UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Juliana Ramos Neves

Hamiltonian Design of Quantum Gates

Belo Horizonte
2022

Juliana Ramos Neves

Hamiltonian Design of Quantum Gates

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Gabriel de Morais Coutinho

Belo Horizonte
2022

Neves, Juliana Ramos.

N518h Hamiltonian design of quantum gates [recurso eletrônico] / Juliana Ramos Neves – 2022.
1 recurso online (63 f. il, color.) : pdf.

Orientador: Gabriel de Morais Coutinho.

Dissertação (Mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f. 62-63

1. Computação – Teses. 2. Computação quântica – Teses. 3. Operadores hamiltonianos - Teses. 4. Autovalores – Teses. I. Coutinho, Gabriel de Morais. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. III. Título.

CDU 519.6*73(043)

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

HAMILTONIAN DESIGN OF QUANTUM GATES

JULIANA RAMOS NEVES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores(a):

Prof. Gabriel de Morais Coutinho - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Leonardo Teixeira Neves
Departamento de Física - UFMG

Prof. Vinícius Fernandes dos Santos
Departamento de Ciência da Computação - UFMG

Dr. Thomás Jung Spier
Projeto Petrobras Inteligência Artificial - DCC - UFMG

Belo Horizonte, 14 de junho de 2022.

Documento assinado eletronicamente por **Vinicius Fernandes dos Santos**, **Professor do Magistério Superior**, em 29/06/2022, às 16:03, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Thomás Jung Spier**, **Usuário Externo**, em 30/06/2022, às 10:47, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Leonardo Teixeira Neves**, **Professor do Magistério Superior**, em 01/07/2022, às 16:23, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1555078** e o código CRC **4B9D8569**.

Acknowledgments

I am very grateful to Gabriel Coutinho for all his guidance and support throughout these years. I am also grateful to my family and friends for supporting me during this time.

Resumo

Um gate quântico de n qubits é um operador unitário $\mathcal{G}$ de dimensão $2^n \times 2^n$. $\mathcal{G}$ é implementado por um Hamiltoniano, i.e., uma matriz Hermitiana $\mathcal{H}$ tal que $\mathcal{G} = \exp(\mathrm{i}\mathcal{H})$. Existem vários modelos de computação quântica: modelo de circuito, caminhada quântica, adiabático e rede quântica, entre outros. Para todos eles, há uma limitação empírica que permite apenas a implementação de Hamiltonianos de 2-corpos. Isto é, Hamiltonianos da forma:

$$\mathcal{H} = h_0\mathrm{I} + \sum_{\substack{a\in\{x,y,z\}\\1\leq i\leq n}} h_a^i\sigma_a^i + \sum_{\substack{a,b\in\{z,y,z\}\\1\leq i<j\leq n}} h_{ab}^{ij}\sigma_a^i\sigma_b^j;$$

onde σ_a^i é a matriz de Pauli σ_a atuando no i-ésimo qubit, $\sigma_a^i = \mathrm{I}^{\otimes i-1} \otimes \sigma_a \otimes \mathrm{I}^{\otimes n-i}$, e os hs são escalares reais. No entanto, dado um gate $\mathcal{G}$ qualquer, o Hamiltoniano $\mathcal{H} = -\mathrm{iLn}(\mathcal{G})$ normalmente contém interações de k corpos com $k > 2$. Nesse cenário, pode-se explorar a propriedade de que o logaritmo complexo é multivalorado para obter um Hamiltoniano de 2 corpos alternativo. Essa estratégia já se mostrou bem-sucedida em trabalhos anteriores, produzindo Hamiltonianos de 2 corpos para gates que são, a princípio, de 3 corpos: os gates de Toffoli e Fredkin de 3 qubits.

Esta dissertação estuda como derivar Hamiltonianos de 2 corpos para gates quânticos, com foco no gate da Transformada Quântica de Fourier (QFT). A escolha do QFT se deve à sua importância no algoritmo de fatoração de Shor, considerado o algoritmo quântico mais relevante da atualidade. Para tanto, utiliza-se uma técnica da literatura que reduz a busca por um Hamiltoniano de 2 corpos a um problema de autovalor inverso. Essa técnica é aplicada na tentativa de gerar Hamiltonianos para os gates QFT_3 e QFT_4, resultando em uma prova formal de que o QFT_3 não possui Hamiltoniano de 2 corpos. No entanto, a existência de tal Hamiltoniano para o QFT_4 permanece uma questão em aberto, devido aos desafios computacionais que acompanham grandes sistemas de equações. Por fim, o texto discute os próximo passos nessa linha de pesquisa, que incluem a busca por métodos mais eficientes para resolver o problema do autovalor inverso. Isso permitiria investigar se o QFT_n admite um gerador de 2 corpos para n>3.

Palavras-chave: porta quântica; Hamiltoniano de dois corpos; problema inverso de autovalores.

Abstract

An n-qubit gate is a $2^n \times 2^n$ unitary operator $\mathcal{G}$. $\mathcal{G}$ is implemented by a Hamiltonian, i.e., an Hermitian matrix $\mathcal{H}$ such that $\mathcal{G} = \exp(\mathrm{i}\mathcal{H})$. There are several models of quantum computation: circuit model, quantum walk, adiabatic, and quantum network, among others. For all of them, there is an empirical limitation that only allows the implementation of 2-body Hamiltonians. I.e., Hamiltonians of the form

$$\mathcal{H} = h_0\mathrm{I} + \sum_{\substack{a\in\{x,y,z\}\\1\leq i\leq n}} h_a^i\sigma_a^i + \sum_{\substack{a,b\in\{z,y,z\}\\1\leq i<j\leq n}} h_{ab}^{ij}\sigma_a^i\sigma_b^j;$$

where σ_a^i is the Pauli matrix σ_a acting on the ith qubit, $\sigma_a^i = \mathrm{I}^{\otimes i-1} \otimes \sigma_a \otimes \mathrm{I}^{\otimes n-i}$, and the hs are real scalars. However, given an arbitrary gate $\mathcal{G}$, the Hamiltonian $\mathcal{H} = -\mathrm{iLn}(\mathcal{G})$ will likely contain k-body interactions with $k > 2$. In this scenario, one can take advantage of the complex logarithm's multivaluedness to obtain a 2-body Hamiltonian. In fact, previous work has produced 2-body Hamiltonians for gates that are, in principle, 3-body: the 3-qubit Toffoli and Fredkin gates.

This dissertation studies how to derive substitute 2-body Hamiltonians for quantum gates, especially the Quantum Fourier Transform (QFT). We focus on the QFT because it is the main gate involved in Shor's factoring algorithm, which is, in turn, the most influential quantum algorithm of our time. We make use of a technique described in the literature, which reduces finding a 2-body Hamiltonian to solving an inverse eigenvalue problem. We try to solve this problem for the QFT_3 and QFT_4 gates. We show that the QFT_3 does not satisfy the necessary conditions for a 2-body Hamiltonian but leave the QFT_4's 2-bodyness as an open question. This is due to computational challenges that arise when dealing with large systems of equations. Finally, we discuss that a next step would be finding a more efficient way to solve the inverse eigenvalue problem. This would allow one to further investigate whether QFT_n admits a 2-body generator for $n > 3$.

Keywords: quantum gate; 2-body Hamiltonian; inverse eigenvalue problem.

List of Figures

Contents

Chapter 1

Introduction

Moore's law predicted that the number of transistors on integrated circuits would double every two years. The exponential evolution of processors, which has been well described by this law since 1965, is now about to meet a physical limit [19]. As designs shrink to 7/5nm and lesser, quantum mechanics takes place, and classical circuits stop working as intended, whereas electrons may pass through a gate via quantum tunneling [21]. One solution for the eventual failure of Moore's law is to move to a different computing paradigm, such as using quantum mechanics to design circuits instead of standard on-off switches.

It is already known [18] that quantum computation is at least as powerful as classical computation, meaning that any problem which can be efficiently solved on a classical computer can also be efficiently solved on a quantum one. What remains an open question is whether quantum computers might efficiently solve problems that have no efficient classical solution. Some theoretic quantum algorithms suggest a positive answer. For example, Shor's quantum factoring algorithm efficiently solves a problem believed to be not efficiently solvable by a classical computer: the prime factorization problem. The best classical complexity known for prime factorization of an n-bit integer is $\exp(\Theta(n^{1/3}\log^{2/3} n))$ [18]. In contrast, Shor demonstrated that this problem could be efficiently solved in the quantum circuit model using $O(n^2 \log n \log\log n)$ operations [20].

Before we can take advantage of this computational power however, we need to build quantum computers and, more specifically, implement the quantum gates involved in Shor's algorithm. Our current experimental technology only supports interactions between two qubits at a time [3]. This limitation poses a serious obstacle to processing relevant inputs for Shor's algorithm, as it would require more interactions. This issue is not exclusive to Shor's algorithm but applies to all quantum gates. This dissertation explores methods for overcoming this limitation.

To provide an overview of the text, we now introduce several terms that are well-established in the field. We will provide thorough definitions in Section 1.1, but here we offer brief explanations. A qubit is the quantum analog of a classical bit. A matrix $\mathcal{H}$ is said to generate a gate $\mathcal{G}$ if $\mathcal{H} = \exp(\mathrm{i}\mathcal{G})$. We refer to $\mathcal{H}$ as having n-body interactions when it simultaneously affects n qubits.

As mentioned earlier, current technology only supports interactions between two

qubits at a time, i.e., generators with at most 2-body interactions. Due to the multi-valued nature of the complex logarithm, given a generator with prohibitive interactions, it may be possible to construct a feasible alternative generator. This is the problem that this dissertation addresses: finding 2-body generators for gates that are initially n-body. We do this while paying special attention to the Quantum Fourier Transform gate, a central part of Shor's algorithm.

In Section 1.1, we present the necessary theory to understand this text. In Section 1.2, we describe Shor's algorithm and the Quantum Fourier Transform gate to motivate our goal. In Chapter 2, we summarize the prior works in the field. First, we discuss a paper by Banchi et al. titled "Quantum Gate Learning in Qubit Networks: Toffoli Gate without Time-Dependent Control" [3]. The text explores the use of auxiliary qubits to avoid undesired interactions. This is done via a supervised learning procedure that maximizes gate fidelity.

We then describe the work done by Innocent et al. in "Supervised learning of time-independent Hamiltonians for gate design" [9]. In this paper, the authors present both an analytical and a machine learning procedure to replace unwanted interactions. This time, the procedures do not require auxiliary qubits. Their research is what motivated our pursuit of a 2-body generator for the Quantum Fourier Transform.

Finally, we discuss the paper published by Geller et al. titled "Universal quantum simulation with prethreshold superconducting qubits: Single-excitation subspace method" [6]. In this work, they utilize a 2-body 2^n-qubit generator to simulate an n-body n-qubit generator. In our research, we build upon and further develop their findings.

In Chapter 3 we present our results, the main one being a proof that the 3-qubit Quantum Fourier Transform gate does not have a 2-body generator. In Chapter 4, we conduct a comprehensive review of our work, where we address the significance of our findings, the primary challenges encountered during our research, and leave open questions to future research.

1.1 Background

The qubit is the basic unit of quantum information, the quantum analog of a classic bit.

Definition 1.1. (Qubit) A qubit is a 2-dimensional complex vector space and its state is a 1-dimensional subspace.

The state of a qubit is represented by a ket.

Definition 1.2. (Bra-ket notation) A ket, $|\psi\rangle$, is an element of the $\mathbb{C}^{2^n}$ vector space. A bra, $\langle\psi|$, is its conjugate transpose, $\langle\psi| = |\psi\rangle^\dagger$.

Saying that a qubit is in state $|\psi\rangle$ means that it is in the subspace spanned by this vector. In the same way that a classic bit assume values 0 or 1, a qubit can be in states

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{or} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Physically, a qubit can be any two-level quantum system, such as a nuclear spin in a magnetic field or a photon polarization. While not measured, this system may be in a superposition of the states $|0\rangle$ and $|1\rangle$. Where, for example, the states $|0\rangle$ and $|1\rangle$ correspond, respectively, to whether a photon is horizontally or vertically polarized. When in superposition, the qubit is in a linear combination $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ of these vectors. After measurement, the system collapses to state $|0\rangle$ or $|1\rangle$, respectively, with probabilities (see [18])

$$\frac{|\alpha|^2}{||\psi\rangle|^2} \quad \text{and} \quad \frac{|\beta|^2}{||\psi\rangle|^2}.$$

Notation. M^T is the transpose of matrix M whereas $M^\dagger$ is the conjugate transpose of M.

Definition 1.3. (Kronecker product) The Kronecker product of matrices $A_{m\times n}$ and $B_{p\times q}$ is the $mp \times nq$ matrix

$$A \otimes B = \begin{bmatrix} A_{11}B & \dots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \dots & A_{mn}B \end{bmatrix}.$$

The Kronecker product is defined in the same way for vectors,

$$v \otimes u = \begin{pmatrix} v_1 u \\ \vdots \\ v_n u \end{pmatrix}.$$

For simplicity, the $\otimes$ operator is sometimes omitted, and the Kronecker product of two vectors is written as $|\psi_1\rangle \otimes |\psi_2\rangle = |\psi_1\rangle|\psi_2\rangle = |\psi_1\psi_2\rangle$. For example,

$$|0\rangle \otimes |1\rangle = |0\rangle|1\rangle = |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

"Quantum system" is an umbrella term to describe either a single qubit or an ensemble of qubits.

Definition 1.4. (Composite state) If a quantum system is composed of n ordered qubits, each in state $|\psi_i\rangle$, then the system's composite state is represented by $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \ldots \otimes |\psi_n\rangle$, i.e., a vector in $\mathbb{C}^{2^n}$.

Definition 1.5. (Computational basis) The set $\{|x_1 x_2 \ldots x_n\rangle \mid x_i \in \{0,1\}\}$ is an orthonormal basis for the $\mathbb{C}^{2^n}$ space, denominated the computational basis [18].

Now we present some basic properties of the Kronecker product [7].

Lemma 1.1. The conjugate transpose distributes over the Kronecker product,

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger.$$

Proof.

$$(A \otimes B)^\dagger = \begin{bmatrix} \overline{A_{11}}B^\dagger & \ldots & \overline{A_{m1}}B^\dagger \\ \vdots & \ddots & \vdots \\ \overline{A_{1n}}B^\dagger & \ldots & \overline{A_{mn}}B^\dagger \end{bmatrix} = A^\dagger \otimes B^\dagger.$$

□

Lemma 1.2. Trace distributes over Kronecker product,

$$\mathrm{tr}(A \otimes B) = \mathrm{tr}(A)\mathrm{tr}(B).$$

Proof.

$$\begin{aligned} \mathrm{tr}(A \otimes B) &= \mathrm{tr} \begin{bmatrix} A_{11}B & \ldots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \ldots & A_{mn}B \end{bmatrix} \\ &= A_{11}\mathrm{tr}(B) + \ldots + A_{mn}\mathrm{tr}(B) \\ &= \mathrm{tr}(A)\mathrm{tr}(B). \end{aligned}$$

□

Lemma 1.3. If A, B, C, and D have the appropriate dimensions, then

$$(A \otimes B)(C \otimes D) = AC \otimes BD.$$

Proof. Let A and C be, respectively, $m \times n$ and $n \times p$ matrices. Then,

$$\begin{aligned} (A \otimes B)(C \otimes D) &= \begin{bmatrix} A_{11}B & \ldots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \ldots & A_{mn}B \end{bmatrix} \begin{bmatrix} C_{11}D & \ldots & C_{1p}D \\ \vdots & \ddots & \vdots \\ C_{n1}D & \ldots & C_{np}D \end{bmatrix} \\ &= \begin{bmatrix} \sum_i (A_{1i}B)(C_{i1}D) & \ldots & \sum_i (A_{1i}B)(C_{ip}D) \\ \vdots & \ddots & \vdots \\ \sum_i (A_{mi}B)(C_{i1}D) & \ldots & \sum_i (A_{mi}B)(C_{ip}D) \end{bmatrix} \end{aligned}$$

$$
\begin{aligned}
&= \begin{bmatrix} \sum_i (A_{1i}C_{i1})(BD) & \dots & \sum_i (A_{1i}C_{ip})(BD) \\ \vdots & \ddots & \vdots \\ \sum_i (A_{mi}C_{i1})(BD) & \dots & \sum_i (A_{mi}C_{ip})(BD) \end{bmatrix} \\
&= \begin{bmatrix} (AC)_{11}(BD) & \dots & (AC)_{1p}(BD) \\ \vdots & \ddots & \vdots \\ (AC)_{m1}(BD) & \dots & (AC)_{mp}(BD) \end{bmatrix} \\
&= AC \otimes BD.
\end{aligned}
$$

□

Definition 1.6. (Density operator) Let $\{|\psi_i\rangle\}$ be an ensemble of states. The density operator of a system prepared in state $|\psi_i\rangle$ with probability p_i is given by

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|.$$

Representing a system state with a density operator is equivalent to using a vector, but density operators are more convenient when working with statistical ensembles.

Definition 1.7. (Pure state) A quantum system is said to be in a pure state if its density operator can be written as $\rho = |\psi\rangle\langle\psi|$. Otherwise it is in a mixed state.

Definition 1.8. (Reduced density operator) Let A and B be quantum systems with, respectively, n and m qubits. Let C denote the composite quantum system formed by A and B, with density operator ρ_C. The partial trace over B is the map

$$\mathrm{tr}_B(\rho_C) = \sum_i^{2^m} (\mathrm{I}_{2^n} \otimes \langle i|)\rho_C(|i\rangle \otimes \mathrm{I}_{2^n}),$$

where $\{|i\rangle\}$ is the computational basis of $\mathbb{C}^{2^m}$. Furthermore, the reduced density operator of system A is

$$\rho_A = \mathrm{tr}_B(\rho_C).$$

To grasp some intuition about the reduced density operator, consider A and B to be, respectively, in states $|\psi_A\rangle$ and $|\psi_B\rangle$. Then, $\rho_C = (|\psi_A\rangle|\psi_B\rangle)(\langle\psi_B|\langle\psi_A|)$ and the reduced density operator of system A is

$$
\begin{aligned}
\rho_A &= \mathrm{tr}_B((|\psi_A\rangle|\psi_B\rangle)(\langle\psi_B|\langle\psi_A|)) \\
&= \sum_i^{2^m} \mathrm{I}_{2^n}|\psi_A\rangle \otimes \langle i||\psi_B\rangle\langle\psi_B||i\rangle \otimes \langle\psi_A|\mathrm{I}_{2^n} \\
&= |\psi_A\rangle \left(\sum_i^{2^m} \langle i||\psi_B\rangle\langle\psi_B||i\rangle \right) \langle\psi_A| \\
&= |\psi_A\rangle\langle\psi_A| \,\mathrm{tr}(|\psi_B\rangle\langle\psi_B|)
\end{aligned} \tag{1.1}
$$

$$= |\psi_A\rangle\langle\psi_A|,$$

which is exactly what we expected it to be. In (1.1), $\mathrm{tr}(|\psi_B\rangle\langle\psi_B|) = 1$ follows from $|\psi_B\rangle\langle\psi_B|$ being an orthogonal projection.

1.1.1 Linear algebra

In this subsection, we state some basic linear algebra theorems that shall be used in the following sections.

Definition 1.9. (Normal matrix) A normal matrix is a complex square matrix M such that $M^\dagger M = MM^\dagger$ [4].

This concept is important because the main matrices involved in quantum computation, which will be discussed in the next subsection, are normal.

Definition 1.10. (Unitary matrix) A unitary matrix is a complex square matrix U such that $UU^\dagger = U^\dagger U = \mathrm{I}$, i.e., its columns and rows form, each, an orthonormal basis [4].

We can also characterize unitary matrices as follows.

Theorem 1. A complex matrix is unitary if and only if it is normal with unimodular eigenvalues [4].

Another important type of matrices for quantum computing are the Hermitian matrices.

Definition 1.11. (Hermitian matrix) An Hermitian matrix is a complex square matrix H such that $H = H^\dagger$ [4].

Note that an Hermitian matrix is trivially normal, but we can characterize it as follows.

Theorem 2. A complex matrix is Hermitian if and only if it is normal with real eigenvalues [4].

All normal matrices are diagonalizable. Furthermore, they are diagonalized by unitary matrices.

Theorem 3. A complex matrix M is normal if and only if it can be written as $M = VDV^\dagger$, where V is unitary and D is diagonal. Moreover, the columns of V and the diagonal entries of D are, respectively, eigenvectors and eigenvalues of M [4].

Definition 1.12. (Subspace preservation) Let $M \in \mathbb{C}^{n\times n}$. Let X be a subspace of $\mathbb{C}^n$. We say that M preserves X (or that X is invariant to M) if $Mv \in X$ for all $v \in X$ [4].

Theorem 4. Two Hermitian matrices commute if, and only if, they preserve each other's eigenspaces [4].

Definition 1.13. (Permutation matrix) A permutation matrix P is a square binary matrix with one 1 in each row and column and 0s elsewhere. For some matrix M, PM is a permutation of its rows and MP of its columns [4].

For a given normal matrix M, there are many pairs V, D that satisfy $M = VDV^\dagger$. Let $\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$ be the set of distinct eigenvalues of M, each with multiplicity n_i. Let $\{v_{i,1}, \ldots, v_{i,n_i}\}$ be an orthonormal set of eigenvectors associated with eigenvalue λ_i. Then,

$$D = \begin{bmatrix} \lambda_{1,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_{1,n_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_{2,1} & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_{d,n_d} \end{bmatrix} \tag{1.2}$$

and

$$V = \begin{bmatrix} v_{1,1} & \ldots & v_{1,n_1} & v_{2,1} & \ldots & v_{d,n_d} \end{bmatrix} \tag{1.3}$$

are one such pair. Also, for any permutation matrix P, $D' = P^\dagger DP$ and $V' = VP$ satisfy $M = V'D'V'^\dagger$. Moreover, if $n_i > 1$, there are infinitely many choices for the set $\{v_{i,1}, \ldots, v_{1,n_i}\}$ and, therefore, for V.

Theorem 5. Two Hermitian matrices M and N commute if, and only if, there exist unitary matrix V and diagonal matrices D_M and D_N such that $M = VD_MV^\dagger$ and $N = VD_NV^\dagger$ [4].

Theorem 6. Let $\{\lambda_1, \lambda_2, \ldots, \lambda_d\}$ be the set of distinct eigenvalues of an Hermitian matrix M, each with multiplicity n_i. Fix specific V and D that diagonalize M, i.e. $M = VDV^\dagger$. Then, all Hermitian matrices N that commute with M can be written as $N = VBV^\dagger$, where B is a diagonal block matrix, and the ith block is $n_i \times n_i$-dimensional [4].

This theorem will be important in Sections 3.2 and 3.3. To get a better intuition about it, consider D and V as defined in (1.2) and (1.3), respectively. Let $s_i = \{v_{i,1}, \ldots, v_{i,n_i}\}$ be an orthonormal set of eigenvectors associated with eigenvalue λ_i. Then, $S_i = \text{span}(s_i)$ is an eigenspace of M and, because s_i is orthonormal, s_i is a basis for it. Theorem 4 implies that if M and N commute, then $Nv_{i,j} \in S_i$ for $1 \leq j \leq n_i$. In

other words, $Nv_{i,j}$ must be a linear combination of the elements in s_i. This is achieved by writing N as

$$N = \underbrace{\begin{bmatrix} v_{1,1} & \dots & v_{1,n_1} & \dots & v_{d,1} & \dots & v_{d,n_d} \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} b_{1,1,1} & \dots & b_{1,1,n_1} & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \dots & 0 & 0 & 0 \\ \overline{b_{1,1,n_1}} & \dots & b_{1,n_1,n_1} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & b_{d,1,1} & \dots & b_{d,1,n_d} \\ 0 & 0 & 0 & 0 & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \overline{b_{d,1,n_d}} & \dots & b_{d,n_d,n_d} \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} v_{1,1}^\dagger \\ \vdots \\ v_{1,n_1}^\dagger \\ \vdots \\ v_{d,1}^\dagger \\ \vdots \\ v_{d,n_d}^\dagger \end{bmatrix}}_{V^\dagger}.$$

This way we have

$$\begin{aligned} Nv_{i,1} &= b_{i,1,1}v_{i,1} + \ldots + \overline{b_{i,1,n_i}}v_{i,n_i}, \\ &\vdots \\ Nv_{i,n_i} &= b_{i,1,n_i}v_{i,1} + \ldots + b_{i,n_i,n_i}v_{i,n_i}. \end{aligned}$$

Definition 1.14. (Matrix exponential) The exponential of a square matrix M is given by the power series

$$\exp(M) = \sum_{k=0}^{\infty} \frac{1}{k!} M^k.$$

Lemma 1.4. For a normal matrix M such that

$$M = V \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n \end{bmatrix} V^\dagger,$$

we have

$$\exp(M) = V \begin{bmatrix} \exp(\lambda_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \exp(\lambda_n) \end{bmatrix} V^\dagger.$$

Proof.

$$\begin{aligned} \exp(M) &= \sum_{k=0}^{\infty} \frac{1}{k!} (VDV^\dagger)^k \\ &= V \left(\sum_{k=0}^{\infty} \frac{1}{k!} D^k \right) V^\dagger \\ &= V \begin{bmatrix} \exp(\lambda_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \exp(\lambda_n) \end{bmatrix} V^\dagger. \end{aligned}$$

□

Proposition 1.1. If Hermitian matrices A and B commute, then $\exp(A)\exp(B) = \exp(A+B)$.

Proof. By theorem 5, let the commuting matrices be diagonalized as

$$A = V \begin{bmatrix} a_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & a_n \end{bmatrix} V^\dagger$$

and

$$B = V \begin{bmatrix} b_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & b_n \end{bmatrix} V^\dagger.$$

Then, by lemma 1.4 we have

$$\begin{aligned} \mathrm{e}^A \mathrm{e}^B &= V \begin{bmatrix} \mathrm{e}^{a_1} & 0 & \dots & 0 \\ 0 & \mathrm{e}^{a_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathrm{e}^{a_n} \end{bmatrix} V^\dagger V \begin{bmatrix} \mathrm{e}^{b_1} & 0 & \dots & 0 \\ 0 & \mathrm{e}^{b_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathrm{e}^{b_n} \end{bmatrix} V^\dagger \\ &= V \begin{bmatrix} \mathrm{e}^{a_1}\mathrm{e}^{b_1} & 0 & \dots & 0 \\ 0 & \mathrm{e}^{a_2}\mathrm{e}^{b_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathrm{e}^{a_n}\mathrm{e}^{b_n} \end{bmatrix} V^\dagger \\ &= V \begin{bmatrix} \mathrm{e}^{a_1+b_1} & 0 & \dots & 0 \\ 0 & \mathrm{e}^{a_2+b_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathrm{e}^{a_n+b_n} \end{bmatrix} V^\dagger \\ &= \mathrm{e}^{A+B}. \end{aligned}$$

□

Theorem 7. (Euler's formula) $\exp(\mathrm{i}x) = \cos x + \mathrm{i}\sin x$, $\forall x \in \mathbb{R}$. [15]

Consider $A = VDV^\dagger$ and let $B = \exp(A)$. Then, A is called a logarithm of B. Euler's formula implies that B has infinitely many logarithms. In fact,

$$C = V \begin{bmatrix} \lambda_1 + 2k_1\pi\mathrm{i} & 0 & \dots & 0 \\ 0 & \lambda_2 + 2k_2\pi\mathrm{i} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n + 2k_n\pi\mathrm{i} \end{bmatrix} V^\dagger$$

is a logarithm of B for any $k_i \in \mathbb{Z}$, since

$$\begin{aligned}\exp(C) &= \exp\left(A + V\begin{bmatrix} 2k_1\pi\mathrm{i} & 0 & \dots & 0 \\ 0 & 2k_2\pi\mathrm{i} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 2k_n\pi\mathrm{i}\end{bmatrix}V^\dagger\right) \\ &= \exp(A)V\begin{bmatrix} \exp(2k_1\pi\mathrm{i}) & 0 & \dots & 0 \\ 0 & \exp(2k_2\pi\mathrm{i}) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \exp(2k_n\pi\mathrm{i})\end{bmatrix}V^\dagger \\ &= B \cdot \mathrm{I}.\end{aligned}$$

To make the logarithm uniquely determined, we use the principal value, $\mathrm{Ln}(x)$, for each eigenvalue x. This is the logarithm of x whose imaginary part lies in the interval $(-\pi, \pi]$.

1.1.2 Hamiltonians

Quantum gates are the building blocks of quantum circuits, responsible for performing calculations and altering qubit states. It is postulated by quantum mechanics that these gates are unitary operators [18]. The state $|\psi\rangle$ of an n-qubit system at time t_1 is, therefore, related to the state $|\psi'\rangle$ of the system at time t_2 by a unitary matrix $\mathcal{G} \in \mathbb{C}^{2^n \times 2^n}$:

$$|\psi'\rangle = \mathcal{G}|\psi\rangle.$$

Due to $\mathcal{G}$'s unitarity, its eigenvalues lie on the complex unit circle and $\mathcal{G}$ can be orthogonally diagonalized as

$$\mathcal{G} = VDV^\dagger = V\begin{bmatrix} \mathrm{e}^{\mathrm{i}k_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \mathrm{e}^{\mathrm{i}k_{2^n}}\end{bmatrix}V^\dagger, \quad k_i \in \mathbb{R}.$$

Let $\mathcal{H}$ be an Hermitian matrix of the form

$$\mathcal{H} = V\begin{bmatrix} k_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & k_{2^n}\end{bmatrix}V^\dagger = -\mathrm{i}V\mathrm{Ln}(D)V^\dagger,$$

then we say that $\mathcal{H} = -\mathrm{i}\mathrm{Ln}(\mathcal{G})$ and $\mathcal{G}$ can be written as $\mathcal{G} = \exp(\mathrm{i}\mathcal{H})$. Any Hermitian matrix $\mathcal{H}'$ such that $\mathcal{G} = \exp(\mathrm{i}\mathcal{H}')$ is called a gate generator or Hamiltonian for $\mathcal{G}$ [2], while $\mathcal{H}$ is called the principal generator.

Definition 1.15. (Pauli matrices) The Pauli matrices are the following four unitary and Hermitian matrices.

$$\begin{aligned} \sigma_0 = \mathrm{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad & \sigma_1 = \sigma_x = \mathrm{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ \sigma_2 = \sigma_y = \mathrm{Y} = \begin{bmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{bmatrix} \qquad & \sigma_3 = \sigma_z = \mathrm{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{aligned} \tag{1.4}$$

Let $\mathcal{S}$ be the set of the 4^n distinct Kronecker products of n Pauli matrices,

$$\mathcal{S} = \{\sigma_{i_1} \otimes \sigma_{i_2} \otimes \ldots \otimes \sigma_{i_n}\}, \;\; i_j \in \{0, 1, 2, 3\}. \tag{1.5}$$

Consider the vector space over $\mathbb{R}$ of complex Hermitian matrices with size $2^n \times 2^n$ and define its inner product as $\langle A, B\rangle = \mathrm{tr}(A^\dagger B)$. Notice that this space is 4^n dimensional: we have 2^n coordinates coming from the real diagonal entries and $4^n - 2^n$ coordinates coming from the non-diagonal complex entries. Then, the inner product of two elements of $\mathcal{S}$ is given by

$$\begin{aligned} \mathrm{tr}((\sigma_{a_1} \otimes \sigma_{a_2} \otimes \ldots \sigma_{a_n})^\dagger(\sigma_{b_1} \otimes \sigma_{b_2} \otimes \ldots \sigma_{b_n})) &= \mathrm{tr}(\sigma_{a_1}\sigma_{b_1} \otimes \sigma_{a_2}\sigma_{b_2} \otimes \ldots \otimes \sigma_{a_n}\sigma_{b_n}) \\ &= \mathrm{tr}(\sigma_{a_1}\sigma_{b_1})\mathrm{tr}(\sigma_{a_2}\sigma_{b_2}) \ldots \mathrm{tr}(\sigma_{a_n}\sigma_{b_n}). \end{aligned}$$

Since $\mathrm{tr}(\sigma_a\sigma_b) = 0 \Leftrightarrow \sigma_a \neq \sigma_b$, $\mathcal{S}$ is an orthogonal basis with real coefficients for this space. Hence, the generator of any gate acting on n qubits can be written as an unique linear combination of the elements in $\mathcal{S}$.

Definition 1.16. (K-body interaction) An element of $\mathcal{S}$ with k factors distinct from I is called a k-body interaction, meaning that it acts non-trivially on k qubits.

Notation. We use σ_a^i to represent the Pauli matrix σ_a acting on the ith qubit. This is the Kronecker product of n Pauli matrices where all factors are the identity matrix, except for the ith one, which is σ_a,

$$\sigma_a^i = \mathrm{I}^{\otimes i-1} \otimes \sigma_a \otimes \mathrm{I}^{\otimes n-i}.$$

Here are some examples to clarify this notation. For $n = 3$, we have

$$\sigma_x^2 = \mathrm{I} \otimes \mathrm{X} \otimes \mathrm{I},$$

which is a 1-body interaction. For $n = 4$ we have

$$\sigma_x^1\sigma_y^3 = \mathrm{X} \otimes \mathrm{I} \otimes \mathrm{Y} \otimes \mathrm{I},$$

which is a 2-body interaction. The main goal of this dissertation is to study how to derive a substitute 2-body generator for a given gate.

1.2 Motivation

Quantum computation can be represented by the diagram in Figure 1.1 [11]. There are several models of quantum computation: circuit model, quantum walk, adiabatic and quantum network, among others.

$$\text{input} \longrightarrow \boxed{\text{encode}} \longrightarrow |\psi_{in}\rangle \longrightarrow U \longrightarrow |\psi_{out}\rangle \longrightarrow \boxed{\text{decode}} \longrightarrow \text{result}$$

Figure 1.1: Diagram for quantum computing.

For all these models, there is an empirical limitation that only allows the implementation of 2-body Hamiltonians. I.e., Hamiltonians of the form

$$h_0 \mathrm{I} + \sum_{\substack{a\in\{x,y,z\}\\ 1\le i\le n}} h_a^i \sigma_a^i + \sum_{\substack{a,b\in\{z,y,z\}\\ 1\le i<j\le n}} h_{ab}^{ij} \sigma_a^i \sigma_b^j;$$

where the hs are the real coefficients of basis $\mathcal{S}$, called coupling strengths [3]. Given an arbitrary gate $\mathcal{G}$, the Hamiltonian $-\mathrm{iLn}(\mathcal{G})$ will likely contain k-body interactions with $k > 2$ [9], which can not be implemented by current technology.

Each model tackles this limitation in a different way. For example, the quantum circuit model, the most prominent one, decomposes an n-qubit gate as a product of universal 2-qubit gates. This way, each unit gate automatically has a 2-body Hamiltonian. To illustrate this, we will use the Quantum Fourier Transform (QFT). The QFT will be thoroughly presented later. For now, just consider the following 3-qubit gate.

$$\mathrm{QFT}_3 = \tfrac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \exp(\mathrm{i}\pi/4) & \mathrm{i} & \exp(3\mathrm{i}\pi/4) & -1 & \exp(-3\mathrm{i}\pi/4) & -\mathrm{i} & \exp(-\mathrm{i}\pi/4) \\ 1 & \mathrm{i} & -1 & -\mathrm{i} & 1 & \mathrm{i} & -1 & -\mathrm{i} \\ 1 & \exp(3\mathrm{i}\pi/4) & -\mathrm{i} & \exp(\mathrm{i}\pi/4) & -1 & \exp(-\mathrm{i}\pi/4) & \mathrm{i} & \exp(-3\mathrm{i}\pi/4) \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \exp(-3\mathrm{i}\pi/4) & \mathrm{i} & \exp(-\mathrm{i}\pi/4) & -1 & \exp(\mathrm{i}\pi/4) & -\mathrm{i} & \exp(3\mathrm{i}\pi/4) \\ 1 & -\mathrm{i} & -1 & \mathrm{i} & 1 & -\mathrm{i} & -1 & \mathrm{i} \\ 1 & \exp(-\mathrm{i}\pi/4) & -\mathrm{i} & \exp(-3\mathrm{i}\pi/4) & -1 & \exp(3\mathrm{i}\pi/4) & \mathrm{i} & \exp(\mathrm{i}\pi/4) \end{bmatrix}$$

This gate can be implemented by the quantum circuit in Figure 1.2, where the H and R nodes are universal gates with 1 and 2 qubits, respectively.

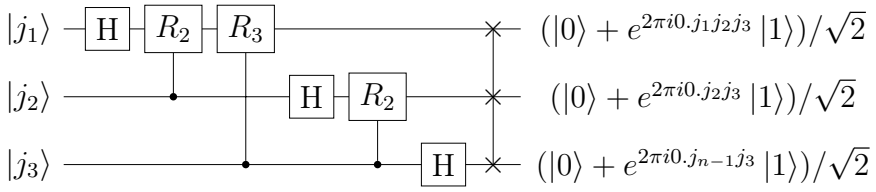

Figure 1.2: Quantum circuit for the QFT_3.

Decomposing an n-qubit gate into this product is a costly process. Therefore, it would be advantageous if we had a 2-body Hamiltonian for the QFT3. This way, we could replace this circuit with a single gate. Unfortunately, QFT3's principal generator contains 3-body interactions,

$$\begin{aligned}\mathcal{H} = \frac{1}{32}(5\mathrm{I} + \sigma_x^1 + (2-\sqrt{2})\sigma_x^2 + (1-\sqrt{2})\sigma_z^2 + \sigma_z^3 - \sqrt{2}\sigma_x^1\sigma_x^2 - (1+\sqrt{2})\sigma_x^1\sigma_z^2 + \sigma_x^1\sigma_z^3 \\ -(2+\sqrt{2})\sigma_x^2\sigma_z^3 + 2\sqrt{2}\sigma_y^1\sigma_y^3 - 2\sqrt{2}\sigma_z^1\sigma_x^3 + (1-\sqrt{2})\sigma_z^2\sigma_z^3 - \sqrt{2}\sigma_x^1\sigma_x^2\sigma_z^3 \\ -(1+\sqrt{2})\sigma_x^1\sigma_z^2\sigma_z^3 + 2\sqrt{2}\sigma_y^1\sigma_y^2\sigma_x^3 + 2\sqrt{2}\sigma_z^1\sigma_y^2\sigma_y^3).\end{aligned}$$

By taking advantage of logarithm's multivaluedness, it might be possible to construct a 2-body $\mathcal{G}$ generator for an initially n-body gate. A procedure that allows general quantum gate synthesis using only 2-body interactions would be a significant progress on the aim of practically applying quantum computation. We will dive into how the literature has tried to achieve such a procedure in the next chapter. For now, we will motivate why we chose to study this problem specifically for the QFT.

If one were to begin by constructing a procedure for a specific gate instead of a general one, the Quantum Fourier Transform (QFT) would be a good starting point because this gate has many interesting applications, including Shor's quantum factoring, arguably the most important quantum algorithm.

The Discrete Fourier Transform (DFT) is a mapping $f : \mathbb{C}^n \to \mathbb{C}^n$ and is the equivalent of the continuous Fourier Transform for a finite sequence of data. The QFT is simply the DFT applied to a quantum state, i.e., a quantum gate for performing a Fourier transform of quantum mechanical amplitudes. Unfortunately, it does not speed up the task of computing the DFT of classical data, which would improve all its applications (signal analysis, data compression, partial differential equations, polynomial multiplication, etc). However, it does allow us to speed up the factorization of an n-bit integer via Shor's Algorithm from $\exp(\Theta(n^{1/3}\log^{2/3} n))$ operations, which is the current best classical complexity [18], to $O(n^3)$.

For $0 \leq j \leq 2^n - 1$, let $|j\rangle_n = |b_{n-1}\rangle \ldots |b_1\rangle|b_0\rangle$, where $b_{n-1}\ldots b_1 b_0$ is the binary representation of j with n bits,

$$j = b_{n-1}2^{n-1} + \ldots + b_1 2^1 + b_0 2^0,\ b \in \{0,1\}.$$

Then, $\{|0\rangle_n, \ldots, |2^n-1\rangle_n\}$ is the computational basis of $\mathbb{C}^{2^n}$ and the action of the n-qubit QFT on a basis state is given by (see [18])

$$\mathrm{QFT}_n|j\rangle_n = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \mathrm{e}^{2\pi \mathrm{i} jk/2^n} |k\rangle_n.$$

Its matrix form,

$$\mathrm{QFT}_n = \frac{1}{\sqrt{2^n}} \begin{bmatrix} F_{0,0} & \ldots & F_{0,2^n-1} \\ \vdots & \ddots & \vdots \\ F_{2^n-1,0} & \ldots & F_{2^n-1,2^n-1} \end{bmatrix}, \; F_{k,j} = \exp(2\pi \mathrm{i} jk/2^n);$$

is the matrix we try to implement with 2-body interactions in Chapter 3.

Now we present Shor's algorithm to motivate why we chose the QFT as starting point. Shor's algorithm solves the problem of finding a prime factor of a composite integer in $O(n^3)$. This is remarkable because RSA cryptography, one of the most relevant cryptosystems, relies on the difficulty of solving this problem. The algorithm is presented in Algorithm 1. Theorems 8 and 9 provide some intuition of why, with high probability, the algorithm works (see [18]).

Theorem 8. *N is an odd composite positive integer and $N = p_1^{\alpha_1} \ldots p_m^{\alpha_m}$ is its prime factorization. Let $1 < x < N$ be a randomly chosen coprime of N. Let r be the smallest positive integer such that $x^r = 1 \pmod N$. The probability that r is even, and that $x^{r/2} \neq N-1 \pmod N$, is greater or equal than $1 - 1/2^m$.*

Theorem 9. *N is a composite integer. If $1 < y < N-1$ is such that $y^2 = 1 \pmod N$, then at least one of $\gcd(y-1, N)$ and $\gcd(y+1, N)$ is a non-trivial factor of N.*

The QFT is used in the order finding procedure inside Shor's algorithm to find the order r of $x \pmod N$, where N is an n-bit odd composite integer and $1 < x < N$ is a randomly chosen coprime of N. The order finding procedure works by defining a unitary operator U such that its eigenvalues have r as denominator and then retrieving r from it. U is defined as

$$U|j\rangle_n = \begin{cases} |xj \pmod N\rangle_n & \text{if } 0 \leq j \leq N-1, \\ |j\rangle_n & \text{if } N \leq j \leq 2^n - 1. \end{cases}$$

For any integer $0 \leq s \leq r-1$, the vector

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \mathrm{e}^{-2\pi \mathrm{i} sk/r} |x^k \pmod N\rangle_n$$

is an eigenvector of U with eigenvalue $\exp(2\pi \mathrm{i} s/r)$, since

$$U|u_s\rangle = \frac{U|1 \pmod N\rangle_n + \ldots + \mathrm{e}^{-2\pi \mathrm{i} s(r-1)/r} U|x^{r-1} \pmod N\rangle_n}{\sqrt{r}}$$

Algorithm 1 Shor's quantum factoring

Input: n-bit composite integer N.
Output: Non-trivial factor of N.
if *N is even* **then**
  return *2*
// The following check is performed classically in $O(n^3)$ operations [18]
if *$N = a^b$ for some $a, b \geq 2$* **then**
  return a
Randomly choose x in the interval $1 < x < N$;
// The gcd is computed classically in $O(n^3)$ operations with Euclid's algorithm.
if $\gcd(x, N) > 1$ **then**
  return $\gcd(x, N)$
Use the order finding procedure (Algorithm 2) to obtain the order r of $x \pmod N$ in $O(n^3)$;
if *r is even and $x^{r/2} \neq N - 1 \pmod N$* **then**
  if $\gcd(x^{r/2} - 1, N) > 1$ **then**
    return $\gcd(x^{r/2} - 1, N)$
  else
    return $\gcd(x^{r/2} + 1, N)$
return *Error*

$$
\begin{aligned}
&= \frac{|x \ (\mathrm{mod} N)\rangle_n + \ldots + \mathrm{e}^{-2\pi \mathrm{i} s(r-1)/r}|x^r \ (\mathrm{mod} N)\rangle_n}{\sqrt{r}} \\
&= \frac{\mathrm{e}^{-2\pi \mathrm{i} s(r-1)/r}|1 \ (\mathrm{mod} N)\rangle_n + \ldots + \mathrm{e}^{-2\pi \mathrm{i} s(r-2)/r}|x^{r-1} \ (\mathrm{mod} N)\rangle_n}{\sqrt{r}} \\
&= \frac{e^{2\pi \mathrm{i} s/r}\mathrm{e}^{-2\pi \mathrm{i} s}|1 \ (\mathrm{mod} N)\rangle_n + \ldots + e^{2\pi \mathrm{i} s/r}\mathrm{e}^{-2\pi \mathrm{i} s(r-1)/r}|x^{r-1} \ (\mathrm{mod} N)\rangle_n}{\sqrt{r}} \\
&= \frac{e^{2\pi \mathrm{i} s/r}|1 \ (\mathrm{mod} N)\rangle_n + \ldots + e^{2\pi \mathrm{i} s/r}\mathrm{e}^{-2\pi \mathrm{i} s(r-1)/r}|x^{r-1} \ (\mathrm{mod} N)\rangle_n}{\sqrt{r}} \\
&= \mathrm{e}^{2\pi \mathrm{i} s/r}|u_s\rangle.
\end{aligned}
$$

Now we determine the denominator of an eigenvalue $\exp(2\pi \mathrm{i} s/r)$ to obtain r. Note that $\sum_{s=0}^{r-1} \mathrm{e}^{-2\pi \mathrm{i} sk/r}$ is a geometric series for $k \neq 0$. Thus,

$$
\sum_{s=0}^{r-1} \mathrm{e}^{-2\pi \mathrm{i} sk/r} = \begin{cases} r & \text{if } k = 0, \\ \dfrac{1 - \mathrm{e}^{-2\pi \mathrm{i} k}}{1 - \mathrm{e}^{-2\pi \mathrm{i} k/r}} = 0 & \text{if } k \neq 0 \end{cases}
$$

and

$$
\begin{aligned}
\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle &= \frac{1}{r} \sum_{k=0}^{r-1} \left(\sum_{s=0}^{r-1} \mathrm{e}^{-2\pi \mathrm{i} sk/r} \right) |x^k \ (\mathrm{mod} N)\rangle_n \\
&= \frac{1}{r} r |x^0 \ (\mathrm{mod} N)\rangle_n \\
&= |1\rangle_n.
\end{aligned}
$$

Therefore, if an n-qubit register is prepared in state $|1\rangle_n$, it is actually in a superposition of eigenvectors of U.

Let I be the 2×2 identity matrix, H be the Hadamard gate,

$$\mathrm{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

and $\mathcal{U}$ be a unitary operator such that

$$\mathcal{U}|j\rangle_p|k\rangle_n = |j\rangle_p(U^j|k\rangle_n).$$

The order finding procedure consists of applying the gate $\mathcal{G} = (\mathrm{QFT}_p^\dagger \otimes \mathrm{I}^{\otimes n})\mathcal{U}(\mathrm{H}^{\otimes p} \otimes \mathrm{I}^{\otimes n})$ to the input state $|\psi_{in}\rangle = |0\rangle_p|1\rangle_n$, where p is selected according to the desired precision,

$$\begin{aligned}
\mathcal{G}|\psi_{in}\rangle &= (\mathrm{QFT}_p^\dagger \otimes \mathrm{I}^{\otimes n})\mathcal{U}(\mathrm{H}^{\otimes p} \otimes \mathrm{I}^{\otimes n})|0\rangle_p|1\rangle_n \\
&= \frac{1}{\sqrt{2^p}}(\mathrm{QFT}_p^\dagger \otimes \mathrm{I}^{\otimes n})\mathcal{U} \sum_{j=0}^{2^p-1} |j\rangle_p|1\rangle_n \\
&= \frac{1}{\sqrt{2^p}}(\mathrm{QFT}_p^\dagger \otimes \mathrm{I}^{\otimes n}) \sum_{j=0}^{2^p-1} |j\rangle_p U^j|1\rangle_n \\
&= \frac{1}{\sqrt{2^p}}(\mathrm{QFT}_p^\dagger \otimes \mathrm{I}^{\otimes n}) \sum_{j=0}^{2^p-1} |j\rangle_p U^j \left(\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle\right) \\
&= \frac{1}{\sqrt{2^p}}\frac{1}{\sqrt{r}}(\mathrm{QFT}_p^\dagger \otimes \mathrm{I}^{\otimes n}) \sum_{j=0}^{2^p-1} |j\rangle_p \sum_{s=0}^{r-1} \mathrm{e}^{2\pi \mathrm{i} sj/r}|u_s\rangle.
\end{aligned}$$

Note that

$$\mathrm{QFT}_p^\dagger \left(\frac{1}{\sqrt{2^p}} \sum_{k=0}^{2^p-1} \mathrm{e}^{2\pi \mathrm{i} jk/2^p}|k\rangle_p\right) = |j\rangle_p.$$

So, by reorganizing the expression we get

$$\begin{aligned}
\mathcal{G}|\psi_{in}\rangle &= \frac{1}{\sqrt{2^p}}\frac{1}{\sqrt{r}}(\mathrm{QFT}_p^\dagger \otimes \mathrm{I}^{\otimes n}) \sum_{j=0}^{2^p-1} |j\rangle_p \sum_{s=0}^{r-1} \mathrm{e}^{2\pi \mathrm{i} sj/r}|u_s\rangle \\
&= \frac{1}{\sqrt{r}}(\mathrm{QFT}_p^\dagger \otimes \mathrm{I}^{\otimes n}) \sum_{s=0}^{r-1} \left(\frac{1}{\sqrt{2^p}} \sum_{j=0}^{2^p-1} \mathrm{e}^{2\pi \mathrm{i} sj/r}|j\rangle_p\right)|u_s\rangle \\
&= \frac{1}{\sqrt{r}}(\mathrm{QFT}_p^\dagger \otimes \mathrm{I}^{\otimes n}) \sum_{s=0}^{r-1} \left(\frac{1}{\sqrt{2^p}} \sum_{j=0}^{2^p-1} \mathrm{e}^{2\pi \mathrm{i}(2^p s/r)j/2^p}|j\rangle_p\right)|u_s\rangle \\
&= \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |2^p s/r\rangle_p|u_s\rangle.
\end{aligned}$$

The final state is

$$|\psi_{out}\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |2^p s/r\rangle_p|u_s\rangle,$$

where $|2^p s/r\rangle_p$ contains the binary representation of s/r with p bits,

$$|2^p s/r\rangle = |b_1\rangle|b_2\rangle \dots |b_p\rangle \implies s/r \approx b_1 2^{-1} + b_2 2^{-2} + \dots + b_p 2^{-p}.$$

Lastly, one can use the continued fractions algorithm [18] to obtain r from $|\psi_{out}\rangle$. The order finding procedure is summarized in Algorithm 2.

Algorithm 2 Order finding

Input: n-bit integer N.
Co-prime x such that $1 < x < N$.
Precision p.
Output: Order r of $x \pmod N$.
$\mathcal{U} \leftarrow \left[|0\rangle_p|x^0 0 \pmod N\rangle_n \quad \dots \quad |2^p - 1\rangle_p|2^n - 1\rangle_n\right]$;
$|\psi_{out}\rangle \leftarrow (\mathrm{QFT}_p^\dagger \otimes \mathrm{I}^{\otimes n})\mathcal{U}(\mathrm{H}^{\otimes p} \otimes \mathrm{I}^{\otimes n})|0\rangle_p|1\rangle_n$;
Use the continued fractions algorithm [18] to obtain r from $|\psi_{out}\rangle$;
return r

Chapter 2

Related work

2.1 Quantum gate learning in qubit networks

In [3] the authors pose the following question: given a quantum gate $\mathcal{G}_Q$ acting on n qubits whose generator $\mathcal{H}$ contains undesired k-body interactions, $k > 2$, is it possible to build a generator $\mathcal{H}'$ comprising only desired interactions such that $e^{i\mathcal{H}'} = \mathcal{G}_Q \otimes \mathcal{G}_A$ (where $\mathcal{G}_A$ is a suitable unitary)? This way we could trade expensive interactions for auxiliary qubits when implementing a quantum circuit. Consider $\mathcal{H}'$ as the parameterized sum

$$\mathcal{H}' = \sum_{\sigma \in \mathcal{I}} h_\sigma \sigma, \tag{2.1}$$

with $\mathcal{I}$ being the set of allowed interactions. Let Q be the set of target qubits and A be the one with auxiliary qubits. Let $\mathcal{E}[|\psi\rangle]$ denote the resulting mixed state on Q after applying $e^{i\mathcal{H}'}$ to the system in a separable initial state $|\psi\rangle = |\psi_Q\rangle \otimes |\psi_A\rangle$,

$$\mathcal{E}[|\psi\rangle] = \mathrm{tr}_A(e^{i\mathcal{H}'}|\psi\rangle\langle\psi|e^{-i\mathcal{H}'}).$$

The authors present a supervised learning inspired procedure to determine the variables h in (2.1) that make $\mathcal{E}[|\psi\rangle]$ as close as possible to $\mathcal{G}_Q|\psi_Q\rangle\langle\psi_Q|\mathcal{G}_Q^\dagger$. They demonstrate its efficacy by finding 2-body generators with one auxiliary qubit for the Toffoli and Fredkin gates. The Toffoli gate is a doubly-controlled not gate, i.e. it flips the third qubit if the first two are in state $|11\rangle$. Its matrix is given by

$$\mathcal{G}_{\text{Toff}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

The Toffoli gate, together with the Hadamard gate, form a universal gate set for quantum computation [1]. The Fredkin gate is a three-qubit gate which swaps the last two qubits conditionally to the first qubit being in state $|1\rangle$. Its matrix is given by

$$\mathcal{G}_{\text{Fred}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The Fredkin gate is universal for reversible classical computation [18]. Toffoli and Fredkin gates in principle require 3-body interactions, since the generators obtained by taking their Ln are

$$\mathcal{H}_{\text{Toff}} = \frac{\pi}{8}(\mathrm{I} - \sigma_x^3 - \sigma_z^1 - \sigma_z^2 + \sigma_z^1\sigma_x^3 + \sigma_z^1\sigma_z^2 + \sigma_z^2\sigma_x^3 - \sigma_z^1\sigma_z^2\sigma_x^3)$$

and

$$\mathcal{H}_{\text{Fred}} = \frac{\pi}{8}(\mathrm{I} - \sigma_z^1 - \sigma_x^2\sigma_x^3 - \sigma_y^2\sigma_y^3 - \sigma_z^2\sigma_z^3 + \sigma_z^1\sigma_x^2\sigma_x^3 + \sigma_z^1\sigma_y^2\sigma_y^3 + \sigma_z^1\sigma_z^2\sigma_z^3).$$

Supervised learning consists of approximating a function f based on a set of data $\{x\}$ for which the value of $f(x)$ is known. The algorithms begin with a parameterized function $g(h)$ and find the set of parameters h that make $g(h, x)$ best approximate $f(x)$. In this fashion, the procedure presented in the article [3] approximates $\mathcal{E}[|\psi\rangle]$ to $\mathcal{G}_Q|\psi_Q\rangle\langle\psi_Q|\mathcal{G}_Q^\dagger$.

The state fidelity between density operators ρ and ϕ is given by

$$\mathrm{F}(\rho, \phi) = \left(\mathrm{tr}\left(\sqrt{\sqrt{\rho}\phi\sqrt{\rho}}\right)\right)^2$$

and is a measure of how distinguishable the states are. $\mathrm{F}(\rho, \phi) = 1$ means they are identical, while $\mathrm{F}(\rho, \phi) = 0$ indicates one measurement can distinguish them perfectly [13]. When ρ is a pure state, $\rho = |\psi\rangle\langle\psi|$, it also happens to be an orthogonal projection. Thus, $\sqrt{\rho} = \rho$ and $\mathrm{tr}(\rho) = 1$. In this case, $\mathrm{F}(\rho, \phi)$ can be rewritten as

$$\begin{aligned} \mathrm{F}(\rho, \phi) &= \left(\mathrm{tr}\left(\sqrt{\sqrt{\rho}\phi\sqrt{\rho}}\right)\right)^2 \\ &= \left(\mathrm{tr}\left(\sqrt{\rho\phi\rho}\right)\right)^2 \\ &= \left(\mathrm{tr}\left(\sqrt{|\psi\rangle\langle\psi|\phi|\psi\rangle\langle\psi|}\right)\right)^2 \\ &= \left(\mathrm{tr}\left(\sqrt{\langle\psi|\phi|\psi\rangle}\sqrt{|\psi\rangle\langle\psi|}\right)\right)^2 \\ &= \langle\psi|\phi|\psi\rangle(\mathrm{tr}\left(\sqrt{\rho}\right))^2 \\ &= \langle\psi|\phi|\psi\rangle(\mathrm{tr}(\rho))^2 \end{aligned}$$

$$= \langle\psi|\phi|\psi\rangle.$$

The gate fidelity,

$$\mathcal{F}_{G,J}(\rho) = \mathrm{F}(G\rho G^\dagger, J\rho J^\dagger), \tag{2.2}$$

measures how similar is the action of gates G and J on state ρ.

The average gate fidelity, $\overline{\mathcal{F}}_{G,J} = \int \mathcal{F}_{G,J}(|\psi\rangle\langle\psi|)d\psi$, is the average of $\mathcal{F}_{G,J}$ over all pure input states $|\psi\rangle$ and is useful for establishing a state independent measure of similarity. We have $\overline{\mathcal{F}}_{G,J} \leq 1$ for any pair G, J; with equality if and only if G implements J perfectly. To simplify the notation, for states of the form $|\psi\rangle = |\psi_Q\rangle \otimes |\psi_A\rangle$, we write $\mathcal{F}_{|\psi\rangle}$ instead of $\mathrm{F}\left(\mathcal{E}[|\psi\rangle],\ \mathcal{G}_Q|\psi_Q\rangle\langle\psi_Q|\mathcal{G}_Q^\dagger\right)$. We also represent the average of $\mathcal{F}_{|\psi\rangle}$ over all $|\psi\rangle$ as $\overline{\mathcal{F}}$. Let $\{|b_i\rangle\}$ denote the computational basis of the 2^n-dimensional qubit space. Then we have the following explicit formula [17] for $\overline{\mathcal{F}}$,

$$\overline{\mathcal{F}} = \frac{1}{2^n} + \frac{1}{2^n(2^n+1)} \sum_{ijkl} \langle b_i|\overline{\mathcal{G}_Q}|b_k\rangle\langle b_i|\mathcal{E}[|b_k\rangle\langle b_l|]|b_j\rangle\langle b_j|\mathcal{G}_Q|b_l\rangle,$$

where $\overline{\mathcal{G}_Q}$ is the complex conjugate of $\mathcal{G}_Q$.

Let $h \in \mathbb{C}^p$ and $x \in \mathbb{C}^n$. Let f be a scalar-valued differentiable function, $f : (\mathbb{C}^n, \mathbb{C}^p) \to \mathbb{C}$. The gradient of $f(x, h)$ with respect to h is the vector function

$$\nabla_h f(x,h) = \begin{pmatrix} \dfrac{\partial f}{\partial h_1}(x,h) \\ \dfrac{\partial f}{\partial h_2}(x,h) \\ \vdots \\ \dfrac{\partial f}{\partial h_p}(x,h) \end{pmatrix}.$$

If it is non-zero at a point (x, h), then the direction of the gradient is the direction in which the function increases most quickly from (x, h), and the magnitude of the gradient is the rate of increase in that direction. The gradient is zero at a point if and only if it is a stationary point.

Gradient ascent is a technique that finds a local maximum of a differentiable function by following the direction of the gradient [5]. Suppose we want to estimate the parameter set h which maximizes a function $f(x, h)$ over all x in a set of inputs X. Then, in each iteration, h is updated as

$$h \leftarrow h + \epsilon \sum_{x \in X} \frac{\nabla_h f(x,h)}{|X|}.$$

The learning rate ϵ determines how fast exploration occurs. If it is too small, climbing might not escape local maxima, whereas if it is too large the algorithm may not converge. Therefore, and given that exploration approaches a global maximum as the iterations take place, ϵ must decrease as a function of the steps.

Gradient ascent could be used to maximize $\overline{\mathcal{F}}$, but it would be inefficient due to the complexity of $\nabla_h \overline{\mathcal{F}}$. Instead, the authors opt to maximize $\mathcal{F}_{|\psi\rangle}$ for a set of random states $X = \{|\psi\rangle\}$. However, using gradient ascent in this task is also impractical because plugging $|X|$ points in $\nabla_h f(x, h)$ at every iteration is expensive. That is why the authors employ stochastic gradient ascent (SGA). SGA drastically reduces the burden of the iteration by computing the gradient for only one randomly picked x.

In the task of maximizing average gate fidelity, SGA also has the advantage of being more prone to escape local maxima than deterministic gradient ascent. This is because the optimal sets of parameters are all and only those for which $\mathcal{F}_{|\psi\rangle} = 1$ for all $|\psi\rangle$. Therefore, if SGA is probing a region of local maximum h for $\mathcal{F}_{|\psi\rangle}$, in the next iteration, when the state changes to $|\psi'\rangle$, h will probably not be a local maximum for $\mathcal{F}_{|\psi'\rangle}$.

Thus, SGA is used to maximize $\mathcal{F}_{|\psi\rangle}$ until convergence, after which $\overline{\mathcal{F}}$ is computed. If $\overline{\mathcal{F}} < 0.95$, the procedure restarts, otherwise it is assumed that h is on the border of a global maximum. In the latter case, the authors use deterministic gradient ascent to further maximize $\overline{\mathcal{F}}$, since h being near a global maximum reduces the number of iterations and guarantees the algorithm will not get stuck in a local maximum. If $\overline{\mathcal{F}} \approx 1$ is achieved, the generator $\mathcal{H}$ is successfully obtained. Algorithm 3 summarizes the procedure.

In [3] the authors use Algorithm 3 to implement the Toffoli gate using only pairwise X and Z interactions and one auxiliary qubit. The average fidelity obtained is $\overline{\mathcal{F}} = 99.98\%$. They start by parameterizing $\mathcal{H}'_{\text{Toff}}$ as a 2-body sum,

$$\mathcal{H}'_{\text{Toff}} = \sum_{\substack{a\in\{x,z\}\\1\le i\le 4}} h_a^i \sigma_a^i + \sum_{\substack{a\in\{x,z\}\\1\le i<j\le 4}} h_{aa}^{ij} \sigma_a^i \sigma_a^j. \tag{2.3}$$

The first two qubits are the control qubits, the third one is the target and the last is the auxiliary qubit. With this is mind, the authors impose the control qubits to be equally coupled to the target and the auxiliary gates, resulting in

$$\begin{aligned} h_x^1 &= h_x^2, \\ h_z^1 &= h_z^2, \\ h_{xx}^{13} &= h_{xx}^{23}, \\ h_{zz}^{13} &= h_{zz}^{23}, \\ h_{xx}^{14} &= h_{xx}^{24}, \\ h_{zz}^{14} &= h_{zz}^{24}. \end{aligned}$$

With this setup the following solution is obtained, with all other parameters from (2.3) being set to zero,

$$h_{zz}^{12} = -2.235,$$

Algorithm 3 Optimization of $\mathcal{F}_{|\psi\rangle}$ through SGA to obtain $\mathcal{H}$

Input: Gate $\mathcal{G}$.
Set of allowed interactions σ.
Output: Generator $\mathcal{H}'$ containing only allowed interactions.
while $\overline{\mathcal{F}} < 0.95$ **do**
  `// SGA`
  Choose an initial parameter set h;
  $\mathcal{H}' \leftarrow \sum_{\sigma\in\mathcal{I}} h_\sigma \sigma$
  Set an initial learning rate ϵ;
  $i = 0$; `// Step counter`
  do
    $i \leftarrow i + 1$;
    Generate a random state $|\psi\rangle = |\psi_Q\rangle \otimes |\psi_A\rangle$;
    Update h according to the gate fidelity gradient,

$$h \leftarrow h + \epsilon \nabla_h \mathcal{F}_{|\psi\rangle};$$

    $\mathcal{H}' \leftarrow \sum_{\sigma\in\mathcal{I}} h_\sigma \sigma$
    Decrease ϵ;
  until Convergence;

`// Deterministic gradient ascent`
Set an initial learning rate ϵ;
$i \leftarrow 0$; `// Step counter`
do
  $i \leftarrow i + 1$;
  Update h according to the average gate fidelity gradient,

$$h \leftarrow h + \epsilon \nabla_h \overline{\mathcal{F}};$$

  $\mathcal{H}' \leftarrow \sum_{\sigma\in\mathcal{I}} h_\sigma \sigma$
  Decrease ϵ;
until Convergence;
if $\overline{\mathcal{F}} \approx 1$ **then**
  return $\mathcal{H}' = \sum_{\sigma\in\mathcal{I}} h_\sigma \sigma$;

$$
\begin{aligned}
h_z^1 &= h_z^2 = -1.214, \\
h_x^3 &= -9.54, \\
h_{zz}^{13} &= h_{zz}^{23} = -1.23925, \\
h_z^3 &= -2.4785, \\
h_x^4 &= -2.1335, \\
h_{zz}^{14} &= h_{zz}^{24} = -1.41425, \\
h_z^4 &= -0.0825, \\
h_{xx}^{34} &= 3.765.
\end{aligned}
$$

Thus,

$$
\begin{aligned}
\mathcal{H}'_{\text{Toff}} = &-1.214(\sigma_z^1 + \sigma_z^2) - 9.54\sigma_x^3 - 2.1335\sigma_x^4 - 0.0825\sigma_z^4 + 3.765\sigma_x^3\sigma_x^4 - 2.235\sigma_z^1\sigma_z^2 \\
&- 2.4785\sigma_z^3 - 1.23925(\sigma_z^1\sigma_z^3 + \sigma_z^2\sigma_z^3) - 1.41425(\sigma_z^1\sigma_z^4 + \sigma_z^2\sigma_z^4).
\end{aligned}
$$

A 2-body Fredkin generator is also presented in the paper. The authors start by parameterizing $\mathcal{H}'_{\text{Fred}}$ as

$$
\mathcal{H}'_{\text{Fred}} = \sum_{\substack{a\in\{x,y,z\} \\ 1\le i\le 4}} h_a^i \sigma_a^i + \sum_{\substack{a,b\in\{z,y,z\} \\ 1\le i<j\le 4}} h_{ab}^{ij} \sigma_a^i \sigma_b^j. \tag{2.4}
$$

Again, the last qubit is the auxiliary one. After using Algorithm 3, they obtain that all variables but

$$
\begin{aligned}
h_{xx}^{12} &= h_{xx}^{13} = 3.4, \\
h_{xx}^{23} &= h_{yy}^{23} = h_{zz}^{23} = -1.178, \\
h_{xx}^{24} &= h_{xx}^{34} = 2.1, \\
h_{zz}^{12} &= h_{zz}^{13} = 2.7875, \\
h_x^4 &= 0.5125, \\
h_z^1 &= \frac{\pi}{2},
\end{aligned}
$$

are set to zero. Thus,

$$
\begin{aligned}
\mathcal{H}'_{\text{Fred}} = &\frac{\pi}{2}\sigma_z^1 + 0.5125\sigma_x^4 + 2.7875(\sigma_z^1\sigma_z^2 + \sigma_z^1\sigma_z^3) + 3.4(\sigma_x^1\sigma_x^2 + \sigma_x^1\sigma_x^3) \\
&- 1.178(\sigma_x^2\sigma_x^3 + \sigma_y^2\sigma_y^3 + \sigma_z^2\sigma_z^3) + 2.1(\sigma_x^2\sigma_x^4 + \sigma_x^3\sigma_x^4)
\end{aligned} \tag{2.5}
$$

and $\mathcal{H}'_{\text{Fred}}$ generates $\mathcal{G}_{\text{Fred}}$ with perfect fidelity up to numerical precision.

2.2 Supervised learning of Hamiltonians

In [9] the authors present two frameworks to, given a quantum gate $\mathcal{G}$ whose Hermitian generator $\mathcal{H} = -\mathrm{iLn}(\mathcal{G})$ contains undesired interactions, produce an Hermitian $\mathcal{H}'$ that, without the aid of auxiliary qubits and using only desired interactions, accomplishes $\exp(\mathrm{i}\mathcal{H}') = \mathcal{G}$. The first framework is an analytical procedure which, although using ad-hoc assumptions, implements the gates perfectly. The second one is an optimization scheme that improves the method described in [3]. It is worth noting that we can not yet determine whether such Hamiltonian exists for a given gate and, even if it does, neither of the methods is guaranteed to find it. With that said, the frameworks still perform well in some cases, as demonstrated by the article's results. The analytical procedure successfully implements a 2-body Toffoli gate and a diagonal 2-body Fredkin gate; while failing to produce a diagonal 2-body Toffoli generator, which, in turn, is achieved by the machine learning method. Diagonal interactions are those whose factors distinct from I are all the same,

$$\{\mathrm{I},\ \sigma_a^{i_1} \dots \sigma_a^{i_m} \mid a \in \{x, y, z\},\ 1 \leq i_j \leq n\}.$$

They are cheaper to implement in some experimental architectures, including superconducting circuits [16].

2.2.1 Analytical procedure

This framework consists of imposing the following conditions:

1. $\mathcal{H}'$ contains only desired interactions.
2. $\mathcal{H}$ and $\mathcal{H}'$ commute. (2.6)
3. The eigenvalues of the operator $\mathcal{H}' - \mathcal{H}$ are $\{2\pi k_j, k_j \in \mathbb{Z}\}$.

These conditions guarantee that $\mathcal{H}'$ generates $\mathcal{G}$ because

$$\begin{aligned} \mathcal{G} &= \mathrm{e}^{\mathrm{i}\mathcal{H}} \\ &= \mathrm{I}\,\mathrm{e}^{\mathrm{i}\mathcal{H}} \\ &= \mathrm{e}^{\mathrm{i}(\mathcal{H}'-\mathcal{H})}\mathrm{e}^{\mathrm{i}\mathcal{H}} \qquad (2.7) \\ &= \mathrm{e}^{\mathrm{i}\mathcal{H}'}. \qquad (2.8) \end{aligned}$$

We need condition 3 for (2.7), while Proposition 1.1 and condition 2 allows (2.8).

The first condition is ensured by writing $\mathcal{H}'$ as a parameterized linear combination of the desired interactions. For example, when pursuing a generic 2-body generator for the Toffoli gate, the equation is

$$\mathcal{H}'_{\text{Toff}} = h_0 \mathrm{I} + \sum_{\substack{a\in\{x,y,z\}\\ 1\le i\le 3}} h_a^i \sigma_a^i + \sum_{\substack{a,b\in\{x,y,z\}\\ 1\le i<j\le 3}} h_{ab}^{ij} \sigma_a^i \sigma_b^j. \tag{2.9}$$

For a Fredkin gate's generator using only diagonal 2-body interactions the equation is

$$\mathcal{H}'_{\text{Fred}} = h_0 \mathrm{I} + \sum_{\substack{a\in\{x,y,z\}\\ 1\le i\le 3}} h_a^i \sigma_a^i + \sum_{\substack{a\in\{x,y,z\}\\ 1\le i<j\le 3}} h_a^{ij} \sigma_a^i \sigma_a^j. \tag{2.10}$$

The second condition is satisfied by solving the linear system $\mathcal{H}'\mathcal{H} - \mathcal{H}\mathcal{H}' = 0$. This step removes some of the variables h from $\mathcal{H}'$. For instance, in the derivation of the Toffoli and Fredkin generators, the number of parameters goes from, respectively, 37 and 19 to 25 and 12. This reduction is of great importance in order to impose the third condition, which is enforced by solving an inverse eigenvalue problem (construct an Hermitian matrix from a set of desired eigenvalues) and can not be realized if there are too many variables. Still, these numbers are yet too high for making the problem amenable. Thus, the authors in [9] further reduce them by making some assumptions about the generators. This is rather an intuitive and guessing step than an algorithmic one, since the authors acknowledge that different conjectures, leading to different solutions, are possible. The assumptions made for $\mathcal{H}'_{\text{Toff}}$ are

$$\begin{aligned} h_{xz}^{12} &= h_{zx}^{12} = 0, \\ h_{zx}^{13} &= h_{zx}^{23} = \frac{\pi}{8}, \\ h_{xx}^{13} &= h_{xx}^{23} = 0, \\ h_{zz}^{23} &= -h_{zz}^{13}, \\ h_z^1 &= h_z^2 = -\frac{\pi}{8}. \end{aligned} \tag{2.11}$$

For $\mathcal{H}'_{\text{Fred}}$ the assumptions are

$$\begin{aligned} h_{yy}^{23} &= h_{zz}^{23} = h_{xx}^{23}, \\ h_z^2 &= h_z^3 = 0, \\ h_y^3 &= 0, \\ h_{yy}^{13} &= 0. \end{aligned} \tag{2.12}$$

The last step is to solve the inverse eigenvalue problem. For this, they parameterize the eigenvalues as $2\pi\nu_i$, where $\nu_i \in \mathbb{Z}$. This is achieved by solving an equation for each remaining variable, as can be seen in Method 4. By choosing values for ν_i, a family of

Method 4 Forcing the eigenvalues of $\mathcal{H}' - \mathcal{H}$ to be integer multiples of 2π.

Input: Matrices $\mathcal{H}$ and $\mathcal{H}'$.
Output: Eigenvalues of $\mathcal{H}' - \mathcal{H}$ parameterized by integer variables ν_i.
eigs $\leftarrow$ `DistinctEigenvalues(`$\mathcal{H}' - \mathcal{H}$`)`;
vals $\leftarrow$ `Variables(`$\mathcal{H}'$`)`;
if |vals| > |eigs| **then**
  return *Error*
for $i \leftarrow 1$ **to** |vals| **do**
  sol $\leftarrow$ `Solve(`eigs[i] == $2\pi\nu_i$, vals[i]`)`;
  eigs $\leftarrow$ `Replace(`eigs, sol`)`;
return eigs

solutions is obtained. Not all values of ν_i work because of the assumptions made in (2.11) and (2.12). However, suitable choices of parameters do produce working solutions.

In what follows, there is a description of how the generators are derived in paper [9]. The principal generator $\mathcal{H}_{\text{Toff}}$ of $\mathcal{G}_{\text{Toff}}$ has the 3-body interaction $\sigma_z^1\sigma_z^2\sigma_x^3$,

$$\mathcal{H}_{\text{Toff}} = -\text{iLn}(\mathcal{G}_{\text{Toff}}) = \frac{\pi}{8}(\text{I} - \sigma_z^1 - \sigma_z^2 - \sigma_x^3 + \sigma_z^1\sigma_z^2 + \sigma_z^1\sigma_x^3 + \sigma_z^2\sigma_x^3 - \sigma_z^1\sigma_z^2\sigma_x^3).$$

After writing $\mathcal{H}'_{\text{Toff}}$ as equation (2.9) and imposing (2.6.2) and (2.11), $\mathcal{H}'_{\text{Toff}}$ is

$$\mathcal{H}'_{\text{Toff}} = h_0\text{I} + h_x^3\sigma_x^3 - \frac{\pi}{8}(\sigma_z^1 - \sigma_z^1\sigma_x^3 + \sigma_z^2 - \sigma_z^2\sigma_x^3) + h_{zz}^{12}\sigma_z^1\sigma_z^2 + h_{zz}^{13}(\sigma_z^1\sigma_z^3 - \sigma_z^2\sigma_z^3).$$

Thus,

$$\mathcal{H}'_{\text{Toff}} - \mathcal{H}_{\text{Toff}} = \frac{\pi}{8}\sigma_z^1\sigma_z^2\sigma_x^3 + (h_0 - \frac{\pi}{8})\text{I} + (h_x^3 + \frac{\pi}{8})\sigma_x^3 + (h_{zz}^{12} - \frac{\pi}{8})\sigma_z^1\sigma_z^2 + h_{zz}^{13}(\sigma_z^1\sigma_z^3 - \sigma_z^2\sigma_z^3),$$

and the eigenvalues of $\mathcal{H}'_{\text{Toff}} - \mathcal{H}_{\text{Toff}}$ are

$$\begin{aligned}
\lambda_1 &= -\pi + h_0 - h_x^3 - 2h_z^1 + 2h_{zx}^{13} + h_{zz}^{12},\\
\lambda_2 &= h_0 - \sqrt{(h_x^3)^2 + 4(h_{zz}^{13})^2} - h_{zz}^{12},\\
\lambda_3 &= h_0 - \sqrt{(h_x^3)^2 + 4(h_{zz}^{13})^2} - h_{zz}^{12},\\
\lambda_4 &= h_0 + \sqrt{(h_x^3)^2 + 4(h_{zz}^{13})^2} - h_{zz}^{12},\\
\lambda_5 &= h_0 + \sqrt{(h_x^3)^2 + 4(h_{zz}^{13})^2} - h_{zz}^{12},\\
\lambda_6 &= h_0 + h_x^3 - 2h_z^1 - 2h_{zx}^{13} + h_{zz}^{12},\\
\lambda_7 &= h_0 + 2h_z^1 - \sqrt{(h_x^3)^2 + 4h_x^3h_{zx}^{13} + 4(h_{zx}^{13})^2 + 16(h_{zz}^{13})^2} + h_{zz}^{12},\\
\lambda_8 &= h_0 + 2h_z^1 + \sqrt{(h_x^3)^2 + 4h_x^3h_{zx}^{13} + 4(h_{zx}^{13})^2 + 16((h_{zz}^{13}))^2} + h_{zz}^{12}.
\end{aligned}$$

To impose condition (2.6.3), one starts by solving equation $\lambda_1 = 2\pi\nu_1$ with respect to variable h_{zz}^{12}. This results in

$$h_{zz}^{12} \leftarrow \pi + 2\pi\nu_1 - h_0 + h_x^3 + 2h_z^1 - 2h_{zx}^{13}.$$

After substitution, the eigenvalues are

$$\begin{aligned}
\lambda_1 &= 2\pi\nu_1,\\
\lambda_2 &= -\pi(1+2\nu_1) - h_x^3 + 2(h_0 - h_z^1 + h_{zx}^{13}) - \sqrt{(h_x^3)^2 + 4(h_{zz}^{13})^2},\\
\lambda_3 &= -\pi(1+2\nu_1) - h_x^3 + 2(h_0 - h_z^1 + h_{zx}^{13}) - \sqrt{(h_x^3)^2 + 4(h_{zz}^{13})^2},\\
\lambda_4 &= -\pi(1+2\nu_1) - h_x^3 + 2(h_0 - h_z^1 + h_{zx}^{13}) + \sqrt{(h_x^3)^2 + 4(h_{zz}^{13})^2},\\
\lambda_5 &= -\pi(1+2\nu_1) - h_x^3 + 2(h_0 - h_z^1 + h_{zx}^{13}) + \sqrt{(h_x^3)^2 + 4(h_{zz}^{13})^2},\\
\lambda_6 &= \pi + 2\pi\nu_1 + 2h_x^3 - 4h_{zx}^{13},\\
\lambda_7 &= \pi + 2\pi\nu_1 + h_x^3 + 4h_z^1 - 2h_{zx}^{13} - \sqrt{(h_x^3 + 2h_{zx}^{13})^2 + 16(h_{zz}^{13})^2},\\
\lambda_8 &= \pi + 2\pi\nu_1 + h_x^3 + 4h_z^1 - 2h_{zx}^{13} + \sqrt{(h_x^3 + 2h_{zx}^{13})^2 + 16(h_{zz}^{13})^2}.
\end{aligned}$$

After doing the same for all eigenvalues, one finally obtains

$$\begin{aligned}
\lambda_1 &= 2\pi\nu_1,\\
\lambda_2 &= 2\pi\nu_2,\\
\lambda_3 &= 2\pi\nu_2,\\
\lambda_4 &= 2\pi\left(\nu_2 + \sqrt{(\nu_2-\nu_3)^3}\right),\\
\lambda_5 &= 2\pi\left(\nu_2 + \sqrt{(\nu_2-\nu_3)^3}\right),\\
\lambda_6 &= 2\pi\nu_4,\\
\lambda_7 &= -\frac{\pi}{2}\sqrt{c + (-1-2\nu_1+2\nu_4)\sqrt{\frac{(c-4(\nu_5-\nu_6)^2)^2}{(1+2\nu_1-2\nu_4)^2}}} + \pi(\nu_5+\nu_6),\\
\lambda_8 &= \frac{\pi}{2}\sqrt{c + (-1-2\nu_1+2\nu_4)\sqrt{\frac{(c-4(\nu_5-\nu_6)^2)^2}{(1+2\nu_1-2\nu_4)^2}}} + \pi(\nu_5+\nu_6),
\end{aligned}$$

where $c = (4\nu_2 - 4\nu_3)^2 + (1 + 2\nu_1 - 2\nu_4)^2$ and the substitutions made are

$$\begin{aligned}
h_{zz}^{12} &\leftarrow \frac{\pi}{8}(1 + 2(\nu_1+\nu_4+\nu_5+\nu_6) - 8\nu_2 - 4\sqrt{(\nu_2-\nu_3)^2}),\\
h_0 &\leftarrow \frac{\pi}{8}(1 + 2(\nu_1+\nu_4+\nu_5+\nu_6) + 8\nu_2 + 4\sqrt{(\nu_2-\nu_3)^2}),\\
h_x^3 &\leftarrow -\frac{\pi}{8}\sqrt{\frac{(c-4(\nu_5-\nu_6)^2)^2}{(1+2\nu_1-2\nu_4)^2}},\\
h_{zx}^{13} &\leftarrow -\frac{\pi}{16}\left(-4 - 8\nu_1 + 8\nu_4 + \sqrt{\frac{(c-4(\nu_5-\nu_6)^2)^2}{(1+2\nu_1-2\nu_4)^2}}\right),\\
h_{zz}^{13} &\leftarrow -\frac{\mathrm{i}\pi}{16\sqrt{(1+2\nu_1-2\nu_4)^2}}\sqrt{-(1+2\nu_1+4\nu_2-4\nu_3-2\nu_4+2\nu_5-2\nu_6)}\\
&\quad \sqrt{1+2\nu_1-4\nu_2+4\nu_3-2\nu_4+2\nu_5-2\nu_6}\sqrt{-(1+2\nu_1+4\nu_2-4\nu_3-2\nu_4-2\nu_5+2\nu_6)}\\
&\quad \sqrt{1+2\nu_1-4\nu_2+4\nu_3-2\nu_4-2\nu_5+2\nu_6},\\
h_z^1 &\leftarrow -\frac{\pi}{8}(1+2\nu_1+2\nu_4-2\nu_5-2\nu_6).
\end{aligned}$$

So the generator is the family of solutions

$$\begin{aligned}\mathcal{H}'_{\text{Toff}} = \frac{\pi}{8}\Big(1 + 4\left(v_1 + v_2 + 2v_3 + \sqrt{(v_3 - v_4)^2}\right) - \sigma_z^1 + \sigma_z^1\sigma_x^3 - \sigma_z^2 + \sigma_z^2\sigma_x^3 \\ +(-2 - 8v_1 + 8v_2)\sigma_x^3 + 4\left(\frac{1}{4} + v_1 + v_2 - 2v_3 - \sqrt{(v_3 - v_4)^2}\right)\sigma_z^1\sigma_z^2 \\ + \sqrt{(4v_3 - 4v_4)^2 - (1 + 4v_1 - 4v_2)^2}(\sigma_z^2\sigma_z^3 - \sigma_z^1\sigma_z^3)\Big),\end{aligned}$$

obtained by replacing $\nu_i \in \mathbb{Z}$ such that $c \geq 0$ and all eigenvalues are integers. As said before, not all values of ν_i work, so a trial and error process is used to find a suitable set. As one can see, for $\lambda_1, \ldots \lambda_6$, is trivial to replace the variables such that all the eigenvalues are integers. The challenging step is to satisfy the condition for the other eigenvalues, which contain complex expressions involving more than one variable. The authors manage to find the following working set $\{\nu_1 = 1, \nu_2 = 0, \nu_3 = 1, \nu_4 = 0, \nu_5 = 1, \nu_6 = 0\}$, resulting in the operator

$$\begin{aligned}\mathcal{H}'_{\text{Toff}} = \frac{\pi}{8}\Big(9\text{I} - 7\sigma_x^3 - \sigma_z^1 - \sigma_z^2 + \sqrt{15}\sigma_z^3 + \frac{5}{2}\sigma_z^1\sigma_x^3 + \sigma_z^1\sigma_z^2 + \frac{\sqrt{15}}{2}\sigma_z^1\sigma_z^3 \\ +\frac{5}{2}\sigma_z^2\sigma_x^3 + \frac{\sqrt{15}}{2}\sigma_z^2\sigma_z^3\Big),\end{aligned}$$

which is 2-body and generates $\mathcal{G}_{\text{Toff}}$ exactly.

The same process produces a Fredkin generator using only diagonal 2-body interactions. The generator $\mathcal{H}_{\text{Fred}}$ obtained by taking the Ln of Fred is neither 2-body nor diagonal,

$$\mathcal{H}_{\text{Fred}} = \frac{\pi}{8}(\text{I} - \sigma_z^1 - \sigma_x^2\sigma_x^3 - \sigma_y^2\sigma_y^3 - \sigma_z^2\sigma_z^3 + \sigma_z^1\sigma_x^2\sigma_x^3 + \sigma_z^1\sigma_y^2\sigma_y^3 + \sigma_z^1\sigma_z^2\sigma_z^3).$$

After writing $\mathcal{H}'_{\text{Fred}}$ as equation (2.10) and imposing (2.6.2) and (2.12), $\mathcal{H}'_{\text{Fred}}$ is

$$\begin{aligned}\mathcal{H}'_{\text{Fred}} = (\frac{\pi}{8} + h_0)\text{I} + h_x^3\sigma_x^2 + h_x^3\sigma_x^3 + (h_z^1 - \frac{\pi}{8})\sigma_z^1 + (h_{xx}^{23} - \frac{\pi}{8})(\sigma_x^2\sigma_x^3 + \sigma_y^2\sigma_y^3 + \sigma_z^2\sigma_z^3) \\ +h_{xx}^{13}(\sigma_x^1\sigma_x^2 + \sigma_x^1\sigma_x^3) + h_{zz}^{13}(\sigma_z^1\sigma_z^2 + \sigma_z^1\sigma_z^3).\end{aligned}$$

After imposing condition (2.6.3), the eigenvalues of $\mathcal{H}'_{\text{Fred}} - \mathcal{H}_{\text{Fred}}$ are

$$\begin{aligned}\lambda_1 &= 0,\\ \lambda_2 &= 0,\\ \lambda_3 &= 2\pi\nu_1,\\ \lambda_4 &= 2\pi\nu_2,\\ \lambda_5 &= -\pi\sqrt{2(\nu_2^2 + \nu_3^2 - \sqrt{(\nu_2^2 - \nu_3^2)^2})},\\ \lambda_6 &= \pi\sqrt{2(\nu_2^2 + \nu_3^2 - \sqrt{(\nu_2^2 - \nu_3^2)^2})},\\ \lambda_7 &= -\pi\sqrt{2(\nu_2^2 + \nu_3^2 + \sqrt{(\nu_2^2 - \nu_3^2)^2})},\\ \lambda_8 &= \pi\sqrt{2(\nu_2^2 + \nu_3^2 + \sqrt{(\nu_2^2 - \nu_3^2)^2})}.\end{aligned}$$

The family of solutions is

$$\mathcal{H}'_{\text{Fred}} = \frac{\pi}{8}\Bigg((1+2\nu_1+2\nu_2)\mathrm{I} + (-4-8\nu_1+8\nu_2)\sigma_z^1 - (1+2\nu_1+2\nu_2)(\sigma_x^2\sigma_x^3+\sigma_y^2\sigma_y^3+\sigma_z^2\sigma_z^3) + \frac{1}{\sqrt{c}}\Bigg(\sqrt{1+2\nu_1-6\nu_2-4\nu_3}\sqrt{1+2\nu_1+2\nu_2-4\nu_3}\sqrt{1+2\nu_1-6\nu_2+4\nu_3} \sqrt{1+2\nu_1+2\nu_2+4\nu_3}(\sigma_x^2+\sigma_x^3) + c\sigma_x^1\sigma_x^2 + c\sigma_x^1\sigma_x^3 + 2\sqrt{(1+2\nu_1-6\nu_2)} \sqrt{(1+2\nu_1+2\nu_2)}\sqrt{(1+2\nu_1-2\nu_2)^2-16\nu_3^2}(\sigma_z^1\sigma_z^2+\sigma_z^1\sigma_z^3) \Bigg)\Bigg),$$

where $c = -5-20\nu_1-20\nu_1^2+20\nu_2+40\nu_1\nu_2-4\nu_2^2+16\nu_3^2$.

An example of generator is the one obtained by $\{\nu_1=1,\nu_2=1,\nu_3=2\}$,

$$\mathcal{H}'_{\text{Fred}} = \frac{\pi}{40}\Bigg(25\mathrm{I} + \sqrt{715}(\sigma_x^2+\sigma_x^3) - 20\sigma_z^1 + 25\sqrt{3}(\sigma_x^1\sigma_x^2+\sigma_x^1\sigma_x^3) -25(\sigma_x^2\sigma_x^3+\sigma_y^2\sigma_y^3+\sigma_z^2\sigma_z^3) - 6\sqrt{35}(\sigma_z^1\sigma_z^2+\sigma_z^1\sigma_z^3) \Bigg).$$

Another interesting application of the framework is showing that a gate can not be implemented with a specific set of interactions. In the paper [9], the authors prove that the CNOT gate cannot be implemented using only one-body interactions. The CNOT is a 2-qubit gate that flips the second qubit if and only if the first qubit is in state $|1\rangle$. Its matrix is given by

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The generator obtained by taking its Ln is

$$\mathcal{H}_{\text{CNOT}} = -\mathrm{i}\text{Ln}(\text{CNOT}) = \frac{\pi}{4}(\mathrm{I}-\sigma_x^2-\sigma_z^1+\sigma_z^1\sigma_x^2).$$

Let $\mathcal{H}'_{\text{CNOT}}$ be a parameterized sum of one-body interactions,

$$\mathcal{H}'_{\text{CNOT}} = h_0\mathrm{I} + \sum_{\substack{a\in\{x,y,z\}\\ 1\le i\le 2}} h_a^i\sigma_a^i.$$

By imposing that $[\mathcal{H}_{\text{CNOT}}, \mathcal{H}'_{\text{CNOT}}] = 0$, $\mathcal{H}'_{\text{CNOT}}$ becomes

$$\mathcal{H}'_{\text{CNOT}} = h_0\mathrm{I} + h_z^1\sigma_z^1 + h_x^2\sigma_x^2.$$

The last step to obtain $\exp(\mathrm{i}\mathcal{H}'_{\text{CNOT}}) = \text{CNOT}$ is ensuring that the eigenvalues of the operator $\mathcal{H}'_{\text{CNOT}} - \mathcal{H}_{\text{CNOT}}$ are integer multiples of 2π. This, in turn, implies that any integer linear combination of them must also be an integer multiple of 2π. The eigenvalues are

$$\lambda_1 = h_0 + h_x^2 - h_z^1,$$

$$\lambda_2 = h_0 - h_x^2 + h_z^1,$$
$$\lambda_3 = h_0 + h_x^2 + h_z^1,$$
$$\lambda_4 = h_0 - h_x^2 - h_z^1 - \pi.$$

But, because

$$\lambda_1 + \lambda_2 - \lambda_3 - \lambda_4 = \pi,$$

a one-body generator for CNOT does not exist.

2.2.2 Supervised learning method

The difficult part of the analytical procedure presented in Section 2.2.1 lies in imposing condition (2.6.3). To overcome this task, the authors trade the algebraic procedure for a machine learning one. This is an improved version of the supervised learning method [3] reviewed in Section 2.1. The improvement is achieved by using condition (2.6.2) to reduce the number of parameters considered in the optimization and by speeding-up the gradient evaluations.

Once again SGA is used to find a parameter set h that maximizes the gate fidelity $\mathcal{F}_{|\psi\rangle}$. In [3], the gradient is numerically approximated, but this is an inefficient approach that can be improved by using Automatic Differentiation (AD). In AD, the function f from which the gradient must be calculated is first decomposed in intermediate terms f_i for which the gradient is known analytically, $f(x,h) = f_m(f_{m-1}(\dots(f_1(x,h))))$.

Let $x \in \mathbb{C}^n$ and $h \in \mathbb{C}^p$. From now on we use (x,h) to represent the vector obtained by concatenating x and h, with $(x,h)_i$ being its ith entry. We have $f_m : \mathbb{C}^{n+p} \to \mathbb{C}$ and $f_i : \mathbb{C}^{n+p} \to \mathbb{C}^{n+p}$, for $1 \leq i \leq m-1$. Let

$$f_i(x,h) = \begin{pmatrix} f_{i,1}(x,h) \\ \vdots \\ f_{i,n+p}(x,h) \end{pmatrix},$$

with $f_{i,j} : \mathbb{C}^{n+p} \to \mathbb{C}$.

With this we can construct a directed acyclic graph (called the computational graph) in which the nodes represent operations and the edges represent which operations are computed from which. I.e., let $v_{0,j} = (x,h)_j$ and $v_{i,j} = f_{i,j}(f_{i-1}(\dots(x,h)))$ for $1 \leq i \leq m-1$ and $1 \leq j \leq n+p$. Then the graph will have a directed edge pointing from $v_{i-1,j}$ to $v_{i,k}$, with $1 \leq k \leq n+p$. It will also have edges from $v_{m-1,j}$ to v_m. This graph can be seen in Figure 2.1.

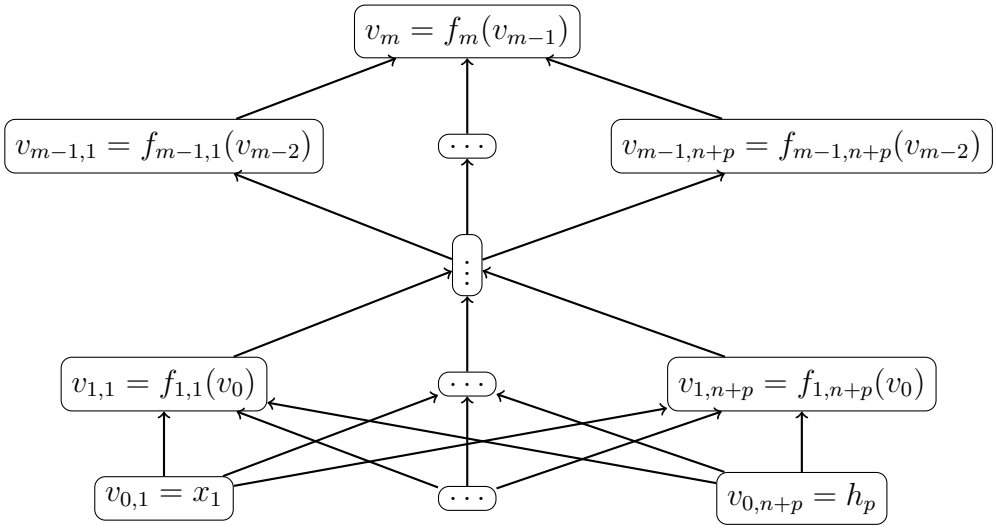

Figure 2.1: Computational graph of Automatic Differentiation after feed-forward phase.

AD uses the chain rule to evaluate the gradient $\nabla_h f(x,h)$. After constructing the Computational Graph, a topological ordering of the vertices is obtained. There are two types of AD; forward AD and reverse AD, also known as backpropagation. Backpropagation is more efficient than forward AD when the function to be derived has more inputs than outputs, this is why the authors employ it on the optimization of $\mathcal{F}_{|\psi\rangle}$. Reverse AD has two phases, feed-forward and backpropagation phase. The feed-forward phase traverses the topological ordering from parent to child nodes, storing in each node $v_{i,j}$ the corresponding intermediate function evaluated on its parent nodes, $v_{i,j} = f_{i,j}(v_{i-1})$. The second phase is the backpropagation phase, in which AD traverses the graph in the opposite direction, storing the partial derivatives $\partial v_m/\partial v_i$. For $0 \leq i < m-1$, this is done using the multivariate chain rule,

$$\frac{\partial v_m}{\partial v_{i,j}}(v_{m-1}) = \sum_{1\leq k\leq n+p} \frac{\partial v_m}{\partial v_{i+1,k}}(v_{m-1}) \frac{\partial v_{i+1,k}}{\partial v_{i,j}}(v_i).$$

When the backpropagation phase is over we have all the partial derivatives $\partial f/\partial h_j(x,h)$ of $\nabla_h f(x,h)$ computed. This process is summarized in Algorithm 5.

Another way the method described in [9] differs from the one in [3] is by using momentum mini-batch SGA to maximize $\mathcal{F}_{|\psi\rangle}$ and not following this with a deterministic optimization of $\overline{\mathcal{F}}$. In mini-batch SGA, instead of updating the parameters h with only one state $|\psi\rangle$, at each iteration a random batch of states $\{|\psi_1\rangle, \ldots, |\psi_b\rangle\}$ is used. Thus, h is updated according to the averaged gradient,

$$h \leftarrow h + \epsilon \sum_{i=1}^{b} \frac{\nabla_h \mathcal{F}_{|\psi_i\rangle}}{b}.$$

Momentum mini-batch SGA, on the other hand, uses a factor γ, called momentum, that determines the relative contribution of the current gradient and earlier gradients to the

Algorithm 5 Automatic differentiation via backpropagation.

Input:
Point (x', h') at which the gradient is to be calculated.
Functions $f_m, \ldots, f_1$ such that $f(x, h) = f_m(\ldots(f_1(x, h)))$.
Output: Gradient $\nabla_h f(x', h')$.

```
// Feed-forward phase
```
for $j \leftarrow 1$ **to** $n + p$ **do**
  $v_{0,j} \leftarrow (x', h')_j$;
for $i \leftarrow 1$ **to** $m - 1$ **do**
  for $j \leftarrow 1$ **to** $n + p$ **do**
    $v_{i,j} \leftarrow f_{i,j}(v_{i-1})$;
$v_m \leftarrow f_m(v_{m-1})$;
```
// Backpropagation phase
```
for $j \leftarrow 1$ **to** $n + p$ **do**
  $d_{m-1,j} \leftarrow \dfrac{\partial f_m}{\partial f_{m-1,j}}(v_{m-1})$;
for $i \leftarrow m - 2$ **to** 1 **do**
  for $j \leftarrow 1$ **to** $n + p$ **do**
    $d_{i,j} \leftarrow 0$;
    for $k \leftarrow 0$ **to** $n + p$ **do**
      $d_{i,j} \leftarrow d_{i,j} + d_{i+1,k} \dfrac{\partial f_{i+1,k}}{\partial f_{i,j}}(v_i)$;
for $j \leftarrow 1$ **to** p **do**
  $d_{0,j} \leftarrow 0$;
  for $k \leftarrow 0$ **to** $n + p$ **do**
    $d_{0,j} \leftarrow d_{0,j} + d_{1,k} \dfrac{\partial f_{1,k}}{\partial h_j}(v_0)$;
return $(d_{0,1}, \ldots, d_{0,p})^T$

weight change. The use of momentum prevents oscillations and accelerates the ascent. Momentum mini-batch SGA has the following updating rule

$$\Delta h \leftarrow \gamma \Delta h + \epsilon \sum_{i=1}^{b} \frac{\nabla_h \mathcal{F}_{|\psi_i\rangle}}{b},$$
$$h \leftarrow h + \Delta h,$$

where $0 \leq \gamma < 1$.

So, first the authors zero the commutator $[\mathcal{H}, \mathcal{H}']$ to reduce the number of variables in $\mathcal{H}'$. Then, they use Algorithm 6 to maximize the fidelity of the reduced expression. This way the authors successfully implement a 2-body diagonal Toffoli gate and a 2-body doubly-controlled swap gate; both with unity fidelity up to numerical precision. There are many possible parameter sets that allow unit fidelity, we show two examples below. For the diagonal $\mathcal{H}'_{\text{Toff}}$ all variables but

$$h_x^3 = 0.397,$$

Algorithm 6 Optimization of $\mathcal{F}_{|\psi\rangle}$ through Momentum mini-batch SGA to obtain $\mathcal{H}'$.

Input: Gate $\mathcal{G}$.
Output: Desired generator $\mathcal{H}'$.
Choose an initial parameter set h;
$\Delta h \leftarrow 0$;
Set the momentum γ;
Set the decay α of the learning rate;
Set the size b of the batches;
$i \leftarrow 0$; `// Step counter`
do
  $i \leftarrow i + 1$;
  Update the learning rate ϵ as

$$\epsilon \leftarrow (1 + i\alpha)^{-1};$$

  $B \leftarrow$ random batch of states $\{|\psi_1\rangle, \ldots, |\psi_b\rangle\}$;
  grad $\leftarrow 0$;
  for $i \leftarrow 1$ **to** b **do**
    `// Compute gradients with backpropagation`
    grad $\leftarrow$ grad+ `BackProp(`$\mathcal{F}_{|\psi_i\rangle}(|\psi_i\rangle, h)$`)`;

  Update h as

$$\Delta h \leftarrow \gamma \Delta h + \epsilon \frac{\text{grad}}{b};$$
$$h \leftarrow h + \Delta h;$$

until Convergence;
if $\overline{\mathcal{F}} \approx 1$ **then**
  return $\mathcal{H}' = \sum_{\sigma \in \mathcal{I}} h_\sigma \sigma$;

$$h_{xx}^{13} = 4.228,$$
$$h_{xx}^{23} = 2.160,$$
$$h_{yy}^{12} = 2.160,$$
$$h_{z}^{1} = 1.650,$$
$$h_{z}^{2} = 6.204,$$
$$h_{zz}^{12} = 2.360,$$
$$h_{zz}^{13} = -0.222,$$
$$h_{zz}^{23} = 0.217.$$

are set to zero. For the doubly-controlled swap gate generator one solution is

$$h_{z}^{2} = 12.465,$$

$$
\begin{aligned}
h_{yy}^{12} &= 8.115,\\
h_{zz}^{12} &= 10.996,\\
h_{zz}^{34} &= 9.725,\\
h_{zz}^{14} &= h_{zz}^{13} = 9.020,\\
h_{y}^{4} &= 8.259,\\
h_{x}^{3} &= 9.138,\\
h_{z}^{3} &= 11.960,\\
h_{xx}^{34} &= 5.679,\\
h_{yy}^{23} &= 7.982,\\
h_{xx}^{13} &= 13.290,\\
h_{yy}^{34} &= 13.263,\\
h_{xx}^{12} &= 9.704,\\
h_{xx}^{23} &= 15.680,\\
h_{zz}^{23} &= 10.571,\\
h_{yy}^{13} &= 7.087,\\
h_{z}^{1} &= 6.318.
\end{aligned}
$$

The great advantage of this machine learning approach is that, even though it is not exact as the one shown in Section 2.2.1, the solutions are obtained without ad-hoc reasoning.

The authors in [9] also tried to produce a 2-body Hamiltonian for the 3-qubit QFT using up to 5 ancillae, but to no success. The best fidelity found was 0.841. As explained in Section 3.4, it does not make sense to try more ancillae because with 5 we can already simulate the gate with perfect fidelity using another framework. The only drawback is that the other framework (contrary to what the authors in [9] tried to accomplish) is not local, i.e., after the computation is done, we need access to all qubits to retrieve the resulting state.

2.3 Gate simulation with single-excitation subspace

The single-excitation subspace (SES) is the subspace of $\mathbb{C}^{2^{2^n}}$ spanned by the basis

$$\{|s_i\rangle = |0\rangle^{\otimes i}|1\rangle|0\rangle^{\otimes 2^n-1-i} \,|\, 0 \leq i \leq 2^n - 1\}.$$

In [6], the authors show how to simulate any n-qubit real Hamiltonian $\mathcal{H}$ in the SES of a 2^n-qubit 2-body Hamiltonian $\mathcal{H}'$. Let

$$\mathcal{H}' = \sum_{0 \leq i < j \leq 2^n - 1} \mathcal{H}_{ii}(\mathrm{I} - \sigma_z^i) + \mathcal{H}_{ij}\sigma_x^i\sigma_x^j. \tag{2.13}$$

The SES is invariant to $\mathcal{H}'$ and we have $\langle s_i|\mathcal{H}'|s_j\rangle = \mathcal{H}_{ij}$. To simulate the action of $\mathcal{H}$ on state $|\psi\rangle \in \mathbb{C}^{2^n}$ we prepare

$$|\psi'\rangle = \sum_{0 \leq i \leq 2^n - 1} |\psi\rangle_i |s_i\rangle$$

and measure the output state $\mathcal{H}'|\psi'\rangle$ in the SES basis. In [10], the authors show how to use (2.13) to simulate a complex Hamiltonian $\mathcal{H}$ by decomposing $\exp(-\mathrm{i}\mathcal{H})$ as

$$\exp(-\mathrm{i}\mathcal{H}) = \exp(-\mathrm{i}A)\exp(-\mathrm{i}B)\exp(\mathrm{i}A), \tag{2.14}$$

with A and B real Hamiltonians. This way any Hamiltonian can be simulated in three steps: A and B are simulated by (2.13) and $\mathcal{H}$ is simulated by executing A, B and $-A$ sequentially.

Let $\mathcal{H}$ be diagonalized as $\mathcal{H} = V\Delta V^\dagger$. To derive (2.14), the authors start by writing V as

$$V = O_1 \exp(-\mathrm{i}D) O_2^T,$$

where D is real diagonal and O_i is real orthogonal. This expression follows from the KAK decomposition [12]. Then

$$\begin{aligned}
\exp(-\mathrm{i}\mathcal{H}) &= V\exp(-\mathrm{i}\Delta)V^\dagger \\
&= O_1\exp(-\mathrm{i}D)O_2^T\exp(-\mathrm{i}\Delta)O_2\exp(\mathrm{i}D)O_1^T \\
&= O_1\exp(-\mathrm{i}D)(O_1^TO_1)O_2^T\exp(-\mathrm{i}\Delta)O_2(O_1^TO_1)\exp(\mathrm{i}D)O_1^T \\
&= \exp(-\mathrm{i}O_1DO_1^T)O_1O_2^T\exp(-\mathrm{i}\Delta)O_2O_1^T\exp(\mathrm{i}O_1DO_1^T).
\end{aligned}$$

By naming $A = O_1DO_1^T$ and $B = (O_1O_2^T)\Delta(O_1O_2^T)^T$ we have

$$\exp(-\mathrm{i}\mathcal{H}) = \exp(-\mathrm{i}A)\exp(-\mathrm{i}B)\exp(\mathrm{i}A),$$

with A and B real symmetric matrices. Let $\chi = VV^T = O_1\exp(-2\mathrm{i}D)O_1^T$. Then $\chi = \chi^T$ and $\chi\chi^\dagger = \chi^\dagger\chi = \mathrm{I}$, from which follows that $\mathrm{Re}\chi$ and $\mathrm{Im}\chi$ commute,

$$\begin{aligned}
\chi\chi^\dagger &= \chi^\dagger\chi \\
\chi\overline{\chi} &= \overline{\chi}\chi \\
(\mathrm{Re}\chi + \mathrm{iIm}\chi)(\mathrm{Re}\chi - \mathrm{iIm}\chi) &= (\mathrm{Re}\chi - \mathrm{iIm}\chi)(\mathrm{Re}\chi + \mathrm{iIm}\chi) \\
\mathrm{Re}\chi\mathrm{Im}\chi &= \mathrm{Im}\chi\mathrm{Re}\chi.
\end{aligned}$$

Therefore, by simultaneously diagonalizing Reχ and Imχ we obtain a real matrix O_1 which, in turn, diagonalizes χ. Then we calculate $D = \mathrm{iLn}(O_1^T \chi O_1)/2$ and $O_2 = V^T O_1 \exp(\mathrm{i}D)$, with which we can obtain A and B.

It is worth noting that SES simulation is not scalable due to the 2^n qubits required to simulate an n-qubit gate. Also, the resulting Hamiltonian is not local, meaning that the ancillae cannot be ignored in the end. Locality is a desirable feature due to technological limitations. In Section 3.4, we present a simpler way to perform SES simulation, one that does not require decomposing a complex Hamiltonian into real Hamiltonians but instead allows it to be simulated in a single step.

As we have seen, we currently have no algorithm to determine if a generic n-qubit gate can be implemented by an n-qubit 2-body Hamiltonian or to assemble such generator. What we do have is an algorithm to implement it as a 2^n-qubit 2-body Hamiltonian and some strategies that, with some luck, might produce an n-qubit 2-body Hamiltonian. However, we would like to have a precise algorithm for this, even if only for the QFT gate, which is the main concern of this dissertation. In the next chapter, we present our attempts to achieve this goal.

Chapter 3

Gate design

Constructing circuits with 3-body interactions or more is an expensive task. Given a quantum gate $\mathcal{G}$ to be implemented, the generator $\mathcal{H} = -\mathrm{iLn}(\mathcal{G})/2\pi$ will likely contain n-body terms with $n > 2$. It is, therefore, desired to realize a 2-body generator for it and for this we seek a Hermitian matrix $\mathcal{H}'$ [9] that

1. commutes with $\mathcal{H}$,
2. is such that $\mathcal{H} + \mathcal{H}'$ is 2-body, (3.1)
3. has integer eigenvalues.

Such an $\mathcal{H}'$ is not guaranteed to exist, but if it does, we would have the 2-body generator $2\pi(\mathcal{H} + \mathcal{H}')$, since

$$\mathcal{G} = \mathrm{e}^{2\pi\mathrm{i}\mathcal{H}} = \mathrm{e}^{2\pi\mathrm{i}\mathcal{H}}\mathrm{I} = \mathrm{e}^{2\pi\mathrm{i}\mathcal{H}}\mathrm{e}^{2\pi\mathrm{i}\mathcal{H}'} = \mathrm{e}^{2\pi\mathrm{i}(\mathcal{H}+\mathcal{H}')}. \tag{3.2}$$

As said in Section 2.2.2, the difficult part of imposing these conditions lies in forcing integer eigenvalues. In an attempt to mitigate this obstacle, we examine three strategies to enforce (3.1). We test them in the search for a QFT 2-body generator. The QFT gate is a natural choice because of its relevance and low eigenspace degeneracy. An eigenvalue is said to be d-fold degenerate if its eigenspace is d dimensional. In Sections 3.2 and 3.3, we describe two different methods to obtain a 2-body generator. In these methods, the number of variables and equations, respectively, is determined by the eigenspaces' dimensions. A low degeneracy is, therefore, a desirable attribute for simplifying the calculations. QFT_n is given by

$$\mathrm{QFT}_n = \frac{1}{\sqrt{2^n}} \begin{bmatrix} F_{0,0} & \cdots & F_{0,2^n-1} \\ \vdots & & \vdots \\ F_{2^n-1,0} & \cdots & F_{2^n-1,2^n-1} \end{bmatrix},$$

where $F_{j,k} = \exp(2\pi\mathrm{i}jk/2^n)$.

All calculations described in this chapter were executed with Wolfram Mathematica [8], a high-level programming language offering a robust library of built-in functions. Due to QFT's convoluted exponents, Mathematica's standard procedures are not able to

compute its Ln or to diagonalize it. Luckily, QFT's spectral decomposition is known [2] to be

$$\mathrm{QFT}_n = 1P_1 - 1P_2 + \mathrm{i}P_3 - \mathrm{i}P_4, \tag{3.3}$$

where P_i is the projection operator of the eigenvalue λ_i,

$$P_i = \prod_{i \neq j} \frac{\mathrm{QFT}_n - \lambda_j \mathrm{I}}{\lambda_i - \lambda_j}, \tag{3.4}$$

with

$$\begin{aligned} \mathrm{rank}(P_1) &= 2^{n-2} + 1, \\ \mathrm{rank}(P_2) &= 2^{n-2}, \\ \mathrm{rank}(P_3) &= 2^{n-2}, \\ \mathrm{rank}(P_4) &= 2^{n-2} - 1. \end{aligned}$$

Thus,

$$\mathcal{H} = -\frac{\mathrm{i}}{2\pi}\mathrm{Ln}(\mathrm{QFT}_n) = 0P_1 + \frac{1}{2}P_2 + \frac{1}{4}P_3 - \frac{1}{4}P_4, \tag{3.5}$$

and we are ready to impose conditions (3.1).

3.1 First method

The first approach, presented in [9], consists of considering the sought-after generator $\mathcal{H}'$ as the parameterized 2-body interaction

$$\mathcal{H}' = h_0 \mathrm{I} + \sum_{\substack{a \in \{x,y,z\} \\ 1 \leq i \leq n}} h_a^i \sigma_a^i + \sum_{\substack{a,b \in \{x,y,z\} \\ 1 \leq i < j \leq n}} h_{ab}^{ij} \sigma_a^i \sigma_b^j - \mathcal{H}. \tag{3.6}$$

The commutativity is then imposed by solving the homogeneous system $\mathcal{H}\mathcal{H}' - \mathcal{H}'\mathcal{H} = 0$. This way, for a gate acting on n qubits, we get $\sum_{k=0}^{2} 3^k \binom{n}{k}$ variables in $\mathcal{H}'$ and at most 2^{2n} independent equations. From all methods tested this was the one best handled by Mathematica. The procedure is summarized in Method 1 and its application to the QFT_3 and QFT_4 gates follows next.

Method 1

Input: Gate $\mathcal{G}$.
$\mathcal{H} \leftarrow -\mathrm{iLn}(\mathcal{G})/2\pi$;
$\mathcal{H}' \leftarrow h_0\mathrm{I} + \sum h_a^i \sigma_a^i + \sum h_{ab}^{ij} \sigma_a^i \sigma_b^j - \mathcal{H}$;
Solve$(\mathcal{H}\mathcal{H}' - \mathcal{H}'\mathcal{H} = 0)$;
Attempt to make the eigenvalues of $\mathcal{H}'$ integers;

3.1.1 2-body Hamiltonian for QFT_3

With 3 qubits we have

$$\mathrm{QFT}_3 = \tfrac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \exp(\mathrm{i}\pi/4) & \mathrm{i} & \exp(3\mathrm{i}\pi/4) & -1 & \exp(-3\mathrm{i}\pi/4) & -\mathrm{i} & \exp(-\mathrm{i}\pi/4) \\ 1 & \mathrm{i} & -1 & -\mathrm{i} & 1 & \mathrm{i} & -1 & -\mathrm{i} \\ 1 & \exp(3\mathrm{i}\pi/4) & -\mathrm{i} & \exp(\mathrm{i}\pi/4) & -1 & \exp(-\mathrm{i}\pi/4) & \mathrm{i} & \exp(-3\mathrm{i}\pi/4) \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \exp(-3\mathrm{i}\pi/4) & \mathrm{i} & \exp(-\mathrm{i}\pi/4) & -1 & \exp(\mathrm{i}\pi/4) & -\mathrm{i} & \exp(3\mathrm{i}\pi/4) \\ 1 & -\mathrm{i} & -1 & \mathrm{i} & 1 & -\mathrm{i} & -1 & \mathrm{i} \\ 1 & \exp(-\mathrm{i}\pi/4) & -\mathrm{i} & \exp(-3\mathrm{i}\pi/4) & -1 & \exp(3\mathrm{i}\pi/4) & \mathrm{i} & \exp(\mathrm{i}\pi/4) \end{bmatrix}$$

and

$$\mathcal{H} = \frac{1}{8\sqrt{2}} \begin{bmatrix} 2\sqrt{2}-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & \sqrt{2} & 1 & \sqrt{2} & 1 & 0 & -1 & 0 \\ -1 & 1 & \sqrt{2}+1 & -1 & -1 & 1 & \sqrt{2}+1 & -1 \\ -1 & \sqrt{2} & -1 & \sqrt{2} & 1 & 0 & 1 & 0 \\ -1 & 1 & -1 & 1 & 2\sqrt{2}-1 & 1 & -1 & 1 \\ -1 & 0 & 1 & 0 & 1 & \sqrt{2} & -1 & \sqrt{2} \\ -1 & -1 & \sqrt{2}+1 & 1 & -1 & -1 & \sqrt{2}+1 & 1 \\ -1 & 0 & -1 & 0 & 1 & \sqrt{2} & 1 & \sqrt{2} \end{bmatrix}.$$

$\mathcal{H}$ has 3-body interactions,

$$\begin{aligned} \mathcal{H} = \frac{1}{32}(5\mathrm{I} + \sigma_x^1 + (2-\sqrt{2})\sigma_x^2 + (1-\sqrt{2})\sigma_z^2 + \sigma_z^3 - \sqrt{2}\sigma_x^1\sigma_x^2 - (1+\sqrt{2})\sigma_x^1\sigma_z^2 + \sigma_x^1\sigma_z^3 \\ -(2+\sqrt{2})\sigma_x^2\sigma_z^3 + 2\sqrt{2}\sigma_y^1\sigma_y^3 - 2\sqrt{2}\sigma_z^1\sigma_x^3 + (1-\sqrt{2})\sigma_z^2\sigma_z^3 - \sqrt{2}\sigma_x^1\sigma_x^2\sigma_z^3 \\ -(1+\sqrt{2})\sigma_x^1\sigma_z^2\sigma_z^3 + 2\sqrt{2}\sigma_y^1\sigma_y^2\sigma_x^3 + 2\sqrt{2}\sigma_z^1\sigma_y^2\sigma_y^3). \end{aligned}$$

We parameterize $\mathcal{H}'$ as (3.6), yielding 37 variables. A priori, solving the system $\mathcal{H}\mathcal{H}' - \mathcal{H}'\mathcal{H} = 0$ should be as simple as using the built-in Mathematica method `Solve`. But this yields the error "Unable to decide whether numeric quantities are equal to zero". This failing approach uses the canonical basis coefficients (i.e., the matrix entries) as equations. If instead we use the Pauli basis coefficients, the same `Solve` method successfully

carries a solution. This is an example of the counter-intuitive steps necessary to work with convoluted expressions in Mathematica.

After solving the system, we find the following solution.

$$
\begin{array}{lll}
h_x^3 \to 0, & h_{zy}^{23} \to 0, & h_{yz}^{13} \to 0, \\
h_y^3 \to 0, & h_{zz}^{23} \to h_x^2 & h_{yx}^{12} \to 0, \\
h_{xx}^{23} \to 0, & h_x^1 \to h_z^3 & h_{yy}^{12} \to 0, \\
h_{xy}^{23} \to 0, & h_{xx}^{13} \to 0, & h_{yz}^{12} \to 0, \\
h_{xz}^{23} \to 0, & h_{xy}^{13} \to 0, & h_z^1 \to 0, \\
h_y^2 \to 0, & h_{xx}^{12} \to h_x^2, & h_{zx}^{13} \to 0, \\
h_{yx}^{23} \to 0, & h_{xy}^{12} \to 0, & h_{zy}^{13} \to 0, \\
h_{yy}^{23} \to 0, & h_{xz}^{12} \to 0, & h_{zz}^{13} \to 0, \\
h_{yz}^{23} \to 0, & h_y^1 \to 0, & h_{zx}^{12} \to 0, \\
h_z^2 \to h_x^2, & h_{yx}^{13} \to 0, & h_{zy}^{12} \to 0, \\
h_{zx}^{23} \to 0, & h_{yy}^{13} \to 0, & h_{zz}^{12} \to 0.
\end{array}
$$

After the substitutions are done, we are left with 4 variables and $\mathcal{H}'$ is

$$
\begin{aligned}
\mathcal{H}' = \frac{1}{32}(&-\sigma_x^1 + (-2+\sqrt{2})\sigma_x^2 + (-1+\sqrt{2})\sigma_z^2 - \sigma_z^3 + \sqrt{2}\sigma_x^1\sigma_x^2 + (1+\sqrt{2})\sigma_x^1\sigma_z^2 + \\
&(2+\sqrt{2})\sigma_x^2\sigma_z^3 - 2\sqrt{2}\sigma_y^1\sigma_y^3 + 2\sqrt{2}\sigma_z^1\sigma_x^3 + (-1+\sqrt{2})\sigma_z^2\sigma_z^3 + \sqrt{2}\sigma_x^1\sigma_x^2\sigma_z^3 + \\
&(1+\sqrt{2})\sigma_x^1\sigma_z^2\sigma_z^3 - 2\sqrt{2}(\sigma_y^1\sigma_y^2\sigma_x^3 + \sigma_z^1\sigma_y^2\sigma_y^3) + (-5+32h_0)\mathrm{I} + \\
&32(\sigma_x^1+\sigma_z^3)h_z^3 + 32(\sigma_x^2+\sigma_z^2+\sigma_x^1\sigma_x^2+\sigma_z^2\sigma_z^3)h_x^2 + \sigma_x^1\sigma_z^3(-1+32h_{xz}^{13})).
\end{aligned}
$$

This is tractable enough for calculating the eigenvalues of $\mathcal{H}'$, which are

$$
\begin{aligned}
\lambda_1 &= -\frac{1}{4} + h_0 - 2h_x^2 - h_{xz}^{13}, \\
\lambda_2 &= \frac{1}{4} + h_0 - 2h_x^2 - h_{xz}^{13}, \\
\lambda_3 &= -\frac{1}{2} + h_0 + 2h_x^2 - h_{xz}^{13}, \\
\lambda_4 &= -\frac{1}{4} + h_0 - 2h_z^3 + h_{xz}^{13}, \\
\lambda_5 &= h_0 + 2h_x^2 - h_{xz}^{13}, \\
\lambda_6 &= h_0 - 2h_z^3 + h_{xz}^{13}, \\
\lambda_7 &= h_0 + 2h_z^3 + 2\sqrt{2}h_x^2 + h_{xz}^{13}, \\
\lambda_8 &= -\frac{1}{2} + h_0 + 2h_z^3 - 2\sqrt{2}h_x^2 + h_{xz}^{13}.
\end{aligned}
$$

The last step is to find values for the parameters such that all eigenvalues are integers. But we have that $\lambda_2 - \lambda_1 = 1/2$ and, therefore, the system does not have solution. This proves that it is impossible to implement the QFT_3 gate using only 2-body interactions.

Theorem 10. QFT_3 cannot be implemented by a 3-qubit 2-body Hamiltonian.

□

3.1.2 2-body Hamiltonian for QFT_4

We have also used this framework to find a 2-body Hamiltonian for the QFT_4. This time the matrices are too large to reproduce. $\mathcal{H}'$ is parameterized with 67 variables, which forces us to solve the system $[\mathcal{H}, \mathcal{H}'] = 0$ numerically to prevent the software from crashing. After solving the system, h_0 is the only variable left and $\mathcal{H}'$ has eigenvalues

$$\begin{aligned}
\lambda_1 &= -0.500067 + h_0, \\
\lambda_2 &= -0.500056 + h_0, \\
\lambda_3 &= -0.499952 + h_0, \\
\lambda_4 &= -0.499925 + h_0, \\
\lambda_5 &= 0.250075 + h_0, \\
\lambda_6 &= -0.250067 + h_0, \\
\lambda_7 &= -0.250056 + h_0, \\
\lambda_8 &= 0.249944 + h_0, \\
\lambda_9 &= -0.249942 + h_0, \\
\lambda_{10} &= 0.249933 + h_0, \\
\lambda_{11} &= -0.249925 + h_0, \\
\lambda_{12} &= 0.0000751233 + h_0, \\
\lambda_{13} &= -0.0000666341 + h_0, \\
\lambda_{14} &= -0.0000564615 + h_0, \\
\lambda_{15} &= 0.0000479723 + h_0, \\
\lambda_{16} &= 0.0000377998 + h_0;
\end{aligned}$$

which can not be made to integers. The procedure failed, but because the system was not solved exactly as we did for QFT_3, this might be due to numerical instability and, therefore, we can not infer a theorem.

3.2 Second method

This method differs from the previous in the order the conditions are enforced. Here we impose commutativity before 2-bodyness. Let $\mathcal{H} = VDV^\dagger$ be a diagonalization of the generator $\mathcal{H}$. Then, Theorem 6 allows us to represent all Hermitian matrices that

commute with $\mathcal{H}$ as $\mathcal{H}' = VBV^\dagger$, where B is a parameterized block diagonal matrix. Thus, for QFT_3, we have

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{4} \end{bmatrix}, \qquad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & 0 & 0 & 0 & 0 & 0 \\ \overline{b_{12}} & b_{22} & b_{23} & 0 & 0 & 0 & 0 & 0 \\ \overline{b_{13}} & \overline{b_{23}} & b_{33} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & b_{44} & b_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & \overline{b_{45}} & b_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & b_{66} & b_{67} & 0 \\ 0 & 0 & 0 & 0 & 0 & \overline{b_{67}} & b_{77} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & b_{88} \end{bmatrix}.$$

Given the spectral decomposition (3.5) of $\mathcal{H}$, we construct the matrix V by orthonormalizing the columns of the P_i projectors through the algorithm described in [14]. To orthogonalize the columns of an orthogonal projection P, first note that P is a positive semi-definite matrix because $P^2 = P = P^\dagger$. A principal submatrix is one obtained by selecting the same subset of rows and columns. In a positive semi-definite matrix all its principal submatrices are positive semi-definite as well. A positive semi-definite matrix has determinant greater than or equal to zero. Suppose P has a diagonal entry $P_{jj} = 0$ and that for some column k we have $P_{jk} \neq 0$. Then the principal submatrix generated by rows and columns $\{j, k\}$ is either

$$\begin{array}{c} \\ j \\ k \end{array} \begin{array}{c} \begin{array}{cc} j & k \end{array} \\ \begin{bmatrix} 0 & P_{jk} \\ P_{jk}^\dagger & P_{kk} \end{bmatrix} \end{array} \quad \text{or} \quad \begin{array}{c} \\ k \\ j \end{array} \begin{array}{c} \begin{array}{cc} k & j \end{array} \\ \begin{bmatrix} P_{kk} & P_{jk}^\dagger \\ P_{jk} & 0 \end{bmatrix} \end{array}.$$

Either way the determinant is $-|P_{jk}|^2$, contradicting the submatrix's positive semi-definiteness. Hence $P_{jj} = 0 \Rightarrow P_{jk} = P_{kj} = 0 \; \forall k$. Also, $P^2 = P$ implies that

$$\sum_l P_{jl} P_{lk} = P_{jk} \tag{3.7}$$

and all diagonal entries of P lie between 0 and 1,

$$\begin{aligned} P_{jj} &= P_{j1} P_{j1}^\dagger + \ldots + P_{jj}^2 + \ldots \\ &\Rightarrow P_{jj} \geq P_{jj}^2 \\ &\Rightarrow 0 \leq P_{jj} \leq 1 \quad \forall j. \end{aligned}$$

Because P is an orthogonal projection, its eigenvalues are either 0 or 1 and thus $\mathrm{tr}(P) = \mathrm{rank}(P)$. Therefore, the number of nonzero diagonal entries of P is at least its rank. By selecting the first nonzero diagonal entry P_{dd} and subtracting P_{dk}/P_{dd} times the dth column from the kth column we get the matrix

$$M = \begin{bmatrix} 0 & 0 & 0 \\ 0 & P_{dd} & 0 \\ 0 & P_{*d} & M' \end{bmatrix},$$

where $M_{dk} = 0$ for $k \neq d$ and $M'_{jk} = P_{jk} - P_{jd}P_{dk}/P_{dd}$. Note that $M'_{jk} = M'^{\dagger}_{kj}$ and thus M' is Hermitian. It is obvious that $\langle M_{*d}, M_{*k}\rangle = 0$ for $k < d$. On the other hand, using (3.7), for $k > d$ we get that

$$\begin{aligned}\langle M_{*d}, M_{*k}\rangle &= \sum_j P^{\dagger}_{jd} M'_{jk} \\ &= \sum_j P^{\dagger}_{jd} P_{jk} - P^{\dagger}_{jd} P_{jd} \frac{P_{dk}}{P_{dd}} \\ &= \sum_j P_{dj} P_{jk} - \frac{P_{dk}}{P_{dd}} \sum_j P_{dj} P_{jd} \\ &= P_{dk} - \frac{P_{dk}}{P_{dd}} P_{dd} \\ &= 0.\end{aligned}$$

Thus, the dth column of M is orthogonal to all other columns. Using (3.7), it is also easy to see that $M'^2 = M'$ and $\mathrm{tr}(M') = \mathrm{tr}(P) - 1$. Because M' has the same properties as P, we can recursively repeat the procedure until we obtain $\mathrm{rank}(P)$ orthogonal columns. After normalization, the $\mathrm{rank}(P_i)$ orthogonal columns of each P_i projector form together an orthonormal basis of eigenvectors of $\mathcal{H}$ and can be assembled in the matrix V. This procedure is summarized in Algorithm 7.

Algorithm 7 Function to orthonormalize the columns of the P_i projectors.

Input: Projectors $\{P_i\}$ from the spectral decomposition of $\mathcal{H}$.
Output: Matrix V such that $\mathcal{H} = VDV^{\dagger}$.
Function `ObtainV(`$\{P_i\}$`):`
  $V \leftarrow \{\}$;
  for $M \in \{P_i\}$ **do**
    for $r \leftarrow 1$ **to** $\mathrm{rank}(M)$ **do**
      $d \leftarrow$ index of the first nonzero diagonal entry M_{dd};
      `Append(`V`, Concat(ZeroVector(`2^n`−Len(`M_{d}`)), `M_{d}`)/|`M_{d}`|);`
      for $k \leftarrow d+1$ **to** 2^n **do**
        $M\_{k} \leftarrow M\_{k} - (M_{dk}/M_{dd})M\_{d}$;
      $M \leftarrow M[d+1:, d+1:]$; `// Submatrix`
  return V

After constructing V, we compute $\mathcal{H}' = VBV^{\dagger}$. Since the basis $\mathcal{S}$ defined in (1.5) is orthogonal, we can obtain the coordinates of $\mathcal{H}' + \mathcal{H}$ on $\mathcal{S}$ by computing the inner product of $\mathcal{H}' + \mathcal{H}$ with each element S_i in $\mathcal{S}$:

$$\mathcal{H}' + \mathcal{H} = \frac{\langle S_1, \mathcal{H}' + \mathcal{H}\rangle}{\langle S_1, S_1\rangle} S_1 + \ldots + \frac{\langle S_{4^n}, \mathcal{H}' + \mathcal{H}\rangle}{\langle S_{4^n}, S_{4^n}\rangle} S_{4^n}.$$

The coefficients are functions of the variables b. Hence we can force $\mathcal{H}' + \mathcal{H}$ 2-bodyness by solving the homogeneous system containing the coefficients of undesired interactions.

If $\mathcal{H}$ has m eigenvalues, each with multiplicity p_i, then the system has $\sum_{i=1}^{m}(p_i^2+p_i)/2$ variables and $\sum_{k=3}^{n} 3^k\binom{n}{k}$ equations. The procedure is summarized in Method 2.

To find a 2-body Hamiltonian for the QFT_3 gate we add the coefficients of the 3-body interactions to the homogeneous system,

$$\frac{\langle \sigma_a^1\sigma_b^2\sigma_c^3, \mathcal{H}'+\mathcal{H}\rangle}{\langle \sigma_a^1\sigma_b^2\sigma_c^3, \sigma_a^1\sigma_b^2\sigma_c^3\rangle} = 0, \text{ for } a,b,c \in \{x,y,z\}.$$

It has to be solved numerically due to the presence of complex conjugate variables that are badly handled by Mathematica's Solve function. In this case the numerical errors make the solution impossible and Mathematica returns empty solution. The same happens for QFT_4.

Method 2

Input: Spectral decomposition of $\mathcal{H}$. Orthogonal basis $\mathcal{S}$.
$V \leftarrow$ `ObtainV(`$\{P_i\}$`)`;
$B \leftarrow$ suitable diagonal block matrix with variables b_{ij};
$\mathcal{H}' \leftarrow VBV^\dagger$;
eqs $\leftarrow \{\}$;
for $S_i \in \mathcal{S}$ **do**
  if S_i *is not* 2*-body* **then**
    $s \leftarrow \dfrac{(S_i, \mathcal{H}'+\mathcal{H})}{(S_i, S_i)}$;
    `Append(`eqs, $s == 0$`)`;
Solve(eqs);
Replace b_{ij} in $\mathcal{H}'$;
Make the eigenvalues of $\mathcal{H}'$ integers;

3.3 Third method

The third approach to imposing conditions (3.1) uses the fact that B has all entries outside the blocks equal zero. So, by parameterizing B as

$$B = V^\dagger \left(h_0 \mathrm{I} + \sum_{\substack{a\in\{x,y,z\}\\ 1\le i\le n}} h_a^i \sigma_a^i + \sum_{\substack{a,b\in\{x,y,z\}\\ 1\le i<j\le n}} h_{ab}^{ij}\sigma_a^i\sigma_b^j \right) V - D, \tag{3.8}$$

we can create a homogeneous system with the entries outside the blocks. This results in $\sum_{k=0}^{2} 3^k\binom{n}{k}$ variables and $2^{2n} - \sum_{i=1}^{m} p_i^2$ equations.

Method 3

Input: Spectral decomposition of $\mathcal{H}$. Orthogonal basis $\mathcal{S}$.
$V \leftarrow$ `ObtainV(`$\{P_i\}$`)`;
$B \leftarrow V^\dagger \left(\sum_{\substack{a\in\{x,y,z\}\\ 1\le i\le n}} h_a^i \sigma_a^i + \sum_{\substack{a,b\in\{x,y,z\}\\ 1\le i<j\le n}} h_{ab}^{ij} \sigma_a^i \sigma_b^j \right) V - D$;
eqs $\leftarrow \{\}$;
for B_{ij} *outside block* **do**
  `Append(`eqs, $B_{ij} == 0$`)`;
Solve(eqs);
Replace h in $\mathcal{H}'$;
Make the eigenvalues of $\mathcal{H}'$ integers;

To obtain a 2-body generator for the QFT_3 gate we have

$$V^\dagger \left(h_0 \mathrm{I} + \sum_{\substack{a\in\{x,y,z\}\\ 1\le i\le n}} h_a^i \sigma_a^i + \sum_{\substack{a,b\in\{x,y,z\}\\ 1\le i<j\le n}} h_{ab}^{ij} \sigma_a^i \sigma_b^j \right) V - D = \begin{bmatrix} * & * & * & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & * \end{bmatrix},$$

where $*$ represents a possibly non zero entry. The system has to be solved numerically, yielding the following solution

$$\begin{array}{lll} h_x^3 \to 0, & h_z^2 \to h_{xx}^{12}, & h_{yz}^{13} \to 0, \\ h_y^3 \to 0, & h_{zx}^{23} \to 0, & h_{yx}^{12} \to 0, \\ h_z^3 \to h_x^1, & h_{zy}^{23} \to 0, & h_{yy}^{12} \to 0, \\ h_x^2 \to h_{xx}^{12}, & h_{zz}^{23} \to h_{xx}^{12}, & h_{yz}^{12} \to 0, \\ h_{xx}^{23} \to 0, & h_{xx}^{13} \to 0, & h_z^1 \to 0, \\ h_{xy}^{23} \to 0, & h_{xy}^{13} \to 0, & h_{zx}^{13} \to 0, \\ h_{xz}^{23} \to 0, & h_{xy}^{12} \to 0, & h_{zy}^{13} \to 0, \\ h_y^2 \to 0, & h_{xz}^{12} \to 0, & h_{zz}^{13} \to 0, \\ h_{yx}^{23} \to 0, & h_y^1 \to 0, & h_{zx}^{12} \to 0, \\ h_{yy}^{23} \to 0, & h_{yx}^{13} \to 0, & h_{zy}^{12} \to 0, \\ h_{yz}^{23} \to 0, & h_{yy}^{13} \to 0, & h_{zz}^{12} \to 0. \end{array}$$

Matrix B after substitution is too convoluted to reproduce. We could only calculate

the eigenvalues from the last three blocks, which are

$$\begin{aligned}
\lambda_4 &= -\frac{1}{2} + h_0 - h_{xz}^{13} + 2h_{xx}^{12} \\
\lambda_5 &= -\frac{1}{2} + h_0 + 2h_x^1 + h_{xz}^{13} - 2\sqrt{2}h_{xx}^{12}, \\
\lambda_6 &= -\frac{1}{4} + h_0 - 2h_x^1 + h_{xz}^{13}, \\
\lambda_7 &= -\frac{1}{4} + h_0 - h_{xz}^{13} - 2h_{xx}^{12}, \\
\lambda_8 &= \quad \frac{1}{4} + h_0 - h_{xz}^{13} - 2h_{xx}^{12}.
\end{aligned}$$

As we already know, it is impossible to make the eigenvalues integers because $\lambda_8 - \lambda_7 = 1/2$. Here, we needed another counter-intuitive workaround. Calculating the eigenvalues should be as simple as using Mathematica's built-in method `Eigenvalues`. But the function would crash and just inform that the desired eigenvalues were the roots of the characteristic polynomial. When, instead, we used `Solve` to find its roots, i.e., `Solve[CharacteristicPolynomi` Mathematica successfully returned the eigenvalues. Again, despite our best attempts, the software failed to solve the system exactly for QFT_4. When solving numerically, all variables were set to zero.

3.4 Hamiltonian simulation in subspaces

In [10], the authors propose a method to simulate a Hamiltonian $\mathcal{H}$ in the SES of a 2-body Hamiltonian $\mathcal{H}'$. Here we state a simpler way to do so. We believe that this result is already known in the community, but could not pinpoint a specific publication.

We index from zero to allow binary indexation. Let $N = 2^n - 1$ and $s_i = |0\rangle^{\otimes i}|1\rangle|0\rangle^{\otimes N-i}$, with $0 \leq i \leq N$. For $0 \leq i < j \leq N$ we have

$$\mathcal{H}' = \frac{\mathrm{I} - \sigma_z^i}{2} \implies s_i^\dagger \mathcal{H}' s_i = 1,$$

$$\mathcal{H}' = \frac{\sigma_x^i \sigma_x^j + \sigma_y^i \sigma_y^j}{2} \implies s_i^\dagger \mathcal{H}' s_j = s_j^\dagger \mathcal{H}' s_i = 1$$

and

$$\mathcal{H}' = \frac{\sigma_x^i \sigma_y^j + \sigma_y^i \sigma_x^j}{2} \implies s_i^\dagger \mathcal{H}' s_j = -s_j^\dagger \mathcal{H}' s_i = -\mathrm{i}.$$

So, by doing

$$\begin{aligned}
\mathcal{H}' = \frac{1}{2} \sum_{0 \leq i < j < N} & \mathcal{H}_{ii}(\mathrm{I} - \sigma_z^{N-i}) + \mathrm{Re}(\mathcal{H}_{ij})(\sigma_x^{N-i}\sigma_x^{N-j} + \sigma_y^{N-i}\sigma_y^{N-j}) \\
& - \mathrm{Im}(\mathcal{H}_{ij})(\sigma_x^{N-i}\sigma_y^{N-j} + \sigma_y^{N-i}\sigma_x^{N-j})
\end{aligned} \tag{3.9}$$

and selecting in $\mathcal{H}'$ the rows and columns indexed by $S = \{2^i \,|\, 0 \leq i \leq N\}$, we obtain $\mathcal{H}$. The SES is invariant to $\mathcal{H}'$, because for $i \in S$ and $j \notin S$ we have $\mathcal{H}'_{ij} = \mathcal{H}'_{ji} = 0$. If we select the rows and columns indexed by S from $\exp(\mathrm{i}\mathcal{H}')$ we obtain $\exp(\mathrm{i}\mathcal{H})$. Thus, if we prepare the state

$$|\psi'\rangle = \sum_{0 \leq i \leq N} |\psi\rangle_i \; s_{N-i}, \tag{3.10}$$

we can calculate the action of $\mathcal{G}$ on $|\psi\rangle$ by computing $\exp(\mathrm{i}\mathcal{H}')|\psi'\rangle$.

We now illustrate this process for an arbitrary 2-qubit Hamiltonian

$$\mathcal{H} = \begin{bmatrix} h_{00} & h_{01} & h_{02} & h_{03} \\ \overline{h_{01}} & h_{11} & h_{12} & h_{13} \\ \overline{h_{02}} & \overline{h_{12}} & h_{22} & h_{23} \\ \overline{h_{03}} & \overline{h_{13}} & \overline{h_{23}} & h_{33} \end{bmatrix}.$$

This example lacks practical application because a 2-qubit Hamiltonian is already 2-body. However, for a bigger Hamiltonian, the matrices would not fit the page. The $\mathcal{H}'$ obtained by (3.9) is

$$\mathcal{H}' = \begin{array}{c|cccccccccccccccc} & 0000 & \mathbf{0001} & \mathbf{0010} & 0011 & \mathbf{0100} & 0101 & 0110 & 0111 & \mathbf{1000} & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ \hline 0000 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ \mathbf{0001} & 0 & h_{00} & h_{01} & 0 & h_{02} & 0 & 0 & 0 & h_{03} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{0010} & 0 & \overline{h_{01}} & h_{11} & 0 & h_{12} & 0 & 0 & 0 & h_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0011 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ \mathbf{0100} & 0 & \overline{h_{02}} & \overline{h_{12}} & 0 & h_{22} & 0 & 0 & 0 & h_{23} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0101 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ 0110 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ 0111 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ \mathbf{1000} & 0 & \overline{h_{03}} & \overline{h_{13}} & 0 & \overline{h_{23}} & 0 & 0 & 0 & h_{33} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1001 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ 1010 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ 1011 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ 1100 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ 1101 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ 1110 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \\ 1111 & * & 0 & 0 & * & 0 & * & * & * & 0 & * & * & * & * & * & * & * \end{array},$$

where the star marks represent an arbitrary value. Let $\mathcal{G} = \exp(\mathrm{i}\mathcal{H})$. Then, to compute

the action of $\mathcal{G}$ on state

$$|\psi\rangle = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix},$$

we prepare the state $|\psi'\rangle$ as described in (3.10),

$$|\psi'\rangle = \begin{pmatrix} 0 & 0000 \\ q_0 & \mathbf{0001} \\ q_1 & \mathbf{0010} \\ 0 & 0011 \\ q_2 & \mathbf{0100} \\ 0 & 0101 \\ 0 & 0110 \\ 0 & 0111 \\ q_3 & \mathbf{1000} \\ 0 & 1001 \\ 0 & 1010 \\ 0 & 1011 \\ 0 & 1100 \\ 0 & 1101 \\ 0 & 1110 \\ 0 & 1111 \end{pmatrix}.$$

Now, we apply $\exp(\mathrm{i}\mathcal{H}')$ to $|\psi'\rangle$. Suppose this results in

$$\exp(\mathrm{i}H')|\psi'\rangle = \begin{pmatrix} 0 & 0000 \\ r_0 & \mathbf{0001} \\ r_1 & \mathbf{0010} \\ 0 & 0011 \\ r_2 & \mathbf{0100} \\ 0 & 0101 \\ 0 & 0110 \\ 0 & 0111 \\ r_3 & \mathbf{1000} \\ 0 & 1001 \\ 0 & 1010 \\ 0 & 1011 \\ 0 & 1100 \\ 0 & 1101 \\ 0 & 1110 \\ 0 & 1111 \end{pmatrix}.$$

Then, it follows that

$$G|\psi\rangle = \begin{pmatrix} r_0 & \mathbf{0001} \\ r_1 & \mathbf{0010} \\ r_2 & \mathbf{0100} \\ r_3 & \mathbf{1000} \end{pmatrix}.$$

In summary, given any n-qubit gate $\mathcal{G}$ with n-body generator $\mathcal{H}$, it is always possible to construct a 2-body 2^n-qubit Hamiltonian $\mathcal{H}'$ that perfectly simulates $\mathcal{G}$. The requirement of 2^n qubits is unscalable, but this technique might be useful for small gates. One example is the QFT_3. In [9], the supervised learning algorithm fails to produce a QFT_3 2-body Hamiltonian with 8 qubits because the best fidelity found was 0.841. On the other hand, the procedure just described produces a perfect fidelity (although not local) 8-qubit gate without the expensive supervised learning task.

Still, it would be useful to simulate the QFT_3 in a Hamiltonian with fewer than 256×256 entries. So we tried to simulate the QFT_3 using up to 4 auxiliary qubits. The method described in [9] failed to find a 2-body Hamiltonian $\mathcal{H}'$ such that $\exp(\mathrm{i}\mathcal{H}') \approx \mathrm{QFT}_3 \otimes U$, but here we do it differently, we try to find a 2-body Hamiltonian $\mathcal{H}'$ with $\mathcal{H} = -\mathrm{iLn}(\mathrm{QFT}_3)$ in some invariant subspace.

Let s be an n-bit binary string. Then, $\{|s\rangle\}$ is the computational basis of $(\mathbb{C}^2)^{\otimes n}$ and the x-excitation subspace is the one spanned by the basis

$$X_x = \{|s\rangle \,|\, s \text{ has } x \text{ 1s and } n-x \text{ 0s}\},\ 0 \leq x \leq n.$$

Denote $X_x \cup X_y$ as X_{xy}. Let $\mathcal{H}' = \sum_{\sigma \in \mathcal{I}} h_\sigma \sigma$ be a $2^m \times 2^m$ Hamiltonian with $n < m < 2^n$. Select a basis X with 2^m dimension and define an ordering $|s_i\rangle$ for its elements. To determine if we can simulate $\mathcal{H}$ in the subspace span(X) of $\mathcal{H}'$ we solve the following system of equations,

$$\begin{cases} \langle s_i|\mathcal{H}'|s_j\rangle = 0 & \text{if } |s_i\rangle \in X,\ |s_j\rangle \notin X, \\ \langle s_i|\mathcal{H}'|s_j\rangle = H_{ij} & \text{if } |s_i\rangle \in X,\ |s_j\rangle \in X. \end{cases}$$

Different orderings of X lead to different solution sets. Here we only consider the ordering inherited from the computational basis.

For QFT_3 with one ancilla span(X_{13}) and span(X_{024}) are candidates to contain $\mathcal{H}$, as they both have dimension 8, but neither produce a consistent system. For two ancillae the subspace dimensions do not add up to 8,

$$\begin{aligned} |X_0| &= |X_5| = 1, \\ |X_1| &= |X_4| = 5, \\ |X_2| &= |X_3| = 10. \end{aligned}$$

With three ancillae we have

$$\begin{aligned} |X_0| &= |X_6| = 1, \\ |X_1| &= |X_5| = 6, \\ |X_2| &= |X_4| = 15, \\ |X_3| &= 20, \end{aligned}$$

so span(X_{016}) and span(X_{056}) are candidates to contain $\mathcal{H}$, but again none of the systems have solution.

As seen in this chapter, we tried to produce 2-body Hamiltonians for the QFT_3 and QFT_4 gates in many ways, but could only achieve so with 2^n-qubit Hamiltonians. We also showed that the QFT_3 gate can not be implemented by a 3-qubit 2-body Hamiltonian. What remains an open question is whether it might be implemented with fewer than 8 qubits and whether the QFT_4 allows a 4-qubit generator.

Chapter 4

Conclusion

Our ultimate goal was to find a closed formula for deriving n-qubit 2-body Hamiltonians for QFT_n with $n \geq 3$. Or, in the case that said Hamiltonians do not exist, to prove a theorem stating so. We tried to solve this problem by using the analytical procedure described in [9]. Given a gate generator $\mathcal{H}$, the procedure consists of enforcing some conditions on a parameterized Hamiltonian $\mathcal{H}'$. Namely,

1. $\mathcal{H}'$ contains only desired interactions.
2. $\mathcal{H}$ and $\mathcal{H}'$ commute.
3. The eigenvalues of the operator $\mathcal{H}' - \mathcal{H}$ are $\{2\pi k_j, k_j \in \mathbb{Z}\}$.

We applied the procedure to QFT_n with $n \in \{3, 4\}$, but it only worked as expected for $n = 3$. For $n > 3$, we faced challenges when attempting to solve the system of equations $\mathcal{H}\mathcal{H}' = \mathcal{H}'\mathcal{H}$ to impose commutativity. In principle, the system is efficiently solvable; however, due to the large number of equations involved, numerical instability became a problem. We tested some variations of the procedure in an attempt to circumvent these difficulties, but it turned out that the alternatives performed poorer than the original version. The differences between them were mainly in the order the conditions were enforced. The inability to solve the system is the main drawback we faced because, for $n > 3$, we could not tell if it had a solution or not. In fact, for $n = 4$, the software would already crash during execution. This left us with no clue whether there was a 2-body Hamiltonian for the gate and, therefore, with little material to work on a closed formula.

What we managed to do instead was to prove that there is no 3-qubit 2-body Hamiltonian for QFT_3. We did this using the aforementioned procedure. In this case, the equation system was solved exactly and resulted in a Hamiltonian that does not accept integer eigenvalues.

Theorem. QFT_3 cannot be implemented by a 3-qubit 2-body Hamiltonian.

While this was not our initial goal, it remains an intriguing result, particularly in light of a previous attempt in [9]. In this paper, the authors explored methods for obtaining a 2-body generator for the QFT_3, but did not prove its nonexistence.

We also presented a closed formula to obtain a 2-body 2^n-qubit Hamiltonian that simulates an n-qubit gate $\mathcal{G}$. An algorithm producing a similar result was already known [10], but our formula is more efficient and straightforward to use. The algorithm in [10] requires the sequential execution of three 2-body Hamiltonians to simulate $\mathcal{G}$, while our method simulates $\mathcal{G}$ in only one step. The formula is given by

$$\begin{aligned}\mathcal{H}' = \frac{1}{2}(\mathcal{H}_{ii}(\mathrm{I} - \sigma_z^{N-i}) + \mathrm{Re}(\mathcal{H}_{ij})(\sigma_x^{N-i}\sigma_x^{N-j} + \sigma_y^{N-i}\sigma_y^{N-j}) \\ - \mathrm{Im}(\mathcal{H}_{ij})(\sigma_x^{N-i}\sigma_y^{N-j} + \sigma_y^{N-i}\sigma_x^{N-j})),\end{aligned}$$

where $\mathcal{H} = -\mathrm{iLn}(\mathcal{G})$.

It remains an open question whether QFT_n with $n > 3$ has a 2-body Hamiltonian or not. Because QFT_3 does not have it, we intuitively expect QFT_n to not have it either. This makes sense when we compare the number of variables and equations involved in the system $\mathcal{H}\mathcal{H}' = \mathcal{H}'\mathcal{H}$. The parameterized 2-body Hamiltonian,

$$\mathcal{H}' = h_0\mathrm{I} + \sum_{\substack{a\in\{x,y,z\}\\ 1\le i\le n}} h_a^i\sigma_a^i + \sum_{\substack{a,b\in\{x,y,z\}\\ 1\le i<j\le n}} h_{ab}^{ij}\sigma_a^i\sigma_b^j,$$

has $\sum_{k=0}^{2} 3^k\binom{n}{k} = 1 + 3n + (9n^2 - 9n)/2$ variables. On the other hand, the system has 2^{2n} equations. So, the number of equations grows exponentially with n while the number of variables doesn't, possibly leaving small room for a solution. Another way of seeing it is that, as n grows, a smaller fraction of the total set of commuting matrices is 2-body.

In all stages, deciding whether or not a parameterized matrix admits only integer eigenvalues seems to be the hardest task. We believe that exploring more sophisticated methods for this problem could lead to interesting results. On that account, general next steps would be to investigate better ways to solve the commutativity system and to impose integer eigenvalues. With these tools one might be able to determine whether QFT_n accepts a 2-body n-qubit generator or not. Another possibility is to study whether the QFT_n (or a generic n-qubit gate) can be simulated in the subspace of a 2-body Hamiltonian (akin to what is done using the SES) with less than 2^n qubits.

Bibliography

[1] Dorit Aharonov. A simple proof that toffoli and hadamard are quantum universal. *arXiv preprint quant-ph/0301040*, 2003.

[2] Takashi Aoki, Yusuke Maesaka, and Mikio Nakahara. Spectral decomposition of quantum fourier transform. *Journal of the Faculty of Science and Technology, Kinki University*, 42:9–12, sep 2006.

[3] Leonardo Banchi, Nicola Pancotti, and Sougato Bose. Quantum gate learning in qubit networks: Toffoli gate without time-dependent control. *npj Quantum Information*, 2(1):1–6, 2016.

[4] Hamilton Prado Bueno. *Álgebra linear*, volume 1. SBM, 2006.

[5] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

[6] Michael R Geller, John M Martinis, Andrew T Sornborger, Phillip C Stancil, Emily J Pritchett, Hao You, and Andrei Galiautdinov. Universal quantum simulation with prethreshold superconducting qubits: Single-excitation subspace method. *Physical Review A*, 91(6):062309, 2015.

[7] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[8] Wolfram Research, Inc. Mathematica, Version 12.2, 2020. Champaign, IL, 2020.

[9] Luca Innocenti, Leonardo Banchi, Alessandro Ferraro, Sougato Bose, and Mauro Paternostro. Supervised learning of time-independent hamiltonians for gate design. In *Quantum Information and Measurement*, pages F5A–28. Optical Society of America, 2019.

[10] Amara Katabarwa and Michael R Geller. Three-step implementation of any n× n unitary with a complete graph of n qubits. *Physical Review A*, 92(3):032306, 2015.

[11] Viv Kendon. How to compute using quantum walks. *arXiv preprint arXiv:2004.01329*, 2020.

[12] Anthony W Knapp and AW Knapp. *Lie groups beyond an introduction*, volume 140. Springer, 1996.

[13] Easwar Magesan, Robin Blume-Kohout, and Joseph Emerson. Gate fidelity fluctuations and quantum process invariants. *Physical Review A*, 84(1):012309, 2011.

[14] Madan Lal Mehta. Eigenvalues and eigenvectors of the finite fourier transform. *Journal of mathematical physics*, 28(4):781–785, 1987.

[15] Martin A Moskowitz. *A course in complex analysis in one variable*. World Scientific, 2002.

[16] Yoshifumi Nakata and Mio Murao. Diagonal quantum circuits: their computational power and applications. *The European Physical Journal Plus*, 129(7):1–14, 2014.

[17] Michael A Nielsen. A simple formula for the average gate fidelity of a quantum dynamical operation. *Physics Letters A*, 303(4):249–252, 2002.

[18] Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information (10th anniv. version), 2010.

[19] James R Powell. The quantum limit to moore's law. *Proceedings of the IEEE*, 96(8):1247–1248, 2008.

[20] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[21] Ed Sperling. Quantum effects at 7/5nm and beyond. *Semiconductor Egineering*, 2018.