\hbar\frac{\partial}{\partial t}\left|\Psi\right\rangle = \hat{H}\left|\Psi\right\rangle \tag{1.1}$$

with the full Hamiltonian

$$\hat{H} = -\sum_{\alpha=1}^{N_n}\frac{\mathbf{P}_\alpha^2}{2M_\alpha} - \sum_{j=1}^{N_e}\frac{\mathbf{p}_j^2}{2m_e} - \sum_{j=1}^{N_e}\sum_{\alpha=1}^{N_n}\frac{Z_\alpha e^2}{|\mathbf{r}_j - \mathbf{R}_\alpha|} + \sum_{j<k}^{N_e}\frac{e^2}{|\mathbf{r}_j - \mathbf{r}_k|} + \sum_{\alpha<\beta}^{N_n}\frac{Z_\alpha Z_\beta e^2}{|\mathbf{R}_\alpha - \mathbf{R}_\beta|}, \tag{1.2}$$

where $\mathbf{P}_\alpha$ is the momentum and $\mathbf{R}_\alpha$ is the position of nucleus α, M_α is the mass and Z_α is the atomic number. $\mathbf{p}_j$ and $\mathbf{r}_j$ correspond to the momentum and position of electron j and the number of electrons and nuclei are donated by N_e and N_n respectively.

Solving the full Hamiltonian of a system is a tough task, and for a large number of particles it is basically impossible. A major part of solid state theory is finding ways to determine electronic properties of solids with sufficient accuracy. That is done by searching for simulation techniques that are applicable to a wide range of problems while keeping the input to a minimum. This is the perspective of so-called first principle approaches, where the starting point is limited to the sole knowledge of the type of atoms in the system, and the computational approach is based on quantum mechanics. In this way, the atomic position and the consequent electronic properties are fully determined. A very successful simplification to this program is to consider the nuclei in their equilibrium position and replace the full Hamiltonian with a simplified one

describing explicitly only electrons. This is the strategy of the Born-Oppenheimer approximation: the masses of the nuclei are much larger than the masses of the electron, and due to that the nuclei move much slower than the electrons. This approximation decouples the nuclei from the electrons, resulting in an electronic Hamiltonian that describes purely electronic systems.

Even with the BO approximation, the full electronic Hamiltonian is still extremely hard to solve exactly. That is why physicists strive to construct even more effective many-body Hamiltonian models, reducing the number of degrees of freedom and making it simpler to solve them numerically. In electronic systems, the interplay of the Coulomb interaction and the kinetic energy of the electrons is central. In the case when the kinetic energy dominates, electrons are delocalized throughout the material. Most of these materials can be thought as having non-interacting electrons, and are decently described by the free-electron model. However, in the case where the Coulomb interaction is dominating, the movement of one electron is influenced by the locations of all other electrons in the material. Hence, one can not describe these materials by a free-electron model. The problem in such systems is treated as a many-body problem, and this regime is said to be correlated or strongly correlated. In this case, even very simple lattice models where electrons move on simple lattices with few (e.g. one) orbital /site can be helpful to gain qualitative insight. For these simple models, a number of techniques are available, which are usually not practically viable for a first-principle description.

Various numerical methods exist for ground-state and dynamical properties calculations for model Hamiltonians, such as Monte Carlo, DMRG, Green's functions and exact diagonalization [1, 2, 3, 4]. Monte Carlo methods statistically evaluate properties of the wave function with no restrictions on the number of dimensions of the problem, unlike DMRG which is only valid for 1D geometries. DMRG, however, treats unusually large systems with outstanding accuracy and does not have the famous "negative-sign problem" that exists in the Monte Carlo method. In the method of Green's functions one shifts the focus from the microscopic states to the behavior of specific correlation functions or average of observables. An example of this approach could be studying the conductivity of the system. In this work, however, we use exact diagonalization as an entry to the many-body problem. Exact diagonalization is a basic technique which constructs the basis states of a system explicitly, then one writes the quantum problem as an eigen problem

$$\hat{H} \left|\psi\right\rangle = E \left|\psi\right\rangle , \tag{1.3}$$

and solves for the eigenvalues and eigenvectors of the Hamiltonian operator, providing the possibility to compute observables in and out of equilibrium to study the properties of a given system. Even though this technique scales dramatically with increasing the size of the system, it is simple and reliable.

Numerical methods have provided the ability to simulate experiments in order to explain materials and explore new ways to improve on them. Typically in experiments, an external force is used to probe systems driving them away from their equilibrium state and measuring their response in order to learn more about them. In the regime where the external probe is weak

compared to the internal interactions of the system, the response can be described within perturbation theory as linearly dependent on the external force applied. This project work is concerned with a code which uses exact diagonalization to simulate electronic systems that will be used to study their properties in this regime.

1.1 Scope of the work

This project focuses on optimizing and expanding a pre-existing configuration-interaction Python code in order to study the behavior of electrons in finite systems. The purpose of this code is to describe such systems in the finite-size, small-particle-number case, with chosen initial conditions and parameters. The aim of this work is to add new functionalities to the code for future applications. The code will be a great addition to the packages available for Python to be used for conducting research as well as a pedagogical learning tool. The code was originally written in FORTRAN, where numerous papers were published using it [5, 6, 7, 8, 9, 10]. The translated Python version aims at encapsulating the features of the original code, and adding the pedagogical side for teaching/learning purposes.

In this diploma project, two new functionalities are added to allow the study of the response of a system to an externally applied force, taken as a perturbation of the basic operator of the system (Hamiltonian). These new functionalities compute the density-density response function in the time and frequency domain for a given system. For this scope, a description of Linear Response Theory and how it applies to the model used will be presented. Also, to show how the new functionalities works, an illustration with three different systems is presented.

A written booklet enriched with worked out examples for the code is created. The booklet can be found in the Appendix attached as a complete document.

As an extra task, we compare real-time evolution and linear response results with a chosen perturbation. This was used to characterize the regime where the two approaches give the same results, and where they differ.

CHAPTER 2

THEORY

2.1 THE HUBBARD MODEL

To study the behavior of electrons in correlated systems, one attempts to work around all the complexity of such systems. John Hubbard simplified this problem in a very insightful fashion when he developed his well-known model. The beauty of his model lies in its simplicity. Since atoms are arranged periodically in a lattice, each lattice site corresponds to an atomic site. The atoms are assumed to have only one orbit, where that orbital state is non-degenerate. In reality, atoms have more than one orbit, but it is reasonable for a qualitative description to assume that the ignored states do not play a significant role in low-energy physics. Figure 2.1 shows the steps of the simplification process.

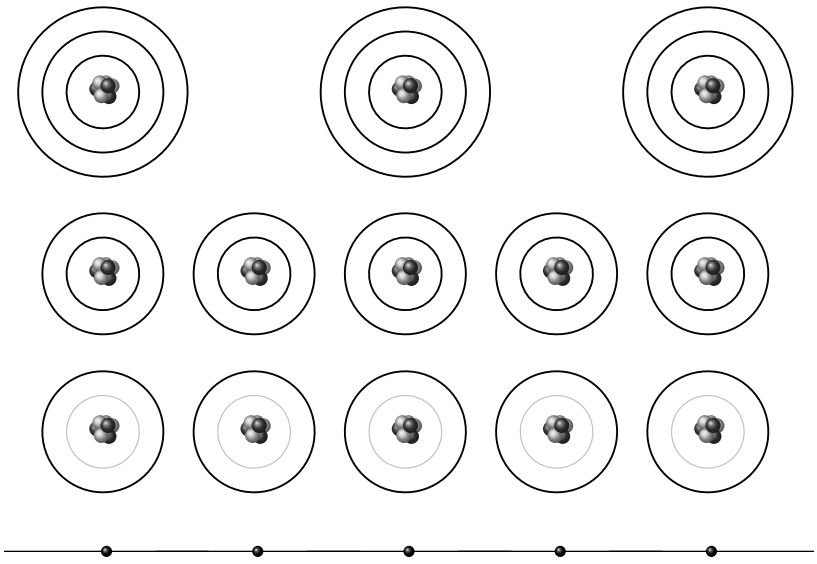


Figure 2.1: A sketch that shows the philosophy of this model, starting with free atoms with multiple electrons and orbits at the top. When atoms are brought together to form solids, electrons in the outermost orbits become delocalized through the solid. The electrons in the inner-most orbits are extremely localized. The electrons in the mid-orbit are mostly localized, but tunnel to nearby orbits with some probability. The electrons in the mid-orbits are the only ones considered in this model. If the mid-orbits are non-degenerate, one gets a lattice model where electrons live on the lattice sites and hop from one site to another. Adapted from [11].

The Hubbard Model [12] is one of the most important models in theoretical physics. It is a simple model that describes strongly correlated electrons and their motion in solids and is used in cold atom physics as an accurate description of ultracold atoms in optical lattices. The model offers a simple way to get insight on how the interaction between electrons gives rise to different effects in solids. Physicists have studied this model using a range of analytic techniques, but analytically one cannot always go very far. Arguably, at present the situations, the Hubbard model is best understood in 1D (where an exact solution is available) and in the limit of infinite dimensions, in terms of many-body non-perturbative treatments. The model has also been attacked with numerical methods such as diagonalization, which is rewarding for cases with a small number of particles.

The Hubbard Hamiltonian is generally presented as the sum of two terms as:

$$\hat{H} = \hat{H}_{kin} + \hat{H}_{int}, \tag{2.1}$$

where $\hat{H}_{kin}$ is the hopping term of the Hamiltonian and $\hat{H}_{int}$ is the Coulomb interaction term. The hopping Hamiltonian in the second quantization language is generally given by:

$$\hat{H}_{kin} = \sum_{i,j} \sum_{\sigma} h_{ij} c^{\dagger}_{i,\sigma} c_{j,\sigma} \quad \text{where} \quad \begin{cases} \text{if } \; i \neq j, & h_{ij} = t \\ \text{if } \; i = j, & h_{ii} = \epsilon_i \end{cases} \tag{2.2}$$

The hopping parameter t is assumed to be real (but it can become complex when magnetic fields are present), and it describes the quantum mechanical amplitude that an electron hops from site i to site j. The parameter ϵ_i is the single-particle potential which is usually called on-site energy. The operator $c^{\dagger}_{i,\sigma}$ creates an electron with spin $\sigma = \uparrow, \downarrow$ at site i, and $c_{j,\sigma}$ is the annihilation operator which destroys an electron at site j with spin σ. These operators obey the canonical anti-commutation relations:

$$\begin{aligned} \left\{ c^{\dagger}_{i,\sigma}, c_{j,\sigma'} \right\} &= \delta_{i,j} \delta_{\sigma,\sigma'} \\ \left\{ c^{\dagger}_{i,\sigma}, c^{\dagger}_{j,\sigma'} \right\} &= 0 \\ \left\{ c_{i,\sigma}, c_{j,\sigma'} \right\} &= 0. \end{aligned} \tag{2.3}$$

The interaction Hamiltonian is given by:

$$\hat{H}_{int} = \sum_{i} U_i \hat{n}_{i,\uparrow} \hat{n}_{i,\downarrow}, \tag{2.4}$$

where $\hat{n}_{i,\sigma}$ is the number operator that is defined to be $\hat{n}_{i,\sigma} = c^{\dagger}_{i,\sigma} c_{i,\sigma}$ and U is a constant that represents the raise in energy when two electrons occupy a single site i. The Coulomb interaction is a long ranged interaction, but for simplicity the only part that is taken into account in this model is the local, intrasite one.

2.2 Linear Response Theory

An interacting many-body system can be described by a time-independent Hamiltonian $\hat{H}_0$. To study how the system responds if driven away from equilibrium, one must examine the case where $\hat{H}_0$ interacts with an external force. The total Hamiltonian for such a case is given by:

$$\hat{H}(t) = \hat{H}_0 + \hat{H}'(t), \tag{2.5}$$

where $\hat{H}_0$ is the Hamiltonian of the unperturbed system and $\hat{H}'(t)$ is the time-dependent external perturbation. In many cases, the perturbation can be expressed as the following:

$$\hat{H}'(t) = -f(t)\hat{A}, \tag{2.6}$$

where $f(t)$ is a generalized force and $\hat{A}$ is an internal operator pertaining to the system which the force couples to. Time-dependent perturbation theory can be formulated in terms of the time-evolution operator. A convenient approach to this problem is to assume that the system is in the ground state of $\hat{H}_0$ at time t_0. The perturbation is then switched on and the state of the system at time t can be given by:

$$\big|\psi(t)\big\rangle = U(t, t_0)\,|\psi_0\rangle\,, \tag{2.7}$$

where $U(t, t_0)$ is the time evolution operator for the full Hamiltonian $\hat{H}(t)$ which has a compact form given by:

$$U(t, t_0) = T\exp\left(-i\int_{t_0}^{t} dt'\hat{H}(t')\right), \tag{2.8}$$

where T is the time ordering operator. The knowledge of $U(t, t_0)$ is needed to determine the time-evolution of any state perturbed by $\hat{H}'(t)$. At t_0 the system is in its ground state, and the expectation value of an arbitrary observable $\hat{B}$ is given by $\langle\psi_0|\hat{B}|\psi_0\rangle$. After switching on the perturbation, the state of the system evolves in time according to (2.7). The expectation value of $\hat{B}$ at time $t > t_0$ is then given by $\big\langle\psi(t)\big|\hat{B}\big|\psi(t)\big\rangle$. The change in this expectation value induced by the external perturbation can then be calculated in the following way:

$$\delta\left\langle\hat{B}(t)\right\rangle = \big\langle\psi(t)\big|\hat{B}\big|\psi(t)\big\rangle - \langle\psi_0|\hat{B}|\psi_0\rangle\,, \tag{2.9}$$

where $\delta\left\langle\hat{B}(t)\right\rangle$ describes the deviation from equilibrium over time under the influence of the external force. Here, we specialize to the linear regime by performing an expansion to first order in $\hat{H}'(t)$. The force $f(t)$ is assumed to be sufficiently weak in order to preserve the applicability of the approach. Noting that the expansion only affects the expression $\big\langle\psi(t)\big|\hat{B}\big|\psi(t)\big\rangle$, the expression can be manipulated to simplify the task, rewriting it as follows:

$$\big\langle\psi(t)\big|\hat{B}\big|\psi(t)\big\rangle = \langle\psi_0|U(t_0, t)\hat{B}U(t, t_0)|\psi_0\rangle \tag{2.10}$$

$$= \langle\psi_0|\hat{B}(t)|\psi_0\rangle\,, \tag{2.11}$$

where $\hat{B}(t)$ is the Heisenberg picture operator. A part of the manipulation process is to move to

the interaction picture. It is suitable to describe a system in that picture when an external time-dependent perturbation is applied, under the assumption that the solutions to the unperturbed system are known. It can be shown that (2.11) can be written in the interaction picture as follows:

$$\langle\psi_0|\hat{B}(t)|\psi_0\rangle = \langle\psi_0|U_I(t_0,t)\hat{B}_I(t)U_I(t,t_0)|\psi_0\rangle\,, \tag{2.12}$$

where $\hat{B}_I(t)$ is the interaction picture operator, and $U_I(t,t_0)$ is the time evolution operator in the interaction picture which is given by:

$$U_I(t,t_0) = T\text{exp}\left(-i\int_{t_0}^{t} dt'\hat{H}'_I(t')\right), \tag{2.13}$$

where $H'_I(t')$ is the perturbation operator also expressed in the interaction picture. At this point it is easier to perform the expansion and carry on with the derivation. Expanding (2.13) to first order in $\hat{H}'$, we get

$$U_I(t,t_0) \approx 1 - i\int_{t_0}^{t} dt'\hat{H}'_I(t'), \tag{2.14}$$

which by definition also means

$$U_I(t_0,t) = \left[U_I(t,t_0)\right]^\dagger \approx 1 + i\int_{t_0}^{t} dt'\hat{H}'_I(t'). \tag{2.15}$$

Inserting both expansions into (2.12) and performing some simple algebra:

$$\begin{aligned}\langle\psi_0|\hat{B}(t)|\psi_0\rangle &= \langle\psi_0|U_I(t_0,t)\hat{B}_I(t)U_I(t,t_0)|\psi_0\rangle \\ &\approx \langle\psi_0|\left(1 + i\int_{t_0}^{t} dt'\hat{H}'_I(t)\right)\hat{B}_I(t)\left(1 - i\int_{t_0}^{t} dt'\hat{H}'_I(t)\right)|\psi_0\rangle \\ &\approx \langle\psi_0|\hat{B}|\psi_0\rangle + i\int_{t_0}^{t} dt'\,\langle\psi_0|\left[\hat{H}'_I(t'),\hat{B}_I(t)\right]|\psi_0\rangle\,.\end{aligned} \tag{2.16}$$

Equation (2.16) describes how the expectation value of some arbitrary operator $\hat{B}$ changes in time under the effect of an external perturbation. Note that, in general,

$$\hat{H}'_I(t) = e^{i\hat{H}_0 t}\hat{H}'(t)e^{-i\hat{H}_0 t} = -f(t)e^{i\hat{H}_0 t}\hat{A}e^{-i\hat{H}_0 t} = -f(t)\hat{A}_I(t). \tag{2.17}$$

One can combine (2.9) and (2.16) to compute the deviation from equilibrium in terms of a linear dependence, which is given by:

$$\delta\left\langle\hat{B}(t)\right\rangle = i\int_{t_0}^{t} dt' f(t')\,\langle\psi_0|\left[\hat{B}_I(t),\hat{A}_I(t')\right]|\psi_0\rangle\,. \tag{2.18}$$

The result shows that the deviation of any physical quantity at time t is an integral over all force history, times a quantity which is completely independent of the perturbation. Through a redefinition,

$$\delta\left\langle\hat{B}(t)\right\rangle = \int_{t_0}^{t} dt' f(t')\phi_{AB}(t-t'), \tag{2.19}$$

where ϕ_{AB} is the response function. A key property of the response function is the preservation of causality: The system cannot show any response before the force is applied. Therefore the response function equals zero for $t' > t$. This statement of causality can also be made explicit by writing the linear response function with a step function where:

$$\phi_{AB}(t-t') = i\theta(t-t') \langle\psi_0|\Big[\hat{B}_I(t), \hat{A}_I(t')\Big]|\psi_0\rangle \,. \tag{2.20}$$

The time dependence of the system is only dependent on the time interval between the application of the force and the observation.

2.3 Density-Density Response Function

Up untill this point, the response function has been kept general in the two operators $\hat{A}$ and $\hat{B}$. We now specialize it to study the density-density response function by considering the exact particle density operator at a specific site of the unperturbed system. Here, what we mean by the density operator is the site-occupation number ($\hat{n}_i = \hat{n}_{i,\uparrow} + \hat{n}_{i,\downarrow}$) which is appropriate for calculations performed on a lattice, i.e. the expectation value of the density operator $n_i \equiv \langle\hat{n}_i\rangle$ at site i can vary from 0 to 2 with half filling at $n_i = 1$.

When specializing to the density-density response function, the general perturbation in (2.6) becomes $\hat{H}_i{}'(t) = -f_i(t)\hat{n}_i$. This local perturbation physically describes an external field which can be e.g. an electric potential that is coupled to the electronic charge density at a specific site i. To proceed in calculating the density-density response function, we substitute the general operators $\hat{A}$ and $\hat{B}$ in (2.20) with the density operators $\hat{n}_i$ and $\hat{n}_j$ respectively:

$$\chi_{ij}(t-t') = -i\theta(t-t') \langle\psi_0|\big[\hat{n}_{I,i}(t'), \hat{n}_{I,j}(t)\big]|\psi_0\rangle \,. \tag{2.21}$$

The subscripts i and j denote the site number, and the consequential minus sign is a result of switching the order of the two operators inside the commutator in equation (2.20). Up to this point, the response function has been in the time domain. We can Fourier-transform $\chi_{ij}(t-t')$ to get the equivalent frequency-dependent equation. This allows studying the response at specific frequencies. Setting $t'' = t' - t$ and taking the Fourier transform

$$\chi_{ij}(\omega) = -i\int_{-\infty}^{0} dt'' e^{-(i\omega-\eta)t''} \langle\psi_0|\big[\hat{n}_{I,i}(0), \hat{n}_{I,j}(t'')\big]|\psi_0\rangle \,, \tag{2.22}$$

where η is a Lorentzian broadening factor and $e^{\eta t''}$ ensures that the external perturbation is switched on in a smooth, adiabatic way, and the unwanted secondary effects may be removed by taking the limit $\eta \longrightarrow 0$. One can choose the excited states of the system to be eigenstates of the unperturbed Hamiltonian $\hat{H}_0$ such that:

$$e^{-i\hat{H}_0 t}|\psi_n\rangle = e^{-iE_n t}|\psi_n\rangle \,. \tag{2.23}$$

Inserting a complete set of the unperturbed states in the commutator in equation (2.22), we get

$$\left[\hat{n}_I(0), \hat{n}_I(t'')\right] = \left[\hat{n}_I(0), \sum_n |\psi_n\rangle \langle\psi_n| \, \hat{n}_I(t'')\right]. \tag{2.24}$$

Noting that $\hat{n_I} = e^{i\hat{H}_0 t}\hat{n}e^{-i\hat{H}_0 t}$, equation (2.22) becomes:

$$\begin{aligned}\chi_{ij}(\omega) = -i\int_{-\infty}^{0} dt'' \, (\sum_n \langle\psi_0|\hat{n}_i|\psi_n\rangle \, \langle\psi_n|\hat{n}_j|\psi_0\rangle \, e^{i(E_n - E_0)t'' - (i\omega - \eta)t''} \\ - \sum_n \langle\psi_0|\hat{n}_j|\psi_n\rangle \, \langle\psi_n|\hat{n}_i|\psi_0\rangle \, e^{-i(E_n - E_0)t'' - (i\omega - \eta)t''} \,).\end{aligned} \tag{2.25}$$

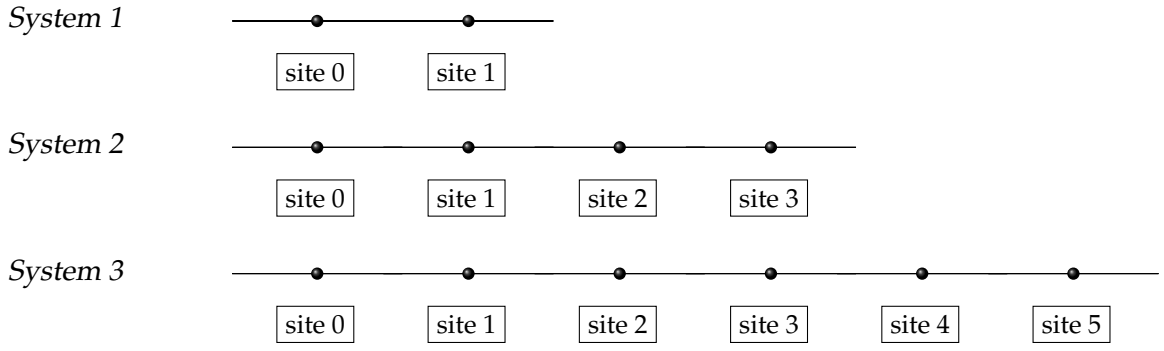
All the time dependence is now contained in the exponentials. Performing the integration over time will yield the density-density response function $\chi_{ij}(\omega)$ in the frequency domain:

$$\chi_{ij}(\omega) = \sum_n \left[\frac{\langle\psi_0|\hat{n}_i|\psi_n\rangle \, \langle\psi_n|\hat{n}_j|\psi_0\rangle}{\omega - (E_n - E_0) + i\eta} - \frac{\langle\psi_0|\hat{n}_j|\psi_n\rangle \, \langle\psi_n|\hat{n}_i|\psi_0\rangle}{\omega + (E_n - E_0) + i\eta} \right]. \tag{2.26}$$

The function has poles when $\omega \to (E_n - E_0)$, the excitation energies. Thus, one can determine the excitation energies of a system by using this equation. The imaginary part of the density-density response function is given by:

$$S_{ij}(\omega) = -\frac{1}{\pi} \operatorname{Im}\{\chi_{ij}(\omega)\}. \tag{2.27}$$

In this thesis work, the new functionality added to the code gives the possibility to compute the density-density response function $\chi_{ij}(\omega)$ for a given system. A demonstration of the new functionality is presented in Chapter 4 by computing the response function for three different clusters at half filling consisting of two, four and six sites respectively.



In *System 1* we considered a dimer with two sites labeled 0 and 1, with the terms $h_{01} = h_{10} = t$, $U_0 = U_1 = U$, while the on-site energy term is set to $h_{00} = h_{11} = \epsilon_0 = 0$. In *System 2*, with four sites labeled 0,1,2 and 3, the sites 0 and 1 represent the previously defined dimer, and the other terms considered are $h_{12} = h_{21} = h_{23} = h_{32} = t$, $U_2 = U_3 = U$ and $h_{22} = h_{33} = \epsilon_1 = 0$. In the last system, *System 3*, the six sites are labeled 0,1,2,3,4 and 5, this cluster consists of the previously defined four-site system plus the two sites 4 and 5. The additional parameters considered are $h_{34} = h_{43} = h_{45} = h_{54} = t$, $U_4 = U_5 = U$ and $h_{44} = h_{55} = \epsilon_2 = 0$.

CHAPTER 3

THE EXCITED CODE

In this chapter, we will provide a brief description of the code/module ExCITeD, which stand for Exact Configuration Interaction Temporal Dynamics, while an equivalent description with more detail and worked out examples can be found in the Appendix. We will also present the two new functions[1] added to the module, and explain their construction process in order to demonstrate how one can use the code and build new functions catered towards a wide range of problems. The long term aim of this work is to provide an open-source package that can be installed and used also by a third party to possibly conduct research, but also for educational purposes.

The purpose of this code is to investigate the dynamics of systems with electron-electron and electron-phonon interactions. So far, the model Hamiltonian considered in this work has been the Hubbard model, focusing purely on electron-electron interactions. We have integrated out the nuclei effects in materials and took them as a passive background that electrons move in. However, in many materials, this is not the case. The lattice vibrations interact with electrons via the electron-phonon interaction. These interactions can give rise to superconductivity [13], spin and charge density waves [14], and many more interesting phenomena. A simple model Hamiltonian that describes those interactions in a specific way is the Holstein model [15]. It was introduced in the 1950's and describes a system of tight-binding electrons coupled to a dispersionless phonon mode. However, the Holstein model is used to study systems in the absence of electron-electron repulsion. A more general situation is where in the lattice we have both the electron-electron and the phonon-electron interactions. The interplay of these two quantities is incorporated in the Hubbard-Holstein model [16], which is a hybrid between the Hubbard model (which has no electron-phonon coupling), and the Holstein model (which has no Coulomb repulsion). The Hubbard-Holstein Hamiltonian is given by:

$$H_{HH} = \underbrace{t \sum_{<i,j>,\sigma} c^{\dagger}_{i,\sigma} c_{j,\sigma}}_{H_{kin}} + \underbrace{\sum_{i} U_i n_{i,\uparrow} n_{i,\downarrow}}_{H_{Hub}} + \underbrace{\omega_0 \sum_{i} b^{\dagger}_i b_i + g \sum_{i,\sigma} n_{i\sigma}(b^{\dagger}_i + b_i)}_{H_{e-ph}}, \tag{3.1}$$

[1] Here we took a simple route and built two functions considering how simple it is to analytically compute them in each domain. In the future, Fourier transform and convolution routines will be used to merge these into one function.

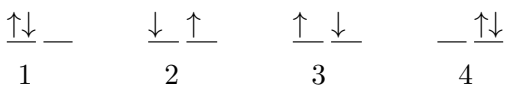
where H_{kin} describes the nearest-neighbor hopping of electrons, H_{Hub} is the Hubbard interaction term and H_{e-ph} describes the kinetic energy of phonons with frequency ω_0 and electron-phonon interaction of strength g. The sum of the first two parts $H_{kin} + H_{Hub}$ are the already defined Hubbard model in Section 2.1. $b_i^\dagger(b_i)$ are the creation (annihilation) operators for phonons at site i. The code can deal with electrons and boson modes (photons, phonons) mutually interacting in a lattice (the HH model is an example of this sort). The code can also be used for a pure Hubbard model, or just a simple tight-binding Hamiltonian.

The code was first developed and written by C. Verdozzi in FORTRAN language. Later, it was augmented with the optimal control functionality and translated to the Python language by his student S. Ydman as part of his work with electron-phonon dynamics in Hubbard rings and chains [17] and magnetic rings [18, 19]. The code was built starting from an existing module for Python called QuTiP [20] (Quantum Toolbox in Python). The Quantum Toolbox in Python is an open-source framework that includes several classes for creating and managing quantum objects; a quantum object is a Class in QuTiP that is capable of encapsulating the properties of a quantum operator and ket/bra vectors. QuTiP has served as a good framework for the code, and offered simple routines such as calculating expectation values. The main features of the code such as defining and constructing the basis and the Hamiltonian of the system, the more efficient Lanczos-adapted time evolution routine and the optimal control functionality, are independent of QuTiP.

In this initial phase of the optimization of the code, only fermionic degrees of freedom will be considered, i.e. bosonic systems and fermion-boson interactions are not discussed. Specifically, we focus on the Hubbard Hamiltonian, with phonons not considered in what follows. Neither we will take into account thermal effects, magnetic fields or optimal control, these are left for future work. Finally, a Hubbard dimer (i.e. a two-site system) will be used as example to illustrate some of the features of the code, e.g. the construction of the set of basis states and of the Hamiltonian.

3.1 Basis states

An important step in solving any problem in quantum mechanics is presenting the basis states for a possible representation of the system's wavefunction. The dimer problem defined in Section 2.3 is simple enough to analytically find the different combinations to organize a spin-up and a spin-down electron in two-sites:



The code allows the user to choose the initial parameters of the system of interest to create a set of basis states. The initial parameters include: number of fermions with the desired spin `(nup, ndw)`, number of sites/orbits `(norb)`, and an optional parameter to include phonons

`(nmode, nph)`. However, as specified earlier, phonons are outside the scope of the present discussion for the Hubbard model. The class

```
Basis( nup, ndw, norb, nmode, nph)
```

returns the set of basis states as an object which has different attributes, such as the density operator, the double density operator and more. For the dimer problem, the configuration above can be described by the `Basis` class in the second quantization language by acting with creation operators on the the vacuum state $|vac\rangle$ in the following manner:

$$
\begin{aligned}
|0\rangle &= \underline{\uparrow\downarrow}\ \_\_ \implies c^\dagger_{0\uparrow} c^\dagger_{0\downarrow} |vac\rangle \\
|1\rangle &= \underline{\downarrow}\ \underline{\uparrow} \implies c^\dagger_{1\uparrow} c^\dagger_{0\downarrow} |vac\rangle \\
|2\rangle &= \underline{\uparrow}\ \underline{\downarrow} \implies c^\dagger_{0\uparrow} c^\dagger_{1\downarrow} |vac\rangle \\
|3\rangle &= \_\_\ \underline{\uparrow\downarrow} \implies c^\dagger_{1\uparrow} c^\dagger_{1\downarrow} |vac\rangle
\end{aligned}
$$

As an example, the first basis state $|0\rangle$ physically describes two electrons with spin $\uparrow$ and $\downarrow$ occupying the site 0. The order of the creation operators can be switched with the consequential negative sign. The choice does not matter as long as one sticks to a convention.

Finding the basis states becomes a tedious task when increasing the number of electrons and sites. For example, a 12 site problem with half filling has 853776 basis states, which is still considered a small system to be studied. The code constructs the basis states by considering the ways of distributing spin up electrons $N_\uparrow$ and spin down electrons $N_\downarrow$ into the sites N_s, i.e $N = \binom{N_s}{N_{\uparrow,\downarrow}}$. The number of ways can be used to construct N-element sets, which after performing a Cartesian product [21] of these sets returns the set of basis states.

3.2 KINETIC ENERGY OPERATOR

The kinetic energy operator in the matrix representation can be found using an attribute of the `Basis` class. The attribute

```
hop_op( [(i,j)] , Value )
```

uses the set of basis states to find the matrix representation of the fermonic hoppings by solving the following matrix elements:

$$\sum_\sigma t_{ij} \langle\lambda| c^\dagger_{i,\sigma} c_{j,\sigma} |\nu\rangle \,, \tag{3.2}$$

where λ and ν represent basis states. The entry `[(i,j)]` specifies the sites that the hopping occurs between, while the argument `Value` specifies the value of the hopping parameter of the nearest neighbor t_{ij}. The attribute includes an optional argument `Spin` where one can specify the spin instead of summing over both spins as in (3.2).

3.3 DENSITY OPERATOR

The density operator $\hat{n}_i$ in the matrix representation is found using another attribute of the `Basis` class. The attribute

```
dens_op( Site )
```

uses the basis states to find the density matrix by solving the following matrix elements:

$$\tilde{n}_{\lambda\nu} \equiv \langle\lambda|\, \hat{n}_i \,|\nu\rangle = \langle\lambda|\, c^\dagger_{i,\uparrow} c_{i,\uparrow} + c^\dagger_{i,\downarrow} c_{i,\downarrow} \,|\nu\rangle\,. \tag{3.3}$$

The attribute requires the entry `Site` which specifies the site i, and returns the density matrix at the chosen site. The attribute also includes an optional argument `Spin`. If the spin is specified, the attribute returns the matrix elements of the number operator

$$\tilde{n}^{(\sigma)}_{\lambda\nu} \equiv \langle\lambda|\, \hat{n}_{i,\sigma} \,|\nu\rangle = \langle\lambda|\, c^\dagger_{i,\sigma} c_{i,\sigma} \,|\nu\rangle\,. \tag{3.4}$$

This attribute can be used to find the on-site energy part of the Hubbard Hamiltonian.

3.4 DOUBLE DENSITY OPERATOR

The double density operator is the product of two density operators with same site index but with opposite spin. The operator in the matrix representation can be found by using an attribute of the `Basis` class. The attribute

```
double_dens_op( Site )
```

uses the basis states to compute the following matrix elements:

$$\langle\lambda|\, n_{i,\uparrow} n_{i,\downarrow} \,|\nu\rangle\,. \tag{3.5}$$

The attribute requires the entry `Site` which specifies the site i, and returns the double density matrix at the chosen site. This attribute can be used to find the interaction part of the Hubbard Hamiltonian H_{int}.

3.5 HAMILTONIAN

The code allows the user to define the Hamiltonian by using the set of basis states. The class

```
Hamiltonian( Basis )
```

takes the set of basis states as an entry and creates an empty matrix with dimensions equal to the number of basis states. The H_{hop} and H_{int} parts are defined individually and then added to form the full Hamiltonian of the system. That is done by using the attribute `add_part` of the `Hamiltonian` class.

The parts needed to assemble the full Hamiltonian can be found using the attributes presented in Sections 3.2, 3.3 and 3.4. The full Hamiltonian for a system is presented as the sum of all the parts as the following:

$$\begin{aligned} H_{\lambda\nu} &= \sum_{ij} \texttt{hop\_op([(i,j)])} + \sum_i h_{ii}(\texttt{dens\_op(i)}) + \sum_i U_i(\texttt{double\_dens\_op(i)}) \\ &= \sum_{\sigma} \sum_{ij} t_{ij} \langle\lambda| c^{\dagger}_{i,\sigma} c_{j,\sigma} |\nu\rangle \quad + \quad \sum_i h_{ii} \langle\lambda| \hat{n}_i |\nu\rangle \quad + \sum_i U_i \langle\lambda| \hat{n}_{i,\uparrow} \hat{n}_{i,\downarrow} |\nu\rangle \,. \end{aligned} \tag{3.6}$$

For the dimer system, the Hamiltonian with the defined parameters in Section 2.3 reads:

$$H = \begin{pmatrix} U & t & t & 0 \\ t & 0 & 0 & t \\ t & 0 & 0 & t \\ 0 & t & t & U \end{pmatrix}. \tag{3.7}$$

The matrices are stored in sparse format, which means that the non-zero elements of the matrix are the only stored elements, thus saving storage space. Another advantage of the sparse matrix storage is when it comes to performing computations, where the zero elements of the matrix are not considered during, for example, matrix-multiplication.

3.6 Exact Diagonlization

The code includes a function that exactly diagonalizes the given Hamiltonian and returns the eigenfunctions $|\psi_i\rangle$ with their respective eigenvalues E_i. The lowest eigenvalue with its respected eigenfunction correspond to the groundstate energy and the groundstate. The function

```
qt_eigs( H , nval , maxiter )
```

is a wrapper to the ARPACK library which uses the implicitly restarted Lanczos method to find the eigenvalues and eigenvectors [22]. `nval` is the number of eigenvalues and eigenvectors to calculate, and `maxiter` is the number of maximum iterations.

3.7 The Response function

A part of this thesis work is to implement a new function to calculate the density-density response function in the frequency domain (2.26). The new implemented function

```
dens_freq( H , basis, w, Site i, Site j, nay)
```

takes as an input the Hamiltonian of the system `H`, the set of basis states `basis`, the range of omega `w`, the choice of sites i, j and the broadening factor `nay` (η) . The function is constructed using the discussed functions and attributes in the code.

The density-density-response function is built from the expectation values $\langle\psi_0|\hat{n}_i|\psi_n\rangle$ and the eigenvalues E_n. One can expand the groundstate and an excited state n in the basis states as:

$$\langle\psi_0| = \sum_{\lambda} m_{\lambda}^{\psi_0} \langle\lambda| \tag{3.8}$$

$$|\psi_n\rangle = \sum_{\nu} m_{\nu}^{\psi_n} |\nu\rangle \tag{3.9}$$

where $m_{\lambda}^{\psi_0}$ is the expansion coefficient for $\langle\psi_0|$ in the basis states $\langle\lambda|$. The expectation value can be written as:

$$\langle\psi_0|\hat{n}_i|\psi_n\rangle = \left(\sum_{\lambda} m_{\lambda}^{\psi_0} \langle\lambda|\right) \hat{n}_i \left(\sum_{\nu} m_{\nu}^{\psi_n} |\nu\rangle\right) \tag{3.10}$$

$$= \sum_{\lambda,\nu} m_{\lambda}^{\psi_0} m_{\nu}^{\psi_n} \langle\lambda| c_{i,\uparrow}^{\dagger} c_{i,\uparrow} + c_{i,\downarrow}^{\dagger} c_{i,\downarrow} |\nu\rangle \,. \tag{3.11}$$

This is an example of how to use the code to construct new functions for different problems. The matrix elements can be found using the `dens_op` attribute

$$\langle\psi_0|\hat{n}_i|\psi_n\rangle = \sum_{\lambda,\nu} m_{\lambda}^{\psi_0} m_{\nu}^{\psi_n} \texttt{dens\_op(i)}\,, \tag{3.12}$$

where the expansion coefficients m_{λ} and the eigenvalues E_n are provided by the Eigensolver `qt_eigs` discussed in Section 3.6. The same procedure is done for the expectation value at the site j, and the density-density response function is then subsequently computed.

3.8 TIME EVOLUTION

The time evolution of a system can be obtained using the function

```
lanczos_te( H , psi0 , tlist , e_ops )
```

implemented in the code. The function solves the time-dependent Schrodinger equation

$$i\frac{\partial\psi(t)}{\partial t} = H(t)\psi(t) \tag{3.13}$$

in an efficient way using the Lanczos-adapted time-evolution method [23]. The function takes as an input the Hamiltonian of the system, the time range, the groundstate of the system and a specified operator. The function then time-evolves the system and returns the expectation value of a specified operator. In this work, this function is used to look at the full dynamics of the density operator for the three systems defined in Section 2.3 under a chosen perturbation and compare that with LRT results. The LRT results are obtained by introducing another new function, which time-evolves the system and returns the expectation value of the density operator following the premise in (2.16) for a specific perturbation.

Substituting the general response function in (2.19) with the density-density response function defined in (2.21), and setting the lower limit $t_0 = 0$ in order to not consider negative time

values:

$$\delta \langle n_j(t) \rangle = \int_0^t dt' \sum_i f_i(t') \chi_{ji}(t - t'). \tag{3.14}$$

Inserting a complete set of unperturbed states in the commutator as in (2.24), the density-density response function becomes:

$$\begin{aligned} \chi_{ij}(t - t') = i \sum_n (& \langle \psi_0 | \hat{n}_i | \psi_n \rangle \langle \psi_n | \hat{n}_j | \psi_0 \rangle e^{i\widetilde{E}(t'-t)} \\ & - \langle \psi_0 | \hat{n}_j | \psi_n \rangle \langle \psi_n | \hat{n}_i | \psi_0 \rangle e^{-i\widetilde{E}(t'-t)}), \end{aligned} \tag{3.15}$$

where $\widetilde{E} = E_n - E_0$. The matrix elements $\langle \psi_0 | \hat{n}_i | \psi_n \rangle$ and the eigenvalues are found following the same procedure discussed in Section 3.7.

The local perturbation $f_0(t')$ at site 0 for both approaches is chosen to be:

$$f_0(t') = \begin{cases} g_0(t') = \dfrac{A}{2} \left(1 - \dfrac{1}{2} \left(e^{i\frac{\pi t'}{T}} + e^{-i\frac{\pi t'}{T}} \right) \right), & \text{if } t' < T \\ A, & \text{if } t' > T. \end{cases} \tag{3.16}$$

The perturbation as shown in Figure 3.1 can be thought to be controlled with a knob that has a maximum value of A, making the parameter T the time it takes for the perturbation to go from $0 \longrightarrow A$.

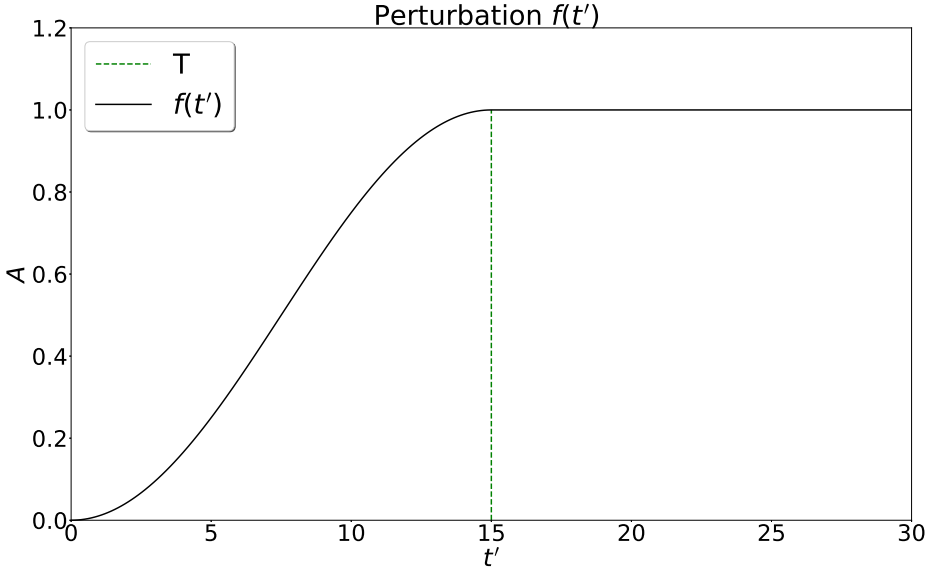


Figure 3.1: A figure that shows the perturbation used for both approaches with the parameters set to $A = 1$ and $T = 15$ as an example.

Here, we specify the local perturbation to site 0 only, and we take the expectation value of the density operator at the same site (site 0). Taking into account the conditions on the perturbation,

the change in the expectation value of the density over time at site 0 in (3.14) becomes:

$$\delta \left\langle n_0(t) \right\rangle = \begin{cases} \int_0^t dt' g_0(t') \chi_{00}(t - t'), & \text{if } t' < T \\ \\ \int_0^T dt' g_0(t') \chi_{00}(t - t') + \int_T^t dt' A \chi_{00}(t - t'), & \text{if } t' > T. \end{cases} \tag{3.17}$$

The change in the expectation value $\delta \left\langle n_0(t) \right\rangle$ is analytically solved in order to construct the function

```
exp_dens_t( H , basis, t , T , A , i , j )
```

which computes $\delta \left\langle n_0(t) \right\rangle$ for the specified perturbation in (3.16). Then, the function calculates the expectation value of the density operator in the ground state $\langle \psi_0 | \hat{n}_0 | \psi_0 \rangle$, which is done following the same procedure in Section 3.7. Finally, the function `exp_dens_t` returns the expectation value over time of the density operator:

$$\left\langle \psi(t) \right| \hat{n}_0 \left| \psi(t) \right\rangle_{LRT} = \langle \psi_0 | \hat{n}_0 | \psi_0 \rangle + \delta \left\langle \hat{n}_0(t) \right\rangle . \tag{3.18}$$

The function `exp_dens_t` takes as an input the Hamiltonian of the system `H`, the set of basis, the range of t, the speed of the perturbation T, the strength of the perturbation `A` and the choice of sites i, j which in this case are set to 0. This could be done for any arbitrary geometry, choice and number of sites, and also in the future, in the presence of electron-phonon interactions.

CHAPTER 4

RESULTS

In this chapter, we present the results produced by the two new functions added to the code. First, we compute the imaginary part of the density-density response function in the frequency domain for the three clusters defined in Section 2.3. Then, we calculate the expectation value of the density operator over time for the dimer system using both full time dynamics and LRT approach. We also expand the computations to the four-site and the six-site system to determine the role that the system size plays. Finally, we conclude this chapter with a sketch characterizing the regime where the two approaches agree.

4.1 DENSITY-DENSITY RESPONSE FUNCTION

In this section, the imaginary part of the density-density response function in the frequency domain $S_{ij}(\omega)$ is computed for the three clusters discussed in Section 2.3. The results are produced using the new implemented function in the code `dens_freq`. In the following, the lorentzian broadening is set to $\eta = 0.1$, and the sites i, j are chosen to be the first site $i = j = 0$. The common hopping parameter for the neighboring sites t is set to 1 and the frequency range is set to $\omega = 10$.

In Figure 4.1, the imaginary part of the density-density response function in the frequency domain $S_{00}(\omega)$ is shown as a function of ω. The parameters that are varied are the number of sites for $L = 2, 4$ and 6, corresponding to the systems previously defined, and the common Coulomb interaction energy for $U = 0, 3$ and 6. As mentioned in Section 2.3, one can determine the excitation energies of the system by using equation (2.26). The poles in Figure 4.1 represent the excitation energies of the system as they correspond to when $\omega \longrightarrow (E_n - E_0)$. For the three clusters considered, we can see that an increase in the value of the common Coulomb interaction energy parameter U results in shifting the poles towards a higher ω value, i.e. increasing the excitation energies of the systems.

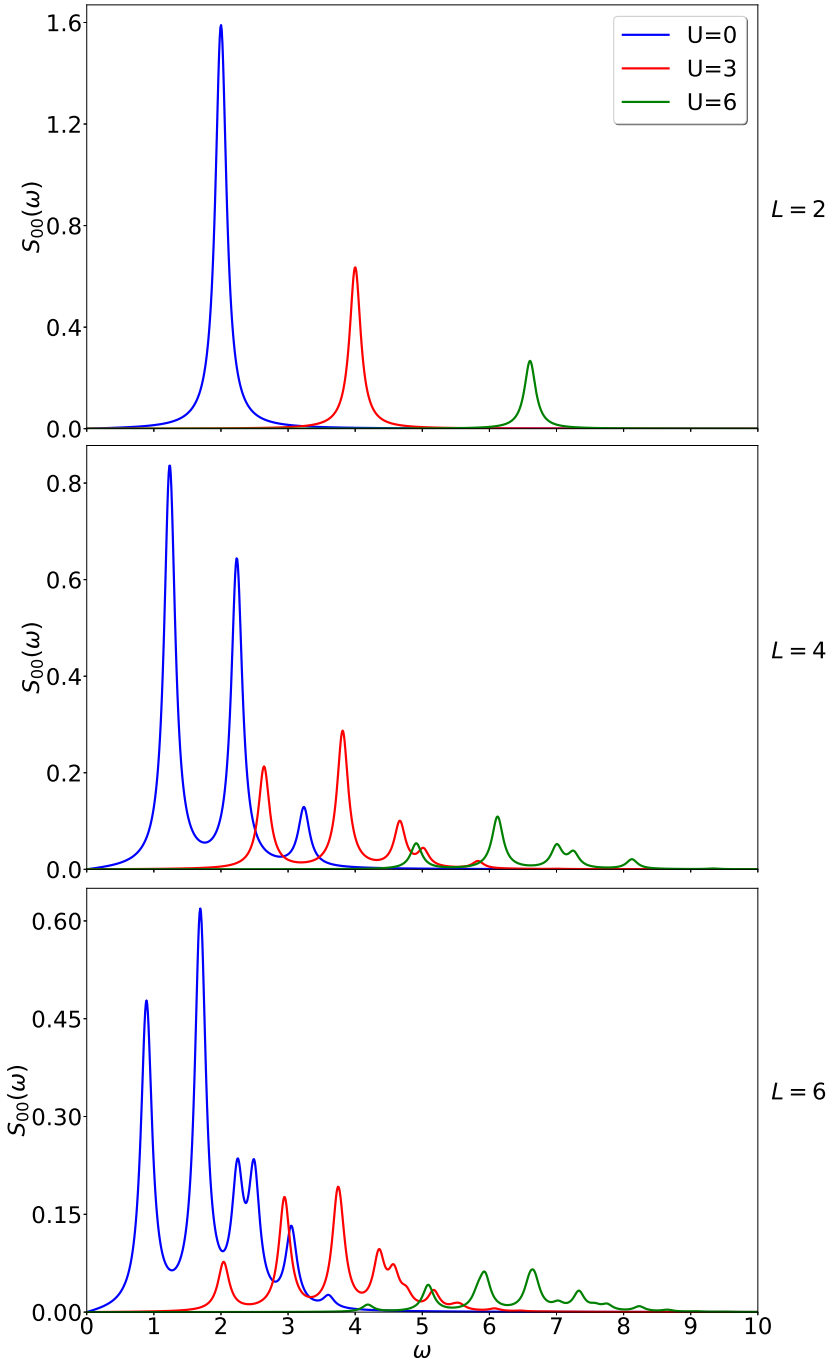


Figure 4.1: A figure that shows the imaginary part of the density-density response function $S_{00}(\omega)$ as a function of ω with $t = 1$ and U=0,3 and 6 for the three systems with $L = 2, 4$ and 6.

In this work we do not wish to further pursue the analysis of these results, since our main aim is to introduce the code and its functionalities. A more detailed analysis is deferred to future work.

4.2 TIME EVOLUTION

In this section, both the full time evolution and LRT results will be presented and compared. The results are produced using the `lanczos_te` function implemented in the code and the new function `exp_dens_t` added. In the following, the site perturbed is site 0, which is also the site that the expectation value of the density operator is taken at. Therefore, in (3.17) the choice of site i and j for χ_{ij} are set to site 0. The hopping parameter between the sites is set to $t = 1$ and the time range is set to 30. The parameters that are varied are U, T and A, which are defined in Section 3.8.

Since there are three parameters to vary, three plots have been produced to cover the comparison over all the parameters for the dimer system at half filling. In Figure 4.2, the parameters T and U are kept as constants at $U = 1$ and $T = 3$ and the expectation value of the density operator is computed for different values for the perturbation strengths $A = 1$, 0.5 and 0.1.

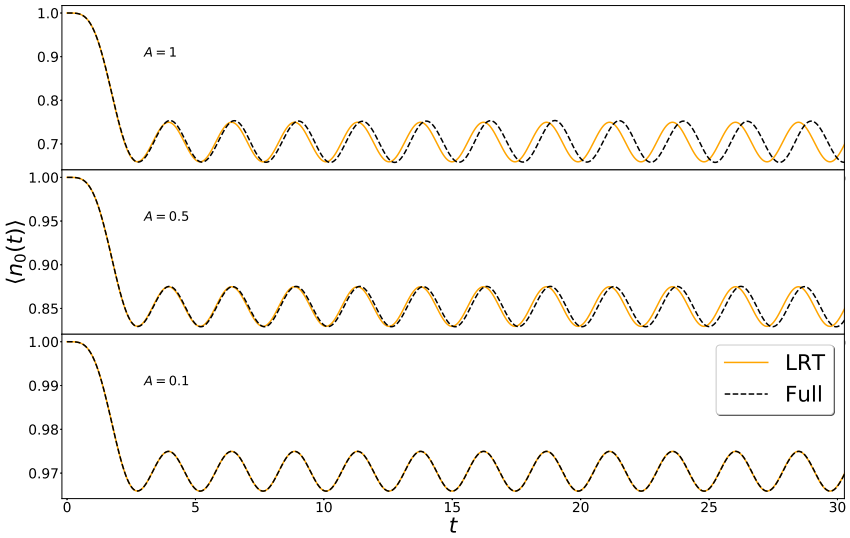


Figure 4.2: A figure that shows the comparison between full dynamics and LRT results of the expectation value of the density operator at site 0 as a function of time. The expectation value was computed for $U = 1$, $T = 3$ and for $A = 1$,0.5 and 0.1.

The two approaches seem to deviate at a relatively high perturbation value $A = 1$, while the agreement increases as the perturbation value is decreased. This is already expected, as it was an assumption of the LRT approach.

In Figure 4.3, the parameters U and A are kept as constants at $U = 1$ and $A = 1$, and the expectation value of the density operator is computed for different values for $T = 3$, 5 and 16, which controls the speed of the perturbation.

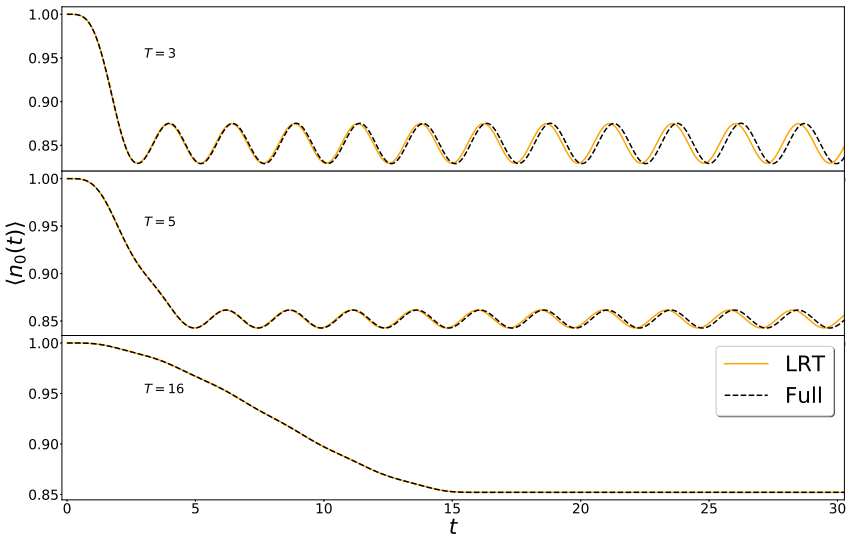


Figure 4.3: A figure that shows the comparison between full dynamics and LRT results of the expectation value of the density operator at site 0 as a function of time. The expectation value was computed for $U = 1$, $A = 1$ and for $T = 3$,5 and 16.

Increasing the parameter T slows the speed of the perturbation and results in an adiabatic smooth switching on of the perturbation. Slowing the perturbation increases the agreement between full dynamics and LRT results.

Finally, in Figure 4.4, the parameters T and A are kept as constants at $T = 3$ and $A = 0.5$ and the expectation value of the density operator is computed for different values for $U = 1$,4 and 8, which is the Coulomb interaction energy parameter.

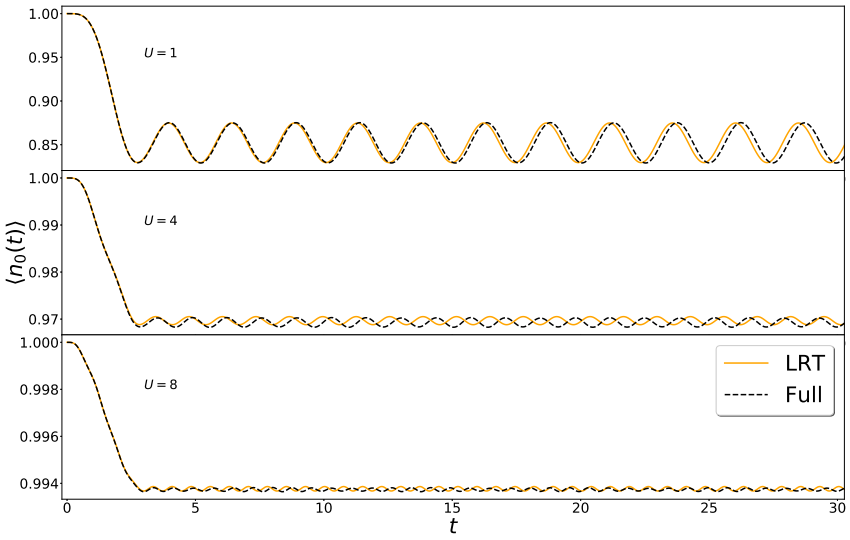


Figure 4.4: A figure that shows the comparison between full dynamics and LRT results of the expectation value of the density operator at site 0 as a function of time. The expectation value was computed for $T = 3$, $A = 0.5$ and for $U = 1$,4 and 8.

The increase in the Coulomb interaction energy parameter U improves the agreement between both approaches by making the perturbation strength A less influent. In that regime, the perturbation is very small compared to the interactions in the system, making it almost insignificant.

One can also see that the fluctuations of the expectation value for both approaches appear more dephased in this regime.

4.2.1 The effect of system size and the role of interactions

Here, we extend our comparison to the four and six site systems defined in Section 2.3. The speed of the perturbation is kept at a constant value $T = 3$, and the time range is set to 50. The parameters that are varied are U, A, and the number of sites L. In Figure 4.5 we compare the results for $U = 1$ and 4, $A = 0.5$ and 0.1 and for $L = 4$ and 6 sites.

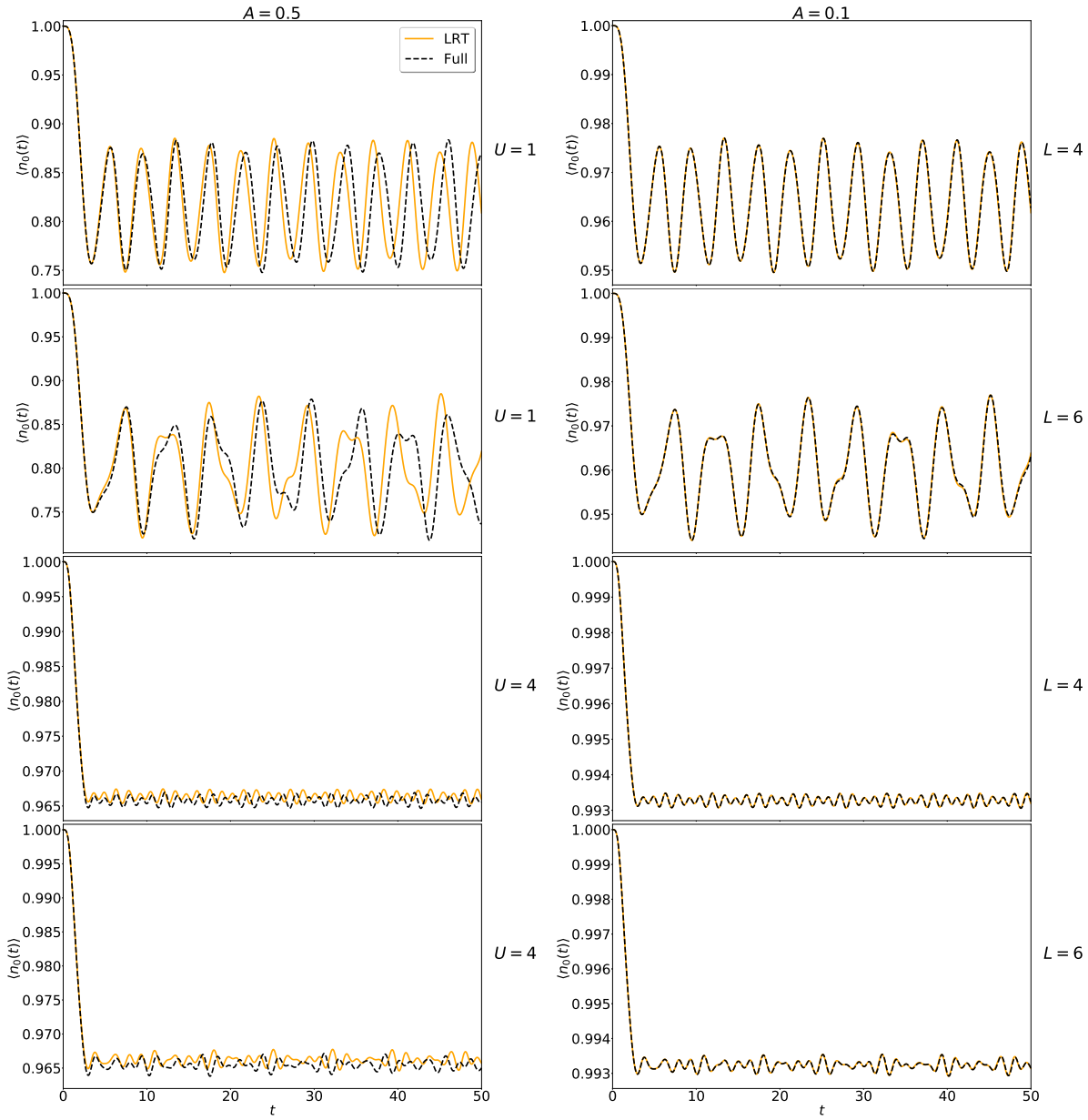


Figure 4.5: A figure that shows the comparison between full dynamics and LRT results of the expectation value of the density operator at site 0 as a function of time. The expectation value was computed for $T = 3$, $U = 1$ and 4, $A = 0.5$ and 0.1 for the two systems with $L = 4$ and 6 at half filling.

One can see that for larger systems, the full dynamic and LRT results also agree well at low perturbation values. We noticed that increasing the number of sites affected the expectation value of the density operator. Looking at Figure 4.5 and 4.2 for $U = 1$, when the perturbation is at $A = 0.5$, the expectation value for the two-site system hovers around 0.8, while for the six and four site system it hovers around 0.8. The perturbation is more effective in a larger system. For a higher interaction value, it seems like a situation where this is the best agreement. However, this is only apparent, since a higher U hinders large density oscillations. Furthermore, the discrepancies noted at $A = 0.5$ are fully consistent with all the other comparisons.

One can see that the agreement between the full dynamics (FD) and the LRT approach favors low perturbation strength, slow perturbation speed and high Coulomb interactions with an increase in the disagreement otherwise. We conclude with characterizing the regime where the two approaches agree for the dimer system only. We determine that by computing the disagreement between the approaches using:

$$\frac{1}{n}\sum^{n}\left|\langle n(t_n)\rangle_{LRT} - \langle n(t_n)\rangle_{FD}\right|, \tag{4.1}$$

where t_n is the time with n being the number of increments taken between the range $0 \longrightarrow t$. Another set of calculations was done for the dimer system considering different values for the parameters. The range of the parameters was increased to cover a bigger regime. The strength of the perturbation is varied between $A = 0.5, 1.5, 3$, while the speed of the perturbation is varied between $T = 3, 10, 26$ and the Coulomb interaction parameter is kept between $U = 1, 4, 8$. Figure 4.6 represents the agreement of the LRT approach taken as a function of parameters A, U and T. If the disagreement is lower than %5, a check sign is placed, while if it is between %5 and %10, a bar is placed, and if it is greater than %10, a cross sign is placed.

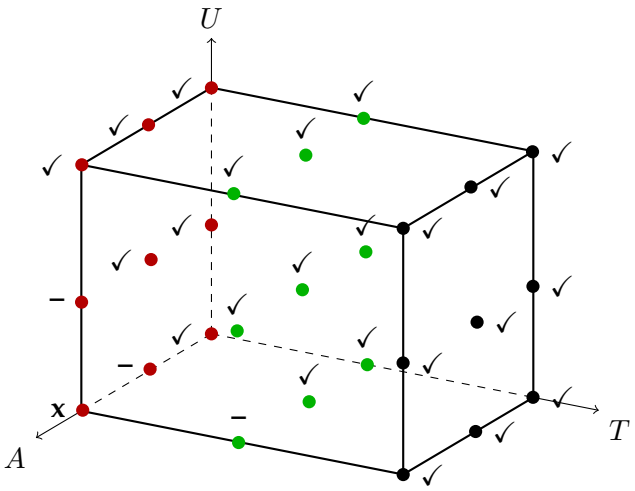


Figure 4.6: A sketch that characterizes the agreement between full dynamics and LRT results in computing the expectation value of the density operator over time.

Chapter 5

Outlook and Future Work

In this Chapter, we will provide a brief summary of the results and the work done with this code during the course of this project. We will also provide some information on how we plan to proceed with future work developments.

In Chapter 2 we presented a brief description of the Hubbard model. Starting from the general response function for any electronic system coupled to an external perturbation, we specialized it to study the density-density response function for the Hubbard Hamiltonian. In Chapter 3 we provided a general description of the code/module and some of its functionalities. We also showed the construction process of the two new functions that were added in the linear sector. The code was used in Chapter 4 to simulate three systems: a Hubbard dimer (two-sites), a four-site and a six-site linear chain, where the density-density response function was computed in the frequency and the time domain for all systems. The time domain density-density response function was used to compute the expectation value of the density operator over time under the specified perturbation in Section 3.8. The results were compared with the full time dynamics characterizing the regime where the two approaches are equivalent. Overall, we saw that the LRT results are applicable only in the presence of a weak perturbation. Lastly, in Appendix A we expanded the preliminary document received by Simon Ydman to create a booklet for the /ExCITeD/ code.

As mentioned in Chapter 3, the /ExCITeD/ code is originally aimed at solving the HH model, while also being capable of solving the pure Hubbard model. The functions included in the linear sector were applied to the Hubbard model, targeting purely electronic systems. A possible expansion of this work could be to generalize the functions in the linear sector to the more general HH Hamiltonian, including electron-phonon interactions.

In physics programs all over the world, simulations are used in classrooms as a tool to help students to understand concepts. The first and most familiar encounter is perhaps in the subject of classical mechanics. Whether it was for projectile motion of objects, elastic collision or just a simple pendulum, the simulations have proven to be a great supplement to the student's understanding. We believe that this code has the potential of being a useful simulation tool used for teaching purposes. The obvious way to proceed on this route would be to construct a graphical user interface (GUI) for this code, allowing teachers and students to use the code without the requirement of having advanced skills in Python programming.

BIBLIOGRAPHY

[1] Gubernatis, J. E., N Kawashima, and P Werner. “Quantum Monte Carlo Methods : Algorithms for Lattice Models.” *Cambridge University Press* (2016).

[2] U. Schollwock. "The density-matrix renormalization group." *arXiv:cond-mat/0409292* [cond-mat.str-el] (2005).

[3] Fetter, Alexander L, and John D. Walecka. "Quantum Theory of Many-Particle Systems." *San Francisco: McGraw-Hill,* (1971).

[4] Zhang, J. M., and R. X. Dong. "Exact diagonalization: the Bose–Hubbard model as an example." *European Journal of Physics* 31.3 (2010): 591.

[5] M. Hopjan, D. Karlsson, S. Ydman, C. Verdozzi, and C.-O. Almbladh. "Merging features from Green’s functions and time dependent density functional theory: A route to the description of correlated ma- terials out of equilibrium?" *Phys. Rev. Lett.*116, 236402 (2016).

[6] E. Boström, A. Mikkelsen, and C. Verdozzi. "Time-resolved spectroscopy at surfaces and adsorbate dynamics: insights from a model-system approach" *Phys. Rev. B*93, 195416 (2016).

[7] D. Karlsson, A. Privitera and C. Verdozzi. "Time Dependent Density Functional Theory meets Dy- namical Mean Field Theory: Real-Time Dynamics for the 3D Hubbard Model" *Phys. Rev. Lett.*106, 116401 (2011).

[8] M. Puig von Friesen, C. Verdozzi, C.-O. Almbladh. "Successes and failures of Kadanoff-Baym dynamics in Hubbard nanoclusters" *Phys. Rev. Lett.*103, 176404 (2009).

[9] C.Verdozzi. "Time-Dependent-Density-Functional-Theory and Strongly Correlated Systems: Insight From Numerical Studies" *Phys. Rev. Lett.*101, 166401 (2008).

[10] P. Samuelsson and C.Verdozzi. "Two-particle Spin Entanglement in Magnetic Anderson Nanoclus- ters" *Phys.Rev. B*75, 132405 (2007).

[11] Tasaki, Hal. "The Hubbard model-an introduction and selected rigorous results." *Journal of Physics: Condensed Matter* 10.20 (1998): 4353.

[12] Hubbard, John. "Electron correlations in narrow energy bands." *Proc. R. Soc. Lond. A* 276.1365 (1963): 238-257.

[13] Ziman, John M. “Electrons and phonons: the theory of transport phenomena in solids.” *Oxford university press,* (1960).

[14] Gruner, George. “Density waves in solids.” *CRC Press,* (2018).

[15] T. Holstein. *Ann. Phys* 8, 325 (1959).

[16] W. von der Linden, E.Berger and P.Valasek. “The Hubbard-Holstein Model." *Journal of Low Temperature Physics.* 99.3-4 (1995): 517-525.

[17] S. Ydman. “Electron-phonon dynamics in Hubbard rings and chains." (2013).

[18] S.Ydman. “Magnetization dynamics in nanorings." (2015).

[19] S. Ydman, M. Hopjan, C. Verdozzi “Nonequilibrium Kondo-vs.-RKKY scenarios in nanoclusters." *(Europhysics Letters) EPL* 123 47001 (2018).

[20] J. R. Johansson, P.D. Nation, and F. Nori, “QuTiP 2: A Python framework for the dynamics of open quantum systems” *Comp. Phys. Comm.* 184, 1234 (2013).

[21] Warner, Seth. Modern algebra.*Courier Corporation*, (1990).

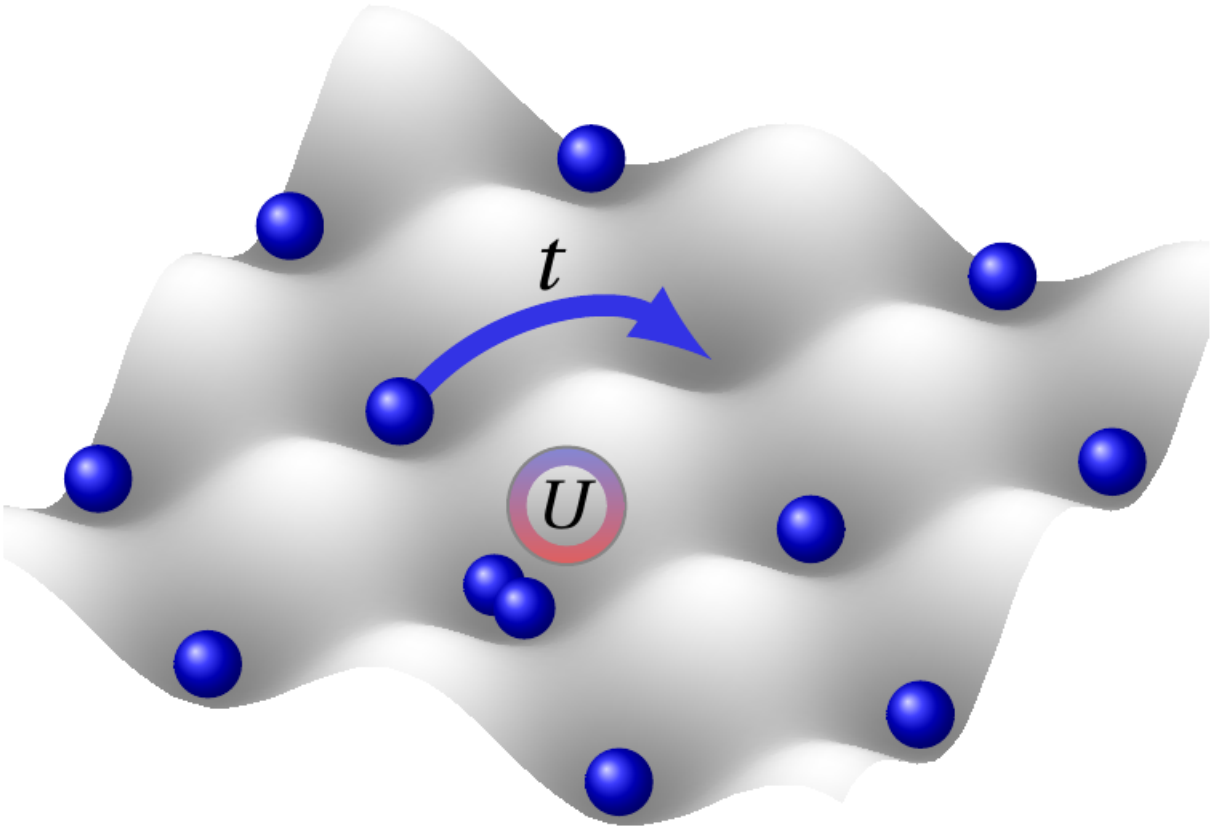
[22] Lehoucq, Richard B., Danny C. Sorensen, and Chao Yang. “ARPACK users’ guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods.” Vol. 6. Siam, (1998).

[23] Tae Jun Park and J. C. Light “Unitary quantum time evolution by iterative Lanczos reduction.” *The Journal of Chemical Physics* 85:10, 5870-5876 (1986).

Appendix A

Booklet

This booklet is a largely expanded version of a very preliminary private document received by Simon Ydman. The aim of the booklet is to guide the user through the installation and running process. It also include worked out examples for the purpose of familiarizing the user with the functionalities of the code. The goal of this booklet is to get the user to use and combine the functionalities of the code and apply them to solve problems.



Booklet of ExCITed

By:
Yasser MAHFOUD

(Adapted and expanded from a preliminary draft from S.Ydman)

CONTENTS

1 About

1.1 About this documentation

This documentation contains a user guide and an API documentation for the ExCITed module.

1.2 About ExCITed

The purpose of this code is to investigate the dynamics of systems with electron-electron and electron-phonon interactions. The target model of ExCITed is the Hubbard-Holstein Hamiltonian, but it is also able to handle a pure Hubbard model or a simple tight-binding Hamiltonian. The code provides the user with the ability to perform different tasks. The following is a brief summary of what ExCITed is capable of providing:

- Set up a system with chosen number of sites, number of electron and phonons.
- Construct different operators such as the density operator, kinetic energy operator, spin operators and more.
- Construct and diagonalize the Hamiltonian of the system.
- Expectation values for various operators.
- Time Evolution using Lanczos-adapted time-evolution method.
- Optimal control.
- Temperature dependent expectation values.

2 Installation

2.1 Requirements

ExCITed depends on several open-source libraries for scientific computing in the Python programming language. The following packages are required:

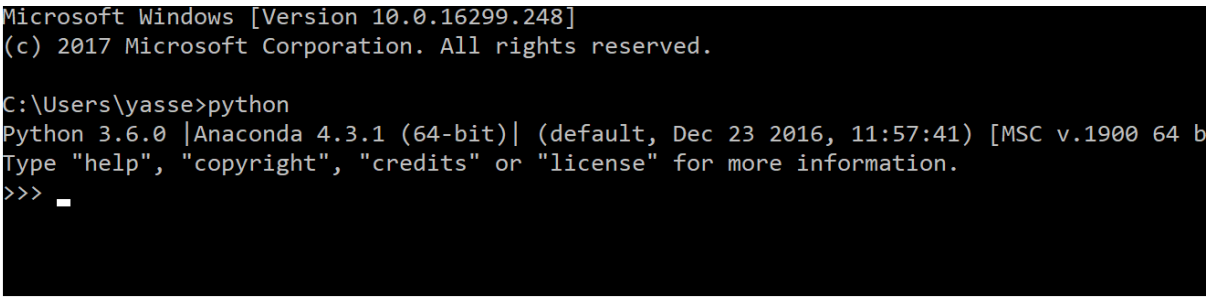
Package	Version
Python	2.7+
NumPy	1.8+
SciPy	0.15+
Matplotlib	1.2.1+
Cython	0.21+
Python Headers	2.7+
QuTiP	4.2+

2.2 Installing on Microsoft Windows

ExCITed is built around the pre-existing open-source package QuTiP. The issue when getting the package ExCITed to work on Microsoft OS is due to the installation of the package QuTiP. Microsoft OS has some issues with its default compiler, and for that reason installing QuTiP is not easy. The recommended way is through installing Anaconda, and using Microsoft Visual Studio as a compiler.

2.2.1 Anaconda

Download the Python 3 Version from Anaconda's official website https://www.anaconda.com/download/. When installing please make sure to check in the box that says "Add Anaconda to my Path environment variable". After finishing the installation, open the Command Prompt and type in `python` to check that your installation was successful.



2.2.2 Microsoft Visual Studio

Microsoft Visual Studio is essential for the installation of the QuTiP package; unfortunately the default compiler on Microsoft OS will not do. Microsoft Visual Studio is free to download from Microsoft's official website. The recommended version is the 2015 version. Make sure to select the option for the C++ compiler.

2.2.3 QuTiP

After installing Anaconda and Microsoft Visual C++ 2015 Build tools, we are now able to install the QuTiP package. We can install the QuTiP package by opening Visual C++ Command Prompt and using the pip command, which is done by typing in `pip install qutip` in the Command Prompt. To check that QuTiP was successfully installed we can open python and import the package by running the following code:

```
In [1]: import qutip as qt
```

2.3 Importing

ExCITed does not currently include a "setup.py" file and so it can not be installed. Importing the ExCITed package can be done by either adding a new path for python to search in, or by moving the package to an existing python path.

2.3.1 Adding a new path directory

We have to add a new path to the list of directories Python goes through when searching for modules. We can do that by using the append method to add a new directory to the path, in the following example the ExCITed file (which contains `__init__.py`) is on the desktop.

```
In [1]: import sys
        sys.path.append("c:/users/yasse/desktop")
```

After that to import ExCITed by running the following code:

```
In [2]: import excited as ex

Pysparse not found -> davidson_gs cant be used!
```

2.3.2 Moving ExCITed to a Python path

To identify the list of directories Python goes through to search for modules we run the following code:

```
In [3]: import sys
        print(sys.path)

['', 'C:\\Users\\yasse\\Anaconda3\\python36.zip',
     'C:\\Users\\yasse\\Anaconda3\\DLLs',
     'C:\\Users\\yasse\\Anaconda3\\lib'...]
```

Python will provide a list of options for us. We can move the ExCITed package to one of those directories. After that, we can continue to import ExCITed in the same way as we did before.

3 GUIDE

The aim of this guide is to teach us how to use the ExCITed package. When solving a problem using matrix representation, the first thing is to choose a basis. ExCITed does this for us and we only have to specify the number of spin up and spin down electrons, the number of electronic orbits and optionally phonon modes and trunkation for each mode. In this guide however, we will only look at electronic systems.

3.1 BASIS

The `Basis` class constructs a complete set of basis states of the Hubbard-Holstein Hamiltonian. It is also possible to use the class for a pure Hubbard or a simple tight-binding Hamiltonian. We will look at a pure Hubbard model, and explain it with a worked out example. The Hubbard model Hamiltonian formally is given by:

$$H = -t \sum_{<i,j>\sigma} \left(c^\dagger_{i\sigma} c_{j\sigma} + c^\dagger_{j\sigma} c_{i\sigma} \right) + U \sum_{i,j} n_{i,j\uparrow} n_{i,j\downarrow} \tag{3.1}$$

where $c^\dagger_{i\sigma}$ ($c_{i\sigma}$) creates (annihilates) a fermion in a single particle orbital localized at site i with a specified spin σ, t is the hopping parameter, U is the Coulomb force parameter and $n_{i\uparrow}$ is the number operator which is defined as $n_{i\uparrow} = c^\dagger_{i\uparrow} c_{i\uparrow}$. The fermionic operators satisfy the anti-commutation relations

$$\begin{aligned} \left\{ c^\dagger_i, c_j \right\} &= \delta_{ij} \\ \left\{ c^\dagger_i, c^\dagger_j \right\} &= 0 \\ \left\{ c_i, c_j \right\} &= 0. \end{aligned} \tag{3.2}$$

The easiest way to describe the essence of this model is by taking an example of the two-site Hubbard model labeled 0,1 as follows:

$$\overbrace{}^{0}\overbrace{}^{1}$$

the `Basis` class allows us to choose how many spin-up electrons and spin-down electrons to add to the sites. In the following example we will stick to 1 spin-up electron and 1 spin-down electron. For the two-site case it is possible to analytically find out the set of basis sates for two electrons in a singlet configuration. The four possible basis states are:

$$\underbrace{\uparrow\downarrow}_{c^\dagger_{0\uparrow} c^\dagger_{0\downarrow}} \qquad \underbrace{\downarrow\ \uparrow}_{c^\dagger_{1\uparrow} c^\dagger_{0\downarrow}} \qquad \underbrace{\uparrow\ \downarrow}_{c^\dagger_{0\uparrow} c^\dagger_{1\downarrow}} \qquad \underbrace{\uparrow\downarrow}_{c^\dagger_{1\uparrow} c^\dagger_{1\downarrow}}$$

in terms of the creation operator. We can get the basis states through ExCITed by running the following code:

```
In [21]: import excited as ex
         nup, ndw = 1, 1
         norb= 2
         basis=ex.Basis(nup, ndw, norb)
         print('#  u d')
         print(basis)
```

```
#  u d
0 [0 0]
1 [1 0]
2 [0 1]
3 [1 1]
```

where the output of the code is equivalent to:

$$|0\rangle = c^\dagger_{0\uparrow} c^\dagger_{0\downarrow} |vac\rangle$$
$$|1\rangle = c^\dagger_{1\uparrow} c^\dagger_{0\downarrow} |vac\rangle$$
$$|2\rangle = c^\dagger_{0\uparrow} c^\dagger_{1\downarrow} |vac\rangle$$
$$|3\rangle = c^\dagger_{1\uparrow} c^\dagger_{1\downarrow} |vac\rangle$$

the basis class contains a number of attributes to create various operators. In the following, we will discuss the first attribute which is the kinetic energy operator.

3.1.1 KINETIC ENERGY OPERATOR

This attribute provides us with the kinetic energy part of the Hamiltonian. Which represents the probability of electrons tunneling from one site to another. We will continue with the two-site example to show how the attribute works. In the two-site case, the hoping can occur from site 0 to site 1 and the oposite. The attribute requires as an input a list which contains the two sites chosen by the user for the allowed hoppings. In the two-site case we choose that the electrons hops from site 0 to site 1. The first step is to define the list.

```
In [3]: hoplist=[(0,1)]
```

now that we have defined the list of possible hoppings we can call on the attribute to present the kinetic operator. The list that we chose is interpreted as the subscripts of $c^\dagger c$. ExCITed then solves the the matrix elements $\langle\lambda|c^\dagger_i c_j|\nu\rangle$ where λ,ν runs over the basis states defined previously. Analytically this is done by solving the following:

$$\langle\lambda|c^\dagger_i c_j|\nu\rangle = [\lambda = 0, \nu = 1,\ i = 1\uparrow, j = 0\uparrow] \tag{3.3}$$
$$= \langle 0|c_{0\downarrow} c_{0\uparrow} c^\dagger_{1\uparrow} c_{0\uparrow} c^\dagger_{1\uparrow} c^\dagger_{0\downarrow}|0\rangle = 1 \tag{3.4}$$

ExCITed solves the matrix elements and provides us with the kinetic energy operator by running the following code:

```
In [4]: h_hop=basis.hop_op(hoplist)
        print(h_hop)
```

```
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper
, isherm = True
Qobj data =
[[ 0. -1. -1.  0.]
 [-1.  0.  0. -1.]
 [-1.  0.  0. -1.]
 [ 0. -1. -1.  0.]]
```

the attribute also includes optional inputs for the user to choose arguments such as spin, symmetry, and the value of the hopping parameters. Check the API Documentation section for details.

3.1.2 Density Operator

The density operator attribute allows the user to choose a specific site for which ExCITed acts on that operator with the defined basis to produce the density matrix. For example, if we choose the site 1 then ExCITed interpret that input as the subscripts of $c_i^\dagger c_i$ and solves the matrix elements $\langle\lambda|c_1^\dagger c_1|\nu\rangle$.

```
In [5]: basis.dens_op(1)
```

```
Out[5]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 2.0 \end{pmatrix}$$

3.1.3 Double density Operator

The double density operator attribute allows the user to choose a specified site for which ExCITed interprets the input as the subscripts of $n_{i\uparrow}n_{i\downarrow}$. For example, if we choose site 0 then ExCITed solves the matrix elements $\langle\lambda|n_0^\dagger n_0|\nu\rangle$.

```
In [21]: basis.double_dens_op(0)
```

```
Out[21]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

This attribute proves helpful when finding the interaction part of the Hubburd Hamiltonian in (3.1).

```
In [22]: U_value=2
         ni_ni=sum([basis.double_dens_op(i) for i in range(norb)])
         H_u=U_value*ni_ni
         print(H_u)
```

```
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, ish
Qobj data =
[[2. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 2.]]
```

3.1.4 Spin Operators

Spin-spin correlations provide information about the magnetic state of a system. ExCITed allows us to choose two sites which then returns a list of operators used to find spin-spin correlations and concurrence.

```
In [6]: basis.spin_ops(0,1)
```

```
Out[6]: [Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4),
         type = oper, isherm = True
         Qobj data =
         [[0. 0. 0. 0.]
          [0. 0. 0. 0.]
          [0. 0. 0. 0.]
          [0. 0. 0. 0.]],
         Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4),
         type = oper, isherm = True
         Qobj data =
         [[0. 0. 0. 0.]
          [0. 0. 0. 0.]
          [0. 0. 1. 0.]
          [0. 0. 0. 0.]],
         Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4),
         type = oper, isherm = True
         Qobj data =
         [[0. 0. 0. 0.]
          [0. 1. 0. 0.]
          [0. 0. 0. 0.]
          [0. 0. 0. 0.]],
         Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4),
         type = oper, isherm = True
         Qobj data =
         [[0. 0. 0. 0.]
          [0. 0. 0. 0.]
          [0. 0. 0. 0.]
          [0. 0. 0. 0.]],
         Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4),
         type = oper, isherm = True
         Qobj data =
         [[ 0.   0.   0.   0. ]
          [ 0.   0.  -0.5  0. ]
          [ 0.  -0.5  0.   0. ]
          [ 0.   0.   0.   0. ]]]
```

where the first four operators correspond to:

$$n_{i\uparrow} \cdot n_{j\uparrow}$$
$$n_{i\uparrow} \cdot n_{j\downarrow}$$
$$n_{i\downarrow} \cdot n_{j\uparrow}$$
$$n_{i\downarrow} \cdot n_{j\downarrow}$$

and last operator is the spin flip operator of sites i and j. For example, we can use the operators above to find the spin-spin correlation operator:

$$S_z^i S_z^j = \frac{1}{4}(n_{i\uparrow} \cdot n_{j\uparrow} + n_{i\uparrow} \cdot n_{j\downarrow} + n_{i\downarrow} \cdot n_{j\uparrow} + n_{i\downarrow} \cdot n_{j\downarrow}) \tag{3.5}$$

```
In [24]: SizSjz=0.25*sum([basis.spin_ops(0,1)[i] for i in range(4)])
         print(SizSjz)
```

Out[24]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.250 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.250 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

3.1.5 Current Operator

The current operator that represents the current from site i to j is given by:

$$J_{ij} = t(-ic_i^\dagger c_j + h.c.) \tag{3.6}$$

ExCITed allows us to choose the two sites i and j and returns the current operator by acting on it with the basis defined.

```
In [7]: basis.curr_op(0,1)
```

Out[7]:
Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = False

$$\begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ -2.0j & 0.0 & 0.0 & 0.0 \\ -2.0j & 0.0 & 0.0 & 0.0 \\ 0.0 & -2.0j & -2.0j & 0.0 \end{pmatrix}$$

3.2 Expectation values

If we are interested in some observable we must turn to the functions in the QuTiP package. In QuTiP, that is done by the function `expect`, which takes two arguments, an operator and a state vector. Up untill this point, we know very well how to define an operator. The state vector on the other hand, can be found using the `groundstate` method in the QuTiP package. The `groundstate` method returns the grounstate energy and the groundstate vector of a Quantum object[1] . For example, if we want to find the groundstate of the kinetic energy operator:

[1] A Quantum object is a class in QutTiP that has many attributes such as `expect` and `groundstate`. For more information please check the package QuTiP.

```
In [10]: e0, psi0=h_hop.groundstate()
         print('Groundstate energy:', e0)
         print('\nGroundstate vector',psi0)

Groundstate energy: -2.0

Groundstate vector Quantum object: dims = [[2, 2], [1, 1]],
shape = (4, 1), type = ket
Qobj data =
[[-0.5]
 [-0.5]
 [-0.5]
 [-0.5]]
```

Now that we have the state vector we can go on to calculate the expectation value using the function `expect`. The simplest case is perhaps the density operator.

```
In [12]: dens_op_0=basis.dens_op(0)
         print('Density at site 0:\t\t',qt.expect(dens_op_0,psi0))

Density at site 0:                  1.0
```

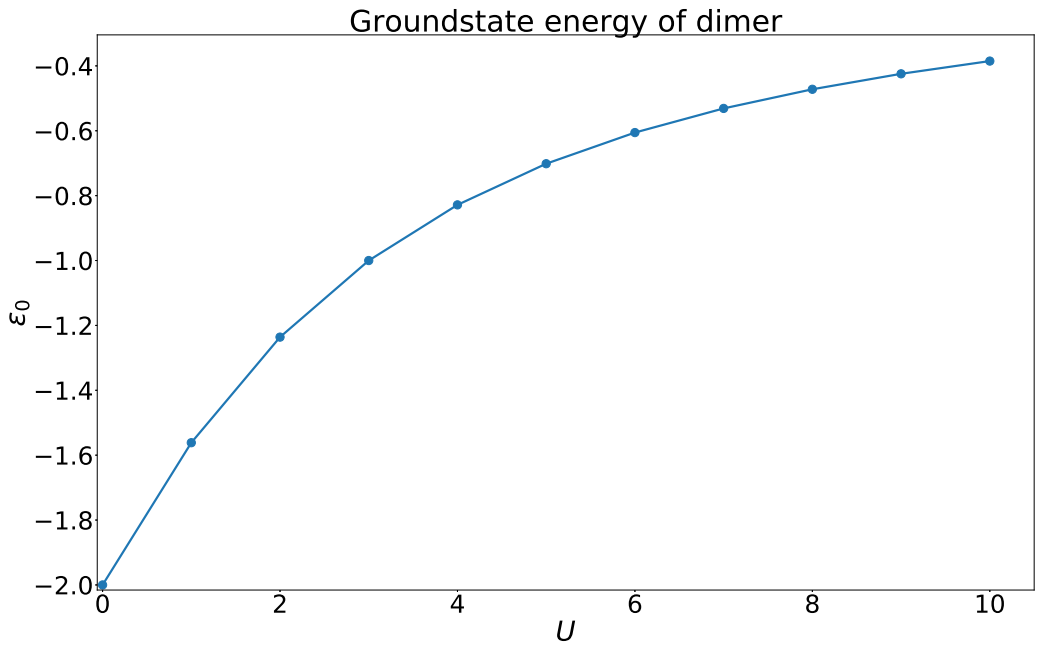
The function `expect` can also take as an input a list of operators.

```
In [14]: dens_ops=[basis.dens_op(i) for i in range(norb)]
         print('Density at [site0 , site 1]:\t',qt.expect(dens_ops,psi0))

Density at [site0 , site 1]:          [1. 1.]
```

We will also demonstrate how useful the function `groundstate` is by a simple example in which we compute the groundstate energy of different values of the on-site Coulomb interaction in the dimer.

```
In [16]: import matplotlib.pyplot as plt   #Importing the plotting packag
         %matplotlib inline
         import numpy as np # Importing numpy for the linspace command

         #Create onsite interaction operator.
         h_u=sum([basis.double_dens_op(i) for i in range(norb)])
         Us=np.linspace(0,10,11)
         e0s=np.empty(11)

         #For all the different U-values
         for i,U in enumerate(Us):
             e0,psi0=(h_hop+U*h_u).groundstate()
             e0s[i]=e0
```

```
plt.figure(figsize=(9,5))
plt.plot(Us,e0s,'-o',linewidth=2,markersize=8)
plt.title('Groundstate energy of dimer',size=16)
plt.xlabel('$U/t$',size=20)
plt.ylabel('$\epsilon_0/t$',size=20)
plt.show()
```



3.3 Constructing the Hamiltonian

The Hamiltonian in ExCITed is a class that is defined from the basis. ExCITed constructs the Hamiltonian by running the following code:

```
In [17]: H=ex.Hamiltonian(basis)
```

After we initialize the Hamiltonian we can start to add parts to it such as the kinetic energy part by using the following attribute:

```
In [18]: H.add_part(h_hop)
```

The Hamiltonian class has the attribute `evalutate` for which ExCITed returns the Hamiltonian evaluated at a specific time as a QuTiP quantum object. For example if we want to evaluate the Hamiltonian at time zero:

```
In [19]: H.evaluate(0)
```

```
Out[19]:
```

Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4), type = oper, isherm = True

$$\begin{pmatrix} 0.0 & -1.0 & -1.0 & 0.0 \\ -1.0 & 0.0 & 0.0 & -1.0 \\ -1.0 & 0.0 & 0.0 & -1.0 \\ 0.0 & -1.0 & -1.0 & 0.0 \end{pmatrix}$$

Now we can us to use a QuTiP method such as the groundstate on the Hamiltonian:

```
In [27]: e0, psi0 = H.evaluate(0).groundstate()
         print('Groundstate energy=',e0)
         print('Groundstate:\n',psi0)

Groundstate energy= -2.0
Groundstate:
 Quantum object: dims = [[2, 2], [1, 1]], shape = (4, 1), type = ket
Qobj data =
[[-0.5]
 [-0.5]
 [-0.5]
 [-0.5]]
```

The Hamiltonian class has one more attribute which is `assemble`. The assemble attribute combines all the added parts to the Hamiltonian and prints out the full Hamiltonian as a quantum object. For example, we can add the kinetic part and the interaction part and then assemble the Hamiltonian by running the code:

```
In [29]: H.add_part(h_hop)
         H.add_part(h_u)
         H.assemble()

Out[29]: [Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4),
          type = oper, isherm = True
          Qobj data =
          [[ 1. -1. -1.  0.]
           [-1.  0.  0. -1.]
           [-1.  0.  0. -1.]
           [ 0. -1. -1.  1.]]]
```

3.4 Time Evolution and Quantum System Dynamics

In this section we will demonstrate how to compute the time evolution of a system with external fields. In the following example we perturb the energy level of the first site in time, and compute the time evolution of :

$$\hat{H}(\tau) = \hat{H}_{hop} + \gamma(\tau)\hat{n}_0 \tag{3.7}$$

We have already defined $\hat{H}_{hop}$ and $\hat{n}_0$. What is left is to define the perturbation γ. The perturbation chosen for this example is :

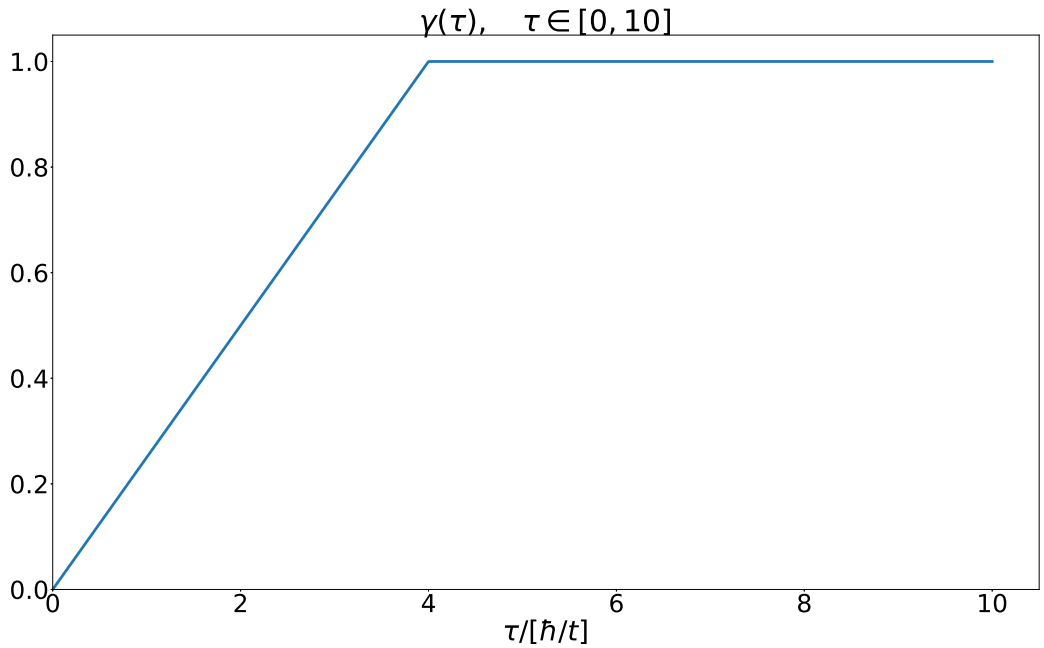
$$\gamma(\tau) = \begin{cases} 0 & \text{if,} \quad \tau \leq 0 \\ \frac{t}{4} & \text{if,} \quad 0 < \tau < 4 \\ 1 & \text{if,} \quad \tau \geq 0 \end{cases} \tag{3.8}$$

Time dependent functions like γ is declared using def keyword in python.

```
In [30]: def linear_quench(t,args={}):
             if t<= 0:
                 return 0.
             if t<4:
                 return t/4.
             else:
                 return 1.
```

we can see how our perturbation looks like by plotting it:

```
In [31]:# Vectorize function for plotting purposes
         vquench=np.vectorize(linear_quench)

         tlist=np.linspace(0,10,1000)
         plt.plot(tlist,vquench(tlist),linewidth=2)
         plt.ylim(0,1.1)
         # Title and labels.
         plt.title('$\gamma(\\tau), \quad \\tau \in [0,10]$',size=20)
         plt.xlabel('$\\tau / [\hbar/t]$',size=18)
         plt.ylabel('',size=18)
         plt.show()
```

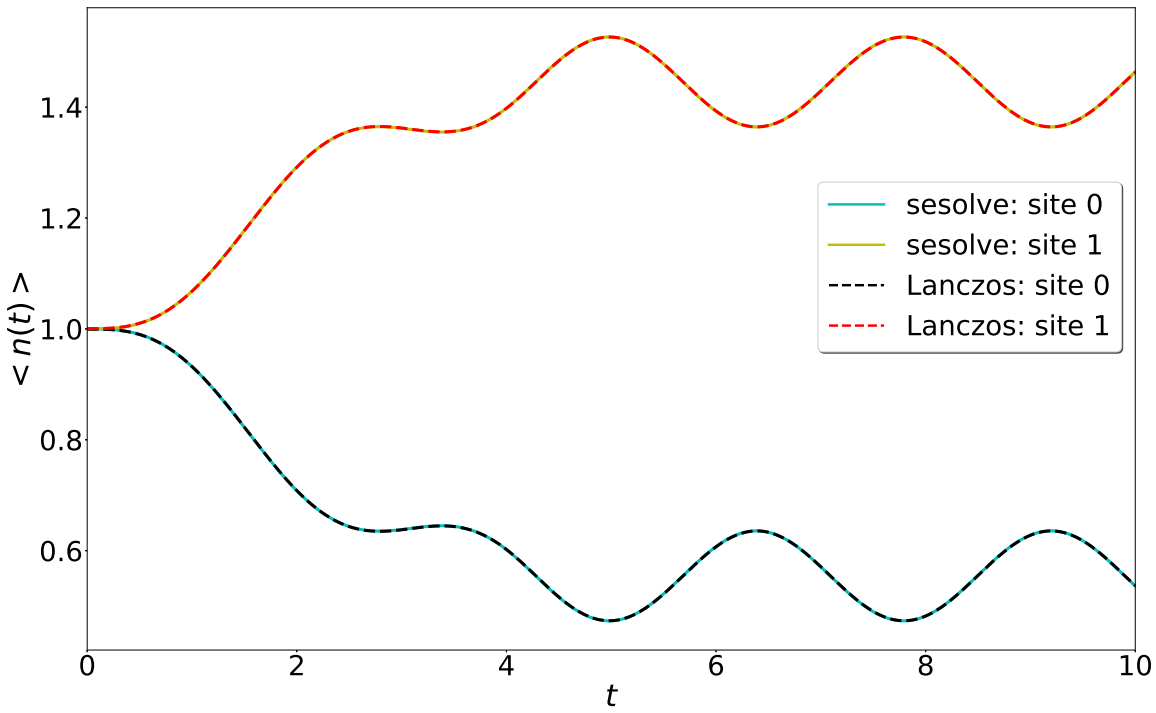


note that linear_quench take two argument (`t` and `args`). This is one of the not so pretty solution in QuTiP (it would have been possible to do this prettier using unpacking dictionaries), but as it is now the extra argument (`args`) is needed to preform the time-evolution.
Now that we have all the components of the full Hamiltonian we can add them:

```
In [32]: H.add_part(h_hop)   # add h_hop to Hamiltonian
         H.add_part(dens_op_0,linear_quench)# add dens as time dependent
```

ExCITed has a time evolution solver function which is `lanczos_te`. The `lancozs_te` function evolves a given state vector using lanczos adapted time-evolution, and returns the expectation values of specified operators.

```
In [55]: e0, psi0 = H.evaluate(0).groundstate()
         #QuTiP Evolver
         res_sesolve=qt.sesolve(H.assemble(),psi0,tlist,dens_ops)
         # Lanczos Evolver
         res_lanczos=ex.evolve.lanczos_te(H.assemble(),psi0,tlist,
                                                    dens_ops)
         #Plot
         plt.plot(tlist,res_sesolve.expect[0],'c-',
                                label='sesolve: site 0',linewidth=3)
         plt.plot(tlist,res_sesolve.expect[1],'y-',
                                label='sesolve: site 1',linewidth=3)
         plt.plot(tlist,res_lanczos.expect[0],'k--',
                                label='sesolve: site 0',linewidth=3)
         plt.plot(tlist,res_lanczos.expect[1],'r--',
                                label='sesolve: site 1',linewidth=3)
         plt.legend(bbox_to_anchor=(1.0,0.75),ncol=1,fancybox=True,
                                shadow=True)
         plt.show()
```



as you can see we have also used QuTiP's evolver and plotted the results. The time evolution solver in QuTiP is becoming very slow for large systems and that is why ExCITed has its own time evolution solver

```
In [56]: print('QuTiP se_solve speed test:')
         %timeit qt.sesolve(H.assemble(),psi0,tlist,dens_ops)
         print('excited lanczos speend test')
         %timeit ex.evolve.lanczos_te(H.assemble(),psi0,tlist,dens_ops)

qutip se_solve speed test:
2.75 s  per loop (mean ± std. dev. of 7 runs, 1 loop each)
excited lanczos speend test:
980 ms  per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

3.5 Density-Density Response function in the frequency domain

In this section we will demonstrate how to compute the density-density response function in the frequency domain for the two-site system that we have been working with. The response function in the frequency domain is given by:

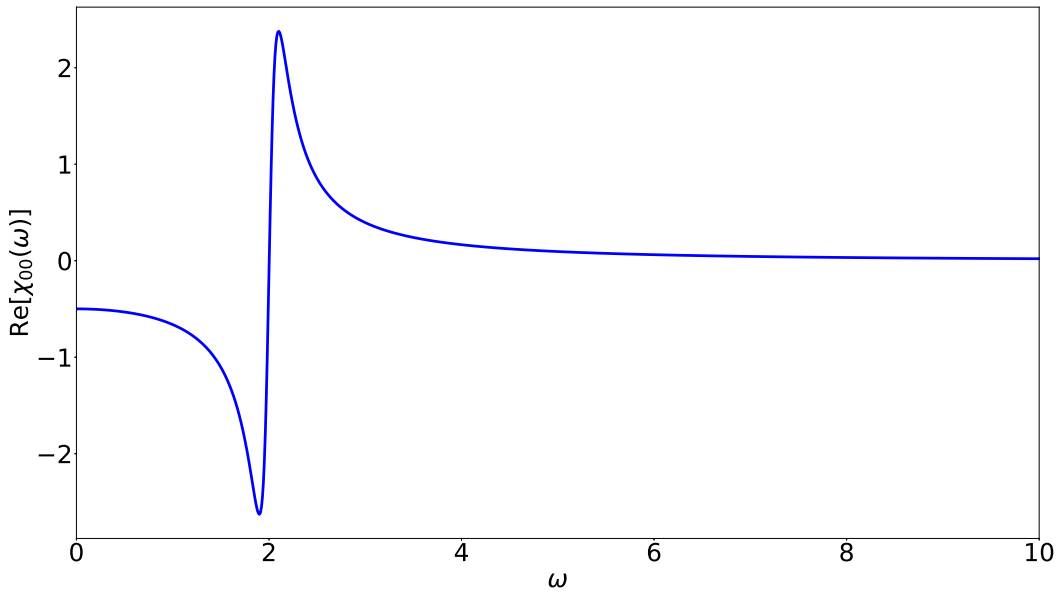
$$\chi_{ij}(\omega) = \sum_n \left[\frac{\langle\psi_0|\hat{n}_i|\psi_n\rangle \langle\psi_n|\hat{n}_j|\psi_0\rangle}{\omega - (E_n - E_0) + i\eta} - \frac{\langle\psi_0|\hat{n}_j|\psi_n\rangle \langle\psi_n|\hat{n}_i|\psi_0\rangle}{\omega + (E_n - E_0) + i\eta} \right] \tag{3.9}$$

where i and j are the sites chosen. The function `dens_freq` computes the imaginary and real part of this quantity for us. As an input we must include the Hamiltonian that describes the system, the basis of the system, the range of ω, the desired broadening value η and the two desired sites. To remind us, we have:

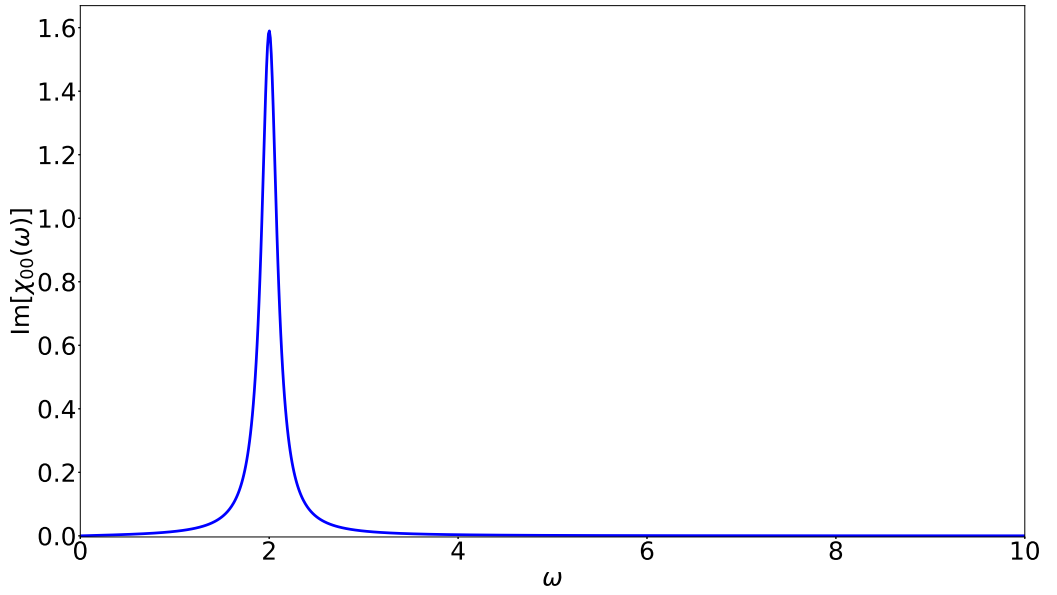
```
In [59]: H=ex.Hamiltonian(basis)
         H.add_part(h_hop)
         H.add_part(h_u)
         H.assemble()

Out[59]: [Quantum object: dims = [[2, 2], [2, 2]], shape = (4, 4),
          type = oper, isherm = True
          Qobj data =
          [[ 1. -1. -1.  0.]
           [-1.  0.  0. -1.]
           [-1.  0.  0. -1.]
           [ 0. -1. -1.  1.]]]
```

In the following example we will compute the response function $\chi_{00}(\omega)$ with the broadening $\eta = 0.1$ and the range of ω goes from $0 \longrightarrow 10$.

```
In [74]: Response_omega=
         ex.response.dens_freq(H.assemble(),basis,10,0,0,0.1)
         plt.plot(np.linspace(0,10,1000),np.real(Response_omega))
         plt.xlabel('$\omega$')
         plt.ylabel('$Re[\chi_{00}(\omega)]$')
```




```
In [75]: plt.plot(np.linspace(0,10,1000),
                  (-1/np.pi)*np.imag(Response_omega))
         plt.xlabel('$\omega$')
         plt.ylabel('$Im[\chi_{00}(\omega)]$')
```



4 API Documentation

4.1 Classes

4.1.1 Basis

`class Basis(nup,ndw,norb,nmode=0,nph=[])`

A class that contains a constructor for the complete basis set as well as a number of attributes that return different components of the Hubbard-Holstein Hamiltonian.

Parameters:

`nup`: Number of spin-up electrons.

`ndw`: Number of spin-down electrons.

`norb`: Number of orbitals (or sites).

`nmodes`: Number of phonon modes.

`nph`: List of phonon subspace dimension for each of the phonon modes.

Attributes:

`.dens_op( site, spin='', mat=False):`

Returns density operator for the site chosen as a QuTiP qobject (Scipy sparse matrix if specified).

Args:

`site`: Site for which to return the density operator

`spin`: Optional argument for spin densities ('up' for spin up density and 'dw' for spin down density). Default is ''.

`mat`: Optional argument. Set to True to return a Scipy sparse matrix instead of QuTiP qobject.

`.hop_op( hoplist, V=-1, spin='', sym=True):`

Returns matrix representation for fermionic hoppings as a QuTiP quantum object.

Args:

`hoplist`: A list containing tuples of the allowed hoppings. For example, [(0, 1), (1, 3)] for fermionic hoppings from site/orbital 0 to 1 and 1 to 3.

`V`: Optional argument for the hopping amplitude.

Default value is -1 for regular fermionic site hoppings.

`spin`: Optional argument. Specify the spin.

`sym`: Optional argument that indicates if the hopping should be symmetric (Default True). Change when different values of the hopping is needed, e.g. 0 -> 1 and 1 -> 0.

`.spin_ops( s1, s2):`

Returns a list of operators used for spin-spin correlations and concurrence.

Args:

`s1`: site i

`s2`: site j

`.curr_op( i, j, V=-1, spin=", func=None):`

Returns the current operator for the bond from site i to site j.

Args:

`i`: Number of the first site.

`j`: Number of the first site.

`V`: Optional argument, hopping amplitude. (Default -1).

`spin`: Optional argument for spin current. 'up' for spin up current and 'dw' for spin down current.

`func`: Optional argument if the spin operator changes in time.

`.double_dens_op(site):`

Returns double density operator for site.

Args:

`site`: Site for which to return the density operator

4.1.2 HAMILTONIAN

`class Hamiltonian(Basis)`

Class object for handling Hamiltonians consisting of time dependent and time independent parts. The class contains attributes for adding parts to the Hamiltonian, evaluate it at a specific time t, and to assemble it for time evolution solvers.

Parameters:

`basis`: excited basis object that contains the basis of the Hamiltonian as well as other useful information.

Attributes:

`.add_part( H_part, w=1):`

Adds a part to the Hamiltonian.

Args:

`H_part`: New part of the Hamiltonian. H_part should be a QuTiP Qobj with a dimension corresponding to the full basis, the bosonic or the fermionic part.

`w`: Optional argument that specifies the weight of H_part. If w is time independent, H_part will be added to H_0 otherwise it will be added to TD_parts. w can be a constant, function or Cython string. (Default:1)

`.evaluate( t,args=):`

Returns the Hamiltonian evaluated at a specific time as a QuTiP qobject.

Args:

`t`: Time.

`args`:Dictionary containing arguments for the time dependent functions in TD_parts.

`.assemble():`

Return the Hamiltonian for TD solvers in QuTiP.

4.2 FUNCTIONS

4.2.1 LANCZOS

`lanczos_te(h, psi0, tlist, e_ops, **kwargs)`

Time-evolve system the specified system and returns the expectation values of specified operators. This function, time evolves psi0 using Lanczos adapted time-evolution. The structure and functionality mimics QuTiP solvers.

Args:

`h`: Hamiltonian of the system. This can be passed as a QuTiP object or as a list of lists [[H1, f1], [H2, f2], ...], with f1, and f2 being the corresponding weight functions to the Hamiltonian parts H1 and H2.

`psi0`: Initial state of the system described by a QuTiP qobject.

`tlist`: List or array of times for which to evolve the system. Typically tlist is the result of a numpy linspace function call.

`e_ops`: List of operators for which to calculate expectation values for. If the operators change with time, it is also possible to pass a list of lists, e.g. [[o1,f1],[o2,f2], ...]. Here f1 and f2 are weight functions to operators o1 and o2

4.2.2 DENSITY-DENSITY RESPONSE FUNCTION IN THE FREQUENCY DOMAIN

`dens_freq(H , basis, w, i, j, nay, plot= ")`

Returns the density-density response function as a function of ω using the Hamiltonian.

Args:

`H`: Hamiltonian object (Qobj or a list containing a Qobj object).

`basis`: An excited object which represents the complete set of Hubbard-Holstein basis vectors.

`w`: range of omega (The frequency).

`i`: Site i.

`j`: Site j.

`nay`: Broadening constant.

`plot=' '`: Optional argument for plotting:
-'real': plots the real part of the function
-'imag': plots the imaginary part of the function