Quantum implementation and simulation of the Hubbard model

Moreillon Gilles-Henry
Supervisors: Professor Giovanni De Micheli and Doctor Mathias Soeken

École Polytechnique Fédérale de Lausanne

January 31, 2025

Abstract

The Hubbard model is an excellent case study for quantum simulation due to its relatively simple definition and its ability to capture the essential physics of specific systems. As a result, it could become one of the first practical applications of fault-tolerant quantum computers for simulations of quantum systems. This significance led to many studies aimed at optimizing this model. A recent publication by Earl T. Campbell [1] introduces new ideas, notably Plaquette Trotterization. This new way of splitting the operator allows for more flexibility in lattice size compared to previous methods and shows an overall reduction in the number of non-Clifford gates. This report presents the theoretical and practical steps taken to implement this approach using Python and the Qiskit library along with some results from the resulting simulations. The code created for this implementation is available publicly on GitHub [2]. It serves as a proof of concept for plaquette Trotterization and provides a foundation for future practical tests on the Hubbard model.

Contents

Chapter 1

Introduction

1.1 Computational chemistry

Computational chemistry is a field that uses mathematical models and computational techniques to understand and predict the properties of materials using numerical simulations. This allows research on chemical reactions, electronic structures, or material behaviors that could not be done via direct experimental observation. The ability to accurately simulate these systems has significant implications for fields ranging from drug discovery to materials science and renewable energy technologies. This leads to the use of many different models, often tailor-made for specific classes of systems, using different approximations. Among these systems are strongly correlated electron systems, where electrons can no longer be considered as non-interacting entities, or described by an average potential. The correlation between the electrons creates quantum effects that cannot be ignored for accurate modeling. These strong correlations lead to complex behaviors such as magnetism, high-temperature superconductivity, and metal-insulator transitions, making their simulations both particularly difficult yet important. New classical methods have been developed to address these challenges.

There are two large families of methods used for such systems:

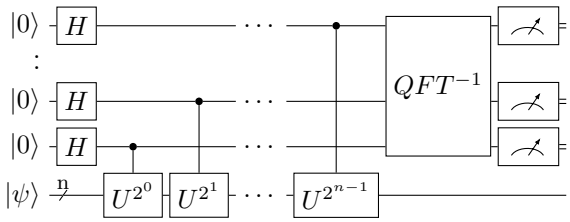
- Methods based on the Hartree-Fock method, where a set number of basis wave-functions N is used to approximate the many-body wave-function of the system and compute the associated energy. These methods, named Post-Hartree-Fock methods, add different correction terms and offer different accuracy and computational cost scaling. For example $\mathcal{O}(N^3)$ for the Hartree-Fock method and $\mathcal{O}(N^5)$ when using the second-order Møller-Plesset Perturbation Theory [3].

- Methods based on electron density, collectively known as density functional methods, share a scaling similar to the Hartree-Fock, though they often require system-specific adaptations for accurate computations. The most well-known among them is Density Functional Theory.

Both post-Hartree-Fock methods and Density Function Theory are still being developed, but the scaling of those methods makes large-scale simulation on classical computers out of reach. Solutions with acceptable precision can only be obtained on small-scale instances.

This is where quantum algorithms offer a novel approach to overcome the complexity of simulations with quantum effects. Quantum algorithms can show exponential speedups over classical methods for certain problems by using the fundamental effects of superposition and entanglement. With this, quantum computers could allow for more efficient simulations in computational chemistry and provide insights into fundamental properties of materials. However, current quantum hardware remains limited in many ways: coherence time, qubit count, error rate and noise. Because of these limitations, designing efficient algorithms is crucial to mitigate noise and ensure that useful computations can be performed on early fault-tolerant devices.

1.2 Energy computation with quantum computers



Showcase of a phase estimation algorithm [4]

Finding the ground state energy of a Hamiltonian is one of the most important problems that numerical simulation attempts to solve. Quantum computing offers another way of retrieving this information via quantum phase estimation algorithms.

These algorithms rely on the time-evolution operator of the Hamiltonian H, defined as $U(t) = e^{-itH}$, to estimate its phase. From this phase, it is possible to retrieve the eigenvalue of the Hamiltonian. However, given that H is generally not a simple diagonal matrix, the time-evolution operator e^{-itH} cannot be implemented exactly. Various approximation techniques have been proposed. One of which is Trotterization, introduced by Hale Trotter [5].

1.3 Trotterization

To approximate time-evolution operators, Trotterization will express H as a sequence of simpler unitaries that can be implemented efficiently. Like any approximation, Trotterization introduces errors, but it remains a practical candidate for simulating quantum systems on early fault-tolerant quantum devices. Consider $H = A + B$, where the operators A and B do not commute ($[A, B] = AB - BA \neq 0$). Then, the Trotter decomposition is given by:

$$e^{\frac{t(A+B)}{n}} = \lim_{n\to\infty}(e^{\frac{tA}{n}}e^{\frac{tB}{n}})^n \tag{1.1}$$

This definition is generalized to higher-order approximations via the Suzuki-Trotter expansion [6]. While this can be extended to arbitrary orders, the second-order Suzuki-Trotter expansion is the most commonly used. This is expressed as:

$$e^{\frac{t(A+B)}{n}} = \lim_{n\to\infty}(e^{\frac{tA}{2n}}e^{\frac{tB}{n}}e^{\frac{tA}{2n}})^n \tag{1.2}$$

In a real implementation, the value for n is finite.

$$e^{t(A+B)} \approx \sum_{k=1}^{n}(e^{\frac{tA}{2k}}e^{\frac{tB}{k}}e^{\frac{tA}{2k}})^k \tag{1.3}$$

In definition (1.3), the number n will be referred to as the number of Trotter steps. The introduced Trotterization error quantifies the deviation between the exact evolution and its Trotterized approximation. Computing its value analytically is challenging as it is based on an infinite series of nested commutators of A and B (coming from the Baker–Campbell–Hausdorff formula [7]). Instead, one typically relies on upper bounds for the Trotterization error and defines W:

$$||e^{t(A+B)} - e^{\frac{tA}{2k}}e^{\frac{tB}{k}}e^{\frac{tA}{2k}}|| \leq Wt^3 \tag{1.4}$$

The value of W depends on the tightness of the upper bound but also on the choice of operators that partition H.

In the context of quantum phase estimation, the total number of repeated unitary operators to be applied depends on W (**Appendix F** in [1]). A smaller Trotterization error allows for fewer repetitions, reducing

the overall number of gates required and improving the feasibility of quantum phase estimation on near-term quantum hardware.

In conclusion, optimizing Trotterization for the Hubbard model involves a trade-off between minimizing the Trotterization error bound W and reducing the cost of implementing each Trotter step. Achieving an optimal balance between these factors is crucial for improving the efficiency of quantum simulations and making them viable for early fault-tolerant quantum computing.

Chapter 2

Background and state-of-the-art

2.1 Quantum phase estimation algorithms

2.1.1 Phase estimation

To retrieve the eigenvalues of the Hamiltonian H, quantum phase estimation can be used. Letting $|\psi_n\rangle$ be an eigenstate of H with associated eigenvalue λ_n.

$$H|\psi_n\rangle = \lambda_n|\psi_n\rangle \qquad \Rightarrow \qquad e^{-itH}|\psi_n\rangle = e^{-it\lambda_n}|\psi_n\rangle = e^{-i2\pi\phi_n}|\psi_n\rangle \qquad \text{with } \phi_n \in [0,1[\tag{2.1}$$

Phase estimation algorithms rely on the time-evolution operator controlled by an ancilla qubit such that:

$$|0\rangle|\psi_n\rangle + e^{-itH}|1\rangle|\psi_n\rangle = |0\rangle|\psi_n\rangle + e^{-it\lambda_n}|1\rangle|\psi_n\rangle = |0\rangle|\psi_n\rangle + e^{-i2\pi\phi_n}|1\rangle|\psi_n\rangle \tag{2.2}$$

Using controlled time-evolution, phase estimation algorithms obtains values of j_i that approximate ϕ_n as a fractional binary expansion:

$$\phi_n = \frac{j_0}{2} + \frac{j_1}{4} + \frac{j_2}{8} + ...$$

Without going into the exact steps occurring in such algorithms, some specific details related to implementation will be presented.

2.1.2 Time of the unitary

For simplicity, consider $t = 1$. Inspection of (2.1) reveals a loss of information if the range of eigenvalues exceeds 2π. If $\lambda_i = \lambda_j + 2k\pi$ with $k \in \mathbb{Z}$, both would share the same phases, $\phi_i = \phi_j$. Additionally, the information associated with the eigenvalue cannot be fully retrieved because measuring a phase ϕ_i only determines that $\lambda_i = 2\pi(\phi_i + k)$. Following the reasoning of **Section 5** in [8], t must be chosen such that $e^{-it\lambda_n}$ is unique for each distinct λ_n. Depending on assumptions about the sign or symmetry of the eigenvalues, different boundaries can be established. The most general condition that ensures $[t\lambda_{\max} - t\lambda_{\min}] \in) - \pi, \pi($ is given by:

$$t \times \max[|\lambda_{max}|, |\lambda_{min}|] < \pi \qquad \Rightarrow \qquad t < \frac{\pi}{\max[|\lambda_{max}|, |\lambda_{min}|]} \tag{2.3}$$

This ensures that the value most distant to 0 is encoded as a phase close to $\pm\pi$ (depending on its sign), and all the others will be in $[-\pi, \pi]$. Setting the optimal t requires knowing $\lambda_{\max}$ and $\lambda_{\min}$, which would mean the energy problem is already solved.

For this reason, t is chosen to be arbitrarily small, assuming that it satisfies (2.3). However, reducing t too much comes with drawbacks. The interval $[t\lambda_{\max} - t\lambda_{\min}]$ becomes smaller, increasing the required precision in phase estimation.. Assuming that t satisfies the condition in (2.3), the following relation can be written:

$$t\lambda_n = \begin{cases} 2\pi\phi_n & \text{if } \phi_n \in [0, 0.5[\\ 2\pi(\phi_n - 1) & \text{if } \phi_n \in]0.5, 1] \end{cases} \tag{2.4}$$

From this, errors on ϕ_n are shown to be amplified as t approaches 0.

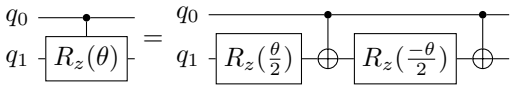
The choice of t is also influenced by the error bounds of the Trotterization, as seen in (1.4). This usually leads to a stricter bound on t. Therefore, t is set as a parameter that at minimum satisfies (2.3), ensuring a bijection from phase to energy. Other bounds related to precision and optimization are beyond the scope of this project.

2.1.3 Uniformly-controlled rotation gates

Any phase estimation algorithm trying to retrieve the phase of a unitary U requires some form of control over that unitary. One could directly apply control to every element of the unitary using a general method, but this significantly increases the number of gates. In general, the transformation of a gate into its controlled version is not trivial.

Two considerations from **Chapter V, Section A** in [9] ensure that the unitary and its controlled version differ only by a limited number of Clifford gates.

The first consideration is that only the Rz-gates, which represent arbitrary Z-rotations, need to be controlled. These gates alone encode the relevant information regarding the phase shift that will be measured. This already greatly reduces the gap between the unitary and the controlled-unitary. In most contexts, controlling an Rz-gate implies doubling the total number of Rz-gates.



(2.5)

Figure 2.1: Basic implementation of a controlled Rz-gate

The second consideration seeks to avoid this doubling of Rz-gates. This is only valid when controlled Rz-gates are used in the context of quantum phase estimation. In such case, the notion of 'uniformly controlled rotation' is introduced in [9]. Here, the method of controlling the unitary during phase estimation is modified. Instead of applying the identity if the ancilla is $|0\rangle$ and $U(t)$ if the ancilla is $|1\rangle$, the approach applies $U(t/2)$ for $|0\rangle$ and $U(-t/2)$ for $|1\rangle$. This is done via the uniformly controlled rotation.

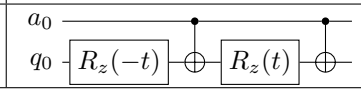
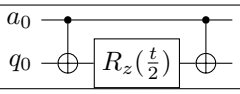
	Controlled	**Uniformly Controlled**
State	$\|0\rangle\|\psi_n\rangle + U(t)\|1\rangle\|\psi_n\rangle$	$U(\frac{t}{2})\|0\rangle\|\psi_n\rangle + U(-\frac{t}{2})\|1\rangle\|\psi_n\rangle$
Rotation implementation		

Table 2.1: Comparison of controlled and uniformly controlled rotations.

With this method, the number of Rz-gates is not doubled. Additionally, the value of t is halved, reducing the overall number of Trotter steps.

2.2 Hubbard model

The Hubbard model, first introduced in 1963 by John Hubbard [10], is used in the study of strongly correlated electron systems. Despite its apparent simplicity, consisting of only two types of interactions, it captures the essential physics of electrons in strongly correlated systems, describing phenomena such as magnetism, metal-insulator transitions, and high-temperature superconductivity. Its versatility and relative simplicity

make it a great starting point for both theoretical exploration and computational simulation, and it has seen many novel ideas and improvements over the years.

The Hubbard model considers a two-dimensional lattice. For simplicity, it is set as a square lattice with a side length of L sites with periodic boundary condition. Each site can host one spin-up and one spin-down electron, hence its association with strongly correlated electron systems, where electron shells are not completely filled.

In the simplest form, the Hubbard model Hamiltonian is expressed as its two main terms:

$$H = H_i + H_h \tag{2.6}$$

These are the two terms representing different interactions. H_h is the hopping term, also called 'off-site interaction'. It describes the interactions between electrons of the same spin but on different lattice sites. H_i is the interaction term, also called 'on-site interaction'. It describes the interactions between electrons of different spins on the same site.

The usual, unshifted, Hamiltonian H_u used for such 'on-site interaction', is defined as:

$$H_u = u \sum_{i}^{L^2} \hat{n}_{i\uparrow} \hat{n}_{i\downarrow} \tag{2.7}$$

As discussed in **Section I** from [1], introducing a potential shift for this term has great benefits. The constant energy shift that this creates can be removed classically in post-processing. The advantages of such a shift will be demonstrated when discussing the implementation later in this report. The relation between the unshifted interaction Hamiltonian H_u and the shifted interaction Hamiltonian H_i is defined as:

$$H_i = H_u - \frac{u}{2}\hat{N} + \frac{u}{4}\mathbb{1} \tag{2.8}$$

Where $\hat{N}$ is the total electron number operator. Since all the terms introduced commute with each other, they share a common eigenbasis. This means that there is no effect on the eigenstates of the model. Using the defined relation between shifted and unshifted interactions (2.8), the shifted interaction Hamiltonian H_i is defined as:

$$H_i = \frac{u}{4} \sum_{i}^{L^2} \hat{z}_{i\uparrow} \hat{z}_{i\downarrow} \tag{2.9}$$

This interaction Hamiltonian uses $\hat{z} := 2\hat{n} - \mathbb{1}$, where $\hat{n} = a^\dagger a$ is the number operator. Hence u is the parameter that influences the strength of the on-site interaction.

The hopping term H_h is defined as:

$$H_h = \sum_{\sigma \in \uparrow,\downarrow} \sum_{i \neq j}^{L^2} R_{i,j} (a^\dagger_{i,\sigma} a_{j,\sigma}) \tag{2.10}$$

Considering that i and j are lattice indices, they both range from 1 to L^2. This operator models the possible tunneling of an electron from site i to site j. Note that these are spin-consistent. $R_{i,j}$ defines the level of complexity of the model. In all cases, it must be that $R_{i,j} = R^*_{j,i}$ for hermiticity of the operator, and also $R_{i,i} = 0$. The simplest scenario, and the one studied here, considers only nearest-neighbor interactions. This means that $R_{i,j} = \tau$ only when i and j are nearest neighbors (with period boundary conditions). Any other case sees $R_{i,j} = 0$, reflecting no interaction. Hence τ is the parameter that influences the strength of the off-site interaction.

The ratio of the on-site and off-site interaction strength, $\frac{u}{\tau}$, leads to different situations. For this model, it is chosen to range between 4 and 12. Within this range, a wide difference of results are observed where none of the effects overtake the other and both need to be considered to get a meaningful result. This is shown in more detail in **Varying the coupling** from [11].

2.3 Trotterization of the Hubbard model

2.3.1 Split operator

The core idea of the split-operator algorithm is to decompose the Hamiltonian into separate terms that can be simulated more efficiently on their own. Once the split is done, the overall Hamiltonian can be

approximated, for example, using the Suzuki-Trotter approximation introduced earlier (1.2). From the definition of the Hubbard Hamiltonian (2.6), the split comes naturally since the terms are already separated into the interaction term H_i and the hopping term H_h. This leads to the first Trotterization, achieved using a second-order Suzuki-Trotter approximation.

$$e^{itH} \approx e^{i\frac{t}{2}H_i} e^{itH_h} e^{i\frac{t}{2}H_i} \tag{2.11}$$

As mentioned in (1.4), the error that this approximation introduces is bounded by Wt^3. For this Trotterization, an upper bound for W is provided in **Appendix C** in [1].

Since the hopping term H_h is the sum of all nearest-neighbor interactions, it presents one of the biggest challenges in the Hubbard model. Direct diagonalization e^{itH_h} can be achieved using the general method of Givens rotations. Givens rotations are unitary transformations used to cancel out specific elements of a matrix while preserving its orthogonality. In this context, they can systematically transform a Hamiltonian into diagonal form by applying a sequence of two-level rotations. For any $N \times N$ matrix, this can be done using at most $\mathcal{O}(N^2)$ Givens rotations. In the case of a hopping Hamiltonian, it requires only $\mathcal{O}(L^4)$ Givens rotations thanks to the sparse and symmetrical nature of nearest neighbors interactions [12]. Even with this, $\mathcal{O}(L^4)$ remains too large in practice.

This opens multiple paths for optimization:

- Find better ways to diagonalize e^{itH_h}. This is what is done by Kivlichan *et al.* [13]. They use Fermionic Fast Fourier Transform (FFFT). While this reduces the cost, it can only be applied to lattices with side lengths that are powers of 2. This method is called Split Operator FFFT (SO-FFFT).
- Further decompose H_h. There are many arbitrary ways to partition the Hopping term. An example, from [14], shows the hopping term split into vertical and horizontal interactions.

Building on the ideas of further decomposition and the use of FFFT, Earl T. Campbell [1] introduced a new decomposition of the hopping term H_h. This method works on any lattice size and reduces the number of non-Clifford gates needed. This is the base of plaquette Trotterization. Improvements were also added to the SO-FFFT method, resulting in the SO-FFFT+ method.

2.3.2 Plaquette Trotterization

Plaquette Trotterization partitions the off-site interaction. It defines smaller 4-sites squares, called plaquettes. Those are split in two categories, pink and gold, such that each edge between two sites belongs only to a single color.

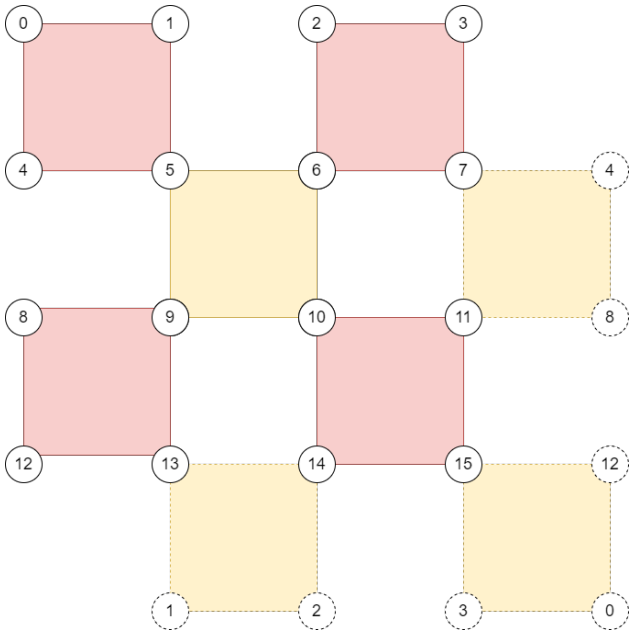


Figure 2.2: Plaquette decomposition on a lattice with $L = 4$ with periodic boundary conditions

With such decomposition, the hopping Hamiltonian becomes:

$$H_h = H_h^p + H_h^g \tag{2.12}$$

Here, H_h^p is the interaction for all pink plaquette edges, and H_h^g is the interaction for all gold plaquette edges. While H_h^p and H_h^g do not commute, they can individually be decomposed as the sum of the independent action on each plaquette of their color.

This is expressed as:

$$H_h^c = \sum_{\sigma \in \{\uparrow,\downarrow\}} \sum_{k=1}^{\frac{L^2}{4}} H_h^{c(k,\sigma)} \qquad \text{where } c \in \{p, g\} \tag{2.13}$$

Here, $H_h^{c(k,\sigma)}$ considers the nearest neighbor interaction taking place on the isolated plaquette of color c and spin σ. Since a square lattice with even side length is partitioned by $L^2/4$ plaquettes for each spin, The index k, used to count all plaquettes, ranges from 1 to $L^2/4$.

Since within a set color c, all $H_h^{c(k,\sigma)}$ act on different and independent elements, the expression for time evolution is expressed without any additional Trotterization error:

$$e^{itH_h^c} = \prod_{\sigma \in \{\uparrow,\downarrow\}} \prod_{k=1}^{\frac{L^2}{4}} e^{itH_h^{c(k,\sigma)}} \qquad \text{where } c \in \{p, g\} \tag{2.14}$$

This does not imply that the plaquette decomposition introduces no Trotterization error. Trotterization error still arises when considering the overall partitioning into pink and gold, which gives:

$$e^{itH_h} \approx e^{itH_h^g/2} e^{itH_h^p} e^{itH_h^g/2} \tag{2.15}$$

With the additional error introduced by plaquette Trotterization, the overall error is upper-bounded by $W_{PLAQ}t^3$. The value and computation of this upper bound can be found in **Appendix D** in [1]. **Table I** in [1] shows the value of this upper bound compared to improved split operator method, SO-FFFT+, and the Plaquette decomposition. For example, on a 8×8 lattice, going from SO-FFFT+ to Plaquette brings the value of W from $3.5 \cdot 10^2$ to $5.3 \cdot 10^2$. But it also significantly reduces the total number of non-Clifford gates, going from $1.7 \cdot 10^3$ to $7.7 \cdot 10^2$ for the number of T-gates for example.

T-gates and arbitrary Z-rotation are the main focus of optimization as these non-Clifford gates represent a significant challenge to implement for fault-tolerant quantum computing. While the increase in Trotterization error would result in more calls to the Trotter step during phase estimation, thereby increasing the overall cost, Plaquette Trotterization can still be beneficial. It may significantly reduce the number of T-gates and arbitrary Z-rotations in a single Trotter step. If the reduction in gate cost compensates for the additional Trotter step calls, the overall implementation may become more efficient. When considering Plaquette Trotterization, many of the arbitrary Z-rotations share the same angle and are applied in parallel, which means it can efficiently use a technique called Hamming weight phasing.

Hamming weight phasing involves introducing additional ancilla qubits that encode the collective effect of multiple rotations. Instead of applying each arbitrary Z-rotation individually, the technique allows these rotations to be executed simultaneously when they share the same angle, leading to a substantial reduction in the overall gate count [15]. Previous decompositions could also use Hamming weight phasing to some extent, but with Plaquette Trotterization, each application of $e^{itH_H^p}$ and $e^{itH_H^g}$ involves arbitrary Z-rotations that share the same angle and can be applied in parallel, making it an ideal scenario for using Hamming weight phasing.

2.4 Project contribution

This project will first expand on the definition of Trotter steps to enable the implementation of the Hubbard model as a simulated quantum circuit, incorporating the previously discussed Plaquette Trotterization. Once implemented, this will serve as a solid proof of concept, demonstrating the feasibility of these improvements and establishing a framework for further investigations into both Plaquette Trotterization and the simulation of the Hubbard model.

Chapter 3

Considerations for implementation

3.1 Theoretical consideration aimed to implementation

This section is greatly inspired by Appendix E.1 in [1]. It will be more explicit in some aspects that were proven challenging, while adding some clarification needed for implementation. Importantly, the following definition will include Trotter steps, which were left out of the discussion in the source as the focus was on resource estimation.

3.1.1 Overall form for the time evolution

Using the most direct definition of the Hubbard model, given in (2.6), and applying a n step second-order Suzuki-Trotter decomposition ((1.2)) over the 3-term Hamiltonian, gives a first definition of the time evolution under the Hubbard model.

$$e^{it(H_i+H_h)} = e^{it(H_i+H_h^p+H_h^g)} \approx (e^{itH_i/2n}e^{itH_h/n}e^{itH_i/2n})^n \tag{3.1}$$

$$\approx (e^{itH_i/2n}e^{itH_h^p/2n}e^{itH_h^g/n}e^{itH_h^p/2n}e^{itH_i/2n})^n \tag{3.2}$$

$$\approx e^{itH_i/2n}(e^{itH_h^p/2n}e^{itH_h^g/n}e^{itH_h^p/2n}e^{itH_i/n})^n e^{-itH_i/2n} \tag{3.3}$$

The last step is obtained by merging the last and first term of a repetition, reducing the number of repeated terms. This form, (3.3), gives a good understanding of the direction to take. But with computational implementation in mind, it can still be refined.

3.1.2 Implemented form of the time evolution

Motivated by computational efficiency, the previous equation can be modified to reach a form more suited for simulations. The first step is to define what is meant by *merging the last and first terms of a repetition*, which was used to justify the previous form (3.3). For clarity, two placeholder operators will be used.

$$\begin{aligned}
[e^{A/2n}e^{B/n}e^{A/2n}]^n &= e^{A/2n}e^{B/n}\underbrace{e^{A/2n}e^{A/2n}}e^{B/n}e^{A/2n}...e^{A/2n}e^{B/n}\underbrace{e^{A/2n}e^{A/2n}}e^{B/n}e^{A/2n} \\
&= e^{A/2n}e^{B/n}\underbrace{e^{A/n}}e^{B/n}\underbrace{e^{A/n}}...e^{B/n}\underbrace{e^{A/n}}e^{B/n}e^{A/2n} \\
&= e^{A/2n}[e^{B/n}e^{A/n}]^n e^{-A/2n} && (3.4) \\
&= e^{A/2n}[e^{B/n}e^{A/n}]^{n-1}e^{B/n}e^{A/2n} && (3.5)
\end{aligned}$$

While (3.4) is used in **Appendix E** in [1] and in (3.3) for a more concise notation, it uses unnecessary operations. Instead, in (3.5), the last iteration of the decomposition is moved outside the bracket to avoid having to compensate with an additional negative application of $e^{-A/2n}$. This reduces the number of operations from $[2n+2]$ to $[2n+1]$.

Using this, (3.3) can be further refined. At the same time, instead of a 3-term second-order Suzuki-Trotter decomposition of the Hamiltonian, it can be considered as two subsequent 2-term decompositions. This lets each of these decompositions use an independent number of Trotter steps, granting more flexibility. The decomposition of $H = H_i + H_h$ is called the outer decomposition, while $H_h = H_h^p + H_h^g$ is called the inner decomposition.

$$
\begin{aligned}
e^{it(H_i+(H_h^p+H_h^g))} &\approx [e^{itH_i/2n} e^{it(H_h^p+H_h^g)/n} e^{itH_i/2n}]^n \\
&\approx [e^{itH_i/2n} (e^{itH_h^p/2nm} e^{itH_h^g/nm} e^{itH_h^p/2nm})^m e^{itH_i/2n}]^n && (3.6)\\
&\approx e^{itH_i/2n} ((e^{itH_h^p/2nm} e^{itH_h^g/nm} e^{itH_h^p/2nm})^m e^{itH_i/n})^n e^{-itH_i/2n} && (3.7)\\
&\approx e^{itH_i/2n} (e^{itH_h^p/2nm} (e^{itH_h^g/nm} e^{itH_h^p/nm})^m e^{-itH_h^p/2nm} e^{itH_i/n})^n e^{-itH_i/2n} && (3.8)
\end{aligned}
$$

The steps from (3.6) to (3.8) use different level of optimizations. For (3.7), the merging was done following (3.4) only on the outer decomposition, while in (3.8) this is done on both the inner and outer decomposition. This brings the number of operations to $[3mn + n + 2]$ for (3.7) and $[2mn + 3n + 2]$ for (3.8). In such case, the choice of the formula to implement to minimize the number of operations would rely on the value for n and m.

Using (3.5) for both the inner and outer decomposition leads to the final definition of the time evolution.

$$
\begin{aligned}
e^{it(H_i+(H_h^p+H_h^g))} &\approx [e^{itH_i/2n} (e^{itH_h^p/2nm} e^{itH_h^g/nm} e^{itH_h^p/2nm})^m e^{itH_i/2n}]^n \\
&\approx [e^{itH_i/2n} e^{itH_h^p/2nm} (e^{itH_h^g/nm} e^{itH_h^p/nm})^{(m-1)} e^{itH_h^g/nm} e^{itH_h^p/2nm} e^{itH_i/2n}]^n \\
&\approx e^{itH_i/2n} [e^{itH_h^p/2nm} (e^{itH_h^g/nm} e^{itH_h^p/nm})^{(m-1)} e^{itH_h^g/nm} e^{itH_h^p/2nm} e^{itH_i/n}]^{(n-1)} \\
&\qquad \times [e^{itH_h^p/2nm} (e^{itH_h^g/nm} e^{itH_h^p/nm})^{(m-1)} e^{itH_h^g/nm} e^{itH_h^p/2nm} e^{itH_i/2n}] && (3.9)
\end{aligned}
$$

This time evolution requires $[2mn + 2n + 1]$ operations, making this preferable for computational implementation, despite its apparent complexity.

3.1.3 The hopping Hamiltonian H_h and plaquette operator

One important element to consider is the Hamiltonian affecting a single plaquette. This is a 4-sites Hamiltonian as defined in (2.14) as $H_h^{c(k,\sigma)}$. The index k and σ are used to define the different plaquettes and their respective spins. For easier notation, these are ignored and the considerations are made over an arbitrary single plaquette operator K, with interaction strength τ.

Considering nearest neighbor interaction over a single plaquette, this is an 8-terms operator. Indexing of the 4-sites is considered a loop, going from 0 to 3. K is then defined as:

$$
K = a_0^\dagger a_1 + a_0^\dagger a_3 + a_1^\dagger a_0 + a_1^\dagger a_2 + a_2^\dagger a_1 + a_2^\dagger a_3 + a_3^\dagger a_0 + a_3^\dagger a_2 = \sum_{i,j} [R_{plaq}]_{i,j} a_i^\dagger a_j \tag{3.10}
$$

$$
\text{with } R_{\text{plaq}} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}. \tag{3.11}
$$

To greatly reduce the number of arbitrary rotations that will be necessary to evolve a system under K, R_{plaq} can be diagonalized. This shows only two non-trivial eigenvalues, $[-2, 2]$, with respective associated eigenvectors b and c:

$$
b = \frac{a_0 + a_1 + a_2 + a_3}{2} \tag{3.12}
$$

$$
c = \frac{a_0 - a_1 + a_2 - a_3}{2} \tag{3.13}
$$

Given this, K can be expressed using these terms:

$$
\begin{aligned}
2b^\dagger b - 2c^\dagger c &= \frac{1}{2}[(a_0 + a_1 + a_2 + a_3)^\dagger (a_0 + a_1 + a_2 + a_3) - (a_0 - a_1 + a_2 - a_3)^\dagger (a_0 - a_1 + a_2 - a_3)] && (3.14)\\
&= \frac{1}{2}[a_0^\dagger a_0 + a_0^\dagger a_1 + a_0^\dagger a_2 + a_0^\dagger a_3 + a_1^\dagger a_0 + a_1^\dagger a_1 + a_1^\dagger a_2 + a_1^\dagger a_3 + a_2^\dagger a_0 + a_2^\dagger a_1 + a_2^\dagger a_2 + a_2^\dagger a_3 + a_3^\dagger a_0 + a_3^\dagger a_1 + a_3^\dagger a_2 + a_3^\dagger a_3] \\
&+ \frac{1}{2}[-a_0^\dagger a_0 + a_0^\dagger a_1 - a_0^\dagger a_2 + a_0^\dagger a_3 + a_1^\dagger a_0 - a_1^\dagger a_1 + a_1^\dagger a_2 - a_1^\dagger a_3 - a_2^\dagger a_0 + a_2^\dagger a_1 - a_2^\dagger a_2 + a_2^\dagger a_3 + a_3^\dagger a_0 - a_3^\dagger a_1 + a_3^\dagger a_2 - a_3^\dagger a_2] \\
&= a_0^\dagger a_1 + a_0^\dagger a_3 + a_1^\dagger a_0 + a_1^\dagger a_2 + a_2^\dagger a_1 + a_2^\dagger a_3 + a_3^\dagger a_0 + a_3^\dagger a_2 = K && (3.15)
\end{aligned}
$$

This will help to reduce the number of rotations needed to apply each plaquette operator. Once in the eigenbasis of R_{plaq}, $e^{it\tau K}$ is diagonal and its effect can be expressed as a direct product of exponentials. Without degeneracy in the eigenvalues, this would require 4 terms ($e^{it\tau(\lambda_0+\lambda_1+\lambda_2+\lambda_3)}$). Here, as there are only 2 non-zero eigenvalue, this only requires 2 terms.

Diagonalizing R_{plaq} would yield the Givens rotations required to diagonalize $e^{it\tau K}$. This would require 3 Givens rotations and then 3 more to revert it. Here, the form of b and c hints at a more specific tool can be used to change the basis: the Fermionic Fourier Transform (FFT). This self-adjoint operator, $F_{i,j} = F_{i,j}^\dagger$, satisfies:

$$\begin{aligned} F_{i,j} a_i F_{i,j} &= \frac{a_i + a_j}{\sqrt{2}} \\ F_{i,j} a_j F_{i,j} &= \frac{a_i - a_j}{\sqrt{2}} \end{aligned} \tag{3.16}$$

With this, the change of basis can be done without the use of Given rotations. This is verified as one can compute the transformation:

$$\begin{aligned} V &= F_{2,0} F_{1,3} F_{1,2} \\ V a_1 V^\dagger &= b \\ V a_2 V^\dagger &= -c \end{aligned} \tag{3.17}$$

While remaining in the framework of creation and annihilation operators, the time evolution under K is defined as:

$$\begin{aligned} e^{it\tau K} &= e^{2it\tau(b^\dagger b - c^\dagger c)} \\ &= V e^{2it\tau(a_1^\dagger a_1 - a_2^\dagger a_2)} V^\dagger \\ &= F_{2,0} F_{1,3} F_{1,2} e^{2it\tau a_1^\dagger a_1} e^{-2it\tau a_2^\dagger a_2} F_{1,2} F_{1,3} F_{2,0} \end{aligned} \tag{3.18}$$

This expression is correct in terms of fermionic operators. It also implies the need for 6 applications of the F operator and 2 arbitrary rotations for each plaquette operator.

There is one more simplification that can be applied. It is more easily shown outside of the frame of purely fermionic operators, where some of the theoretical rigor and generality can be ignored. This abstraction is possible only when two adjacent terms are taken into consideration. In such cases, this can be considered as an isolated 2 qubits system with 4 number states. This givse that the operators can be realized as:

$$F = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \qquad \text{and} \qquad e^{2it\tau a_0^\dagger a_0} e^{-2it\tau a_1^\dagger a_1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{2it\tau} & 0 & 0 \\ 0 & 0 & e^{-2it\tau} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.19}$$

This can be computed as:

$$\begin{aligned} F e^{2it\tau a_0^\dagger a_0} e^{-2it\tau a_1^\dagger a_1} F &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{2it\tau} & 0 & 0 \\ 0 & 0 & e^{-2it\tau} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{e^{2it\tau}+e^{-2it\tau}}{2} & \frac{e^{2it\tau}-e^{-2it\tau}}{2} & 0 \\ 0 & \frac{e^{2it\tau}-e^{-2it\tau}}{2} & \frac{e^{2it\tau}+e^{-2it\tau}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(2t\tau) & i\sin(2t\tau) & 0 \\ 0 & i\sin(2t\tau) & \cos(2t\tau) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= e^{it\tau X\otimes X} e^{it\tau Y\otimes Y} \end{aligned} \tag{3.20}$$

With this final consideration, the total number of required operations to implement the plaquette operator is reduced. Only 4 applications of F and 2 arbitrary Z-rotations are needed for (3.18). This will be shown in more details when considering computational implementation.

3.2 Repeated unitary for phase estimation

Any phase estimation algorithm trying to retrieve the phase of a unitary U also requires different powers of the time-evolution unitary. This could be seen as the same unitary, where the time is multiplied by the factor r: $U(t)^r = U(rt)$, or as r repetition of the unitary with a set time. Increasing t implies increasing the number of Trotter steps, n, to keep the precision constant, and repeating the unitary also increases the length of the circuit. In most cases, repetition of the unitary while keeping t constant is preferred. This claim can be verified for a such circuit, first theoretically and then experimentally.

Here, directly applying r successive application of e^{-itH} as defined in (3.9) leads to $[2mnr + 2nr + r]$ operations. Considering the specifics and using (3.5) allow a final optimization.

$$\begin{aligned}
\left[e^{itH}\right]^r \approx \Big[& e^{itH_i/2n}\left[e^{itH_h^p/2nm}\left(e^{itH_h^g/nm}e^{itH_h^p/nm}\right)^{(m-1)}e^{itH_h^g/nm}e^{itH_h^p/2nm}e^{itH_i/n}\right]^{(n-1)} \\
& \left[e^{itH_h^p/2nm}\left(e^{itH_h^g/nm}e^{itH_h^p/nm}\right)^{(m-1)}e^{itH_h^g/nm}e^{itH_h^p/2nm}e^{itH_i/2n}\right]\Big]^r \\
\approx\ & e^{itH_i/2n}\left[e^{itH_h^p/2nm}\left(e^{itH_h^g/nm}e^{itH_h^p/nm}\right)^{(m-1)}e^{itH_h^g/nm}e^{itH_h^p/2nm}e^{itH_i/n}\right]^{n(r-1)} \\
& \left[e^{itH_h^p/2nm}\left(e^{itH_h^g/nm}e^{itH_h^p/nm}\right)^{(m-1)}e^{itH_h^g/nm}e^{itH_h^p/2nm}e^{itH_i/n}\right]^{(n-1)} \\
& \left[e^{itH_h^p/2nm}\left(e^{itH_h^g/nm}e^{itH_h^p/nm}\right)^{(m-1)}e^{itH_h^g/nm}e^{itH_h^p/2nm}e^{itH_i/2n}\right] \\
\approx\ & e^{itH_i/2n}\left[e^{itH_h^p/2nm}\left(e^{itH_h^g/nm}e^{itH_h^p/nm}\right)^{(m-1)}e^{itH_h^g/nm}e^{itH_h^p/2nm}e^{itH_i/n}\right]^{nr-1} \\
& \left[e^{itH_h^p/2nm}\left(e^{itH_h^g/nm}e^{itH_h^p/nm}\right)^{(m-1)}e^{itH_h^g/nm}e^{itH_h^p/2nm}e^{itH_i/2n}\right]
\end{aligned} \tag{3.21}$$

This repetition of the unitary now requires only $[2mnr + 2nr + 1]$. This is the final expression used for simulations. It requires the least amount of operations and offers the most flexibility. Setting $r = 1$ yields (3.9), and using $m = 1$ brings it to the more direct 3-term second-order decomposition seen in (3.3).

Regarding the choice of whether to apply $U(t)^r$ as repetitions with a fixed t, or as a single application with scaled t and n. Consider a setting where n_0 and m_0 yield results with a satisfying precision for a set t_0 without repetition. This is written as$U^r_{n_0,m_0}(t_0)$, a shorthand notation for (3.21). The relation between the operators is such that:

$$U^a_{n_0,m_0}(t_0) = U^1_{an_0,m_0}(at_0) \tag{3.22}$$

Hence, increasing r or increasing t and n accordingly are equivalent. Given that the implementation requires $[2mnr + 2nr + 1]$ operations, there is no preference between the two. This claim will also be supported with results.

Chapter 4

Computational implementation

The theoretical framework is now defined and explicit. This chapter will show the details of the computational implementation. The simulation of the quantum circuit is done in Python using the **Qiskit** library. To link creation and annihilation operator notation to circuits and matrices, **OpenFermions** is used. While all of the gates are directly available in the code, this section can explain some of the shortcuts and reasoning explicitly.

Indices are set row-wise, as shown in Figure 2.2, with spin-up and spin-down being separated. Spins can be seen as two different lattices. Computationally, these are two different registers, both with qubits ranging from 0 to $L^2 - 1$.

The code is publicly available on GitHub [2], and the source of every figure can be found in the *figures.ipybn* notebook. This allows for better comprehension, a direct link between this report and the code as well as a source of examples.

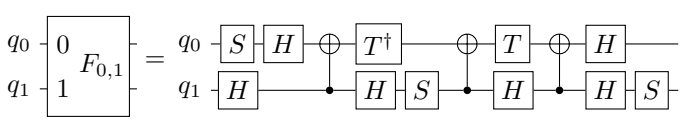
4.1 Building blocks

In this section, the building blocks of the circuit are explicitly defined. Each operator is defined as an independent quantum circuit. **Qiskit** allows to draw circuits as Matplotlib figures or to save the code required to include those in LaTeX using Qcircuit, both of which are obtainable from *figures.ipynb*. Note that the quantum circuit visible in this chapter are not directly obtained from this, but instead were re-drawn using the Yquant package, which allows for more flexibility.

4.1.1 Single plaquette operator, F and Fermionic Swap

As hinted at after (3.18) when discussing the general form of a single plaquette operator, there are some non-trivial factors to take into account when working with $F_{i,j}$, and more generally with the step from creation and annihilation operator to gate based implementation.

Starting by the F gate as defined in (3.19). There are many ways of creating such operator but the one used is based on the 2 qubits Fast Fermionic Fourier Transform (FFFT).



$$\equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{4.1}$$

Figure 4.1: Implementation of the F gate. From Figure 8 in [13]

The rotations shown in (3.20), respectively, one $X \otimes X$ and one $Y \otimes Y$, also need specification. This is done by rotation $X \otimes X$ to $Z \otimes \mathbb{I}$ using Clifford operation and then applying an arbitrary Z rotation and reverting the basis rotation. Same for $Y \otimes Y$ to $\mathbb{I} \otimes Z$.

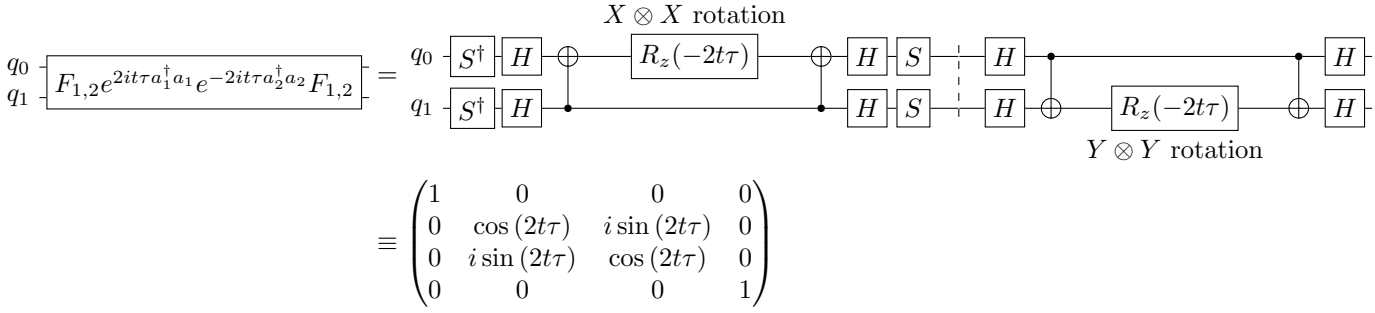
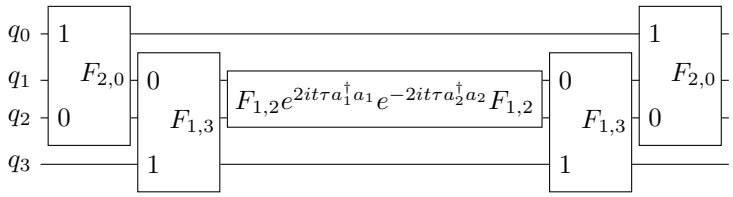


Figure 4.2: Implementation of the 2 rotations from (3.20)

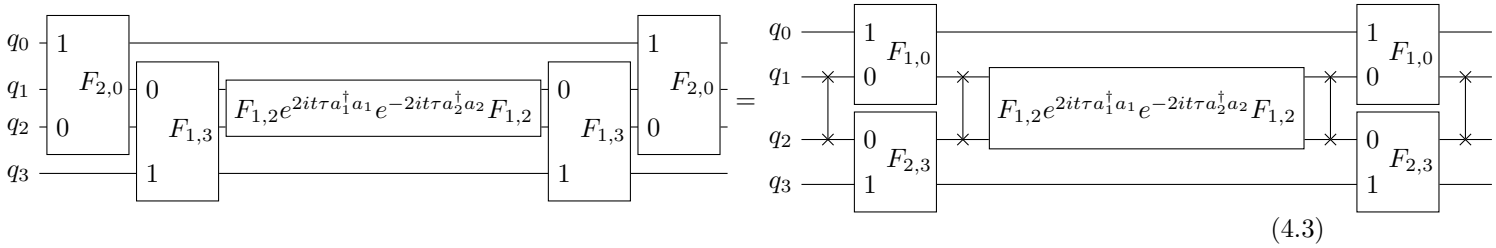
Combining these, a direct implementation of (3.18) would be:



(4.2)

Here, $F_{2,0}$ is still a 2-qubit gate that takes as the first input q_2 and q_0 as the second input and maps it to the previously defined circuit in 4.1. To verify the implementation, it is no longer possible to rely on the explicit matrix definition of operators. This is where **OpenFermions** is used. It offers great features and, most importantly here, the option of mapping creation and annihilation operators to matrices that can be compared with circuit matrices. This shows that (4.2) is not correct.

Further tests show that F acts non-trivially when used over a larger system, and that locality of the application of gates matters. Trying to swap qubits for the F-gates to always apply on adjacent bits shows that:



(4.3)

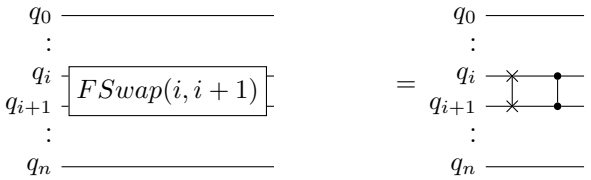
Figure 4.3: Failed implementation via exchanging q_1 and q_2 to satisfy F adjacency

Given the equivalence, this does not yet implement the plaquette operator as needed. This issue requires the introduction of the Fermionic Swap (FSwap) gate, as this implementation is based on Fermionic operators. In fermionic systems, swapping two fermions involves more than just exchanging their quantum states. It also introduces a phase factor due to the anti-commutation property of fermionic operators.

$$a_i^\dagger a_j^\dagger = -a_j^\dagger a_i^\dagger \tag{4.4}$$

This sign change does not appear in standard swap operations. Hence, implementing fermionic swap requires an additional phase correction. When i and j are adjacent, this is defined and implemented as:

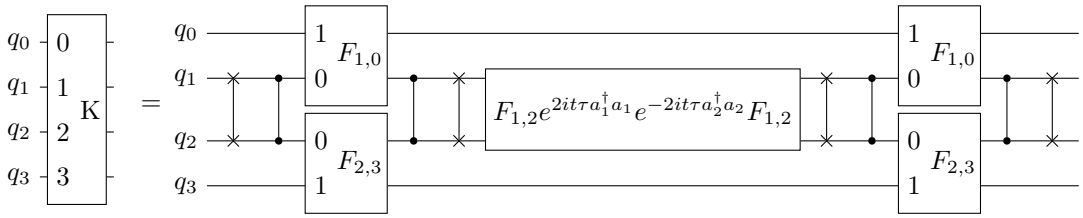
$$FSWAP_{i,j} = SWAP_{i,j} \cdot e^{i\frac{\pi}{4}Z_iZ_j} \Rightarrow$$



(4.5)

This Fermionic swap over 2 adjacent qubits is enough to solve the implementation of the plaquette operator. When needed, the definition of the Fswap gate will be generalized. Using **OpenFermion** to verify the result, the single plaquette operator (defined in (3.18)), is correctly simulated via the following quantum circuit:

$$e^{itK} \equiv$$



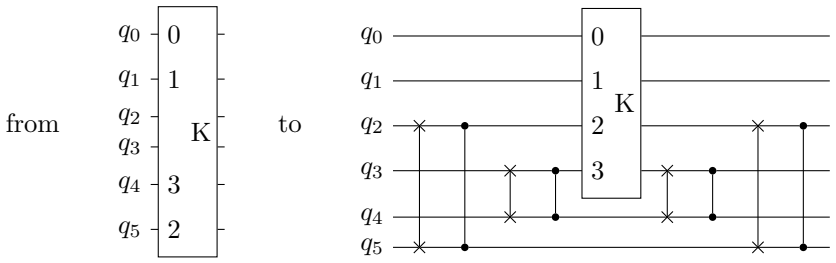
(4.6)

Figure 4.4: Implementation of the single plaquette circuit.

4.1.2 Plaquettes

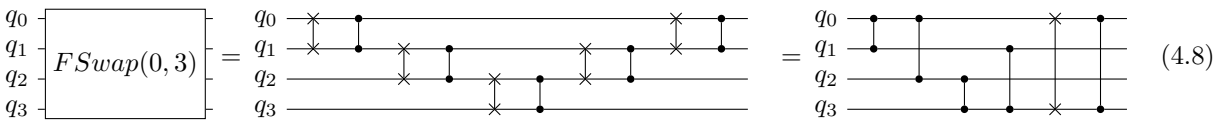
Knowing the operator for a single plaquette, these need to be arranged to partition the full lattice following (2.14). For simplicity, spins are not considered during this discussion as there are no interactions between different spins in such cases. As seen previously, the confusing part of this implementation arises from the need of adjacency when applying gates. Following the chosen row-wise mapping of indices, the single plaquette operation affects none-adjacent qubits. For example, in a $L = 2$ lattice (see Figure 2.2), the first plaquette acts on site $[0, 1, 5, 4]$. Index 4 and 5 are swapped since the single plaquette operator is designed considering the 4 site indices as a loop, not row-wise.

Guided by tests, applying the single plaquette operator on qubit $[0, 1, 5, 4]$ does not yield the correct result. Using previous insight, swapping indices to ensure that the operator acts on 4 adjacent qubits might be the solution. This requires mapping $[0, 1, 5, 4]$ to $[0, 1, 2, 3]$, which requires 2 indices swaps: $(2, 5)$ and $(3, 4)$.



(4.7)

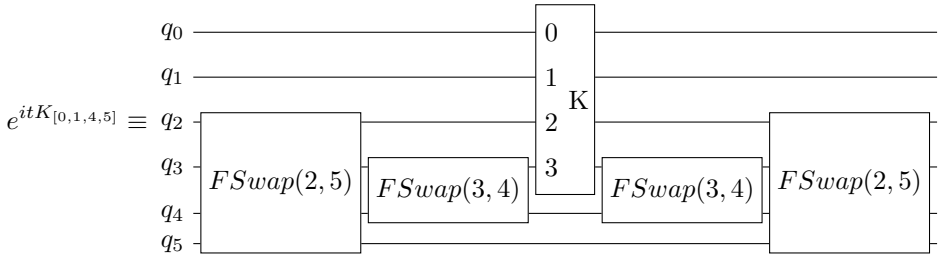
From tests, this does not yield the expected result. This comes from the Fermionic swap, which requires more consideration when applied over non-adjacent terms. This is fixed by defining it as a chain of adjacent term operations, giving a more general definition of the Fswap gate. For simplicity, this is shown over 4 qubits as it is a circuit that spans the full range of qubits.



(4.8)

Figure 4.5: General definition of the fermionic swap for distant qubits.

With that, it is verified that the following circuit correctly implements the single plaquette operator showing the nearest neighbors interactions between site $[0, 1, 4, 5]$:



(4.9)

Figure 4.6: Plaquette operator over [0,1,5,4] plaquette.

In fact, with the correct definition of the Fswap gate, it becomes possible to arbitrarily re-index the system without creating any error. Given a correct sequence of fermionic swaps, it is possible to re-order the qubits so that all of the pink plaquettes are applied in parallel, each spanning 4 adjacent qubits. The same is also true of the gold plaquette.

For example, with $L = 4$ and while still ignoring spin consideration and only showing pink plaquettes, there are four independent plaquettes: $[0, 1, 5, 4]$, $[2, 3, 7, 6]$, $[8, 9, 13, 12]$, $[10, 11, 15, 14]$. A series of Fswap gates can be used to map:

$$[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] \Rightarrow [0, 1, 5, 4, 2, 3, 7, 6, 8, 9, 13, 12, 10, 11, 15, 14]$$

Computationally, it is found to be done by 8 sequential swaps: (2, 5), (3, 4), (4, 5), (6, 7), (10, 13), (11, 12), (12, 13), (14, 15). Following the notation defined in (2.14), this leads to:

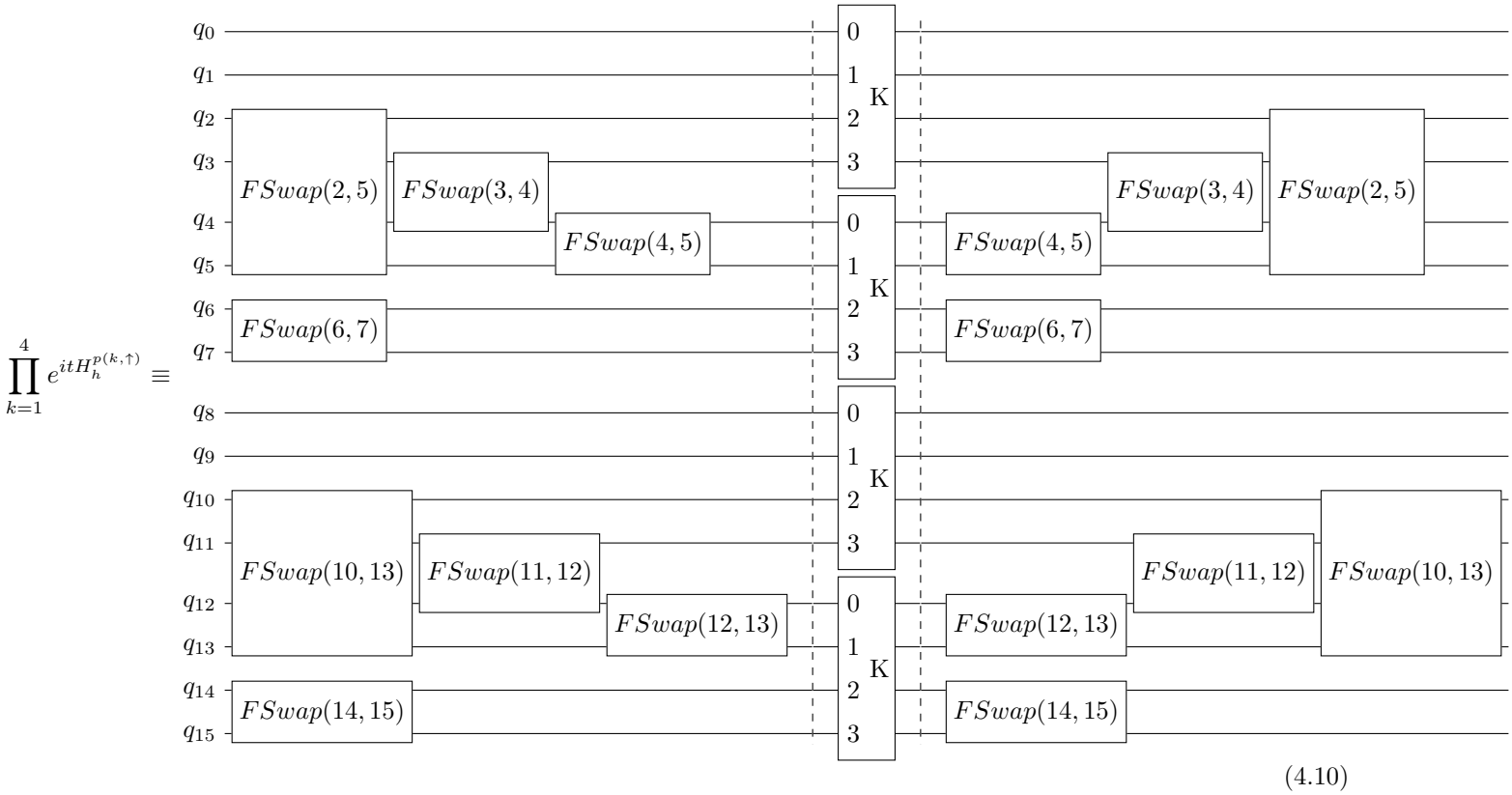


Figure 4.7: Hopping pink operator (up spin only).

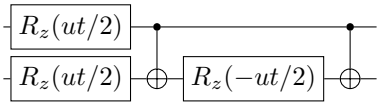
This finalizes the definition of the Hopping terms, as the idea remains the same between gold and pink plaquettes. Note that gold usually requires more Fswap gates since it has more non-adjacent terms as a consequence of periodic boundary conditions.

4.1.3 The interaction term

Fortunately, on-site interaction terms are much more straightforward to implement. This will rather illustrate why the added potential shift, resulting in the definition of the shifted interaction H_i seen in (2.9), allows for a significant reduction in gate counts.

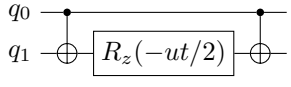
Comparing the circuit implementation of the unshifted interaction Hamiltonian H_u, as defined in (2.7) and the circuit implementation of the shifted interaction Hamiltonian H_i will highlight the upside of the chemical shift. Consider a 2-qubits system for now.

$$e^{itH_u} = e^{iut(\hat{n}_0\hat{n}_1)} \equiv e^{itu/4} \times$$



(4.11)

$$e^{itH_i} = e^{iut(\hat{z}_0\hat{z}_1)/4} \equiv$$

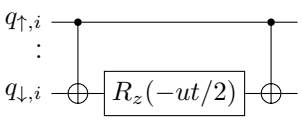


(4.12)

Figure 4.8: Unshifted and unshifted Hamitonian implementation

Knowing the problem of interaction between distant terms, a test for a more general approach is needed. It shows that for the interaction term, there is no further swapping of indices needed. The definition (4.12) can be generalized to:

$$e^{itH_i} = e^{iut(\hat{z}_{\uparrow,i}\hat{z}_{\downarrow,i})/4} \equiv$$

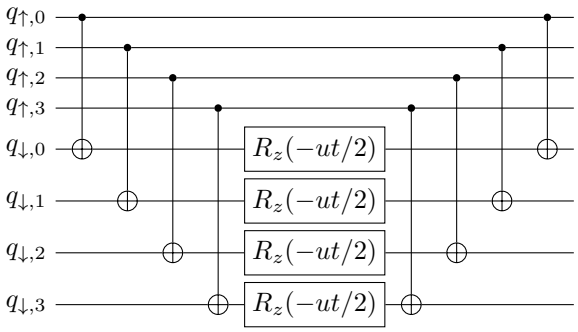


(4.13)

Figure 4.9: General definition of the on site interaction between distant site.

Hence, on a $L = 2$ lattice, it is verified that the following circuit implements the interaction term correctly:

$$e^{itH_i} = e^{i\frac{ut}{4}\sum_i^{L^2}(\hat{z}_{\uparrow,i}\hat{z}_{\downarrow,i})} \equiv$$



(4.14)

Figure 4.10: Example of the interaction term H_i with $L = 2$.

4.2 Overall implementation

With all the operations defined, they can be assembled in a circuit that simulates the time evolution under the Hubbard model. This creates a final gate that encapsulates all previous ones and uses the two Trotterizations and the repetitions as described in (3.9).

This is the gate called *hubbard_unitary*.

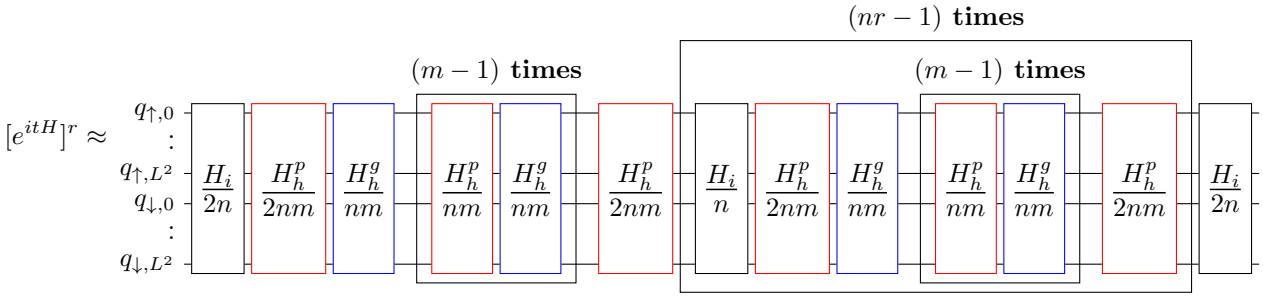


Figure 4.11: Implementation of the Hubbard unitary.

Chapter 5

Results

This chapter will show results obtained directly from simulations. As previously, the code used to generate each figure is available in *figure.ipynb* for better transparency and also to serve for example for anyone willing to conduct further tests.

5.1 Technical limitation

The simulation of quantum circuit on classical computers comes with various limitations. One of which is that different libraries use different techniques when trying to run a circuit. Some allow for a perfect simulation of the system, acting as a large matrices multiplication which requires exponentially scaling memory, while others make use of approximations to increase the upper limit of computation at the cost of precision.

Given that for any L, one would require $2L^2$ qubits to represent the sites accordingly and that the lattice side length has been defined as even, this greatly limits the scale upon which results can be obtained.

Method	**Max qubits**	**Max L**
Automatic	30	2
Statevector	30	2
Density Matrix	15	2
Matrix Product State	63	4
Extended Stabilizer	63	4
Unitary	15	2
Superoperator	7	0

Table 5.1: Maximum number of qubits supported by different simulation methods.

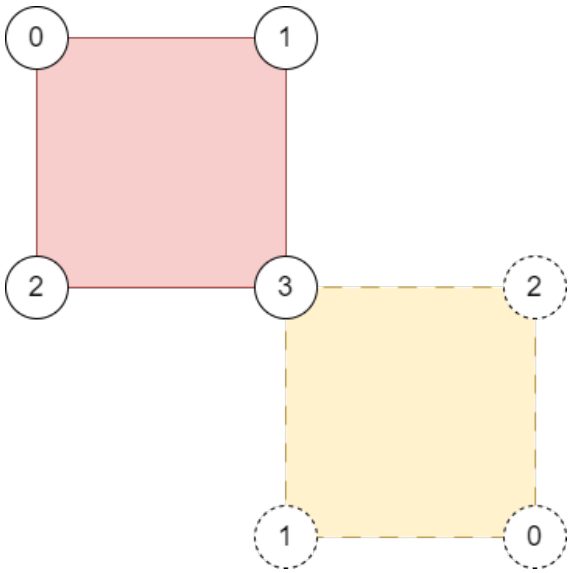
Considering only the simulation methods limits, this would allows for simulations up to $L = 4$. But to verify that the implementations are correct, a reference frame is needed. This is done using **OpenFermions**, which allows to obtain a matrix representation of the operator from its fermionic operator definition. With a matrix representation, this also gives access to all eigenstates $|\psi_n\rangle$ and their associated eigenvalues λ_n. Those eigenstates are needed to bypass state preparation and have a perfect scenario to test the phase estimation algorithms. Without access to these eigenstates, one would need to implement state preparation to extract useful phases. This was left outside of the scope of this project.

Given these limitations, computational verification of theoretical error scaling over larger system (**Appendix F** in [1]) is not possible. The focus will instead be on results that can be directly obtained from the implementation of the plaquette method. It will demonstrate that the implementation converges to expected results. Once this is established, the framework can serve as a foundation for further testing or applications.

5.2 Trotter steps and convergence

The first issue from technical limitation arises here. It is not possible to verify the correct behavior of the complete equation (3.9) with all parameter at once. Those are m, the number of Trotter steps for the plaquette operator decomposition ($H_h = H_h^p + H_h^g$), and n, the number of Trotter steps for the overall decomposition ($H = H_i + H_h$). This comes from the fact that when considering $L = 2$, the decomposition of the lattice into pink and gold plaquettes does not apply.

Considering a single pink and a single gold plaquette with period boundary condition would double the interaction between site.



$$\Rightarrow$$

$$\begin{aligned} H_h^p &= a_0^\dagger a_1 + a_0^\dagger a_2 + a_1^\dagger a_0 + a_1^\dagger a_2 \\ &\quad + a_2^\dagger a_0 + a_2^\dagger a_3 + a_3^\dagger a_1 + a_3^\dagger a_2 \\ H_h^g &= a_3^\dagger a_2 + a_3^\dagger a_1 + a_2^\dagger a_3 + a_2^\dagger a_0 \\ &\quad + a_1^\dagger a_3 + a_1^\dagger a_0 + a_0^\dagger a_2 + a_0^\dagger a_1 \end{aligned}$$

Figure 5.1: Pink and gold plaquette with periodic boundary condition if $L = 2$.

Here, the two plaquette operators are equal and applying in sequence pink and gold would lead to incorrect results. This means that for $L = 2$, the plaquette operator is a singular plaquette operator. For $L = 4$, the matrix representation of the operator grows too large and access to a reference point is no longer possible.

This is not a problem for the interaction term as it's overall effect can be verified with $L = 2$. This results in either using a made-up lattice to show convergence of the plaquette decomposition with regard to m while ignoring the interaction term, or using a $L = 2$ lattice to show the overall convergence while ignoring the specific nature of the plaquette decomposition.

Different metrics will be used to show convergence to expected results. These are the Hilbert-Schmidt norm (also known as the Frobenius norm) and the fidelity.

The Hilbert-Schmidt norm is used to show the element-wise difference between two matrices:

$$||A - B||_{H-S} = \sqrt{\sum_i \sum_j |A_{ij} - B_{ij}|^2} \tag{5.1}$$

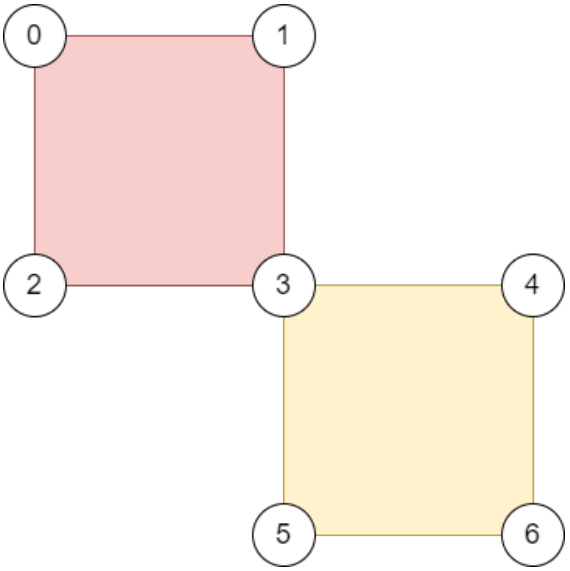
The fidelity is a more commonly used metric for quantum circuit. The closer it is to 1, the closer is the effect of the two operators. Note that in the following results, the value of $(1 - F)$ is shown instead, as it allows better visualization.

$$F(A, B) = \frac{1}{d}|Tr(A^\dagger B)| \tag{5.2}$$

In these definitions, A will be the reference matrix obtained via **OpenFermions**, while B will be the matrix retrieved from the circuit. The factor $\frac{1}{d}$ uses d, which is the side length of our square matrices, in our case $2^{(2L^2)}$.

5.2.1 Number of Trotter steps m

To still showcase that the partitioning of the lattice into gold and pink plaquettes leads to a correct result, a fake lattice without boundary condition is defined as follows:



$$\Rightarrow$$

$$\begin{aligned} H_h^p &= a_0^\dagger a_1 + a_0^\dagger a_2 + a_1^\dagger a_0 + a_1^\dagger a_2 \\ &\quad + a_2^\dagger a_0 + a_2^\dagger a_3 + a_3^\dagger a_1 + a_3^\dagger a_2 \\ H_h^g &= a_3^\dagger a_4 + a_3^\dagger a_5 + a_4^\dagger a_3 + a_4^\dagger a_6 \\ &\quad + a_5^\dagger a_3 + a_5^\dagger a_6 + a_6^\dagger a_4 + a_6^\dagger a_0 \end{aligned}$$

Figure 5.2: Fake lattice with one pink and one gold plaquette and not periodic boundary conditions.

In this lattice, the alternating application of the pink and gold plaquettes faces the same challenge of approximating all nearest-neighbor interactions through successive smaller interactions, as site 3 is shared between both plaquettes. While this is not a perfect representation of the complex behavior of plaquette decomposition on a larger lattice, it should still capture the essential characteristics of the process.

The Trotterization, with regard to the number of Trotter steps m, is defined as:

$$\begin{aligned} e^{itH_h} &\approx \left(e^{itH_h^p/2m} e^{itH_h^g/m} e^{itH_h^p/2m}\right)^m \\ \text{(as coded)} &\approx e^{itH_h^p/2m} \left[e^{itH_h^g/m} e^{itH_h^p/m}\right]^{(m-1)} e^{itH_h^g/m} e^{itH_h^p/2m} \end{aligned}$$

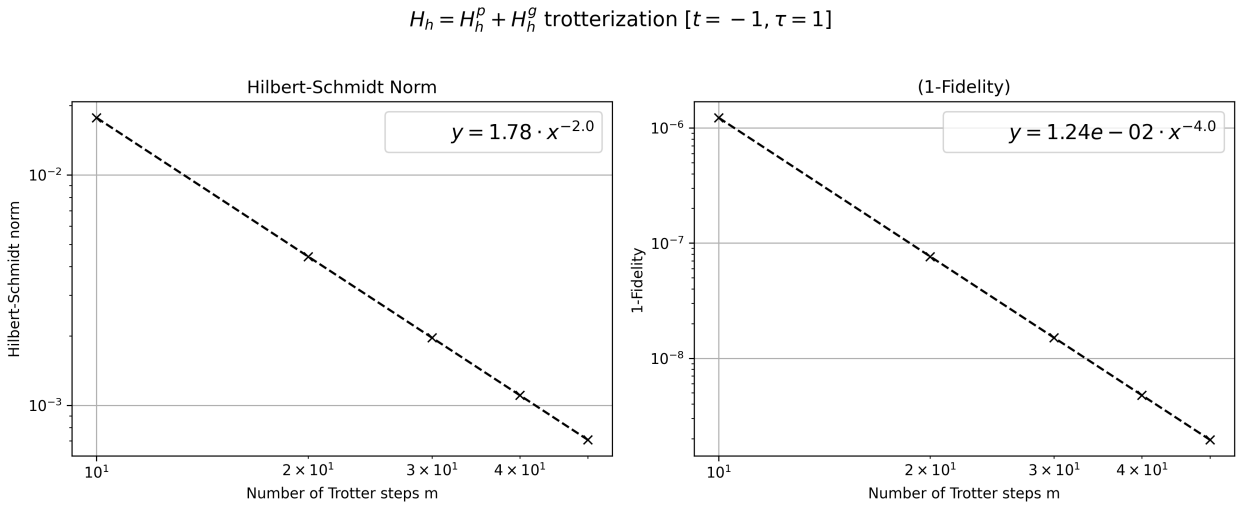


Figure 5.3: Convergence to perfect operator with regard to the number of Trotter step m

The results in Figure 5.3 illustrate the convergence of the action of the circuit to the expected model.

This provides strong evidence that the implementation of the plaquette decomposition correctly captures the behavior of the hopping term of the Hubbard model.

Guided by the appearance of linear behavior on a $log - log$ scale, it is possible to fit the results to a power law.

$$(1 - F) = c(t, \tau, u)m^{k(t,\tau,t)} \tag{5.3}$$

$$F = 1 - c(t, \tau, u)m^{k(t,\tau,t)} \tag{5.4}$$

Without going into much detail, additional tests show that the value of c and k is influenced at least by τ and u. Given that extensive tests cannot be performed nor verified for $L > 2$, some results regarding c and k will be shown later, but no general conclusions regarding their definition as functions of the model parameters will be drawn.

5.2.2 Number of Trotter steps n

For the decomposition of $H = H_i + H_h$, a $L = 2$ lattice can be used. As discussed, in such cases, the plaquette decomposition has no meaning and the hopping term is just a singular operator that implements all of the nearest neighbor interaction.

This decomposition, with regard to the number of Trotter steps n, becomes:

$$e^{itH} \approx (e^{itH_i/2n}e^{itH_h/n}e^{itH_i/2n})^n$$

$$\text{(as coded)} \approx e^{itH_i/2n}[e^{itH_h/n}e^{itH_i/n}]^{(n-1)}e^{itH_h/n}e^{itH_i/2m}$$

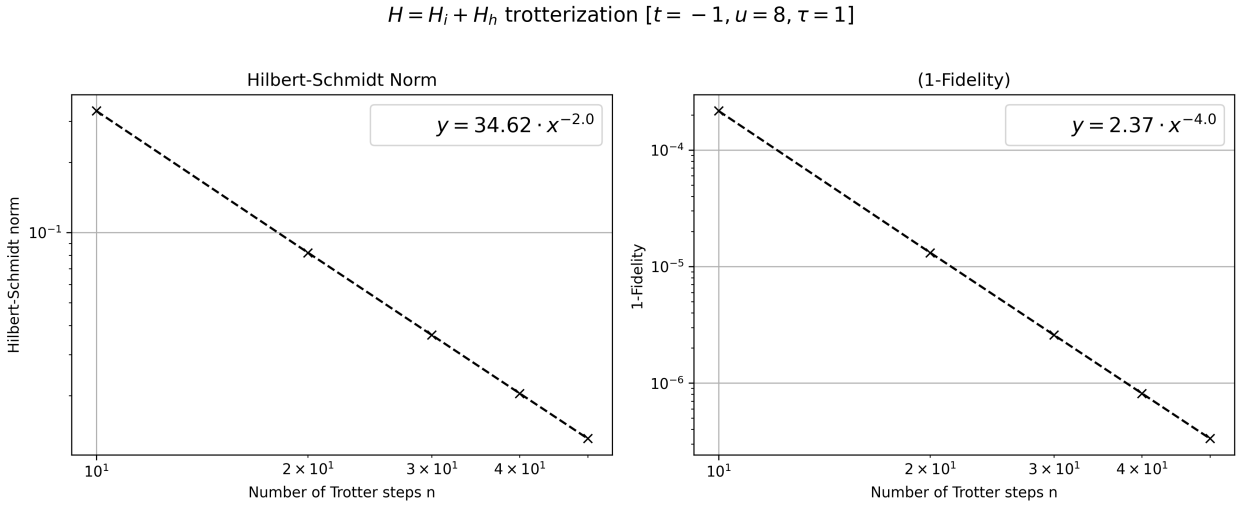


Figure 5.4: Convergence to perfect operator with regard to the number of Trotter step n

The results in Figure 5.4 illustrate the convergence of the action of the circuit to the expected model. This provides strong evidence that the implementation of the on-site interaction is correct and that the Trotter decomposition captures the overall behavior expected from the Hubbard model.

5.3 Increasing n and t or r

Following the claim made in (3.22), evidence that considering $U(t)^k$ as k repetition of the unitary $U(t)$ or as a single $U(kt)$ (with scaled n) is equivalent, both in terms of resource scaling and precision, are provided here.

The difference between the two circuits is defined as the Hilbert-Schmidt norm of their matrix representation.

$$\Delta = ||U^{kr_0}_{n_0,1}(t_0) - U^1_{kn_0,1}(kt_0)||_{H-S} \tag{5.5}$$

To also verify the scaling of the complexity of the circuit, the number of Rz-gates present in both versions is shown.

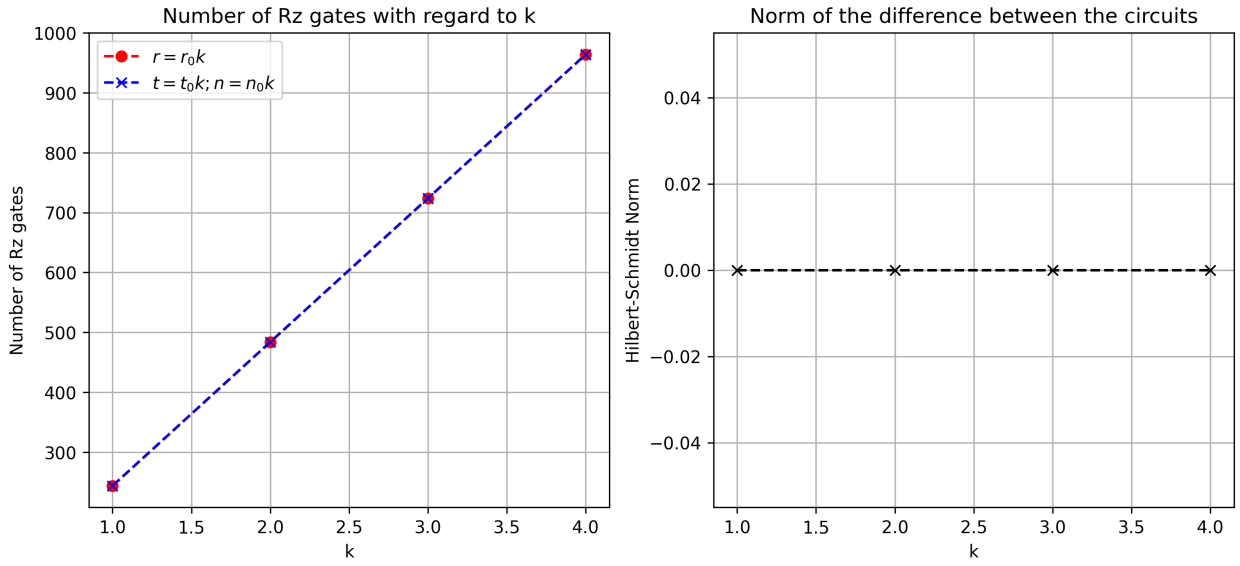


Figure 5.5: Effect of increasing k on the two circuits $[r_0 = 1, n_0 = 30]$

The results seen in Figure 5.5 confirm the claim made in (3.22).

5.4 Phase estimation

As a uniformly controlled version of the unitary, following the definition in Section 2.1.3, is supported, one can implement all sorts of phase estimation algorithm. Since the focus of this project is not on the quantum phase estimation algorithm itself, a simple version of phase estimation is used. The implementation is based on this documentation [16], but many other sources discuss this algorithm as it is considered the simplest for phase estimation.

The results will be presented for a configuration in which the interaction term strength is set to $u = 8$ and the hopping term strength to $\tau = 1$. The lattice size is limited to $L = 2$, and the number of Trotter steps is fixed at $n = 10$ to ensure a reasonable computational time.

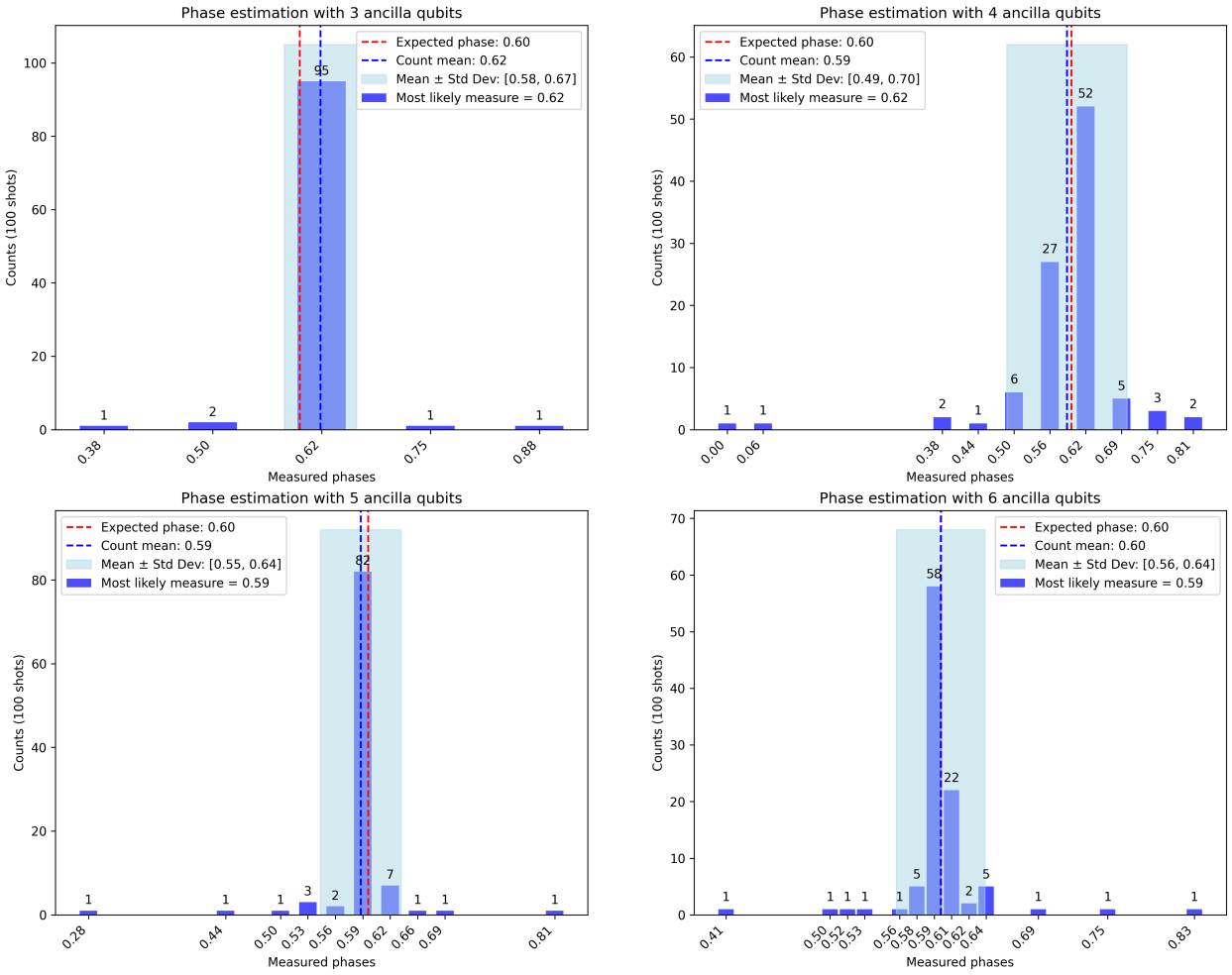


Figure 5.6: Results of the quantum phase estimation algorithm

As explained in Section 2.1.2, the choice of t is such that a bijection from phase to energy value is possible (2.4).

In the given example, the circuit was initialized with an eigenstate retrieved using **OpenFermions**. The expected phase measurement of this eigenstate was approximately $\phi_n \approx 0.60$. Since the eigenvalues of H are all such that $\lambda_n \in [-10, 10]$, the value of t was set to $t = \frac{2\pi}{20}$. Using (2.4) yields the following:

$$\begin{aligned} \lambda_n &= \frac{2\pi(\phi_n - 1)}{t} \\ &= 20(\phi_n - 1) \approx -8 \end{aligned} \tag{5.6}$$

This value is confirmed to be part of the energy spectrum of the model, as the spectrum of the matrix form of H is available for verification.

A larger study will now be conducted to evaluate the accuracy of the energy measurement depending on the number of Trotter steps and the precision of the phase estimation. Here, the notion of precision of the phase estimation reflects the number of ancilla qubits used for phase estimation, directly linked with the number of bits in the fractional binary expansion of the phase.

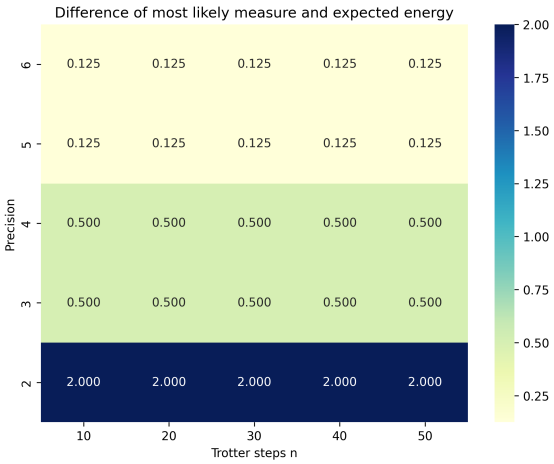


Figure 5.7: Most likely measurement

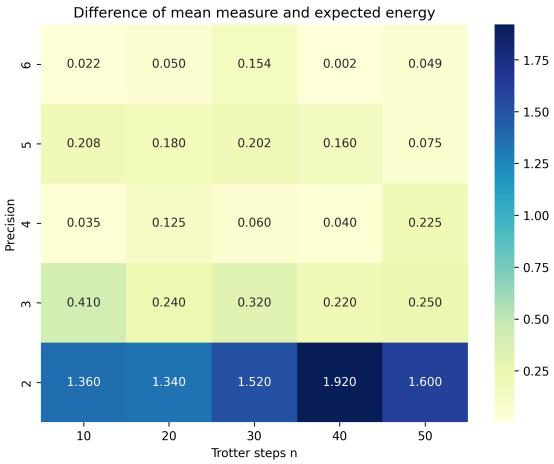


Figure 5.8: Mean measurement

From Figure 5.7, it appears that increasing the number of Trotter steps does not change the accuracy of the phase estimation. This comes from the fact that the increased precision is not enough to change which measure is the most likely. But it influences the probability of making such a measurement. In Figure 5.8, the mean measure is considered to see more subtle influences. This shows that increasing the number of ancilla qubits for the phase estimation does reduce the error on the measure, but the effect of the Trotter step increase is not always what would be expected. This can be attributed to the small sample size of the simulation rather than to an effect of the model.

5.5 Additional tests

With the arguments validating the correctness of the implementation now established, the designed circuit can be used to conduct additional tests as desired.

5.5.1 Increasing precision

The phase estimation algorithm uses a ancilla qubits. With this, the resulting phase is encoded as a information bits, leading to a numerical precision of at most $p = 2^{-a}$ in its fractional binary expansion.

The number of operations needed to apply r repetitions of the unitary is known from (3.9). From this, the number of operations as a function of a:

Table 5.2: Cost calculation for different values of a.

Number of ancillas for PE	Unitary application	Total Cost
1	U^1	$[2mn + 2n + 1]$
2	$U^1 + U^2$	$[2mn + 2n + 1] + [2 \cdot 2mn + 2 \cdot 2n + 1]$
:	:	:
a	$\sum_{k=0}^{a-1} U^{2^k}$	$(2^a - 1)[2mn + 2n] + a$

The last results comes from

$$\sum_{k=0}^{a-1}[2^k \cdot 2mn + 2^k \cdot n + 1] = (\sum_{k=0}^{a-1} 2^k)[2mn + n] + a = (2^a - 1)[2mn + 2n] + a = C(a, n, m) \tag{5.7}$$

The cost of increasing by one the number of ancilla qubits for phase estimation is given by $C(a+1, n, m) - C(a, n, m)$. C is defined as a cost function in (5.7), to lighten the notation.

$$\begin{aligned} C(a+1, n, m) - C(a, n, m) &= (2^{(a+1)} - 1)[2mn + 2n] + (a + 1) - (2^{(a)} - 1)[2mn + 2n] - a \\ &= 2^a[2mn + 2n] + 1 \end{aligned}$$

It is now possible to determine how much the number of Trotter steps can be increased while maintaining the same cost increase as the that caused by adding one ancilla qubits for phase estimation. Following the introduced notation, this is equivalent to finding x such that $C(a+1,n,1) - C(a,n,1) = C(a,n+x,1) - C(a,n,1)$. Note that from now on, $m = 1$, as this is the only case where results can be obtained given the current limitations.

$$\begin{aligned} C(a+1,n,1) - C(a,n,1) &= C(a,n+x,1) - C(a,n,1) \\ (2^a)\cdot 4n + 1 &= (2^a - 1)[4(n+x)] + a - (2^a - 1)[4n] - a \\ 2^{a+2}n + 1 &= (2^{a+2} - 4)x \\ x &= \frac{2^{a+2}n + 1}{2^{a+2} - 4} \\ \Rightarrow n + x(a,n) &= \frac{2^{a+2}n - 4n + 2^{a+2}n + 1}{2^{a+2} - 4} = \frac{2^{a+1}n - n + 0.25}{2^a - 1} \end{aligned} \tag{5.8}$$

This creates two choices that have an equivalent effect on the total number of operations but result in different precision outcomes. Using the framework, it is possible to sample, starting from specific initial conditions, which of these two choices brings the results closer to the ideal measurement.

- Set the starting condition n_0 and a_0.
- Measure with starting condition $n = n_0$, $a = a_0$.
- Repeat:
  - Measure the most likely result with $n_1 = \lfloor n_0 + x(a_0, n_0) \rfloor$, $a_1 = a_0$ (Measure N).
  - Measure the most likely result with $n_2 = n_0$, $a_2 = a_0 + 1$ (Measure A).
  - If Measure N is better, set $n_0 = n_1$, else set $a_0 = a_2$. (If equal, use the mean measurement.)

With $n_0 = 20$ and $a_0 = 2$, this gives:

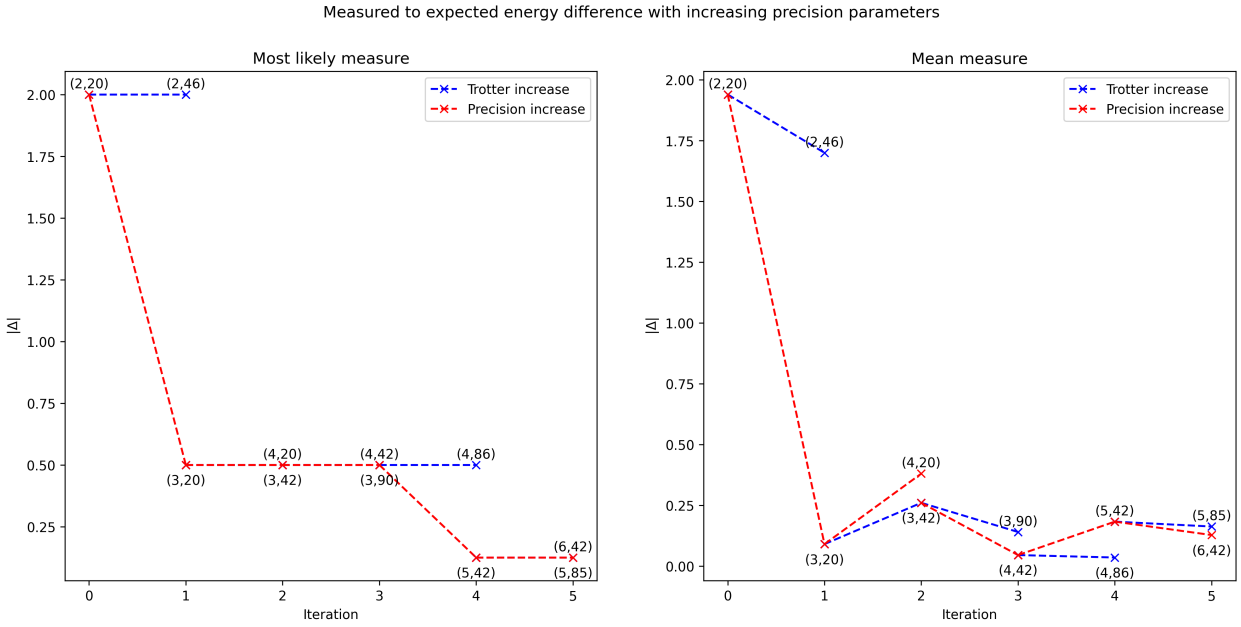


Figure 5.9

The result of this test, shown in Figure 5.9, demonstrates that the path to optimization is not trivial. While this was an interesting consideration, the first step should focus more on identifying and selecting a more efficient phase estimation algorithm, rather than attempting to optimize with this basic approach. Phase estimation algorithms are plentiful, each offering different trade-offs in terms of qubit requirements and precision (**Table I** in [17])

5.5.2 Power-law parameters

To see if there exists a simple relation to express the constant and exponent terms in the power law observed in the convergence as n increases (Figure 5.4), the same convergence test using different values of u and τ is conducted. The results are as follows:

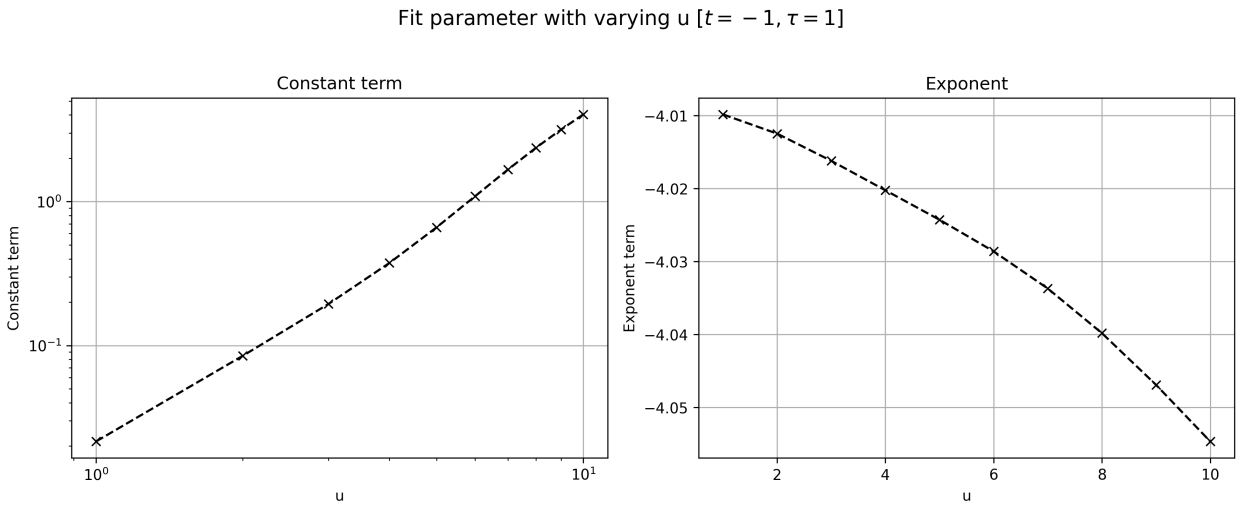


Figure 5.10: Evolution of the power low parameters observed during convergence, with regard to the interaction strength u

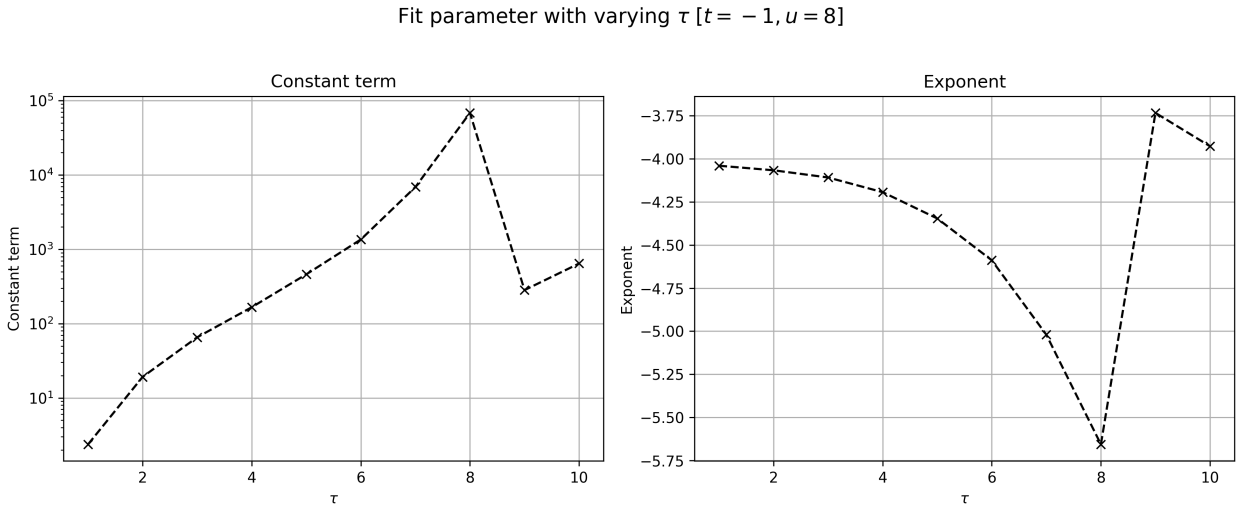


Figure 5.11: Evolution of the power low parameters observed during convergence, with regard to the hopping strength τ

When considering the variation of the interaction strength parameter u (Figure 5.10), the results could indicate another power-law scaling. But when considering the same iterative process over the hopping strength τ (Figure 5.11), the non-trivial nature of the result will leave a more general conclusion outside the scope of this project.

5.5.3 Gate counts

The claims from **Appendix E,1** [1] regarding the number of gates required can be verified, focusing primarily on the number of T and Rz gates, as these are the most difficult to implement. In the reference, there is no consideration for multiple Trotter steps, and only the repeated terms are taken into account. Adjusting

by adding the two non-repeated terms leads to the implementation requiring $12rL^2$ T-gates and $(4r+2)L^2$ Rz-gates. This is reflected as '*Expected value from reference*'.

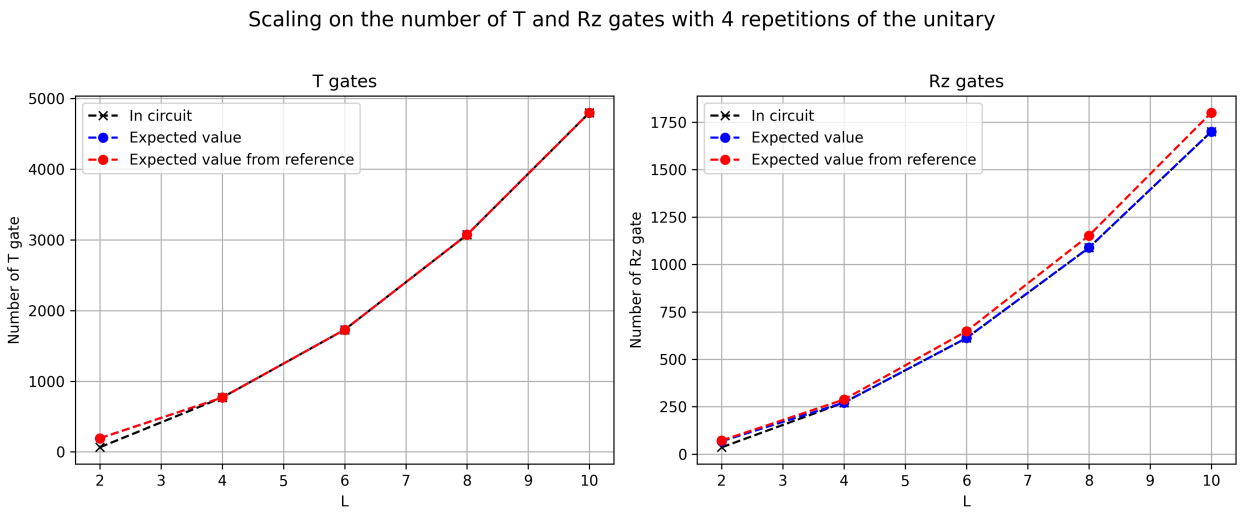


Figure 5.12: Number of Rz and T gates needed with regard to lattice size $[n = 1, m = 1]$

The other curve, '*expected value*', comes from (3.9) when considering $n = m = 1$. In such a case, $[(8mnr + 4nr)L^2 \to 12rL^2]$ T-gates and $[(2mnr + 2mr + 1)L^2 \to (4r + 1)L^2]$ Rz-gate are expected. This is verified via direct count from the circuit. One difference remains when $L = 2$, but is explained because the previous count does not consider the absence of gold plaquette in such a case. The reduction of the number of Rz-gates when comparing circuit results and expected counts from reference comes from the reduction of one application of the interaction terms. This number would scale according to the number of Trotter steps m and n and could still represent a significant reduction when working on more precise system.

5.5.4 Number of Fswap

It was observed that a significant number of Fswap gates are required to re-index the registers when applying the plaquette operator (Figure 4.7). While Fswap can be implemented using Clifford gates only, its scaling for larger lattices warrants further attention.

Since the exact number of Fswap gates depends on specific choices of indexation, a formal mathematical proof for the minimal number of permutations remains an open problem. Instead, a practical approach is taken. By analyzing the circuit as the lattice size L increases, the number of Fswap gates needed for a single application of the pink (gold) plaquettes operator can be directly retrieved.

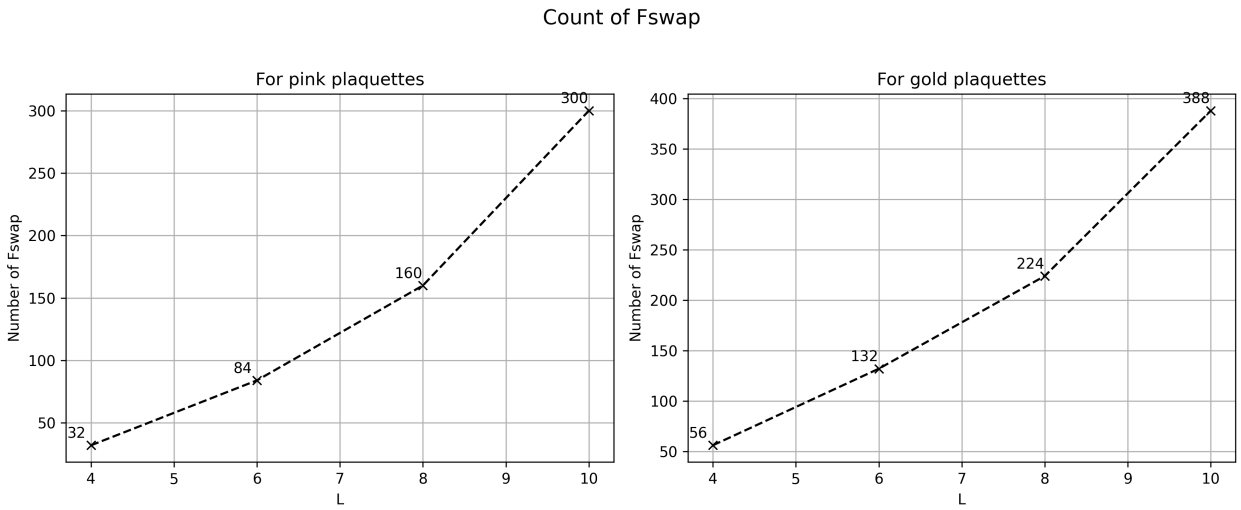


Figure 5.13: Number of Fswap gates used with regard to lattice size $[n = 1, m = 1]$

Looking at the implemented time evolution (3.9), there are $[mnr + nr]$ applications of the pink plaquettes operator and $[mnr]$ applications of the gold plaquettes operator. For $L = 8$, the total number of Fswap gates needed will reach $[384mnr + 160nr]$.

While this scales linearly with each parameter of Trotter steps and repetitions, the results from Figure 5.13 indicate a non-linear scaling regarding the size of the lattice, which could be a source of optimization for larger lattices.

5.5.5 Hardware error sensibility

The results from phase estimation seen in Figure 5.6 show very promising results with a small number of Trotter steps. This comes from the idea situation in which the simulation are done. Using **Qiskit**, it is possible to set noise models based on real hardware. Running the phase estimation algorithm using this noisy model instead of the idea model gives the following results.

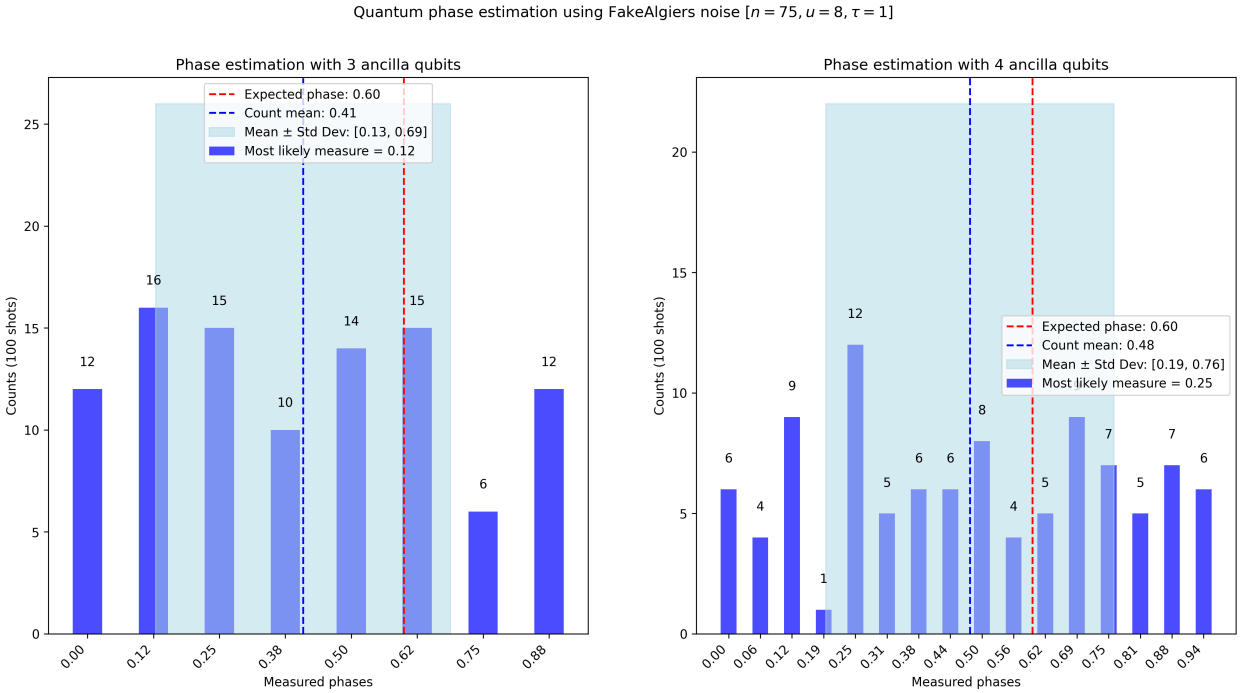


Figure 5.14: Phase estimation results with simulated noise

In Figure 5.14, the number of Trotter steps n was increased to 75 compared to the previous results using only 20 (Figure 5.6). Even with this increase, the results can be attributed to random noise. It might be possible to reduce that noise with more shots for each simulation or by increasing the number of Trotters steps. Simulations of noisy hardware are much more computationally demanding and it is not possible to see results using larger values for n while keeping a reasonable computational time. This confirms that, while the Hubbard model can be implemented as a quantum algorithm, quantum computer technologies still need to reach the creation of a fault-tolerant device to obtain usable results.

Chapter 6

Discussion

6.1 Next steps and open questions

While the framework for simulation of quantum circuit that implements the Hubbard model under plaquette Trotterization is now established and available, there are still points that could be addressed for a more complete analysis of the model.

Independent studies of the Trotterization of the plaquette operator with m steps and of the overall Trotterization with n steps has been used to show convergence of the circuit to the expected time evolution. However, no tests could be performed to analyze their combined effects. This is due to technical limitations with two factors. Trying to directly compare matrices forms of those operators creates matrices that cannot be handled on local available machines. Such a test could be performed on machines with more available memory, but the computational power and time required might exceed what is acceptable. To circumvent this, it would be possible to study the results from phase estimation instead. In such cases, there would be no need to handle huge matrices. This brings to light an important missing part of the implementation: state preparation.

In fact, to retrieve information relative to phases, the current approach and the results presented in Figure 5.6 assume access to an eigenstate. Initializing the circuit to this eigenstate leads to a measure that converges only to the phase of this specific eigenstate. In a real scenario, the initial state would have to be prepared, either via classical means or additional quantum circuit, to maximize the overlap with one eigenstate. For more details on the effect of such preparation on the expected measurement output of phase estimation, refer to **Section 5.1** in [8].

On the topic of phase estimation, the algorithm implemented in this work is quite basic. It requires increasing the number of ancilla qubits to increase the precision of the measurement, and is not flexible when it comes to the total number of unitary applications, as they are always $2^n - 1$ with $n \in \mathbb{N}^+$. The use of iterative phase estimation was tested but abandoned due to time constraints and added complexity. Many other phase estimation algorithms could be used and implemented, as phase estimation is a widely used tool for quantum computing. But the specific choice and implementation for this case were left out of the scope of the project.

More specific to the Hubbard model and the plaquette Trotterization, it was discussed in section 5.5.4 that the number of Fermionic swap operations could become a problem with larger systems. While the priority remains on reducing the number of T and Rz gates, Fermionic swap operation can be optimized. As an example, by choosing a different convention for the lattice indices for the pink plaquettes operator to require no re-indexing, or by reducing the number of Fswap between the application of the pink plaquette operator and the gold plaquette operator. For now, the indices are reset to the base position and then changed again, which might introduce useless swapping.

While the results confirm expectations regarding the scaling number of gates regarding gate number (as seen in Figure 5.12), a more explicit connection with the established error bounds for the plaquette Trotterization (**Appendix D** in[1]) and the associated link with phase estimation precision (**Appendix F** in [1]) would strengthen the validity of both plaquettes decomposition and its implementation. This could also allow for a more general conclusion in terms of scaling, specifically regarding the implementation rather than the theoretical model.

The use of Hamming weight phasing has also been referenced in section 2.3.2 as an upside to the use of plaquette Trotterization. This other way of applying parallel same angle rotation would reduce the number

the Rz-gates and T-gates needed at the cost of adding some ancilla qubits. This is not present in the implementation, but would be an essential component to consider. This would require, among other things, a rework of the gate seen in Figure 4.2. Here, the two rotations need to be applied in parallel. It is known to be possible and is explicitly shown in **Figure 4.c** from [18].

This new publication of 20.01.2025, shows extensions of the plaquette Trotterization concept over more general shapes of lattices. In doing so, some results are more explicitly stated than in the source on which this project is based [1]. Given the timing of this publication [18], none of the results presented here were considered in the creation of either the code and the report.

6.2 Conclusion

This project explored the implementation and validity of the Hubbard model under plaquette Trotterization. The model was implemented from the ground up in Python with **Qiskit**, enabling simulations that demonstrate how the proposed plaquette decomposition can effectively approximate time evolution under the Hubbard model using quantum algorithms. Throughout the implementation process, key definitions were expanded to optimize gate counts while providing more flexible parameters for computation. This includes two distinct Trotter step parameters and a more detailed formulation for repeated unitary application. Operators and gates were clearly defined, and simple phase estimation was used to confirm that the retrieval of phase and energy eigenvalues is possible.

While the simulation framework can still be refined, it serves as a solid proof of concept and provides a foundation for further exploration. This work establishes plaquette Trotterization as a promising approach for optimizing the Hubbard model and positions it as a viable candidate for use on future fault-tolerant quantum computers.

Bibliography

[1] Earl T. Campbell. Early fault-tolerant simulations of the Hubbard model, July 2024. arXiv:2012.09238.

[2] GHMoreillon. GHMoreillon/QuantumSimulationHubbard, January 2025. original-date: 2025-01-30T16:12:26Z.

[3] Ursula Röthlisberger. Introduction to Electronic Structure Methods.

[4] A. Yu Kitaev. Quantum measurements and the Abelian Stabilizer Problem, November 1995. arXiv:quant-ph/9511026.

[5] H. F. Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.

[6] Naomichi Hatano and Masuo Suzuki. Finding Exponential Product Formulas of Higher Orders. volume 679, pages 37–68. November 2005. arXiv:math-ph/0506007.

[7] Robert S Strichartz. The Campbell-Baker-Hausdorff-Dynkin formula and solutions of differential equations. *Journal of Functional Analysis*, 72(2):320–345, June 1987.

[8] James D. Whitfield, Jacob Biamonte, and Alán Aspuru-Guzik. Simulation of electronic structure Hamiltonians using quantum computers. *Molecular Physics*, 109(5):735–750, March 2011. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00268976.2011.552441.

[9] Dave Wecker, Matthew B. Hastings, Nathan Wiebe, Bryan K. Clark, Chetan Nayak, and Matthias Troyer. Solving strongly correlated electron models on a quantum computer, August 2015. arXiv:1506.05135.

[10] J. Hubbard. http://www.jstor.org Electron Correlations in Narrow Energy Bands. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 276(1365):238–257, 1963.

[11] Bo-Xiao Zheng, Chia-Min Chung, Philippe Corboz, Georg Ehlers, Ming-Pu Qin, Reinhard M. Noack, Hao Shi, Steven R. White, Shiwei Zhang, and Garnet Kin-Lic Chan. Stripe order in the underdoped region of the two-dimensional Hubbard model. *Science*, 358(6367):1155–1160, December 2017. arXiv:1701.00054 [cond-mat].

[12] Ian D. Kivlichan, Jarrod McClean, Nathan Wiebe, Craig Gidney, Alán Aspuru-Guzik, Garnet Kin-Lic Chan, and Ryan Babbush. Quantum Simulation of Electronic Structure with Linear Depth and Connectivity. *Physical Review Letters*, 120(11):110501, March 2018. arXiv:1711.04789 [quant-ph].

[13] Ian D. Kivlichan, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Wei Sun, Zhang Jiang, Nicholas Rubin, Austin Fowler, Alán Aspuru-Guzik, Hartmut Neven, and Ryan Babbush. Improved Fault-Tolerant Quantum Simulation of Condensed-Phase Correlated Electrons via Trotterization, July 2020. arXiv:1902.10673.

[14] qsharp/samples/estimation/estimation-hubbard-2D.ipynb at main · microsoft/qsharp · GitHub.

[15] Craig Gidney. Halving the cost of quantum addition. *Quantum*, 2:74, June 2018. arXiv:1709.06648 [quant-ph].

[16] 2-4. Phase Estimation Algorithm (Introduction) — Quantum Native Dojo documentation.

[17] D. W. Berry, B. L. Higgins, S. D. Bartlett, M. W. Mitchell, G. J. Pryde, and H. M. Wiseman. How to perform the most accurate possible phase measurements. *Physical Review A*, 80(5):052114, November 2009. arXiv:0907.0014 [quant-ph].

[18] Andreas Juul Bay-Smidt, Frederik Ravn Klausen, Christoph Sünderhauf, Róbert Izsák, Gemma C. Solomon, and Nick S. Blunt. Fault-tolerant quantum simulation of generalized Hubbard models, January 2025. arXiv:2501.10314 [quant-ph].