Computer aided Analytical Calculations and Numerical Experimentation on the spin-dependent two-dimensional square Hubbard Model in an Electric Field

Practical Training Report

submitted by

Jonas Kell

on 15th of Mai 2024

University of Augsburg
Faculty of Mathematics, Natural Sciences, and Materials Engineering
Institute of Physics
Chair for theoretical Physics III

1st Corrector: Prof. Dr. Markus Heyl

Table of Contents

List of Abbreviations

Abbreviation	Meaning
TP III	Chair for theoretical Physics III
MC	Monte Carlo

1 Introduction

In preparation for the master thesis, as well as to get accustomed to the process of working in a scientific context, the module *Practical Training* ("Fachpraktikum") is one of the key modules that need to be completed at the end of the masters degree program in physics at the University of Augsburg. In this case, work has been performed at the Chair for theoretical Physics III under the guidance of Prof. Dr. Markus Heyl. To fulfill the module requirements, this report is supposed to give an overview of my work during the past six months, understanding and preparing the basics of the target subject. Following this report, the research on the subject will be continued, in order to be able to generate scientific content that possibly is of interest to the established community in the scope of my master thesis.
Several, different projects were attempted in the scope of this Practical Training. Central element were the analytical calculations performed on the Hubbard Model, that will be presented in subsection 2.1.1. Subsequent sections will deal with the theory of time evolution and measurement of the operators in the interaction picture necessary for the final implementation in section 3.2.

Several of the required analytical calculations seemed to be to repetitive, error-prone or time-consuming to be done by hand. During my work and studies at TP III, my background in Computer Science made me question multiple times, whether the established sentiment of doing thins "by hand" was really an acceptable use of the staffs time. Therefore in the past six months, I tried employing computers to aid with completing analytical calculations of such kind in a more timely and less frustrating way. In this spirit was one calculation performed with help of the package *SymPy*. Also a bigger chunk of my time was spent, writing a custom application to aid in performing certain "written-on-paper-like" calculations with computational assistance. This application is called *Math-Manipulator*.
The methods are touched superficial in subsection 3.1.1 in this limited report. However they became central part of my *Project Work Presentation*, that is available alongside this very LaTeX document [1] and was presented at the University of Augsburg in 2024.

2 Theory

2.1 Physical Theory

To achieve the goal of calculating observables for a specified Hamiltonian and system geometry, mathematical simplifications are required. While mathematically doing so is quite straight forward, the computational complexity for solving the time-evolution for arbitrary quantum-mechanical problems is too large, in relation to the system size.
Here, the goal is to bring down the necessary number of sampled states from $\mathcal{O}(2^n)$ to something around the $\mathcal{O}(n)$-range (depending on the required precision) with the use of Monte-Carlo sampling.
The time-evolution will be made by employing an expansion in the Interaction Picture to first degree, which results in a computational complexity of $\mathcal{O}(n^2)$ per evaluated state in the worst case. Depending on the geometry (number of nearest neighbors c), this can be reduced to $\mathcal{O}(c \cdot n)$ in general.
By placing further restrictions on the Hamiltonian, the act of choosing the next state in the Monte-Carlo-Markov chain can be brought down from $\mathcal{O}(c \cdot n)$ to $\mathcal{O}(c)$ and similarly also the cost of evaluating the most expensive observable drops to $\mathcal{O}(c)$.

2.1.1 Hubbard Model and discussed Hamiltonian

The system in question is described by a *Hubbard Model Hamiltonian*, that includes an electrical field. It is shown in Equation 2.1.

$$\mathcal{H} = \mathcal{H}_0 + \hat{\mathrm{V}} \tag{2.1}$$

$$\mathcal{H}_0 = U \cdot \sum_l \hat{\mathrm{n}}_{l,\uparrow}\hat{\mathrm{n}}_{l,\downarrow} + \sum_{l,\sigma} \underbrace{\left(\vec{E} \cdot \vec{r}_l\right)}_{\varepsilon_l} \hat{\mathrm{n}}_{l,\sigma} \tag{2.2}$$

$$\hat{\mathrm{V}} = -J \cdot \sum_{\langle l,m\rangle,\sigma} \left(\hat{\mathrm{c}}^\dagger_{l,\sigma}\hat{\mathrm{c}}_{m,\sigma} + \hat{\mathrm{c}}^\dagger_{m,\sigma}\hat{\mathrm{c}}_{l,\sigma}\right) \tag{2.3}$$

$\vec{E}$ describes the vector of said electrical field and $\vec{r}_l$ the position of the site with index l. The number operators $\hat{\mathrm{n}}_{l,\sigma} = \hat{\mathrm{c}}^\dagger_{l,\sigma}\hat{\mathrm{c}}_{l,\sigma}$ measure the occupation on site l with the respective spin σ.
U, J and ε are constants that describe the interaction strength. They all have the unit of energy. The scalar Product that is defined as ε can be evaluated, based on the system geometry. In this case, the system described by a regular square pattern that can be seen in Figure 2.1. For such a system, the energy difference that is acquired from hopping from site l to m ($\Delta E_{l\to m}$) (provided, l and m are nearest neighbors) is described by Equation 2.4. By assuming a default

value for ε_0 one gets the relation from Equation 2.5 for ε_l (using the telescoping sum over nearest-neighbors-hopping along the path from $\vec{r}_0$ to $\vec{r}_l$).

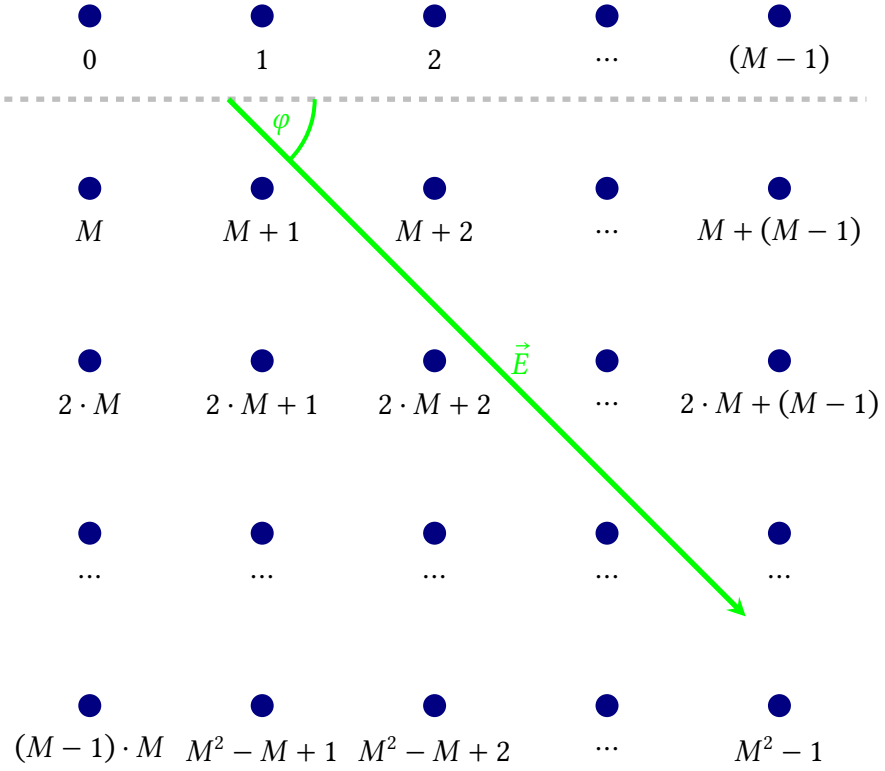

Figure 2.1: Graphical representation of how the examined square system is laid out. The sites are labeled with the index l they later can be identified by. Each of the sites has 4 nearest neighbors, except the ones on the borders, that have 3 or only 2 nearest neighbors (as the system is not periodic). A system of side-length M with size N is depicted, which means that it has $M^2 = N$ sites. In green, the Electrical field vector $\vec{E}$ is depicted, it is parametrized by the field strength E and the angle φ.

$$\begin{aligned}\Delta E_{l\to m} &= \varepsilon_m - \varepsilon_l = \vec{E}\cdot\left(\vec{r}_m - \vec{r}_l\right)\\ &= E\cdot\left[\cos(\varphi)\cdot(m\%M - l\%M) + \sin(\varphi)\cdot\left(\left\lfloor\frac{m}{M}\right\rfloor - \left\lfloor\frac{l}{M}\right\rfloor\right)\right]\end{aligned} \tag{2.4}$$

$$\begin{aligned}\varepsilon_0 = 0 \Rightarrow \varepsilon_l &= \varepsilon_l - \varepsilon_0 = \Delta E_{0\to x} + \cdots + \Delta E_{y\to l}\\ &= E\cdot\left[\cos(\varphi)\cdot l\%M + \sin(\varphi)\cdot\left\lfloor\frac{l}{M}\right\rfloor\right]\end{aligned} \tag{2.5}$$

The Hubbard Model [2] is a spin-depending quantum-mechanical model. Because of this, the Hamiltonian includes terms for both the spin directions ↑ and ↓ (most of the time denoted by a summation over σ).
In principle, the operators $\hat{c}^{(\dagger)}_{l,\sigma}$ can be any type of quantum-mechanical ladder operator, like e.g. *bosonic* or *fermionic* ones. In this application, the desicion was made to use *hard-core bosons*

[3]. Typically for bosons, the Hubbard Model is tranformed into the *Bose-Hubbard Model* [4]. However hard-core bosons have the added property, that the can only have the occupation numbers of 0 and 1 per-spin-and-site in the investigated energy range. This means they behave basically like fermions, only that their wave-function symmetrizes instead of anti-symmetrizes [5]. Because of the restriction one term of the Bose-Hubbard Model drops out and the studied model in equations 2.1, 2.2 and 2.3 is produced. Also the typical Bose-Hubbard Model is not spin-dependent, while this property is kept here. Hard-core bosons on their own have relevant properties that makes them worth studying. Experimental research on hard-core bosons can give insight into e.g. *ultra-cold gasses* and *Bose-Einstein condensation* [3]. For this reason, the calculations are performed on hard-core bosons in this application and the operators obey the *commutation-relations* described in Equation 2.6, with δ the *Kronecker-Delta* [6]. But in principle, the calculations may be performed on fermions analogously.

$$\begin{aligned}\left[\hat{c}_{l,\sigma}, \hat{c}_{l',\sigma'}\right] &= \left[\hat{c}^\dagger_{l,\sigma}, \hat{c}^\dagger_{l',\sigma'}\right] = 0 \\ \left[\hat{c}_{l,\sigma}, \hat{c}^\dagger_{l',\sigma'}\right] &= \delta(l,l') \cdot \delta(\sigma,\sigma')\end{aligned} \tag{2.6}$$

For ease of notation (most importantly in the code), the two spin degrees of freedom may be described by the alternate naming scheme provided in Equation 2.7.

$$\hat{c}^{(\dagger)}_{l,\uparrow} \leftrightarrow \hat{c}^{(\dagger)}_{l} \qquad\qquad \hat{c}^{(\dagger)}_{l,\downarrow} \leftrightarrow \hat{d}^{(\dagger)}_{l} \tag{2.7}$$

For simplification purposes, the two spin directions are assumed to have the same coupling constants (U, J and ε_l). This means, all later results must be symmetrical in terms of the spin directions $\uparrow$ and $\downarrow$.
Translated into the alternative notation, Equation 2.2 reads like Equation 2.8 and Equation 2.3 reads like Equation 2.9.

$$\mathcal{H}_0 = U \cdot \sum_l \hat{c}^\dagger_l \hat{c}_l \hat{d}^\dagger_l \hat{d}_l + \sum_l \varepsilon_l \hat{c}^\dagger_l \hat{c}_l + \sum_l \varepsilon_l \hat{d}^\dagger_l \hat{d}_l \tag{2.8}$$

$$\hat{V} = -J \cdot \sum_{\langle l,m \rangle} \left(\hat{c}^\dagger_l \hat{c}_m + \hat{c}^\dagger_m \hat{c}_l + \hat{d}^\dagger_l \hat{d}_m + \hat{d}^\dagger_m \hat{d}_l \right) \tag{2.9}$$

Several of the previous equations feature the *nearest neighbor* summing notation with pointy brackets $\langle\rangle$. This is supposed to express, that in the case of $\sum_{\langle l,m \rangle}$ for each possible l, m will take on the indices of all neighbor sites of l. However, if a pair $l = x, m = y$ is part of the sum, then the inverse $l = y, m = x$ will not be. To be more in-line with the code implementation, Equation 2.10 gives an alternative notation using []. In such sums, each pair $l = x, m = y$ and $l = y, m = x$ will be included in the sum for each nearest-neighbor-relation x, y.

$$\begin{aligned}
&\sum_{\langle l,m\rangle} \left(F(l,m)\,\hat{c}_l^\dagger \hat{c}_m + F(m,l)\,\hat{c}_m^\dagger \hat{c}_l\right) \overset{\substack{\text{rename-swap}\\ l \text{ and } m}}{=} \\
&\sum_{\langle l,m\rangle} F(l,m)\,\hat{c}_l^\dagger \hat{c}_m + \sum_{\langle m,l\rangle} F(l,m)\,\hat{c}_l^\dagger \hat{c}_m = \\
&\sum_{[l,m]} F(l,m)\,\hat{c}_l^\dagger \hat{c}_m \overset{\text{! caution !}}{\neq} 2 \cdot \sum_{\langle l,m\rangle} F(l,m)\,\hat{c}_l^\dagger \hat{c}_m
\end{aligned} \tag{2.10}$$

With that convention, you can rewrite Equation 2.9 to Equation 2.11.

$$\hat{V} = -J \cdot \sum_{[l,m]} \left(\hat{c}_l^\dagger \hat{c}_m + \hat{d}_l^\dagger \hat{d}_m\right) \tag{2.11}$$

2.1.2 Interaction Picture and Time Evolution

Solving the quantum-mechanical many-body time-evolution problem in general requires at least a time-complexity of $\mathcal{O}(2^n)$. For some special cases it is possible to solve analytically. E.g. for 1-dimensional systems, the Hubbard Model can be solved analytically even with the addition of an electrical field [7].
For arbitrary geometry in two or even more dimensions, this is no longer possible. In that case, approximations are required to lower the computational complexity to get it to a reasonable level to obtain the required measurements. Here, the calculation is performed in the *Interaction Picture* (relevant equations from [5]), along the lines of the calculation performed in [8].
The equations that were provided so far, show operators from the *Schrödinger Picture*. For this section, the operators and states will be labeled whether they are Schrödinger- or Interaction Picture. All un-labeled elements are assumed to be Schrödinger Picture.
A general state in the Schrödinger Picture can be expanded in a basis like shown in Equation 2.12. It is time-evolved using the full Hamiltonian in the Schrödinger Picture (Equation 2.13, all units so that $\hbar = 1$).

$$\left|\Psi^S\right\rangle = \left|\Psi^S(t=0)\right\rangle = \sum_N |N\rangle \underbrace{\left\langle N\middle|\Psi^S\right\rangle}_{\Psi_N} = \sum_N \Psi_N |N\rangle \tag{2.12}$$

$$\left|\Psi^S(t)\right\rangle = e^{-i\mathcal{H}^S t}\left|\Psi^S(t=0)\right\rangle = e^{-i\mathcal{H}^S t}\left|\Psi^S\right\rangle \tag{2.13}$$

In the Interaction Picture, operators have a time evolution, that depends only on the $\mathcal{H}_0^S$-part, shown in Equation 2.14 (assuming the operators are time-independent in the Schrödinger Picture).

$$\begin{aligned} &\hat{\mathrm{A}}^{\mathrm{I}}(t) = e^{i\mathcal{H}_0^{\mathrm{S}}t}\hat{\mathrm{A}}^{\mathrm{S}}e^{-i\mathcal{H}_0^{\mathrm{S}}t} \quad \Rightarrow \quad \mathcal{H}_0^{\mathrm{S}} = \mathcal{H}_0^{\mathrm{I}} = \mathcal{H}_0 \\ \text{especially:} \quad &\hat{\mathrm{V}}^{\mathrm{I}}(t) = e^{i\mathcal{H}_0^{\mathrm{S}}t}\hat{\mathrm{V}}^{\mathrm{S}}e^{-i\mathcal{H}_0^{\mathrm{S}}t} \end{aligned} \tag{2.14}$$

The time-evolution of a general state in the Interaction Picture on the other hand is obtained like Equation 2.15.

$$\left|\Psi^{\mathrm{I}}(t)\right\rangle = e^{i\mathcal{H}_0 t}\left|\Psi^{\mathrm{S}}(t)\right\rangle \overset{2.13}{=} e^{i\mathcal{H}_0 t}e^{-i\mathcal{H}^{\mathrm{S}}t}\left|\Psi^{\mathrm{S}}\right\rangle \quad \Rightarrow \quad \left|\Psi^{\mathrm{S}}(t)\right\rangle = e^{-i\mathcal{H}_0 t}\left|\Psi^{\mathrm{I}}(t)\right\rangle \tag{2.15}$$

Differentiating Equation 2.15 and substituting the special case in Equation 2.14 along the way, one gets Equation 2.16.

$$\begin{aligned} \frac{\mathrm{d}\left|\Psi^{\mathrm{I}}(t)\right\rangle}{\mathrm{d}t} &\overset{2.15}{=} \frac{\mathrm{d}}{\mathrm{d}t}\left(e^{i\mathcal{H}_0 t}e^{-i\mathcal{H}^{\mathrm{S}}t}\right)\left|\Psi^{\mathrm{S}}\right\rangle \\ &= -i\cdot e^{i\mathcal{H}_0 t}\left(\mathcal{H}^{\mathrm{S}} - \mathcal{H}_0\right)e^{-i\mathcal{H}^{\mathrm{S}}t}\left|\Psi^{\mathrm{S}}\right\rangle \\ &\overset{2.1}{=} -i\cdot e^{i\mathcal{H}_0 t}\hat{\mathrm{V}}^{\mathrm{S}}\cdot 1\cdot e^{-i\mathcal{H}^{\mathrm{S}}t}\left|\Psi^{\mathrm{S}}\right\rangle \\ &= -i\cdot e^{i\mathcal{H}_0 t}\hat{\mathrm{V}}^{\mathrm{S}}e^{-i\mathcal{H}_0 t}e^{i\mathcal{H}_0 t}e^{-i\mathcal{H}^{\mathrm{S}}t}\left|\Psi^{\mathrm{S}}\right\rangle \\ &\overset{2.15}{=} -i\cdot e^{i\mathcal{H}_0 t}\hat{\mathrm{V}}^{\mathrm{S}}e^{-i\mathcal{H}_0 t}\left|\Psi^{\mathrm{I}}(t)\right\rangle \\ &\overset{2.14}{=} -i\cdot\hat{\mathrm{V}}^{\mathrm{I}}(t)\left|\Psi^{\mathrm{I}}(t)\right\rangle \end{aligned} \tag{2.16}$$

The equation of motion in Equation 2.16, is solved by the ansatz in Equation 2.17 with the *time-ordering-operator* $\mathbb{T}$ and the *time-evolution-operator* $\hat{\mathrm{P}}^{\mathrm{I}}(t)$.

$$\begin{aligned} \left|\Psi^{\mathrm{I}}(t)\right\rangle &= \mathbb{T}\left\{e^{-i\int_0^t \mathrm{d}t'\hat{\mathrm{V}}^{\mathrm{I}}(t')}\right\}\left|\Psi^{\mathrm{I}}(0)\right\rangle \\ &= \hat{\mathrm{P}}^{\mathrm{I}}(t)\left|\Psi^{\mathrm{I}}(0)\right\rangle \overset{2.15}{=} \hat{\mathrm{P}}^{\mathrm{I}}(t)\left|\Psi^{\mathrm{S}}(0)\right\rangle \end{aligned} \tag{2.17}$$

Plugging this in, one finally obtains the general time-evolution for the state in the Schrödinger Picture in Equation 2.18. This requires the *base-energies* $E_0(N)$, where $e^{-i\mathcal{H}_0 t}\left|N\right\rangle = e^{-iE_0(N)t}\left|N\right\rangle$.

$$
\begin{aligned}
\left|\Psi^{\mathrm{S}}(t)\right\rangle &\overset{2.15}{=} e^{-i\mathcal{H}_0 t}\left|\Psi^{\mathrm{I}}(t)\right\rangle \overset{2.17}{=} e^{-i\mathcal{H}_0 t}\hat{\mathrm{P}}^{\mathrm{I}}(t)\left|\Psi^{\mathrm{S}}\right\rangle \\
&= e^{-i\mathcal{H}_0 t}\cdot 1\cdot \hat{\mathrm{P}}^{\mathrm{I}}(t)\left|\Psi^{\mathrm{S}}\right\rangle \cdot 1 \\
&= e^{-i\mathcal{H}_0 t}\sum_N |N\rangle \left\langle N\middle|\hat{\mathrm{P}}^{\mathrm{I}}(t)\middle|\Psi^{\mathrm{S}}\right\rangle \cdot \frac{\left\langle N\middle|\Psi^{\mathrm{S}}\right\rangle}{\langle N|\Psi^{\mathrm{S}}\rangle} \\
&\overset{2.12}{=} \sum_N \underbrace{e^{-i\mathcal{H}_0 t}}_{e^{-iE_0 t}} \underbrace{\frac{\left\langle N\middle|\hat{\mathrm{P}}^{\mathrm{I}}(t)\middle|\Psi^{\mathrm{S}}\right\rangle}{\langle N|\Psi^{\mathrm{S}}\rangle}}_{e^{\mathcal{H}_N(t)}} \Psi_N |N\rangle \\
&= \sum_N e^{-iE_0(N)t} e^{\mathcal{H}_N(t)}\Psi_N |N\rangle = \sum_N e^{\mathcal{H}_{\mathrm{eff}}(N,t)}\Psi_N |N\rangle
\end{aligned}
\tag{2.18}
$$

Up to this point, the calculation is still exact. Now the effective hamiltonian $\mathcal{H}_N(t)$ could be approximated in different ways. Typically $e^{\mathcal{H}_N(t)}$ is *Taylor-expanded*, but in this context this approximation would converge quite poorly. To combat this and require a smaller order of expansion, one could employ the *cumulant-expansion* [9] to expand the object in terms of its exponent $\mathcal{H}_N(t)$ and not the whole exponential $e^{\mathcal{H}_N(t)}$. The first and second order cumulants for the cumulant-expansion are provided in Equation 2.19 [8] with the *cumulant-average* in respect to the state $|N\rangle$, $\langle\cdot\rangle_{\mathrm{c}(N)}$.

$$
\begin{aligned}
\left\langle \hat{\mathrm{A}}\right\rangle_{\mathrm{c}(N)} &= \frac{\left\langle N\middle|\hat{\mathrm{A}}\middle|\Psi^{\mathrm{S}}\right\rangle}{\langle N|\Psi^{\mathrm{S}}\rangle} \\
\left\langle \hat{\mathrm{A}}\hat{\mathrm{B}}\right\rangle_{\mathrm{c}(N)} &= \frac{\left\langle N\middle|\hat{\mathrm{A}}\hat{\mathrm{B}}\middle|\Psi^{\mathrm{S}}\right\rangle}{\langle N|\Psi^{\mathrm{S}}\rangle} - \frac{\left\langle N\middle|\hat{\mathrm{A}}\middle|\Psi^{\mathrm{S}}\right\rangle \cdot \left\langle N\middle|\hat{\mathrm{B}}\middle|\Psi^{\mathrm{S}}\right\rangle}{\langle N|\Psi^{\mathrm{S}}\rangle^2}
\end{aligned}
\tag{2.19}
$$

Plugging this into the rearranged result from Equation 2.18 following the derivation in [8], one obtains an expansion for $\mathcal{H}_N(t)$ in Equation 2.20.

$$
\begin{aligned}
\left\langle N\middle|\Psi^{\mathrm{S}}\right\rangle \cdot e^{\mathcal{H}_N(t)} &\overset{2.18}{=} \frac{\left\langle N\middle|\hat{\mathrm{P}}^{\mathrm{I}}(t)\middle|\Psi^{\mathrm{S}}\right\rangle}{\langle N|\Psi^{\mathrm{S}}\rangle} \overset{2.17}{=} \left\langle N\middle|\mathbb{T}e^{-i\int_0^t \mathrm{d}t'\hat{\mathrm{V}}^{\mathrm{I}}(t')}\middle|\Psi^{\mathrm{S}}\right\rangle \\
&\overset{[8]}{\Longleftrightarrow} \\
\mathcal{H}_N(t) &= \sum_{v=1}^{\infty}\frac{(-i)^v}{v!}\int_0^t \mathrm{d}t_1\int_0^t \mathrm{d}t_2\cdots\int_0^t \mathrm{d}t_v \left\langle \mathbb{T}\,\hat{\mathrm{V}}^{\mathrm{I}}(t_1)\hat{\mathrm{V}}^{\mathrm{I}}(t_2)\cdots\hat{\mathrm{V}}^{\mathrm{I}}(t_v)\right\rangle_{\mathrm{c}(N)} \\
&= -i\int_0^t \mathrm{d}t' \frac{\left\langle N\middle|\hat{\mathrm{V}}^{\mathrm{I}}(t')\middle|\Psi^{\mathrm{S}}\right\rangle}{\langle N|\Psi^{\mathrm{S}}\rangle} + \cdots
\end{aligned}
\tag{2.20}
$$

As $\hat{V}^{I}(t') \propto J$, if J is a small perturbation (in comparison to U) this quickly converges for few terms.

2.1.3 Operators in the Interaction Picture

As Equation 2.20 reveals, by this process that problem was converted into finding $\hat{V}^{I}(t')$ to be able to calculate the first order of the expansion.
Operators in the Interaction Picture can be derived from their Schrödinger Picture variants like previously shown in Equation 2.14. With this, one can derive a useful expression for their *equation of motion* (Equation 2.21).

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\hat{A}^{I}(t) &= i\left[\mathcal{H}_0^{S}, \hat{A}^{I}(t)\right] \overset{2.14}{=} i\left[\mathcal{H}_0^{S}, e^{i\mathcal{H}_0^{S}t}\hat{A}^{S}e^{-i\mathcal{H}_0^{S}t}\right] \\ &= ie^{i\mathcal{H}_0^{S}t}\left[\mathcal{H}_0^{S}, \hat{A}^{S}\right]e^{-i\mathcal{H}_0^{S}t} = i\left\{\left[\mathcal{H}_0^{S}, \hat{A}^{S}\right]\right\}(t) \end{aligned} \tag{2.21}$$

Equation 2.21 also defines the operator $\{\cdot\}(t)$, which simply is $e^{i\mathcal{H}_0^{S}t}\{\cdot\}e^{-i\mathcal{H}_0^{S}t}$. This section uses (t) for indicating that operators in the Interaction Picture have a time dependence. This is superficial and only for better readability, as in fact $\hat{c}_m^{\dagger I}(t) = \hat{c}_m^{\dagger I} = \left\{\hat{c}_m^{\dagger S}\right\}(t)$.
If now $\left[\mathcal{H}_0^{S}, \hat{A}^{S}\right]$ now is a function of $\hat{A}^{S}$, one gets a *differential equation* that can be solved to obtain $\hat{A}^{I}(t)$.
For the repetitive calculation of the objects of type $\left[\mathcal{H}_0^{S}, \hat{A}^{S}\right]$ for various $\hat{A}^{S}$, the tool *Math-Manipulator* was specifically developed. Later in *subsection 3.1.2 Math-Manipulator Results* an explanation will be given, where the derivations that are used here can be checked with this tool from scratch. The equations 2.22 and 2.23 list exemplary results that were computed with the tool.

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\hat{d}_m^{\dagger I}\hat{d}_m^{I}\right)(t) = i\left\{\left[\mathcal{H}_0, \hat{d}_m^{\dagger S}\hat{d}_m^{S}\right]\right\}(t) \overset{MM}{=} 0 \quad \Rightarrow \quad \left(\hat{d}_m^{\dagger I}\hat{d}_m^{I}\right)(t) = \hat{d}_m^{\dagger S}\hat{d}_m^{S} \tag{2.22}$$

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\hat{c}_m^{\dagger I}(t) &= i\left\{\left[\mathcal{H}_0, \hat{c}_m^{\dagger S}\right]\right\}(t) \overset{MM}{=} i\left\{\left(\varepsilon_m + U\hat{d}_m^{\dagger S}\hat{d}_m^{S}\right)\hat{c}_m^{S}\right\}(t) \\ &= ie^{i\mathcal{H}_0^{S}t}\left(\varepsilon_m + U\hat{d}_m^{\dagger S}\hat{d}_m^{S}\right)\hat{c}_m^{S}e^{-i\mathcal{H}_0^{S}t} \\ &= i\left(\varepsilon_m + Ue^{i\mathcal{H}_0^{S}t}\hat{d}_m^{\dagger S}\hat{d}_m^{S}e^{-i\mathcal{H}_0^{S}t}\right)e^{i\mathcal{H}_0^{S}t}\hat{c}_m^{S}e^{-i\mathcal{H}_0^{S}t} \\ &\overset{2.14}{=} i\left(\varepsilon_m + U\left(\hat{d}_m^{\dagger I}\hat{d}_m^{I}\right)(t)\right)\hat{c}_m^{\dagger I}(t) \end{aligned} \tag{2.23}$$

Additionally, for fermions and hard-core bosons Equation 2.24 holds for all $z \in \mathbb{N}$.

$$\left(\hat{n}_{l,\sigma}^{S}\right)^{z} = \left(\hat{c}_{l,\sigma}^{\dagger S}\hat{c}_{l,\sigma}^{S}\right)^{z} = \hat{c}_{l,\sigma}^{\dagger S}\hat{c}_{l,\sigma}^{S} = \hat{n}_{l,\sigma}^{S} \tag{2.24}$$

With this, one can first derive Equation 2.25 and finally use an exponential function as a natural ansatz to solve Equation 2.23 with Equation 2.26.

$$\begin{aligned} e^{a\cdot\hat{c}^{\dagger S}_{l,\sigma}\hat{c}^{S}_{l,\sigma}} &= \sum_{m=0}^{\infty}\frac{a^m\left(\hat{c}^{\dagger S}_{l,\sigma}\hat{c}^{S}_{l,\sigma}\right)^m}{m!} \overset{2.24}{=} 1+\left[\sum_{m=1}^{\infty}\frac{a^m}{m!}\right]\cdot\left(\hat{c}^{\dagger S}_{l,\sigma}\hat{c}^{S}_{l,\sigma}\right) \\ &= 1+\left[\sum_{m=0}^{\infty}\frac{a^m}{m!}-1\right]\cdot\left(\hat{c}^{\dagger S}_{l,\sigma}\hat{c}^{S}_{l,\sigma}\right) = 1+(e^a-1)\cdot\hat{c}^{\dagger S}_{l,\sigma}\hat{c}^{S}_{l,\sigma} \end{aligned} \tag{2.25}$$

$$\begin{aligned} \overset{2.23,\,2.22}{\Longrightarrow}\ \hat{c}^{\dagger I}_{m}(t) &= e^{i\cdot\varepsilon_m\cdot t+i\cdot U\left(\hat{d}^{\dagger I}_{m}\hat{d}^{I}_{m}\right)(t)\cdot t}\hat{c}^{\dagger S}_{m} \overset{2.22}{=} e^{i\cdot\varepsilon_m\cdot t+i\cdot U\cdot\hat{d}^{\dagger S}_{m}\hat{d}^{S}_{m}\cdot t}\hat{c}^{\dagger S}_{m} \\ &\overset{2.25}{=} e^{i\cdot\varepsilon_m\cdot t}\left(1+\left(e^{i\cdot U\cdot t}-1\right)\hat{d}^{\dagger S}_{m}\hat{d}^{S}_{m}\right)\hat{c}^{\dagger S}_{m} \end{aligned} \tag{2.26}$$

From analogous calculations follows:

$$\begin{aligned} \hat{c}^{\dagger I}_{m}(t) &= e^{i\cdot\varepsilon_m\cdot t}\left(1+\left(e^{i\cdot U\cdot t}-1\right)\hat{d}^{\dagger S}_{m}\hat{d}^{S}_{m}\right)\hat{c}^{\dagger S}_{m} \\ \hat{c}^{I}_{m}(t) &= e^{-i\cdot\varepsilon_m\cdot t}\left(1+\left(e^{-i\cdot U\cdot t}-1\right)\hat{d}^{\dagger S}_{m}\hat{d}^{S}_{m}\right)\hat{c}^{S}_{m} \\ \hat{d}^{\dagger I}_{m}(t) &= e^{i\cdot\varepsilon_m\cdot t}\left(1+\left(e^{i\cdot U\cdot t}-1\right)\hat{c}^{\dagger S}_{m}\hat{c}^{S}_{m}\right)\hat{d}^{\dagger S}_{m} \\ \hat{d}^{I}_{m}(t) &= e^{-i\cdot\varepsilon_m\cdot t}\left(1+\left(e^{-i\cdot U\cdot t}-1\right)\hat{c}^{\dagger S}_{m}\hat{c}^{S}_{m}\right)\hat{d}^{S}_{m} \end{aligned}$$

Injecting these derivations into Equation 2.11, finally $\hat{V}^{I}(t)$ can be derived in Equation 2.27.

$$\begin{aligned} \hat{V}^{I}(t) &= \left\{\hat{V}^{S}\right\}(t) \overset{2.11}{=} -J\cdot\sum_{[l,m]}\left\{\left(\hat{c}^{\dagger S}_{l}\hat{c}^{S}_{m}+\hat{d}^{\dagger S}_{l}\hat{d}^{S}_{m}\right)\right\}(t) \\ &= -J\cdot\sum_{[l,m]}\left(\hat{c}^{\dagger I}_{l}(t)\hat{c}^{I}_{m}(t)+\hat{d}^{\dagger I}_{l}(t)\hat{d}^{I}_{m}(t)\right) \\ &\overset{\text{MM}}{=} -J\cdot\sum_{[l,m]}\Big[e^{i\cdot(\varepsilon_m-\varepsilon_l)\cdot t}\cdot\hat{V}_{\text{Part A}}(l,m)+ \\ &\qquad e^{i\cdot(\varepsilon_m-\varepsilon_l+U)\cdot t}\cdot\hat{V}_{\text{Part B}}(l,m)+e^{i\cdot(\varepsilon_m-\varepsilon_l-U)\cdot t}\cdot\hat{V}_{\text{Part C}}(l,m)\Big] \end{aligned} \tag{2.27}$$

$$
\begin{aligned}
\hat{V}_{\text{Part A}}(l,m) &\overset{\text{MM}}{=} \left(5 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S}\right) + \left(5 \cdot \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(2 \cdot \hat{c}_l^S \hat{c}_m^S \hat{c}_l^{\dagger S} \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(2 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_m^S \hat{d}_l^{\dagger S} \hat{d}_m^{\dagger S}\right) + \\
&+ \left(-3 \cdot \hat{c}_l^S \hat{c}_l^{\dagger S} \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(-3 \cdot \hat{c}_m^S \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(-3 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_l^{\dagger S}\right) + \left(-3 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S} \hat{d}_m^S \hat{d}_m^{\dagger S}\right) \\
\hat{V}_{\text{Part B}}(l,m) &\overset{\text{MM}}{=} \left(-2 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S}\right) + \left(-2 \cdot \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(-1 \cdot \hat{c}_l^S \hat{c}_m^S \hat{c}_l^{\dagger S} \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(-1 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_m^S \hat{d}_l^{\dagger S} \hat{d}_m^{\dagger S}\right) + \\
&+ \left(1 \cdot \hat{c}_l^S \hat{c}_l^{\dagger S} \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(2 \cdot \hat{c}_m^S \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(1 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_l^{\dagger S}\right) + \left(2 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S} \hat{d}_m^S \hat{d}_m^{\dagger S}\right) \\
\hat{V}_{\text{Part C}}(l,m) &\overset{\text{MM}}{=} \left(-2 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S}\right) + \left(-2 \cdot \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(-1 \cdot \hat{c}_l^S \hat{c}_m^S \hat{c}_l^{\dagger S} \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(-1 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_m^S \hat{d}_l^{\dagger S} \hat{d}_m^{\dagger S}\right) + \\
&+ \left(2 \cdot \hat{c}_l^S \hat{c}_l^{\dagger S} \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(1 \cdot \hat{c}_m^S \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_m^{\dagger S}\right) + \left(2 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S} \hat{d}_l^S \hat{d}_l^{\dagger S}\right) + \left(1 \cdot \hat{c}_l^S \hat{c}_m^{\dagger S} \hat{d}_m^S \hat{d}_m^{\dagger S}\right)
\end{aligned}
\tag{2.28}
$$

It is possible to rewrite the components from Equation 2.28 and write them as single hoppings, decorated with number operators. One receives Equation 2.29, that is symmetrical in $\uparrow$ and $\downarrow$ (with $\overline{\uparrow} = \downarrow$ and $\overline{\downarrow} = \uparrow$).

$$
\begin{aligned}
\hat{V}_{\text{Part A}}(l,m) &\overset{\text{MM}}{=} \sum_{\sigma \in \{\uparrow,\downarrow\}} \hat{c}_{l,\sigma}^S \hat{c}_{m,\sigma}^{\dagger S} \left(1 + 2 \cdot \hat{n}_{l,\overline{\sigma}}^S \hat{n}_{m,\overline{\sigma}}^S - \hat{n}_{l,\overline{\sigma}}^S - \hat{n}_{m,\overline{\sigma}}^S\right) \\
\hat{V}_{\text{Part B}}(l,m) &\overset{\text{MM}}{=} \sum_{\sigma \in \{\uparrow,\downarrow\}} \hat{c}_{l,\sigma}^S \hat{c}_{m,\sigma}^{\dagger S} \left(\hat{n}_{m,\overline{\sigma}}^S - \hat{n}_{l,\overline{\sigma}}^S \hat{n}_{m,\overline{\sigma}}^S\right) \\
\hat{V}_{\text{Part C}}(l,m) &\overset{\text{MM}}{=} \sum_{\sigma \in \{\uparrow,\downarrow\}} \hat{c}_{l,\sigma}^S \hat{c}_{m,\sigma}^{\dagger S} \left(\hat{n}_{l,\overline{\sigma}}^S - \hat{n}_{l,\overline{\sigma}}^S \hat{n}_{m,\overline{\sigma}}^S\right)
\end{aligned}
\tag{2.29}
$$

Base Energy

In Equation 2.18, the effective Hamiltonian $\mathcal{H}_{\text{eff}}(N,t) = -iE_0(N)t + \mathcal{H}_N(t)$ was defined. This construct will be necessary in the following sections. $E_0(N)$ is luckily calculated quite swiftly, as one can see in Equation 2.30 (with $n_{l,\sigma}$ - the occupation-number in $|N\rangle$, not the operator).

$$
\begin{aligned}
E_0(N) &= \frac{\langle N|\mathcal{H}_0|N\rangle}{\langle N|N\rangle} \\
&\overset{2.2}{=} \langle N|U \cdot \sum_l \hat{n}_{l,\uparrow}\hat{n}_{l,\downarrow}|N\rangle + \langle N|\sum_{l,\sigma} \varepsilon_l \hat{n}_{l,\sigma}|N\rangle \\
&= U \cdot \sum_l n_{l,\uparrow} n_{l,\downarrow} + \sum_l \varepsilon_l n_{l,\sigma}
\end{aligned}
\tag{2.30}
$$

Calculate $\mathcal{H}_N(t)$ to get the full effective Hamiltonian

The last missing element for $\mathcal{H}_{\text{eff}}(N,t) = -iE_0(N)t + \mathcal{H}_N(t)$ now is $\mathcal{H}_N(t)$. At least for the first order, the integration in Equation 2.20 is easily doable, with now knowing the form of $\hat{V}^{\text{I}}(t)$.

The result of the integration can be seen in Equation 2.31.

$$
\begin{aligned}
\mathcal{H}_N(t) &\overset{2.20}{\approx} -i \int_0^t \mathrm{d}t' \, \frac{\left\langle N \middle| \hat{\mathrm{V}}^{\mathrm{I}}(t') \middle| \Psi^{\mathrm{S}} \right\rangle}{\langle N | \Psi^{\mathrm{S}} \rangle} \\
&\overset{2.12}{=} -i \frac{1}{\Psi_N} \int_0^t \mathrm{d}t' \sum_K \langle N | \hat{\mathrm{V}}^{\mathrm{I}}(t') | K \rangle \, \Psi_K \\
&\overset{2.27}{=} i \cdot J \sum_K \sum_{[l,m]} \frac{\Psi_K}{\Psi_N} \int_0^t \mathrm{d}t' \left[e^{i \cdot (\varepsilon_m - \varepsilon_l) \cdot t'} \cdot \langle N | \hat{\mathrm{V}}_{\text{Part A}}(l,m) | K \rangle + \right. \\
&\qquad \left. e^{i \cdot (\varepsilon_m - \varepsilon_l + U) \cdot t'} \cdot \langle N | \hat{\mathrm{V}}_{\text{Part B}}(l,m) | K \rangle + e^{i \cdot (\varepsilon_m - \varepsilon_l - U) \cdot t'} \cdot \langle N | \hat{\mathrm{V}}_{\text{Part C}}(l,m) | K \rangle \right] \\
&= J \sum_K \sum_{[l,m]} \frac{\Psi_K}{\Psi_N} \left[\frac{e^{i \cdot (\varepsilon_m - \varepsilon_l) \cdot t} - 1}{\varepsilon_m - \varepsilon_l} \cdot \langle N | \hat{\mathrm{V}}_{\text{Part A}}(l,m) | K \rangle + \right. \\
&\qquad \left. \frac{e^{i \cdot (\varepsilon_m - \varepsilon_l + U) \cdot t} - 1}{\varepsilon_m - \varepsilon_l + U} \cdot \langle N | \hat{\mathrm{V}}_{\text{Part B}}(l,m) | K \rangle + \frac{e^{i \cdot (\varepsilon_m - \varepsilon_l - U) \cdot t} - 1}{\varepsilon_m - \varepsilon_l - U} \cdot \langle N | \hat{\mathrm{V}}_{\text{Part C}}(l,m) | K \rangle \right]
\end{aligned}
\tag{2.31}
$$

While it seems that $\sum_K \sum_{[l,m]}$ is even a larger summation than the previously problematic ones, because the operators $\hat{\mathrm{V}}_{\text{Part A}}(l,m)$, $\hat{\mathrm{V}}_{\text{Part B}}(l,m)$ and $\hat{\mathrm{V}}_{\text{Part C}}(l,m)$ have interaction range limited to nearest neighbors, actually the summation is really sparse and most entries are 0. This summation can be efficiently evaluated in $\mathcal{O}(\#(\text{sites}) \cdot \#(\text{nearest neighbors}))$. How this could be done, can be looked up in the implementation [10]: `/computation-scripts/hamiltonian.py` in the specialization class *HardcoreBosonicHamiltonian*.

2.1.4 Observable Measurement

The evaluation of observables is a primary requirement for this application. First it is necessary to acquire a more general formula, that one later can plug specific observables into.
From Equation 2.18 and applying *dagger* † to said equation, Equation 2.32 follows.

$$
\begin{aligned}
\left\langle N \middle| \Psi^{\mathrm{S}}(t) \right\rangle &\overset{2.18}{=} \langle N | \sum_K e^{\mathcal{H}_{\text{eff}}(K,t)} \Psi_K \, | K \rangle \\
&= \sum_K e^{\mathcal{H}_{\text{eff}}(K,t)} \Psi_K \underbrace{\langle N | K \rangle}_{\delta_{N,K}} = e^{\mathcal{H}_{\text{eff}}(N,t)} \Psi_N \\
\overset{\dagger}{\Rightarrow} \left\langle \Psi^{\mathrm{S}}(t) \middle| N \right\rangle &= e^{\mathcal{H}_{\text{eff}}{}^*(N,t)} \Psi_N^*
\end{aligned}
\tag{2.32}
$$

In Equation 2.33 by starting with the default way of calculating a normalized *expectation-value* for an observable $\hat{\mathcal{O}}$ [11] and applying multiple modifications, one gets a representation that can be effectively sampled later.

$$
\begin{aligned}
\frac{\langle \Psi^S(t)|\hat{\mathcal{O}}|\Psi^S(t)\rangle}{\langle \Psi^S(t)|\Psi^S(t)\rangle} &= \frac{\sum_N \langle \Psi^S(t)|N\rangle \langle N|\hat{\mathcal{O}}|\Psi^S(t)\rangle}{\sum_K \langle \Psi^S(t)|K\rangle \langle K|\Psi^S(t)\rangle} \\
&\overset{2.32}{=} \frac{\sum_N e^{\mathcal{H}_{\text{eff}}^*(N,t)} \Psi_N^* \langle N|\hat{\mathcal{O}}|\Psi^S(t)\rangle}{\sum_K e^{\mathcal{H}_{\text{eff}}^*(K,t)} \Psi_K^* e^{\mathcal{H}_{\text{eff}}(K,t)} \Psi_K} \\
&= \frac{\sum_N e^{\mathcal{H}_{\text{eff}}^*(N,t)} \Psi_N^* \langle N|\hat{\mathcal{O}}|\Psi^S(t)\rangle}{\sum_K |e^{\mathcal{H}_{\text{eff}}(K,t)}|^2 |\Psi_K|^2} \cdot \frac{\langle N|\Psi^S(t)\rangle}{\langle N|\Psi^S(t)\rangle} \\
&= \sum_N \frac{e^{\mathcal{H}_{\text{eff}}^*(N,t)} \Psi_N^* \langle N|\Psi^S(t)\rangle}{\sum_K |e^{\mathcal{H}_{\text{eff}}(K,t)}|^2 |\Psi_K|^2} \cdot \frac{\langle N|\hat{\mathcal{O}}|\Psi^S(t)\rangle}{\langle N|\Psi^S(t)\rangle} \\
&\overset{2.32}{=} \sum_N \underbrace{\frac{|e^{\mathcal{H}_{\text{eff}}(N,t)}|^2 |\Psi_N|^2}{\sum_K |e^{\mathcal{H}_{\text{eff}}(K,t)}|^2 |\Psi_K|^2}}_{\mathrm{P}(N,t)} \cdot \frac{\langle N|\hat{\mathcal{O}}|\Psi^S(t)\rangle}{\langle N|\Psi^S(t)\rangle} \\
&= \sum_N \mathrm{P}(N,t) \frac{\langle N|\hat{\mathcal{O}}|\Psi^S(t)\rangle}{\langle N|\Psi^S(t)\rangle} = \sum_N \mathrm{P}(N,t) \frac{\sum_K \langle N|\hat{\mathcal{O}}|K\rangle \langle K|\Psi^S(t)\rangle}{\langle N|\Psi^S(t)\rangle} \\
&\overset{2.32}{=} \sum_N \mathrm{P}(N,t) \frac{\sum_K \langle N|\hat{\mathcal{O}}|K\rangle e^{\mathcal{H}_{\text{eff}}(K,t)} \Psi_K}{e^{\mathcal{H}_{\text{eff}}(N,t)} \Psi_N} \\
&= \sum_N \mathrm{P}(N,t) \underbrace{\sum_K \langle N|\hat{\mathcal{O}}|K\rangle e^{\mathcal{H}_{\text{eff}}(K,t) - \mathcal{H}_{\text{eff}}(N,t)} \frac{\Psi_K}{\Psi_N}}_{\hat{\mathcal{O}}_{\text{loc}}(N,t)} = \sum_N \mathrm{P}(N,t) \hat{\mathcal{O}}_{\text{loc}}(N,t)
\end{aligned}
\tag{2.33}
$$

By inserting Equation 2.32 into Equation 2.33, a probability-dependent formula is obtained, that can be used to plug in specific observables. The probability of a base-state at a specified time $\mathrm{P}(N,t)$ is important for this calculation. The *subsection 2.1.5 Monte Carlo Sampling* will discuss the viability of obtaining these probabilities and strategies of doing so. The matrix-element $\langle N|\hat{\mathcal{O}}|K\rangle$ has the important property of being 0 for almost all combinations of $\langle N|$ and $|K\rangle$, depending on the chosen observable $\hat{\mathcal{O}}$, which is what makes the calculation of the *local-observable* $\hat{\mathcal{O}}_{\text{loc}}(N,t)$ possible in the first place.

Double-Occupation

One possibly interesting observable would be the measurement of *double-occupation* (one particle of spin-up and one of spin-down) per site.

The operator $\hat{\mathcal{O}}_{\text{do}}(l)$ that measure the double-occupation on site l can be easily written as $\hat{\mathcal{O}}_{\text{do}}(l) = \hat{\text{n}}_{l,\uparrow}\hat{\text{n}}_{l,\downarrow}$.
As the used basis-states are eigenstates of the number-operators, the evaluation of this operator is quite straight forward. $\langle N|\hat{\mathcal{O}}_{\text{do}}(l)|K\rangle$ becomes $\delta_{N,K} \cdot n_{l,\uparrow} \cdot n_{l,\downarrow}$ with the occupation-number (not operator) of the specific site and spin in $\langle N|$, $n_{l,\sigma}$. With that $\hat{\mathcal{O}}_{\text{loc}}(N,t)$ can be simply evaluated to the result in Equation 2.34.

$$\hat{\mathcal{O}}_{\text{loc}}(N,t) = \sum_K \langle N|\hat{\mathcal{O}}_{\text{do}}(l)|K\rangle\, e^{\mathcal{H}_{\text{eff}}(K,t)-\mathcal{H}_{\text{eff}}(N,t)} \frac{\Psi_K}{\Psi_N} = n_{l,\uparrow} \cdot n_{l,\downarrow} \tag{2.34}$$

Spin-Polarized Kinetics

For measuring the spin-polarized flow of particles, a slightly more complex observable must be employed. In Equation 2.35 a possible observable is listed, that measures such kinetics direction in-dependent ($\hat{\mathcal{O}}_{\text{sp-kin}}(l \leftrightarrow m,\sigma)$) or direction dependent ($\hat{\mathcal{O}}_{\text{sp-kin, dir}}(l \to m,\sigma)$, caution: needs i to obtain hermitian operator).

$$\begin{aligned}\hat{\mathcal{O}}_{\text{sp-kin}}(l \leftrightarrow m,\sigma) &= -J\left(\hat{\text{c}}^\dagger_{l,\sigma}\hat{\text{c}}_{m,\sigma} + \hat{\text{c}}^\dagger_{m,\sigma}\hat{\text{c}}_{l,\sigma}\right)\\ \hat{\mathcal{O}}_{\text{sp-kin, dir}}(l \to m,\sigma) &= iJ\left(\hat{\text{c}}^\dagger_{m,\sigma}\hat{\text{c}}_{l,\sigma} - \hat{\text{c}}^\dagger_{l,\sigma}\hat{\text{c}}_{m,\sigma}\right)\end{aligned} \tag{2.35}$$

In this case, the used basis-states are not eigenstates of the operators. $\langle N|\hat{\text{c}}^\dagger_{m,\sigma}\hat{\text{c}}_{l,\sigma}|K\rangle$ becomes $\delta_{N,\widetilde{N}} \cdot (1 - n_{l,\sigma}) \cdot n_{m,\sigma}$, where $|\widetilde{N}\rangle$ is the state obtained when the particle number on site m,σ is transferred to site l,σ (this *hopping* is only possible, when there is a particle on the original site and no particle yet on the target site, which is ensured by the occupation numbers). Evaluating the whole operator with the signs and is correctly and in a efficient manner can best be looked up in the implementation [10]: `/computation-scripts/observables.py`.
Overall, the evaluation of this observable requires knowing the value of an object of the form presented in Equation 2.36.

$$\frac{\Psi_{\widetilde{N}}}{\Psi_N} e^{\mathcal{H}_{\text{eff}}(\widetilde{N},t)-\mathcal{H}_{\text{eff}}(N,t)} \tag{2.36}$$

The *subsection 2.1.6 Analytical Simplifications for Special Cases* will go into how to compute this efficiently for states $|N\rangle$ and $|\widetilde{N}\rangle$ that are connected via the hopping between nearest-neighbor lattice sites.
It is finally important to notice, that the objects from Equation 2.36 are not real-valued, but complex. Only the complete observable from the full sum over all basis-states $|N\rangle$ over $\hat{\mathcal{O}}_{\text{loc}}(N,t)$ has a fully vanishing imaginary part. Especially when the observable is approximated with an incomplete set of basis-states, computationally an imaginary component remains. It is important to monitor the magnitude of this value and monitor it going to 0. If it doesn't fully vanish, this indicates an error in the sampling strategy or the implementation or the number of sampled states is not great enough.

2.1.5 Monte Carlo Sampling

Mathematically, the problem of finding the solution to such a quantum-mechanical many-body model is solved by the calculations provided up to this point. However when looking at Equation 2.33, one can easily see why in practice this is not viable with larger systems. The calculation requires summing over all base-states. Their number is given in Equation 2.37 (factor 2 in the exponent for the spin-degree) to be of exponential size in regards to the number of lattice-sites. Considering a physical material has at least in the order of 10^{23} (from *Avogadros-Constant*) degrees of freedom, an $\mathcal{O}(2^{\#(\text{sites})})$ computational-complexity is not good enough to simulate interestingly-sized systems.

$$\#(\text{states}) = 2^{\#(\text{sites})\cdot 2} \tag{2.37}$$

Multiple strategies have already been suggested to circumvent this limitation. Here, the focus will be put onto randomized algorithms that use *Monte-Carlo sampling* in order generate a limited number of states, representative of the real probability-distribution, in order to not having to sum over the complete *Hilbert-Space*. The strategy of solving problems of this kind with the strategy of *variational Monte-Carlo* is e.g. suggested in [12]. This means starting with a random state and applying modifications to it (= variation). When the accepting/rejecting of these modifications is coupled to the probability the state has, based on its energy in the systems energy-distribution, the resulting states will have an energy/probability distribution, that resembles the one as if they were sampled from the full Hilbert-Space. The derivation for the whole process is shown in [11].
However computing the normalized probability for the states is still required, in order to compare against a (e.g. thermal) reference-distribution. As one sees in the probability in Equation 2.38 that was defined in Equation 2.33, still a full sum over all states is required to get the normalization factor.

$$\mathrm{P}(N,t) = \frac{\left|e^{\mathcal{H}_{\text{eff}}(N,t)}\right|^2 |\Psi_N|^2}{\sum_K \left|e^{\mathcal{H}_{\text{eff}}(K,t)}\right|^2 |\Psi_K|^2} \tag{2.38}$$

However by employing the *Metropolis-Hastings Algorithm* instead of only the *Metropolis-Algorithm* to decide on the acceptance/rejection of proposed states, one can drop the requirement of normalizing the probability. In Equation 2.40 the *acceptance-ratio* α is derived that is necessary for the Metropolis-Hastings Algorithm to work (as per [13]) and decide on the transition from state $|N\rangle$ to $|\widetilde{N}\rangle$. The algorithm proposes to generate a new state $|\widetilde{N}\rangle$ and calculate the corresponding α. Then it requires generating a random number u between 0 and 1, pulled from a uniform distribution. If $u \leq \alpha$, $|\widetilde{N}\rangle$ is accepted and will be used as the new current state in the next step of the sampling. If $u > \alpha$, the proposed state is rejected and $|N\rangle$ will be used again as the next state. The function $f(N,t)$, provided in Equation 2.39, fulfills the necessary property of being proportional to the probability $\mathrm{P}(N,t)$.

$$f(N,t) = \left|e^{\mathcal{H}_{\text{eff}}(N,t)}\right|^2 |\Psi_N|^2 \propto \mathrm{P}(N,t) \tag{2.39}$$

$$\begin{aligned}
\alpha &= \frac{\mathrm{P}(\widetilde{N},t)}{\mathrm{P}(N,t)} = \frac{f(\widetilde{N},t)}{f(N,t)} \overset{2.39}{=} \frac{\left|e^{\mathcal{H}_{\mathrm{eff}}(\widetilde{N},t)}\right|^2 \left|\Psi_{\widetilde{N}}\right|^2}{\left|e^{\mathcal{H}_{\mathrm{eff}}(N,t)}\right|^2 \left|\Psi_N\right|^2} \\
&= \frac{\left|\Psi_{\widetilde{N}}\right|^2}{\left|\Psi_N\right|^2} \frac{e^{\Re\left(\mathcal{H}_{\mathrm{eff}}(\widetilde{N},t)\right)+i\Im\left(\mathcal{H}_{\mathrm{eff}}(\widetilde{N},t)\right)+\Re\left(\mathcal{H}_{\mathrm{eff}}(\widetilde{N},t)\right)-i\Im\left(\mathcal{H}_{\mathrm{eff}}(\widetilde{N},t)\right)}}{e^{\Re(\mathcal{H}_{\mathrm{eff}}(N,t))+i\Im(\mathcal{H}_{\mathrm{eff}}(N,t))+\Re(\mathcal{H}_{\mathrm{eff}}(N,t))-i\Im(\mathcal{H}_{\mathrm{eff}}(N,t))}} \\
&= \frac{\left|\Psi_{\widetilde{N}}\right|^2}{\left|\Psi_N\right|^2} e^{2\cdot\Re\left(\mathcal{H}_{\mathrm{eff}}(\widetilde{N},t)\right)-2\cdot\Re(\mathcal{H}_{\mathrm{eff}}(N,t))} \\
&= \frac{\left|\Psi_{\widetilde{N}}\right|^2}{\left|\Psi_N\right|^2} e^{2\cdot\Re\left(\mathcal{H}_{\mathrm{eff}}(\widetilde{N},t)-\mathcal{H}_{\mathrm{eff}}(N,t)\right)}
\end{aligned} \tag{2.40}$$

Multiple factors must be considered, which kind of variation is applied to the states in this schema. However probably the most important criteria is how quickly one can evaluate Equation 2.40, as this will be done tens to hundreds of times between each sampled state.
Finally, the method of sampling a selection of distributed states results in Equation 2.41 as a rewrite to Equation 2.33 [11].

$$\frac{\left\langle\Psi^{\mathrm{S}}(t)\middle|\hat{\mathcal{O}}\middle|\Psi^{\mathrm{S}}(t)\right\rangle}{\left\langle\Psi^{\mathrm{S}}(t)|\Psi^{\mathrm{S}}(t)\right\rangle} \overset{2.33}{=} \sum_N \mathrm{P}(N,t)\hat{\mathcal{O}}_{\mathrm{loc}}(N,t) \overset{[11]}{\approx} \frac{1}{|\{N\}_{\mathrm{MC}}|} \sum_{\{N\}_{\mathrm{MC}}} \hat{\mathcal{O}}_{\mathrm{loc}}(N,t) \tag{2.41}$$

2.1.6 Analytical Simplifications for Special Cases

Generally, handling the computation of the difference of the effective Hamiltonian for two states $\mathcal{H}_{\mathrm{eff}}(\widetilde{N},t) - \mathcal{H}_{\mathrm{eff}}(N,t)$ is the most expensive computation per Monte-Carlo-step. It is required in Equation 2.36 for the calculation of the spin-polarized kinetics. As stated in *subsection 2.1.5 Monte Carlo Sampling* it is also needed to compute the transition probability between two sampled states in Equation 2.40. The calculation of $\mathcal{H}_{\mathrm{eff}}(N,t)$ on its own requires an effort of $\mathcal{O}(\#(\text{sites}) \cdot \#(\text{nearest neighbors}))$. However, for the states $|N\rangle$ and $|\widetilde{N}\rangle$ that are connected with only small changes, most of the elements in the sum cancel and the complexity may be here even reduced to $\mathcal{O}(\#(\text{nearest neighbors}))$. This is quite attractive, because it de-couples the per-step computational costs from the number of lattice sites - a vital step for simulating large systems efficiency - however this requires extra analytical calculations. In this section analytical simplifications are presented to get the objects $E_0(N) - E_0(\widetilde{N})$ and $\mathcal{H}_N(t) - \mathcal{H}_{\widetilde{N}}(t)$ (Caution, do not miss the extra *minus* required, to calculate the required $\mathcal{H}_{\mathrm{eff}}(\widetilde{N},t) - \mathcal{H}_{\mathrm{eff}}(N,t)$ from this!).

Initial State

The initial state of the system before the start of the time-evolution is encoded in Ψ_N, defined by Equation 2.12. Different configurations can be considered here.

One natural choice would be the preparation of a single defined base-state which would result in all Ψ_N being 0 but one that would have the value of 1. This case needs to be considered with a specialized analytical calculation, because there are many instances in this calculation, where a division by Ψ_N is performed.
However the current calculation is compatible with the concept of one base-state having a probability y that is x times (e.g. $x = 100$-times) as large as the probability $z = y/x$ of all the other states. This effectively also concentrates the initial state to one base-state.
With the additional restriction of normalization from Equation 2.42 the individual values for Ψ_N can be calculated (see Equation 2.43).

$$\sum_N {\Psi_N}^2 = 1 \tag{2.42}$$

$$\begin{aligned} &\overset{2.42}{\Rightarrow} y^2 + [\#(\text{states}) - 1]\, z^2 = 1 \\ &y^2 + [\#(\text{states}) - 1]\, \frac{y^2}{x^2} = 1 \\ &y = \sqrt{\left(1 + [\#(\text{states}) - 1]\, \frac{1}{x^2}\right)^{-1}} \end{aligned} \tag{2.43}$$

This configuration was tested and verified for small systems with summation over the full Hilbert-space. However when Monte-Carlo sampled, the configuration produced different results than for exact sampling. This can probably be explained by the fact that the energy landscape of base-states for this configuration is not smooth enough. Causing the Metropolis-Algorithm not to perform as intended for the number of samples. As stated, treating non-uniform distribution requires a different analytical approach. Therefore no further experimentation on this was done.

A second natural choice would be the perfectly uniform distribution of probability for all the base-states. This results in a smooth energy-landscape, that can be properly sampled.
The calculation for this is straight-forward and listed in Equation 2.44

$$\Psi_N = \frac{1}{\sqrt{\#(\text{states})}} \overset{2.37}{=} \frac{1}{\sqrt{2^{\#(\text{sites})\cdot 2}}} = \frac{1}{2^{\#(\text{sites})}} \tag{2.44}$$

One last advantage of the uniform distribution of base-state-probabilities is, that all Ψ_N are equal, which makes it possible to optimize sums, as $\Psi_N/\Psi_{\widetilde{N}} = 1$. The following two calculations require this assumption.
As the states under consideration are $|N\rangle$ and $|\widetilde{N}\rangle$, the occupation-numbers $n_{l,\sigma}$ will describe the occupation of $|N\rangle$ and $\widetilde{n}_{l,\sigma}$ the occupation of $|\widetilde{N}\rangle$. Both the following modifications restrict the values $\widetilde{n}_{l,\sigma}$ to make them dependent on $n_{l,\sigma}$.

Hopping-only modification

In this simplification, the maximum difference that can happen, is that a *hopping* event occurs from site i and spin σ_i to the site j and spin σ_j. This can only happen if $n_{i,\sigma_i} \neq n_{j,\sigma_j}$, because if they are the same, the swapping doesn't change the state. This check can be performed beforehand, to save on computational resources (if they are equal, $\mathcal{H}_{\text{eff}}(\widetilde{N},t) - \mathcal{H}_{\text{eff}}(N,t) = 0$). Because the inverse (hopping backwards) is also an equivalently valid change of state, this really looks at the *swapping* of the two occupations.
The hopping results in the values for $\widetilde{n}_{l,\sigma}$ are given in Equation 2.45.

$$\widetilde{n}_{l,\sigma} = \begin{cases} n_{i,\sigma_i} & : l = j \wedge \sigma = \sigma_j \\ n_{j,\sigma_j} & : l = i \wedge \sigma = \sigma_i \\ n_{l,\sigma} & : \text{else} \end{cases} \tag{2.45}$$

This simplifies the energy difference $E_0(N) - E_0(\widetilde{N})$ like equation Equation 2.46 presents.

$$\begin{aligned} E_0(N) - E_0(\widetilde{N}) &= U \sum_l n_{l,\downarrow} n_{l,\uparrow} - U \sum_l \widetilde{n}_{l,\downarrow} \widetilde{n}_{l,\uparrow} + \sum_{l,\sigma} \varepsilon_l n_{l,\sigma} - \sum_{l,\sigma} \varepsilon_l \widetilde{n}_{l,\sigma} \\ &\overset{2.45}{=} \left(\varepsilon_i - \varepsilon_j\right)\left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) + U \cdot \begin{cases} \left(n_{i,\uparrow} - n_{j,\uparrow}\right)\left(n_{i,\downarrow} - n_{j,\downarrow}\right) & : \sigma_i = \sigma_j \\ \left(n_{i,\uparrow} - n_{j,\downarrow}\right)\left(n_{i,\downarrow} - n_{j,\uparrow}\right) & : \sigma_i \neq \sigma_j \end{cases} \\ &= \left(\varepsilon_i - \varepsilon_j\right)\left(n_{i,\sigma_i} - n_{j,\sigma_j}\right) + U\left(n_{i,\sigma_i} - n_{j,\sigma_j}\right)\left(n_{i,\overline{\sigma_i}} - n_{j,\overline{\sigma_j}}\right) \end{aligned} \tag{2.46}$$

The simplification of $\mathcal{H}_N(t) - \mathcal{H}_{\widetilde{N}}(t)$ can not be calculated as quickly as the difference of base-energies. Primarily this is because of the number of terms in this calculation. Each of the three $\hat{\mathrm{V}}_{\text{Part A}}(N,t)$, $\hat{\mathrm{V}}_{\text{Part B}}(N,t)$ and $\hat{\mathrm{V}}_{\text{Part C}}(N,t)$ is a sum of 8 summands, as one can see in Equation 2.28. Each of those must be treated for for cases: ($\sigma_i = \uparrow \wedge \sigma_j = \uparrow$), ($\sigma_i = \uparrow \wedge \sigma_j = \downarrow$), ($\sigma_i = \downarrow \wedge \sigma_j = \uparrow$) and ($\sigma_i = \downarrow \wedge \sigma_j = \downarrow$). This makes 32 differently-shaped terms, or combined with the pre-factors 96 terms to fully write down.
Different notation could be used, to lessen the number of calculations, but this seemed unnecessarily complex.
A simpler solution was in fact, to let a python script generate all the final simplified analytical terms and output them as usable functions into a python script to include in the rest of the code. The generating script can be found here:
[10]: `/calculation-helpers/simplificationtermhelper.py`
and the output with the analytically fully optimized calculations in a unified format here: [10]: `/computation-scripts/analyticalcalcfunctions.py`.
As this needs to be run quite often, it is useful to reduce the number of calculations as much as possible and avoid running unnecessary calculations on every iteration (e.g. $n_{l,\uparrow} + 1 - (1 + n_{l,\uparrow})$ which is definitely 0, no matter the value of $n_{l,\uparrow}$). For this reason, the terms were pre-optimized

as much as possible, by trying to pre-evaluate to definite zeros or constants terms. This was performed automatically with the python package *SymPy* [14].
More details on how this was calculated can be found in *section 3.2 Python Scrips implementing Analytical Calculations* and further resources, that that section redirects towards.

Flipping-only modification

In this simplification, the maximum difference that can happen, is that a *flipping* event occurs on site i and spin σ_i.
The flipping results in the values for $\widetilde{n}_{l,\sigma}$ are given in Equation 2.47.

$$\widetilde{n}_{l,\sigma} = \begin{cases} (1 - n_{i,\sigma_i}) & : l = i \wedge \sigma = \sigma_i \\ n_{l,\sigma} & : \text{else} \end{cases} \tag{2.47}$$

This simplifies the energy difference $E_0(N) - E_0(\widetilde{N})$ like equation Equation 2.48 presents.

$$\begin{aligned} E_0(N) - E_0(\widetilde{N}) &= U \sum_l n_{l,\downarrow} n_{l,\uparrow} - U \sum_l \widetilde{n}_{l,\downarrow} \widetilde{n}_{l,\uparrow} + \sum_{l,\sigma} \varepsilon_l n_{l,\sigma} - \sum_{l,\sigma} \varepsilon_l \widetilde{n}_{l,\sigma} \\ &\overset{2.47}{=} \varepsilon_i \left(2 n_{i,\sigma_i} - 1\right) + U \cdot \begin{cases} n_{i,\downarrow}(2 n_{i,\uparrow} - 1) & : \sigma_i = \uparrow \\ n_{i,\uparrow}(2 n_{i,\downarrow} - 1) & : \sigma_i = \downarrow \end{cases} \end{aligned} \tag{2.48}$$

Like for the hopping problem, the calculation of $\mathcal{H}_N(t) - \mathcal{H}_{\widetilde{N}}(t)$ is more involved in the number of terms.
More details on how this was calculated can be found in *section 3.2 Python Scrips implementing Analytical Calculations* and further resources, that that section redirects towards.

All presented simplifications to the Hamiltonian and corresponding MC-modification strategies in this section are implemented as options that are selectable in the control-script [10]: `/computation-scripts/script.py`.

2.2 Handling Operator Trees

This theoretical part is not relevant for the overall understanding of the physical calculation that was performed. It only serves as an introductory helper for the ones that are interested in the inner workings of the tools that were developed to calculate the analytical expressions that are necessary to generate meaningful physical simulation results.
The tool *Math-Manipulator* was developed as a tool to aid doing calculations that are repetitive and impractical to do by hand, but still complex enough in the sense that they can not be simply put into a computer algebra system (at least not with out extra effort and knowledge).
Thereby the operation of the tool can be broken down into three regimes:

1. The *lexing*/*parsing* of a string input (the equation/term that should be handled as a text input from the keyboard) into a structural *operator-tree*
2. The *modification* of existing operator-trees by applying algorithms to reshape parts of the operator tree into a simplified/preferred form
3. The *rendering*/*displaying* of the operator tree to give direct visual feedback, user-interface and export to LaTeX

The first two are touched upon in the following sections. The third comparably easily implemented as the stored structure allows for direct conversion to hierarchical *LaTeX* code, which lets the established rendering engine take care of all the complexity of rendering mathematical formulae.

2.2.1 Operator Tree Structures and Parsing

The problem of *parsing* text into relevant and specified structures and performing operations on them is one of the most fundamental problems in computer science. Many problems in computer use can be broken down to fit such a description.
Prime example for a physicist could be the source-code of this very document, written in LaTeX, or the instruction in the Python-script that runs the simulations. Both are just text, that needs to be parsed, dissected into relevant parts, re-arranged in computer-memory and connected to instructions, that make the computer form the simulation or output document.
The problem is known as first *lexing* (sectioning a continuous stream of *characters*/*symbols* into *words* with a different possible properties). Directly after this, one needs to *parse* these words to detect structures and extra information, derived from this structure (like e.g. parentheses, that change the order of operations for some operators).
The fine-details of such a process can vary wildly from application to application. General overview and guidelines to how such a process may be designed, could be found in sources like [15].
In general, any mathematical equation can be represented as a *tree* of operators. So in this application, a stream of characters needs to be lexed into mathematical operators, numbers and supplementary symbols. Then these symbols need to be parsed into an operator tree, following the *grammar* on mathematical notation and extra (convenience) features one desires/requires.
Why the act of converting a character stream into a tree is not quite straight forward and why the tree is the correct data-structure to represent terms, can be seen on the simple example presented in Figure 2.2
It shows that depending on the notation (here examples shown for *postfix*, *prefix* and *infix*), unique operator-tree can have multiple valid notations. General math notation uses mainly infix notation, however some operators like faculty are postfix and functions are a type of infix. This coupled with special/convenience features, like multiplications that could be omitted or not, implied zeros, brackets and many more, make this an interesting and challenging problem to tackle.

	Left Example		Right Example	
Infix	1 * 4 + 2 !		4 * (3 - 2)	Infix
Postfix	2 ! 1 4 * +		4 3 2 - *	Postfix
Prefix	+ * 1 4 ! 2		* 4 - 3 2	Prefix

Figure 2.2: Display of two operator-trees for two different mathematical expressions. The nodes in the branches of the tree hold operators, while the leaves generally hold numbers. These examples could be expanded through the introduction of variables and many more features that are implemented in Math-Manipulator. But the general structure is the same as in the implemented program.

2.2.2 Outlook: Modifying Operator Trees to get "simplified" Results

Doing math on paper is mostly two things. Trying and thinking what to do and executing steps by writing them down.
The problem of "what to do" can become basically of unlimited difficulty. There are programs that attempt to provide complex computations like integral evaluation or something similar. Most of them are purpose-built calculation machines that require at the minimum some expertise on how to operate them. This program is not supposed to take any of this load, nor could it. The program Math-Manipulator is designed to help with the "writing down" and "executing steps" part of the calculation.

- How to make the denominator of the fraction real-valued?
- How to expand a product of sums to a sums of product?
- How to remove al zero- and canceling terms from a sum?
- How to rename this one variable and copy all the rest of the equation?

All these problems are computationally easy, just tedious and error-prone if done by hand for many terms. This is the problem the program is designed to solve. A helper that mimics how math would be done by hand, but with the extra "Expand these 50 terms, cancel and order for me" built in.
This requires many small algorithms and functions that are purpose built for every operator. Operations can be simple - like returning the numerical value of a constant 5.1. They become more difficult, when recursion over children may be required - computing a percent-sing operator 5.1. And can even become gigantic algorithms like for the general distribution operation 5.2.
This report is not a place to present them in any shape or form, but just give an outlook, where some of the very interesting implementations can be found if one is interested.

3 Implementation and Numerical Calculations

The report will not be able to go in-depth in terms of the programming or results from the code. For that purpose it is suggested to look at the master thesis and the accompanying Project Work Presentation that can be both found in the same repository [1].

3.1 Math-Manipulator

3.1.1 Math-Manipulator Implementation

The Math-Manipulator project was implemented in the programming Language *Typescript* and fully compiles down to an application that can be natively used in every modern browser. More information on the project can be found on its GitHub-page [16]. The application is hosted on GitHub-pages, so no installation knowledge is required to try it. The program includes an interactive-playground tutorial and is also available as an extension in the code-editor *Visual Studio Code* as an extension.

3.1.2 Math-Manipulator Results

Many repetitive calculations have not been performed by hand, but with the Math-Manipulator tool. To be able to check the derivations, that were presented in this report, one can look at the stored calculation files in the calculation-repository [17].

3.2 Python Scrips implementing Analytical Calculations

A missing part of the report is, how and what results were obtained that describe the $\mathcal{O}(\#(\text{nearest neighbors}))$ calculation of $\mathcal{H}_{\text{eff}}(\widetilde{N},t)-\mathcal{H}_{\text{eff}}(N,t)$ needed in *subsection 2.1.6 Analytical Simplifications for Special Cases*.
Like previously hinted, they consist of many repetitive terms. Therefore listing them here would just unnecessarily bloat the report and serve no further purpose. It is better to look at the generating script [10]: `/calculation-helpers/simplificationtermhelper.py` and the generated output script if one desires a specific term from it:
[10]: `/computation-scripts/analyticalcalcfunctions.py`.
The algorithm that the script operates on, is explained in the accompanying Project Work Presentation [1]. The validity of the analytical simplifications was tested by direct comparison to the non-simplified terms. The checking script can be verified here:
[10]: `/computation-scripts/comparehamiltonians.py`.

4 Bibliography

[1] J. Kell, *LaTeX sources for this report, the thesis and presentations*, (2024) https://github.com/jonas-kell/master-thesis-documents.

[2] J. Hubbard, "Electron correlations in narrow energy bands", *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 10.1098/rspa.1963.0204 (1963).

[3] F. Tennie, V. Vedral, and C. Schilling, "Universal upper bounds on the Bose-Einstein condensate and the Hubbard star", *Physical Review B* **96**, 064502 (2017).

[4] H. A. Gersch and G. C. Knollman, "Quantum Cell Model for Bosons", *Phys. Rev.* **129**, 959–967 (1963).

[5] F. Schwabl, *Quantenmechanik (QM I)*, Springer-Lehrbuch (Springer, Berlin, Heidelberg, 2007).

[6] F. Schwabl, *Quantenmechanik für Fortgeschrittene (QM II)*, Springer-Lehrbuch (Springer, Berlin, Heidelberg, 2008).

[7] Y. Zheng, "Exact solution of $U \to \infty$ 1D Hubbard model in electric field", *Physics Letters A* **394**, 127112 (2021).

[8] R. Verdel, M. Schmitt, Y.-P. Huang, P. Karpov, and M. Heyl, "Variational classical networks for dynamics in interacting quantum matter", *Physical Review B* **103**, 165103 (2021).

[9] R. Kubo, "Generalized Cumulant Expansion Method", *Journal of the Physical Society of Japan* **17**, 1100–1120 (1962).

[10] J. Kell, *Thesis Code implementation and Reference*, (2024) https://github.com/jonas-kell/master-thesis-code.

[11] M. Schmitt and M. Reh, "jVMC: Versatile and performant variational Monte Carlo leveraging automated differentiation and GPU acceleration", *SciPost Physics Codebases*, 10.21468/scipostphyscodeb.2 (2022).

[12] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal, "Quantum Monte Carlo simulations of solids", *Rev. Mod. Phys.* **73**, 33–83 (2001).

[13] S. Chib and E. Greenberg, "Understanding the Metropolis-Hastings Algorithm", *The American Statistician* **49**, 327–335 (1995).

[14] *SymPy 1.12.1rc1 documentation*, https://docs.sympy.org/latest/index.html (visited on 04/12/2024).

[15] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, 2nd ed. (2006).

[16] J. Kell, *Math-Manipulator Tool - Info and Repository*, (2024) https://github.com/jonas-kell/math-manipulator.

[17] J. Kell, *Supplementary Calculations Perfomed with Math-Manipulator*, (2024) https://github.com/jonas-kell/master-thesis-mm-calculations.

[18] M. Schmitt and M. Heyl, "Quantum dynamics in transverse-field Ising models from classical networks", *SciPost Phys.* **4**, 013 (2018).

[19] M. Schulz, C. A. Hooley, R. Moessner, and F. Pollmann, "Stark Many-Body Localization", *Phys. Rev. Lett.* **122**, 040606 (2019).

5 Appendix

5.1 Calculate Numerical Values

[16]: /src/functions/implementedOperators.ts

```
        });

        res = res.replaceAll("#", "\\#");
        return res;
    }

    static getNumberOfIntendedChildren(config: OperatorConfig,
    ↪ trigger: string) {
        let matches = DefinedMacro.getOutputString(config,
    ↪ trigger).matchAll(DefinedMacro.ARG_REGEX) /* c8 ignore next */
    ↪ || [];
```

```
    }
}

export class EmptyArgument extends Operator {
    constructor(config: OperatorConfig) {
        super(config, OperatorType.EmptyArgument, "{", "", "}", [],
    ↪ "");
    }
}

export class Equals extends Operator {
    constructor(config: OperatorConfig) {
        super(config, OperatorType.Equals, "\\eq", "", "", [], "");
    ↪ // custom macro check
    }
}

export class NotEquals extends Operator {
    constructor(config: OperatorConfig) {
```

5.2 Distribute Operation

[16]: /src/functions/implementedOperators.ts

```
1254            ),
1255            new ComplexOperatorConstruct(
1256                config,
1257                constructContainerOrFirstChild(config,
 ↪  OperatorType.BracketedSum, newChildrenRealPullingOut),
1258                constructContainerOrFirstChild(config,
 ↪  OperatorType.BracketedSum, newChildrenImaginaryPullingOut)
1259            ),
1260        ];
1261    }
1262
1263    /**
1264     * Calculate the two cases, where a total minus was pulled out and
 ↪  where it wasn't
1265     * @returns [allChildrenPulledOutOddNumber: boolean,
 ↪  newOperatorAfterNotPullingOut: Operator,
 ↪  newOperatorAfterPullingOut: Operator]
1266     */
1267    function generalPullOutMinusHandler(
1268        config: OperatorConfig,
1269        childrenArrayArray: Operator[][]
1270    ): [boolean, Operator[][], Operator[][]] {
1271        let allChildrenPulledOutOddNumber = true;
1272        let newChildrenNotPullingOut = [] as Operator[][];
1273        let newChildrenPullingOut = [] as Operator[][];
1274
1275        childrenArrayArray.forEach((childArray) => {
1276            let newChildrenNotPullingOutInner = [] as Operator[];
1277            let newChildrenPullingOutInner = [] as Operator[];
1278
1279            childArray.forEach((child) => {
1280                if (implementsMinusPulloutManagement(child)) {
1281                    const [childEvenNumberMinusPulledOut,
 ↪  childResultingOperator] = child.minusCanBePulledOut();
1282
1283                    if (childEvenNumberMinusPulledOut) {
1284                        allChildrenPulledOutOddNumber = false;
1285                    }
```

```
1286
1287                if (childEvenNumberMinusPulledOut) {
1288
1289                    newChildrenPullingOutInner.push(new
    ↪ Negation(config, childResultingOperator));
1290                } else {
1291                    newChildrenNotPullingOutInner.push(new
    ↪ Negation(config, childResultingOperator));
1292
    ↪ newChildrenPullingOutInner.push(childResultingOperator);
1293                }
1294            } else {
1295                allChildrenPulledOutOddNumber = false;
1296
1297                newChildrenNotPullingOutInner.push(child);
1298                newChildrenPullingOutInner.push(new Negation(config,
    ↪ child));
1299            }
1300        });
1301
1302      newChildrenNotPullingOut.push(newChildrenNotPullingOutInner);
```