DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Dual Unitary Gates for Digital Quantum Circuits

Lisa Scheller

DEPARTMENT OF INFORMATICS

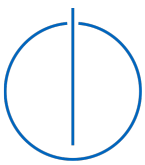
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Dual Unitary Gates for Digital Quantum Circuits

Dual-unitäre Gatter für digitale Quantenschaltkreise

Author:	Lisa Scheller
Supervisor:	Prof. Dr. Christian Mendl
Advisor:	Prof. Dr. Christian Mendl
Submission Date:	15.07.2021



I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.07.2021 Lisa Scheller

Acknowledgments

I would like to thank my supervisor Prof. Dr. Christian Mendl for proposing the topic of dual unitary gates and for his support, ideas and advice during the creation of this thesis. Also, I would like to thank all the people that proofread the thesis for their willingness to help and their valuable feedback.

Abstract

This thesis gives an overview over the usage of dual unitary gates, a special subset of quantum gates, in quantum circuits, especially with the goal of using dual unitary circuits as benchmark for quantum computers in mind. Since benchmarking a quantum computer against a simulator can only be done for a low number of qubits before the expontential nature of the statevector representation makes this too computationally expensive, there is a need for more efficient benchmarks. Circuits of dual unitary gates are a good candidate for this, because for certain circuits there exists an analytical solution whose computational complexity does not scale with the number of qubits and only involves matrix operations on 4×4 matrices. To extend the usefulness of this property to more circuits, dual unitary circuits are further generalized to include circuits of mixed dual unitaries as well as higher dimensional multi-unitaries. Indeed, a self-ternary four-qubit gate - i.e. a quantum gate unitary in three directions - is found. To check if a useful benchmarking scenario can be built from these circuits, an implementation comparing dual unitary quantum circuits against the analytical solution on a quantum computer simulator is analyzed and found to confirm the suitability of dual unitary circuits as a benchmark. To take the step from an idealized infinite grid to the simulator, periodic boundary conditions have to be added to the original grid. To reach the goal of using the benchmark on a quantum computer, some changes to the implementation are necessary in the step from simulator to quantum computer. This includes changes to the implementation of periodic boundary conditions. Also, in contrary to the implementation on the simulator, a way to evaluate the trace on a quantum computer has to be found. Some approaches for this are discussed. In conclusion, dual unitary circuits are found to be good candidates for a benchmark scenario to run on a quantum computer, even though for some issues - especially the trace evaluation - a more efficient approach has to be found before this benchmark scenario can be efficiently applied.

Nomenclature and abbreviations

List of Abbreviations

fSim gate fermionic simulation gate, a special two-qubit gate

SU(2) special unitary group of 2×2 matrices

SVD singular value decomposition

Nomenclature

CNOT gate controlled NOT gate, flips target qubit if control qubit is 1, matrix representation $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Correlation function In this thesis, used for dynamical spatiotemporal correlation functions of local operators as defined in [2]

Hermitian adjoint Conjugate transpose of a complex matrix

Hilbert space A vector space that has an inner product and additionally fulfills the condition of completeness

Kronecker product $A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \dots & & \dots \\ a_{1n}B & \dots & a_{nn}B \end{pmatrix}$

Pauli matrices Pauli X $= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, Pauli Y $= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$, Pauli Z $= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

qubit quantum bit, $|\Psi\rangle = a\,|0\rangle + b\,|1\rangle$

SWAP gate two-qubit quantum gate to exchange two qubits,
matrix representation $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Contents

Acknowledgments iii

Abstract iv

Nomenclature and abbreviations v

1. Introduction 1

2. Overview of dual unitaries 2
2.1. Definition of dual unitary quantum gates . . . 2
2.1.1. Explicit parametrization of two-qubit dual unitaries . . . 3
2.1.2. Examples of two-qubit dual unitaries . . . 5
2.1.3. Further properties of dual unitary gates . . . 6
2.2. Analytically solving circuits of dual unitaries . . . 7
2.2.1. Setup . . . 7
2.2.2. Solution . . . 8

3. Extension of patterns 11
3.1. Circuits of multiple dual unitaries . . . 11
3.1.1. Nomenclature of grid points . . . 11
3.1.2. Simplification of multi dual unitary circuits . . . 13
3.1.3. Changes to the analytical solution . . . 17
3.2. Higher dimensional dual unitaries . . . 17
3.2.1. Ternary unitaries . . . 17
3.2.2. 2D circuits of alternating 1D dual unitaries . . . 23

4. Implementation 28
4.1. Boundary conditions . . . 28
4.1.1. Placement limits for a_y^β and a_x^α . . . 29
4.1.2. Maximum t for a given circuit width . . . 29
4.2. Single gate . . . 29
4.2.1. SWAP . . . 30
4.2.2. Random . . . 31

4.3. Multi-gate . . . 32
4.3.1. Non-random gates . . . 33
4.3.2. Mixture of iSWAP and U_{rdm} . . . 34
4.3.3. Random . . . 34
4.3.4. Interpretation of results . . . 35

5. Adapting the scenario to run on a real quantum computer 36
5.1. Spatial periodic boundary conditions . . . 36
5.1.1. Singular value decomposition . . . 36
5.1.2. Utilizing 2D qubit setup . . . 37
5.2. Temporal periodic boundary conditions . . . 40
5.2.1. Computing the trace of a quantum circuit on a quantum computer 41
5.3. Building the gates . . . 41
5.3.1. IBM Q . . . 42
5.3.2. Sycamore . . . 43
5.4. Benchmark scenario . . . 44

6. Conclusion 45

A. Appendix 46
A.1. List of randomly generated dual unitaries used in the implementation . 46
A.1.1. Single gate . . . 46
A.1.2. Multi-gate . . . 47

List of Figures 53

List of Tables 55

Bibliography 56

1. Introduction

This thesis gives an overview over dual unitary quantum circuits and the usage of dual unitary gates in quantum computing. Dual unitary gates are quantum gates that are unitary not only in time direction but also in space direction. They were originally introduced in the context of condensed matter physics to compute dynamical correlation functions [2]. However, they are also useful in the context of quantum computing. One of the main benefits is that for a certain kind of quantum circuit of dual unitaries the effect of the quantum circuit on an input state can be computed by an efficient analytical solution. This also works for circuits utilizing a higher number of qubits without the exponential blowup that is inevitable when computing the statevector evolution of larger quantum circuits. Therefore such circuits are good candidates for a benchmark scenario that quantum computers can be tested against. However, since the original equation was developed for a grid of unlimited size, the grid needs to be adapted to be solvable on finite width quantum circuits by introducing periodic boundary conditions. These periodic boundary conditions have to be revisited again when stepping from a quantum circuit simulator to a quantum computer, since the qubit topology of a quantum computer is more restricted than a simulator and in general does not allow connections between remote qubits. Also, since the solution requires the evaluation of a trace, there needs to be an adaptation in order to facilitate this on quantum hardware as well.

First a definition of dual unitaries and an overview over them and their properties will be given, and their application in quantum computing will be described. Also, the equations originally defined for correlation functions will be introduced and some examples of two-qubit dual unitaries will be listed. The next chapter contains an extension of the concept of dual unitary circuits, namely to circuits of multiple different dual unitaries as well as to higher dimensions. Afterwards the implementation of the previously defined concepts will be discussed and the results of these implementations are compared against the analytical solutions. This is then used to judge the suitability of the respective circuits in a benchmarking scenario. Finally, the necessary changes to such benchmarking scenarios when applied on a quantum computer, as opposed to a digital simulator, are discussed and the resulting adaptations to the algorithm are described.

2. Overview of dual unitaries

2.1. Definition of dual unitary quantum gates

Dual unitary gates are a subset of unitary quantum gates. Unitary gates fulfill the following condition:

$$UU^\dagger = I \tag{2.1}$$

where $U^\dagger$ is the Hermitian adjoint of U. Any U that fulfills this condition is referred to as unitary in time direction from here on. Dual unitary gates are not only unitary in time direction, but also in space direction (for a one-dimensional space) [2]. This means that a certain reshuffling of U is also unitary. So the second unitarity condition is given by:

$$\tilde{U}\tilde{U}^\dagger = I \tag{2.2}$$

where $\tilde{U}$ is the reshuffled version of U. The reshuffling is defined by equation 2.3 [2].

$$\langle k| \otimes \langle l|\tilde{U}|i\rangle \otimes |j\rangle = \langle j| \otimes \langle l|U|i\rangle \otimes |k\rangle \tag{2.3}$$

The reshuffling defined by this equation can be visualized by viewing the gate in its tensor network representation, in which it is drawn as a square with four legs, where two legs are the input and two legs are the output in each direction. This is illustrated in figure 2.1, following the diagrammatic representation of [2], where the legs forming the input and the output for each direction are marked with curly brackets.

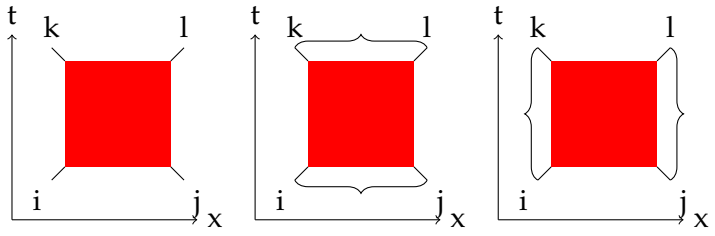
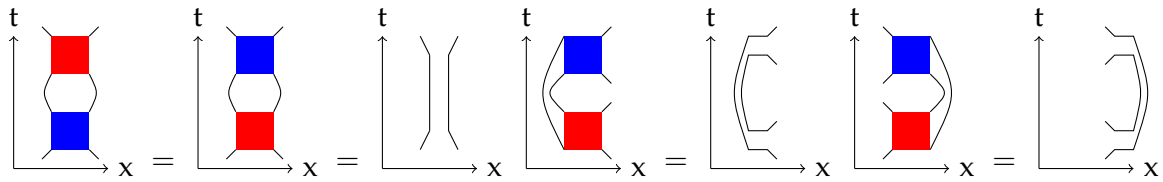


Figure 2.1.: Tensor network visualization of gate U, adapted from [2]

So for t-direction, i and j form the input and k and l form the output, while in x-direction i and k form the input and j and l form the output.
The property of dual unitarity can also be visualized in the same way, as shown in figure 2.2. Here and in all following illustrations, red is used for a gate U and blue is



(2.4)

Figure 2.2.: Visualization of unitarity in t and x direction, adapted from [2]

used for its Hermitian adjoint $U^\dagger$.
It can be seen from equation 2.3 that the dual gate $\tilde{U}$ is obtained by an index reshuffling. For a two-qubit gate of size 4×4 as in equation 2.5

$$U = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{pmatrix} \tag{2.5}$$

this means that this gate has to be transformed into a tensor of dimension $2 \times 2 \times 2 \times 2$ and then the two middle indices are interchanged. This leads to the explicit reshuffling given in equation 2.6.

$$\tilde{U} = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{2,1} & x_{2,2} \\ x_{1,3} & x_{1,4} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{4,1} & x_{4,2} \\ x_{3,3} & x_{3,4} & x_{4,3} & x_{4,4} \end{pmatrix} \tag{2.6}$$

2.1.1. Explicit parametrization of two-qubit dual unitaries

One important result of [2] is the discovery of the existence of an explicit parametrization of dual unitary two-qubit gates. Therefore it is possible to simply generate random dual unitaries of this size. The parametrization is given by [2]:

$$U = e^{i\phi}(u_+ \otimes u_-)V(J)(v_- \otimes v_+) \tag{2.7}$$

The parameters ϕ and J are real numbers and the matrices $u_\pm$, $v_\pm$ are randomly chosen from $SU(2)$. $V(J)$ is defined as in equation 2.8 in [2], where σ_i are the Pauli matrices.

$$V(J) = exp\left[-i\left(\frac{\pi}{4}\sigma_x \otimes \sigma_x + \frac{\pi}{4}\sigma_y \otimes \sigma_y + J\sigma_z \otimes \sigma_z\right)\right] \tag{2.8}$$

To see which gates might be candidates for being dual unitary, it is helpful to look at the matrix representation of $V(J)$.

$$V(J) = \begin{pmatrix} e^{-iJ} & 0 & 0 & 0 \\ 0 & 0 & -ie^{iJ} & 0 \\ 0 & -ie^{iJ} & 0 & 0 \\ 0 & 0 & 0 & e^{-iJ} \end{pmatrix} = e^{-iJ} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -ie^{2iJ} & 0 \\ 0 & -ie^{2iJ} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.9}$$

$V(J)$ is a self-dual matrix, meaning $V(J) = \tilde{V}(J)$.
It can be seen that $V(J)$ has a structure similar to that of the SWAP-gate.

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.10}$$

It is even closer to the iSWAP gate.

$$iSWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.11}$$

The exact relationship between the iSWAP gate and $V(J)$ for a generic $J \in \mathbb{R}$ is

$$V(J) = e^{-iJ} \cdot \text{iSWAP} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -e^{2iJ} & 0 & 0 \\ 0 & 0 & -e^{2iJ} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.12}$$

Another interesting aspect is that $V(J)$ is also very similar to the fSim gate, defined in [1] and used extensively on Google's sycamore architecture.

$$f_{\text{sim}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{i(\Delta_+ + \Delta_-)} \cos\theta & -ie^{i(\Delta_+ - \Delta_{-,\text{off}})} \sin\theta & 0 \\ 0 & -ie^{i(\Delta_+ + \Delta_{-,\text{off}})} \sin\theta & e^{i(\Delta_+ - \Delta_-)} \cos\theta & 0 \\ 0 & 0 & 0 & e^{i(2\Delta_+ - \Phi)} \end{pmatrix} \tag{2.13}$$

So $V(J)$ can be also written as:

$$V(J) = e^{-iJ} f_{\text{sim}}(\theta = \frac{\pi}{2}, \Delta_+ = 2J, \Delta_{-,\text{off}} = 0, \Phi = 4J - 2\pi) \tag{2.14}$$

The choice of the parameter Δ_- does not matter because it appears only in matrix entries that are multiplied by $cos(\theta)$, which is zero for the choice of θ given above.

2.1.2. Examples of two-qubit dual unitaries

With the parametrization of equation 2.7 or alternatively by checking if the reshuffling given by equation 2.6 is unitary, it can be checked whether a given gate is dual unitary or not. Table 2.1 lists notable two-qubit gates and states whether they are dual unitary or not.

Gate	**Dual unitary?**
Identity	No
Controlled U	No
$X \otimes X$	No
$Y \otimes Y$	No
$Z \otimes Z$	No
SWAP	Yes
$\sqrt{\text{SWAP}}$	No
iSWAP	Yes
$\sqrt{\text{iSWAP}}$	No
$R_{xx}(\phi)$	No
$R_{yy}(\phi)$	No
$R_{zz}(\phi)$	No
$V(J)$	Yes
$V(J_1)V(J_2)$	No
$V(J_1)V(J_2)V(J_3)$	Yes
$H \otimes H$	No
$H \otimes I$	No
$H \otimes X$	No
$H \otimes Y$	No
$H \otimes Z$	No
Identity permutations	Some

Table 2.1.: List of notable two-qubit gates and their dual unitarity

Table 2.2 lists which of the 24 permuations of the 4×4 identity matrix are dual unitary.

Whether a permutation is dual unitary or not can be explained by the following criteria: First of all, the SWAP gate (permutation (1,3,2,4)) and all cyclic permutations of this gate are dual unitary. In contrast, the identity gate and all its cyclic permutations are not dual unitary. To understand the common ground between the other dual unitaries, it is helpful to look at the tensor network representation of those gates and how it is

Dual unitary permutations	Non dual unitary permutations
{1,3,2,4}	{1,2,3,4}
{1,3,4,2}	{1,2,4,3}
{1,4,2,3}	{1,4,3,2}
{2,3,1,4}	{2,1,3,4}
{2,4,1,3}	{2,1,4,3}
{2,4,3,1}	{2,3,4,1}
{3,1,2,4}	{3,2,1,4}
{3,1,4,2}	{3,4,1,2}
{3,2,4,1}	{3,4,2,1}
{4,1,3,2}	{4,1,2,3}
{4,2,1,3}	{4,3,1,2}
{4,2,3,1}	{4,3,2,1}

Table 2.2.: Classification of permutations of the 4×4 identity matrix regarding their dual unitarity

affected by the application of a SWAP gate. If a non-permuted two-qubit gate with legs labeled a (left lower corner), b (right lower corner), c (left upper corner), and d (right upper corner) is looked at, then in t-direction a is connected to c, and b to d. Swapping b and c, which is exactly what the Swap gate does, leads to a situation where a and c are now connected in x-direction and b and d analogously. The main conclusion is that any permutation resulting in a and c (or in reversed order) as well as b and d (reverse order possible here as well) being connected, is dual unitary. This explains all remaining dual unitary permutations. Interestingly, the permutation (1,2,4,3), which is a cyclic permutation of the dual unitary permutation (3,1,2,4) is not dual unitary. This is precisely because of the non-fulfillment of the above criterion: After the permutation a is still connected to b in x-direction and not to c. Therefore, being a cyclic permutation of such an allowed gate is not sufficient to be dual unitary.

2.1.3. Further properties of dual unitary gates

Two-qubit dual unitaries do not form a group. This can be easily seen by the consecutive application of two SWAP gates after each other, which results in the identity matrix. As discussed earlier, the identity gate is not dual unitary. In fact any product $V = V(J_1)V(J_2)$ of two $V(J_i)$ is not dual unitary, which leads to the conclusion that there is also no subset of dual unitaries that can form a group. However it is interesting to note that any product of three $V(J_i)$ is again dual unitary.

2.2. Analytically solving circuits of dual unitaries

One of the main benefits of using circuits of dual unitary gates is the existence of an analytical equation describing the behavior of the circuit that is very efficient to compute and doesn't suffer from the exponential blowup which is inevitable when simulating quantum circuits on classical computers. Statevectors of a n-qubit system have size 2^n, while the matrices describing n-qubit gates are of size $2^n \times 2^n$. This becomes unfeasible quickly for increasing number of qubits n. The method described here for certain dual unitary circuits does not scale directly with number of qubits. This makes it highly efficient, even for very large circuits.

2.2.1. Setup

The original method from [2] works on a chain of size $2L$ in a grid of infinite size where one axis corresponds to x and the other axis corresponds to t. x can take values within $\frac{1}{2}\mathbb{Z}_{2L}$ and $t \in \mathbb{N}$. Each row corresponds to half a timestep. This is done because the time evolution of the system can be split into two parts, an even and an odd part. This is a common technique called Trotterization or Suzuki-Trotter splitting [20]. The Hamiltonian of the whole system is split into the sum of smaller Hamiltonians, each describing only part of the system. With this partition the time evolution of the system described by the full Hamiltonian can be numerically approximated. The main property of Trotterization is given in equation 2.15, where A and B are operators, the full Hamiltonian is given by $H = A + B$ and δ is sufficiently small [20].

$$e^{\delta(A+B)} = e^{\delta A} e^{\delta B} + \mathcal{O}(\delta^2) \tag{2.15}$$

In the case of dual unitary circuits, the two operators are the even and the odd part of the system. The even part has a simple transfer matrix $\mathbb{U}^e = U^{\otimes L}$, while the odd part is translated by one site $\mathbb{U}^o = \mathbb{T}_{2L} U^{\otimes L} \mathbb{T}_{2L}^\dagger$[2]. The whole timestep is then described by $\mathbb{U} = \mathbb{U}^o \mathbb{U}^e$.
Following the same diagrammatic representation of [2] as above, the expression $tr[\mathbb{U}^t]$ can then be drawn as in figure 2.3. This representation uses periodic boundary conditions in both directions.
Equation 2.16, which describes a dynamical correlation function, can be obtained from this.

$$D^{\alpha\beta}(x, y, t) = \frac{1}{d^{2L}} \mathrm{tr}\left[a_x^\alpha \mathbb{U}^{-t} a_y^\beta \mathbb{U}^t\right] \tag{2.16}$$

In [2] an analytical solution was found for this correlation function.
In equation 2.16 d is the dimension of the local Hilbert space, i.e. $d = 1 + 1$ for a system

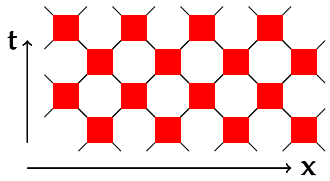


Figure 2.3.: Grid representation of $tr\left(\mathbb{U}^t\right)$, adapted from [2]

of qubits. a_x^α and a_y^β are elements of a basis of this local Hilbert space. The notation means that a_x^α is the α-th element of this basis and the gate is applied on position x. The elements of this basis have to fulfill some additional constraints: The first element a^0 is set as the identity matrix and all basis elements fulfill $tr\left[(a^\alpha)^\dagger a^\beta\right] = d\delta_{\alpha,\beta}$. For $d = 1+1$ an obvious choice of basis are the Pauli matrices, i.e. $a^0 = I, a^1 = X, a^2 = Y, a^3 = Z$ [2]. The tensor network representation (again with periodic boundary conditions) of this equation is pictured in figure 2.4.

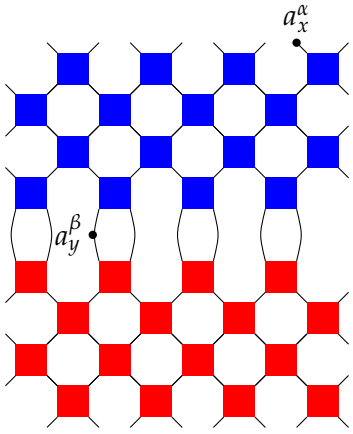


Figure 2.4.: Tensor network representation of equation 2.16 (without prefactor $\frac{1}{d^{2L}}$), adapted from [2]

2.2.2. Solution

The main result of [2] is the existence of an analytical solution for the correlation function defined in equation 2.16, which can be computed efficiently. The only possibility of non-zero correlation function values is in a light cone shaped area, meaning that the difference between the sites of the local operators $x - y$ has to be $\pm t$. Otherwise the function result will always be zero. Equation 2.17 is an intermediate step in computing

the correlation function. The grid becomes simplified further, leading to the final result equation 2.18.

$$D^{\alpha\beta}(x,y,t) = \begin{cases} C_+^{\alpha\beta}(x-y,t) & \text{if 2y odd} \\ C_-^{\alpha\beta}(x-y,t) & \text{if 2y even} \end{cases} \tag{2.17}$$

$$C_\nu^{\alpha\beta}(vt,t) = \frac{1}{d} tr\left[M_\nu^{2t}(a^\beta)a^\alpha\right] \tag{2.18}$$

It is important to note that the shorthand notation $M_\nu^{2t}(a^\beta)$ does not mean exponentiation here, but rather the nested application of function M_ν for $2t$ times. These functions are defined as followed, where tr_1 and tr_2 are the two partial traces:

$$M_+(a) = \frac{1}{d} tr_1\left[U^\dagger(a\otimes I)U\right] \tag{2.19}$$

$$M_-(a) = \frac{1}{d} tr_2\left[U^\dagger(I\otimes a)U\right] \tag{2.20}$$

The grid representation of M_ν is given in figures 2.5 and 2.6 and the grid representation of equation 2.18 is given in figure 2.7. Here the remaining circuit for $C_+^{\alpha\beta}$ was drawn. The circuit for $C_-^{\alpha\beta}$ is mirrored along the vertical axis, but looks the same otherwise.

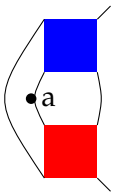


Figure 2.5.: Grid representation of M_+ (missing a prefactor of $\frac{1}{d}$), adapted from [2]

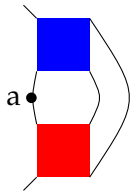


Figure 2.6.: Grid representation of M_- (missing a prefactor of $\frac{1}{d}$), adapted from [2]

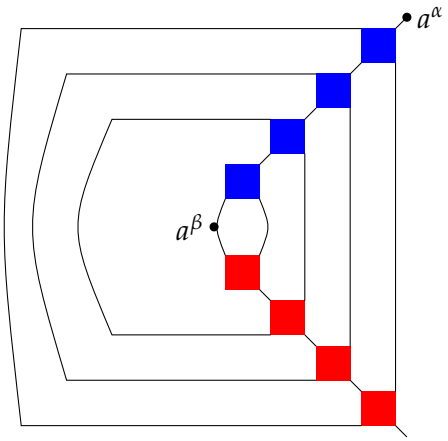


Figure 2.7.: Grid representation of the simplified correlation function (here $C_{+}^{\alpha\beta}$), adapted from [2]

A benefit of this solution is that the evaluation still stays simple for bigger grids. The dimensionality of M_ν does not increase with increasing grid size. The additional function calls are not very computationally expensive, since they still operate on 4×4 matrices. So even for large grids which cannot be simulated on a quantum circuit simulator in reasonable time, a solution can be quickly computed. This makes those circuits good candidates for application as a benchmark scenario for quantum computers where the results obtained by the quantum computer are compared against this analytical solution.
Chapter 4 compares results computed by equation 2.18 with results obtained by using IBM's qiskit simulator. In chapter 5 some problems when setting up such dual unitary circuits on a real quantum computer are discussed.

3. Extension of patterns

3.1. Circuits of multiple dual unitaries

The circuits discussed in chapter 2 are made up of only one dual unitary and its Hermitian conjugate. This section discusses whether the same formalism for getting an analytical solution like equation 2.18 can be applied to circuits using multiple different dual unitaries and what restrictions on the pattern have to be respected.
The main structure of the grid has to stay the same, so the upper part where a^{α} is applied still has to be the Hermitian conjugate of the lower part. However, in this lower part, it turns out that there are no further restrictions on the pattern, so each gate can be a different dual unitary. Therefore it is possible to fill this part with any random dual unitaries, irrespective of whether some of them are identical or not. The following section shows that such a circuit can still be simplified in the same way as a circuit using the same dual unitary on all grid locations.
The initial grid of a scenario with different dual unitaries on each grid point is shown in figure 3.1. Here different shades of red represent different dual unitaries and different shades of blue for their respective Hermitian conjugate. The whole blue part is the Hermitian conjugate of the red part.

3.1.1. Nomenclature of grid points

To discuss this topic it is necessary to first introduce a nomenclature of locations in the grid to make it possible to describe what happens on which grid point. The following nomenclature was selected to be used in this thesis.
The grid has a horizontal size of $2L$ grid sites. In vertical direction there are $2t$ rows for the lower part and $2t$ rows for the Hermitian conjugate part. Each row corresponds to a timestep of size 0.5 in the original correlation function picture. x denotes the horizontal location of a^{α}, y denotes the horizontal position of a^{β}. Both x and y are $\in \frac{1}{2}\mathbb{Z}_{2L}$. Currently in the implementation used in this thesis, zero is in the middle on qubit $\frac{n}{2} - 1$, where n is the number of qubits in the circuit.
The position of each dual unitary within the grid is given by its horizontal and vertical coordinates, called s and t throughout the remainder of this thesis. There are several

possibilities for choosing the horizontal coordinate s. One of them is to simply label the leftmost column with $s = 0$ and then increase s in steps of 1 when going further right. If s is chosen this way, it directly corresponds to one of the qubits the gate works on. However, for scenarios that allow negative x and y, s would not correspond to the values for x and y given to the correlation function. For example, the dual unitary below a^β would not be labeled U_y, t. So to stay consistent with the input values for the correlation function, the leftmost position is labeled as $\frac{-L+1}{2}$ and the rightmost position as $\frac{L}{2}$. The step size along the horizontal axis is 0.5. Another possible labeling would be defining y as zero, so that the dual unitaries on the diagonal between a^β and a^α would always have $s = 1$, $s = 2$ and so on. However, this would induce a new labeling for every different y, whereas describing the elements on the diagonal is not a difficult task and not worth the confusion caused by this and is thus not used. Another possibility would be to use the two indices of the qubits that the gate operates on instead of just the horizontal position. However, since this also does not correspond to the values given as input to the correlation function, it does not seem to be the most intuitive way to describe the gates in the grid.
For the vertical direction, only the range t is used. Once for the "normal" part and once for the complex conjugate part. The stepsize is 1. Since t is strictly positive and never zero, the smallest value for t is 1. Since each row is only half a timestep, the same t is used on two rows each.
Taken together, the leftmost, highest position of the original part is labeled as $U_{\frac{-L+1}{2},1}$ (and thus the lowest leftmost part of the Hermitian conjugate part is labeled $U^\dagger_{\frac{-L+1}{2},1}$). Accordingly, the rightmost, lowest position is labelled $U_{\frac{L}{2},t}$ and the upper right corner is $U^\dagger_{\frac{L}{2},t}$. For $L = 4, t = 1$ the grid would be labeled as depicted in figure 3.1.

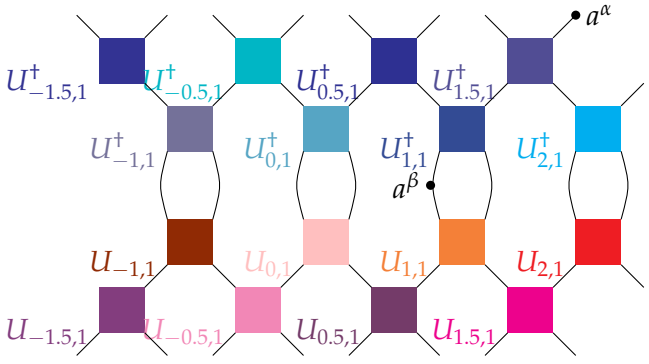


Figure 3.1.: Initial grid of multiple dual unitaries ($L = 4$, $t = 1$)

3.1.2. Simplification of multi dual unitary circuits

To check if a grid of multiple different dual unitaries like figure 3.1 can be simplified in the same way as a grid of multiple identical dual unitaries, it is necessary to look at the individual simplifications that were made on the original grid and check if they are still possible if the grid consists of different dual unitaries. The goal is again to simplify the circuit to a form similar to figure 2.7. To show how the different simplification steps can be applied to a grid of multiple different dual unitaries, an example grid of those will be simplified. The starting point of this simplification example is a grid of size $L = 3$ as specified in figure 3.2. The same simplification steps as shown below can be generalized to any dual unitary grid following the same basic structure of one half being the Hermitian conjugate of the other half.

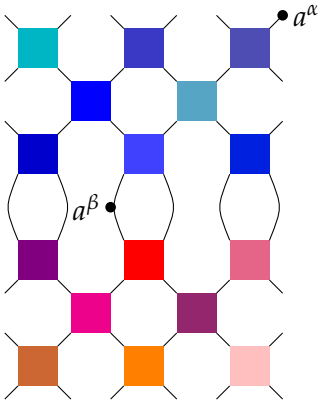


Figure 3.2.: Initial grid of multiple dual unitaries

The first simplification step is the elimination of pairs of U_i and $U_i^{\dagger}$ at all locations on the border between the regular part and the complex conjugate part, except at the location where a^{β} is in between them, by utilizing their unitarity. Since the whole upper part is the Hermitian conjugate of the lower part, every dual unitary gate faces its individual Hermitian conjugate and all those pairs simplify to the identity. This simplification step does not depend on whether all dual unitaries are identical or not. This leads to the circuit as shown in figure 3.3

The next step uses the dual unitary property. First, this leads to the elimination of the pair of light blue in the upper left and brown-red in the lower left. Their two left legs are connected. Remember that for the upper left leg of light blue and the lower left

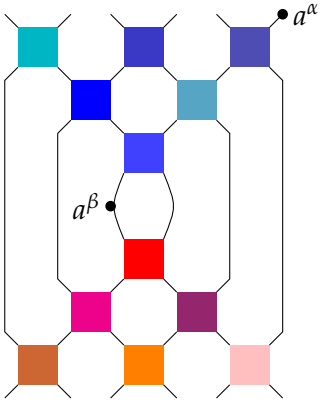


Figure 3.3.: Grid of multiple dual unitaries after first simplifications

leg of brown-red in all those circuits periodic boundary conditions are applied in both directions. Then afterwards, the exact same simplification can be done for royal blue and dark pink as well as for orange and its Hermitian conjugate. Those simplifications make use of equation 2.2 and figure 2.2 and therefore utilize the dual unitarity of both involved gates. The result is pictured in figure 3.4. It can be seen that the resulting

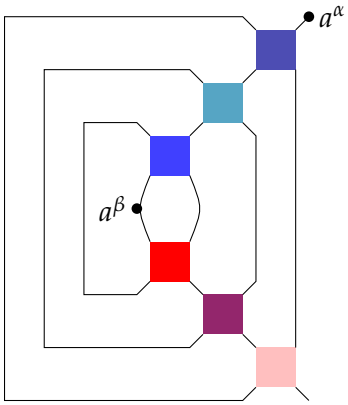


Figure 3.4.: Grid of multiple dual unitaries after simplifcations

circuit is similar to the one obtained from circuits of the same dual unitary. Again, only grid points on the line $x = \pm t$, i.e. on the light cone, remain.
Even for larger circuits the application of the dual unitary property to remove a gate outside the light cone and its Hermitian conjugate still works. This is shown in figures 3.5, 3.6, 3.7 and 3.8. It will be shown that all gates in between $U^\dagger_{0.5,2}$ and $U_{0.5,2}$ can be removed by repeated application of dual unitary and unitary properties and thus

$U_{0.5,2}$ and $U^\dagger_{0.5,2}$ eliminate themselves because of their dual unitarity. The initial setup is shown in figure 3.5. The red lines indicate which legs of the gates shall be connected to eliminate $U^\dagger_{0.5,2}$ and $U_{0.5,2}$ because of their dual unitarity. Again it starts by eliminating

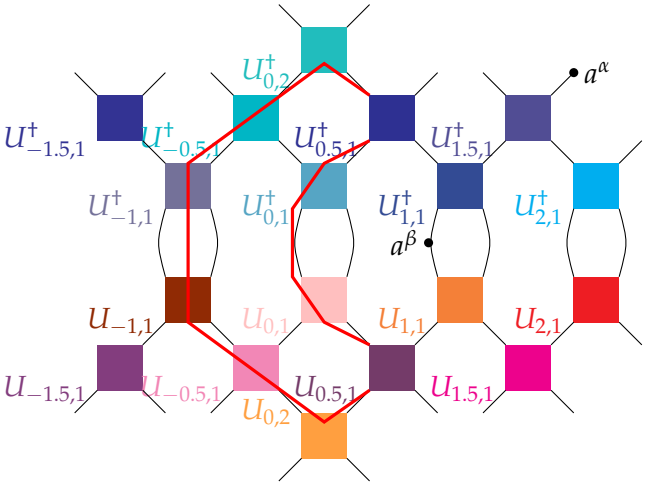


Figure 3.5.: Initial setup for grid simplification example

the direct pairs of U_i and $U_i^\dagger$ by using their unitarity. This affects $U_{0,1}$ and $U_{-1,1}$ as well as their respective Hermitian conjugate here. The resulting grid is pictured in figure 3.6.

Then $U_{-3,1}$ and its Hermitian conjugate are eliminated the same way. $U_{-0.5,1}$ and

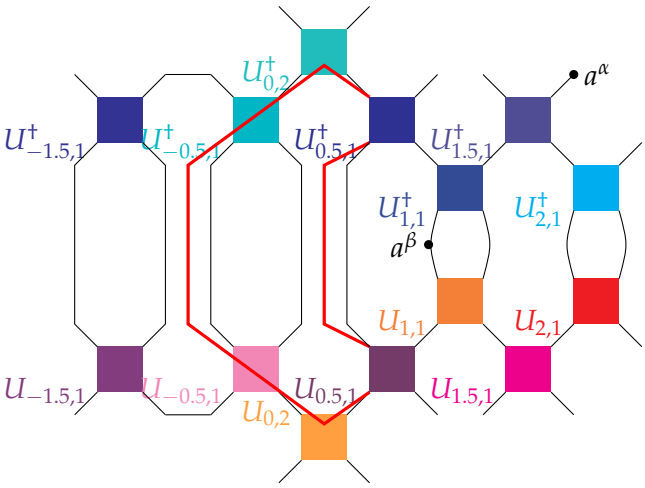


Figure 3.6.: Step 1 of grid simplification example

$U^\dagger_{-0.5,1}$ can then be eliminated by using either their unitarity or their dual unitarity. The remaining circuit is pictured in figure 3.7.

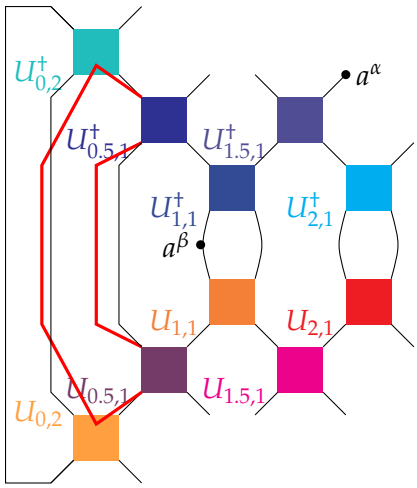


Figure 3.7.: Step 2 of grid simplification example

Now $U_{0,2}$ and $U^\dagger_{0,2}$ can be eliminated because of their dual unitarity. This leads to the desired result that $U_{0.5,1}$ and $U^\dagger_{0.5,1}$ are now directly connected in a way that their dual unitarity can be exploited. This is pictured in figure 3.8.

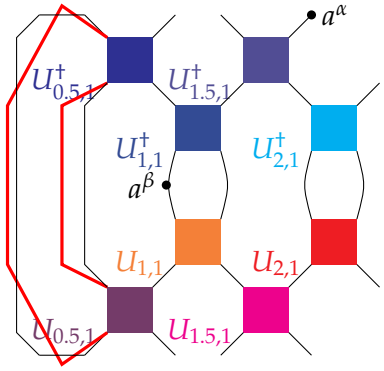


Figure 3.8.: Step 3 of grid simplification example

3.1.3. Changes to the analytical solution

From the previous section it can be seen that the resulting pattern of a grid of random dual unitaries has exactly the same structure as for a grid identical dual unitaries. Only the gates on the diagonal between a^β and a^α remain, if $x - y = \pm t$. So in principle the same equation as for the single dual unitary case can be used. Of course, equation 2.18 for the analytical solution has to be adapted in order to comply with using different dual unitaries. This means that M_ν is now evaluated for a different U in each step and not only a changes. The adapted equation is given by equation 3.1.

$$C_\nu^{\alpha\beta}(\nu t, t) = \frac{1}{d} tr \left[M_\nu(U_{x,t}, M_\nu(U_{x-1,t}, M_\nu(...M_\nu(U_{y+1,1}, M_\nu(U_{y,1}, a^\beta))))) a^\alpha \right] \tag{3.1}$$

3.2. Higher dimensional dual unitaries

Another possibility to extend the original model is to look at dual unitaries of higher dimensions. There are several possibilities to do so. One possible direction is to extend the property of dual unitarity to higher dimensions, i.e. defining unitarity not only in t- and x-direction, but also for example in y-direction and then finding multi-qubit gates that fulfill all three unitarity constraints. The other route is to apply two-qubit dual unitaries to a three-dimensional grid in alternating order, so for example alternating x-t- and y-t-dual unitaries. Of course, it is also possible to simply take a closer look on higher dimensional gates that are still only unitary in x- and t-direction, but not in y-direction.
From here on, the dual unitaries discussed before with $d = 1 + 1$, i.e. two-qubit gates or matrices of size 4×4 are called one-dimensional dual unitaries, since they have one space dimension, in addition to one time dimension. Gates with two space dimensions x and y are thus called two-dimensional dual unitaries.

3.2.1. Ternary unitaries

Gates fulfilling three unitarity constraints, one in t-, one in x- and one in y-direction, from here on called ternary unitaries, will be analyzed first. For one-dimensional dual unitaries, which can be drawn as a two-qubit gate (so as a rectangle with four legs), equation 2.3 applies. Therefore for the following gate pictured in figure 3.9, equation 3.2 has to be fulfilled for the gate to be a dual unitary.

$$\langle c| \otimes \langle d|\tilde{U}|a\rangle \otimes |b\rangle = \langle b| \otimes \langle d|U|a\rangle \otimes |c\rangle \tag{3.2}$$

This can be visualized by considering which legs are input legs and which are output legs. In this example a and b are the input in t-direction, and c and d are the output in

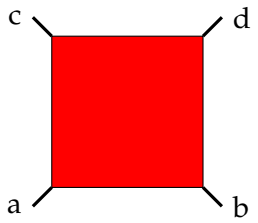


Figure 3.9.: Example of a one-dimensional dual unitary with legs a, b, c, d

t-direction. Looking at the evolution in x-direction, a and c are the input and b and d are the output.

If this scenario is extended by adding a third coordinate axis y, the gate becomes a four-qubit gate and the corresponding graphical representation (as a cube, no longer a square) is shown in figure 3.10. Now the equation system 3.3 has to be fulfilled to make the gate a ternary unitary.

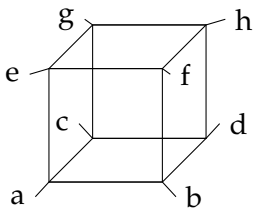


Figure 3.10.: Example of a two-dimensional gate with legs a, b, c, d, e, f, g, h

$$\langle abef|U|cdgh\rangle = \langle aceg|\tilde{U}|bdfh\rangle = \langle abcd|\hat{U}|efgh\rangle \tag{3.3}$$

The argument which gates are grouped together can again be constructed by considering which half of the legs are input and which are output. For t-direction, the front face of the cube is the input (so a, b, e, f) and the back face is the output (c, d, g, h). For x-direction, the left face is the input (a, c, e and g) and the right face is the output (b, d, f, h). And for y-direction, the lower face is the input (a, b, c, d) and the upper face is the output (e, f, g, h).
$\hat{U}$ is another reshuffling of U that has to be unitary for the gate to be unitary in y-direction.

Criteria for unitarity in each direction This section lists the criteria for unitarity in each direction for a four-qubit gate (so a 16×16 matrix). Since the objective is to

find gates fulfilling all three criteria, it is instructive to understand how each of the reshufflings is defined and changes the original gate.

t-direction The matrix U itself has to be unitary. To visualize the following reshufflings a colorcoded version of U is shown in equation 3.4

$$\begin{pmatrix}
x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} & x_{1,6} & x_{1,7} & x_{1,8} & x_{1,9} & x_{1,10} & x_{1,11} & x_{1,12} & x_{1,13} & x_{1,14} & x_{1,15} & x_{1,16} \\
x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} & x_{2,6} & x_{2,7} & x_{2,8} & x_{2,9} & x_{2,10} & x_{2,11} & x_{2,12} & x_{2,13} & x_{2,14} & x_{2,15} & x_{2,16} \\
x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{3,5} & x_{3,6} & x_{3,7} & x_{3,8} & x_{3,9} & x_{3,10} & x_{3,11} & x_{3,12} & x_{3,13} & x_{3,14} & x_{3,15} & x_{3,16} \\
x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} & x_{4,6} & x_{4,7} & x_{4,8} & x_{4,9} & x_{4,10} & x_{4,11} & x_{4,12} & x_{4,13} & x_{4,14} & x_{4,15} & x_{4,16} \\
x_{5,1} & x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} & x_{5,6} & x_{5,7} & x_{5,8} & x_{5,9} & x_{5,10} & x_{5,11} & x_{5,12} & x_{5,13} & x_{5,14} & x_{5,15} & x_{5,16} \\
x_{6,1} & x_{6,2} & x_{6,3} & x_{6,4} & x_{6,5} & x_{6,6} & x_{6,7} & x_{6,8} & x_{6,9} & x_{6,10} & x_{6,11} & x_{6,12} & x_{6,13} & x_{6,14} & x_{6,15} & x_{6,16} \\
x_{7,1} & x_{7,2} & x_{7,3} & x_{7,4} & x_{7,5} & x_{7,6} & x_{7,7} & x_{7,8} & x_{7,9} & x_{7,10} & x_{7,11} & x_{7,12} & x_{7,13} & x_{7,14} & x_{7,15} & x_{7,16} \\
x_{8,1} & x_{8,2} & x_{8,3} & x_{8,4} & x_{8,5} & x_{8,6} & x_{8,7} & x_{8,8} & x_{8,9} & x_{8,10} & x_{8,11} & x_{8,12} & x_{8,13} & x_{8,14} & x_{8,15} & x_{8,16} \\
x_{9,1} & x_{9,2} & x_{9,3} & x_{9,4} & x_{9,5} & x_{9,6} & x_{9,7} & x_{9,8} & x_{9,9} & x_{9,10} & x_{9,11} & x_{9,12} & x_{9,13} & x_{9,14} & x_{9,15} & x_{9,16} \\
x_{10,1} & x_{10,2} & x_{10,3} & x_{10,4} & x_{10,5} & x_{10,6} & x_{10,7} & x_{10,8} & x_{10,9} & x_{10,10} & x_{10,11} & x_{10,12} & x_{10,13} & x_{10,14} & x_{10,15} & x_{10,16} \\
x_{11,1} & x_{11,2} & x_{11,3} & x_{11,4} & x_{11,5} & x_{11,6} & x_{11,7} & x_{11,8} & x_{11,9} & x_{11,10} & x_{11,11} & x_{11,12} & x_{11,13} & x_{11,14} & x_{11,15} & x_{11,16} \\
x_{12,1} & x_{12,2} & x_{12,3} & x_{12,4} & x_{12,5} & x_{12,6} & x_{12,7} & x_{12,8} & x_{12,9} & x_{12,10} & x_{12,11} & x_{12,12} & x_{12,13} & x_{12,14} & x_{12,15} & x_{12,16} \\
x_{13,1} & x_{13,2} & x_{13,3} & x_{13,4} & x_{13,5} & x_{13,6} & x_{13,7} & x_{13,8} & x_{13,9} & x_{13,10} & x_{13,11} & x_{13,12} & x_{13,13} & x_{13,14} & x_{13,15} & x_{13,16} \\
x_{14,1} & x_{14,2} & x_{14,3} & x_{14,4} & x_{14,5} & x_{14,6} & x_{14,7} & x_{14,8} & x_{14,9} & x_{14,10} & x_{14,11} & x_{14,12} & x_{14,13} & x_{14,14} & x_{14,15} & x_{14,16} \\
x_{15,1} & x_{15,2} & x_{15,3} & x_{15,4} & x_{15,5} & x_{15,6} & x_{15,7} & x_{15,8} & x_{15,9} & x_{15,10} & x_{15,11} & x_{15,12} & x_{15,13} & x_{15,14} & x_{15,15} & x_{15,16} \\
x_{16,1} & x_{16,2} & x_{16,3} & x_{16,4} & x_{16,5} & x_{16,6} & x_{16,7} & x_{16,8} & x_{16,9} & x_{16,10} & x_{16,11} & x_{16,12} & x_{16,13} & x_{16,14} & x_{16,15} & x_{16,16}
\end{pmatrix} \tag{3.4}$$

x-direction The reshuffled matrix $\tilde{U}$ has to be unitary. $\tilde{U}$ is defined as follows:

$$\begin{pmatrix}
x_{1,1} & x_{1,2} & x_{2,1} & x_{2,2} & x_{1,5} & x_{1,6} & x_{2,5} & x_{2,6} & x_{5,1} & x_{5,2} & x_{6,1} & x_{6,2} & x_{5,5} & x_{5,6} & x_{6,5} & x_{6,6} \\
x_{1,3} & x_{1,4} & x_{2,3} & x_{2,4} & x_{1,7} & x_{1,8} & x_{2,7} & x_{2,8} & x_{5,3} & x_{5,4} & x_{6,3} & x_{6,4} & x_{5,7} & x_{5,8} & x_{6,7} & x_{6,8} \\
x_{3,1} & x_{3,2} & x_{4,1} & x_{4,2} & x_{3,5} & x_{3,6} & x_{4,5} & x_{4,6} & x_{7,1} & x_{7,2} & x_{8,1} & x_{8,2} & x_{7,5} & x_{7,6} & x_{8,5} & x_{8,6} \\
x_{3,3} & x_{3,4} & x_{4,3} & x_{4,4} & x_{3,7} & x_{3,8} & x_{4,7} & x_{4,8} & x_{7,3} & x_{7,4} & x_{8,3} & x_{8,4} & x_{7,7} & x_{7,8} & x_{8,7} & x_{8,8} \\
x_{1,9} & x_{1,10} & x_{2,9} & x_{2,10} & x_{1,13} & x_{1,14} & x_{2,13} & x_{2,14} & x_{5,9} & x_{5,10} & x_{6,9} & x_{6,10} & x_{5,13} & x_{5,14} & x_{6,13} & x_{6,14} \\
x_{1,11} & x_{1,12} & x_{2,11} & x_{2,12} & x_{1,15} & x_{1,16} & x_{2,15} & x_{2,16} & x_{5,11} & x_{5,12} & x_{6,11} & x_{6,12} & x_{5,15} & x_{5,16} & x_{6,15} & x_{6,16} \\
x_{3,9} & x_{3,10} & x_{4,9} & x_{4,10} & x_{3,13} & x_{3,14} & x_{4,13} & x_{4,14} & x_{7,9} & x_{7,10} & x_{8,9} & x_{8,10} & x_{7,13} & x_{7,14} & x_{8,13} & x_{8,14} \\
x_{3,11} & x_{3,12} & x_{4,11} & x_{4,12} & x_{3,15} & x_{3,16} & x_{4,15} & x_{4,16} & x_{7,11} & x_{7,12} & x_{8,11} & x_{8,12} & x_{7,15} & x_{7,16} & x_{8,15} & x_{8,16} \\
x_{9,1} & x_{9,2} & x_{10,1} & x_{10,2} & x_{9,5} & x_{9,6} & x_{10,5} & x_{10,6} & x_{13,1} & x_{13,2} & x_{14,1} & x_{14,2} & x_{13,5} & x_{13,6} & x_{14,5} & x_{14,6} \\
x_{9,3} & x_{9,4} & x_{10,3} & x_{10,4} & x_{9,7} & x_{9,8} & x_{10,7} & x_{10,8} & x_{13,3} & x_{13,4} & x_{14,3} & x_{14,4} & x_{13,7} & x_{13,8} & x_{14,7} & x_{14,8} \\
x_{11,1} & x_{11,2} & x_{12,1} & x_{12,2} & x_{11,5} & x_{11,6} & x_{12,5} & x_{12,6} & x_{15,1} & x_{15,2} & x_{16,1} & x_{16,2} & x_{15,5} & x_{15,6} & x_{16,5} & x_{16,6} \\
x_{11,3} & x_{11,4} & x_{12,3} & x_{12,4} & x_{11,7} & x_{11,8} & x_{12,7} & x_{12,8} & x_{15,3} & x_{15,4} & x_{16,3} & x_{16,4} & x_{15,7} & x_{15,8} & x_{16,7} & x_{16,8} \\
x_{9,9} & x_{9,10} & x_{10,9} & x_{10,10} & x_{9,13} & x_{9,14} & x_{10,13} & x_{10,14} & x_{13,9} & x_{13,10} & x_{14,9} & x_{14,10} & x_{13,13} & x_{13,14} & x_{14,13} & x_{14,14} \\
x_{9,11} & x_{9,12} & x_{10,11} & x_{10,12} & x_{9,15} & x_{9,16} & x_{10,15} & x_{10,16} & x_{13,11} & x_{13,12} & x_{14,11} & x_{14,12} & x_{13,15} & x_{13,16} & x_{14,15} & x_{14,16} \\
x_{11,9} & x_{11,10} & x_{12,9} & x_{12,10} & x_{11,13} & x_{11,14} & x_{12,13} & x_{12,14} & x_{15,9} & x_{15,10} & x_{16,9} & x_{16,10} & x_{15,13} & x_{15,14} & x_{16,13} & x_{16,14} \\
x_{11,11} & x_{11,12} & x_{12,11} & x_{12,12} & x_{11,15} & x_{11,16} & x_{12,15} & x_{12,16} & x_{15,11} & x_{15,12} & x_{16,11} & x_{16,12} & x_{15,15} & x_{15,16} & x_{16,15} & x_{16,16}
\end{pmatrix} \tag{3.5}$$

So each row of the original U is split into multiple parts of length 4. These 4 length parts are then arranged as 2×2 blocks as can be seen in the colorcoded matrix representation.

y-direction The reshuffled matrix $\hat{U}$ has to be unitary. It is defined as follows:

$$\begin{pmatrix}
x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \\
x_{1,5} & x_{1,6} & x_{1,7} & x_{1,8} & x_{2,5} & x_{2,6} & x_{2,7} & x_{2,8} & x_{3,5} & x_{3,6} & x_{3,7} & x_{3,8} & x_{4,5} & x_{4,6} & x_{4,7} & x_{4,8} \\
x_{1,9} & x_{1,10} & x_{1,11} & x_{1,12} & x_{2,9} & x_{2,10} & x_{2,11} & x_{2,12} & x_{3,9} & x_{3,10} & x_{3,11} & x_{3,12} & x_{4,9} & x_{4,10} & x_{4,11} & x_{4,12} \\
x_{1,13} & x_{1,14} & x_{1,15} & x_{1,16} & x_{2,13} & x_{2,14} & x_{2,15} & x_{2,16} & x_{3,13} & x_{3,14} & x_{3,15} & x_{3,16} & x_{4,13} & x_{4,14} & x_{4,15} & x_{4,16} \\
x_{5,1} & x_{5,2} & x_{5,3} & x_{5,4} & x_{6,1} & x_{6,2} & x_{6,3} & x_{6,4} & x_{7,1} & x_{7,2} & x_{7,3} & x_{7,4} & x_{8,1} & x_{8,2} & x_{8,3} & x_{8,4} \\
x_{5,5} & x_{5,6} & x_{5,7} & x_{5,8} & x_{6,5} & x_{6,6} & x_{6,7} & x_{6,8} & x_{7,5} & x_{7,6} & x_{7,7} & x_{7,8} & x_{8,5} & x_{8,6} & x_{8,7} & x_{8,8} \\
x_{5,9} & x_{5,10} & x_{5,11} & x_{5,12} & x_{6,9} & x_{6,10} & x_{6,11} & x_{6,12} & x_{7,9} & x_{7,10} & x_{7,11} & x_{7,12} & x_{8,9} & x_{8,10} & x_{8,11} & x_{8,12} \\
x_{5,13} & x_{5,14} & x_{5,15} & x_{5,16} & x_{6,13} & x_{6,14} & x_{6,15} & x_{6,16} & x_{7,13} & x_{7,14} & x_{7,15} & x_{7,16} & x_{8,13} & x_{8,14} & x_{8,15} & x_{8,16} \\
x_{9,1} & x_{9,2} & x_{9,3} & x_{9,4} & x_{10,1} & x_{10,2} & x_{10,3} & x_{10,4} & x_{11,1} & x_{11,2} & x_{11,3} & x_{11,4} & x_{12,1} & x_{12,2} & x_{12,3} & x_{12,4} \\
x_{9,5} & x_{9,6} & x_{9,7} & x_{9,8} & x_{10,5} & x_{10,6} & x_{10,7} & x_{10,8} & x_{11,5} & x_{11,6} & x_{11,7} & x_{11,8} & x_{12,5} & x_{12,6} & x_{12,7} & x_{12,8} \\
x_{9,9} & x_{9,10} & x_{9,11} & x_{9,12} & x_{10,9} & x_{10,10} & x_{10,11} & x_{10,12} & x_{11,9} & x_{11,10} & x_{11,11} & x_{11,12} & x_{12,9} & x_{12,10} & x_{12,11} & x_{12,12} \\
x_{9,13} & x_{9,14} & x_{9,15} & x_{9,16} & x_{10,13} & x_{10,14} & x_{10,15} & x_{10,16} & x_{11,13} & x_{11,14} & x_{11,15} & x_{11,16} & x_{12,13} & x_{12,14} & x_{12,15} & x_{12,16} \\
x_{13,1} & x_{13,2} & x_{13,3} & x_{13,4} & x_{14,1} & x_{14,2} & x_{14,3} & x_{14,4} & x_{15,1} & x_{15,2} & x_{15,3} & x_{15,4} & x_{16,1} & x_{16,2} & x_{16,3} & x_{16,4} \\
x_{13,5} & x_{13,6} & x_{13,7} & x_{13,8} & x_{14,5} & x_{14,6} & x_{14,7} & x_{14,8} & x_{15,5} & x_{15,6} & x_{15,7} & x_{15,8} & x_{16,5} & x_{16,6} & x_{16,7} & x_{16,8} \\
x_{13,9} & x_{13,10} & x_{13,11} & x_{13,12} & x_{14,9} & x_{14,10} & x_{14,11} & x_{14,12} & x_{15,9} & x_{15,10} & x_{15,11} & x_{15,12} & x_{16,9} & x_{16,10} & x_{16,11} & x_{16,12} \\
x_{13,13} & x_{13,14} & x_{13,15} & x_{13,16} & x_{14,13} & x_{14,14} & x_{14,15} & x_{14,16} & x_{15,13} & x_{15,14} & x_{15,15} & x_{15,16} & x_{16,13} & x_{16,14} & x_{16,15} & x_{16,16}
\end{pmatrix} \tag{3.6}$$

Here each row of U is converted into a 4×4 block and those blocks are arranged according to row order.

Finding a ternary unitary

It is not feasible to randomly search for ternary unitaries in the group of all potential 16×16 unitary matrices. Even when each matrix entry is constrained to either zero or one there are $16^2 = 256$ positions with 2 possible values each, so 2^{256} combinations in total. Constraining the matrices further to only allow permutation matrices (permutations of the identity matrix) there are still $16! = 2.09 \cdot 10^{13}$ different matrices. This is still too much to check the potential ternary unitarity of each of them on a regular computer. So it is necessary to look at the problem in a more systematic way.
What happens if one takes the Kronecker product of two one-dimensional dual unitaries? If the result of this operation was always or sometimes a ternary unitarity, it would be very helpful, since for two-qubit dual unitaries an explicit parametrization (see equation 2.7) exists. However, this is not the case. Instead, although all matrices generated this way are still unitary in t- as well as in x-direction, none of them can ever be unitary in y-direction. This becomes clearer when looking at the parametrization of two-qubit dual unitaries and more precisely at the main defining matrix $V(J)$ (equation 2.9). The Kronecker product of two of these matrices $V(J_1) \otimes V(J_2)$ is given below in

equation 3.7 with matrix entries defined in equations 3.8, 3.9, 3.10 and 3.11.

$$\begin{pmatrix}
A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -iB & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -iB & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -iD & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -C & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -C & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -iD & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -iD & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -iD & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -iB & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -iB & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A
\end{pmatrix} \tag{3.7}$$

$$A = e^{-iJ_1 - iJ_2} \tag{3.8}$$

$$B = e^{iJ_2 - iJ_1} \tag{3.9}$$

$$C = e^{iJ_1 + iJ_2} \tag{3.10}$$

$$D = e^{iJ_1 - iJ_2} \tag{3.11}$$

The reshuffling that has to be unitary for unitarity in x-direction is in fact identical to the original matrix 3.7 (compare the matrix representation of the reshuffling in equation 3.12, which was computed via Wolfram Mathematica [12]).

$$\begin{pmatrix}
A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -iB & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -iB & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -iD & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -C & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -C & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -iD & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -iD & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -iD & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -iB & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -iB & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A
\end{pmatrix} \tag{3.12}$$

However, the reshuffling for y-direction can never be unitary (compare equation 3.13)

$$\begin{pmatrix}
e^{-iJ_1-iJ_2} & 0 & 0 & 0 & 0 & 0 & -ie^{iJ_2-iJ_1} & 0 & 0 & -ie^{iJ_2-iJ_1} & 0 & 0 & 0 & 0 & 0 & e^{-iJ_1-iJ_2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-ie^{iJ_1-iJ_2} & 0 & 0 & 0 & 0 & 0 & -e^{iJ_1+iJ_2} & 0 & 0 & -e^{iJ_1+iJ_2} & 0 & 0 & 0 & 0 & 0 & -ie^{iJ_1-iJ_2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-ie^{iJ_1-iJ_2} & 0 & 0 & 0 & 0 & 0 & -e^{iJ_1+iJ_2} & 0 & 0 & -e^{iJ_1+iJ_2} & 0 & 0 & 0 & 0 & 0 & -ie^{iJ_1-iJ_2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
e^{-iJ_1-iJ_2} & 0 & 0 & 0 & 0 & 0 & -ie^{iJ_2-iJ_1} & 0 & 0 & -ie^{iJ_2-iJ_1} & 0 & 0 & 0 & 0 & 0 & e^{-iJ_1-iJ_2}
\end{pmatrix} \tag{3.13}$$

This construction is thus not able to find a ternary unitary.
It is interesting to note that if the potential values of all matrix entries are limited to either zero or one, the only possible way for a matrix to be unitary as well as unitary in y-direction is that $\hat{U}$ has to be a kind of "sudoku matrix", meaning that there can only be a single value of one per row, column and 4×4 block. In [3] those matrices are named "S-permutation matrices". Thus, when searching for a "self-ternary" gate within permutation matrices, it suffices to limit the permutation matrices to S-permutation matrices. However, there are still $4!^{2\cdot 4} = 24^8 = 1.1 \cdot 10^{11}$ 16×16 S-permutation matrices with block size 4 [3]. So simply testing all those S-permutation matrices is not numerically feasible, considering that their generation is also non-trivial.

However, when considering the SWAP gate and its role as self-dual gate for two-qubit gates, this raises the question if there is such a gate for the four-qubit case. The main property of the SWAP gate is that it swaps two qubits. In fact it swaps exactly b and c in the graphical interpretation figure 3.9. Its effect on the input can be visualized as "crossing the wires" inside the gate. It seems reasonable to assume that the same method applied to a cube should lead to a self-ternary two dimensional gate. Therefore the goal is to connect each corner of the cube representation with its opposite corner.
To do this, firstly a and d have to be swapped. Then b and c have to be swapped. This sets the input up in a way that now the opposite corners are connected in y direction. For example a is opposite of h. When a is swapped with d, a is directly below h and thus they are connected in y-direction. The same applies to the other 3 lower corners. The resulting quantum gate can be built by the circuit pictured in figure 3.11 and its matrix representation is given in equation 3.14.

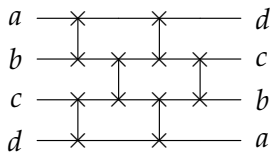


Figure 3.11.: Quantum circuit to build the self-ternary four-qubit gate

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \tag{3.14}$$

This gate is in indeed a ternary unitary. Also, it is a S-permutation matrix and also self-ternary. It can be interpreted as the equivalent to the SWAP gate for ternary unitaries. In terms of permutation matrices, it is the permutation Perm(0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15).
Also, in [8] a permutation matrix that is ternary unitary but not self-ternary was found. This is the permutation matrix Perm(4, 3, 13, 10, 14, 9, 7, 0, 11, 12, 2, 5, 1, 6, 8, 15).

3.2.2. 2D circuits of alternating 1D dual unitaries

Another option to extend dual unitary circuits into more than one space dimension is to look at circuits made of alternating dual unitaries. For example, a circuit could be built that is like figure 3.1 in x-t-direction for all y and the different y-levels are also connected by dual unitaries. The resulting circuit is visualized in figure 3.12. As before, red tones visualize dual unitary gates and blue tones their Hermitian conjugates.

For readability purposes all two-qubit gates between different y-levels are visualized as lines instead of rectangles. Dark blue is on y-level $y = 0$, light blue on $y = 1$. Analogously, red is for $y = 0$ and orange for $y = 1$. The purple gates connect the different y-levels, the green gates do the same in the Hermitian conjugate part.

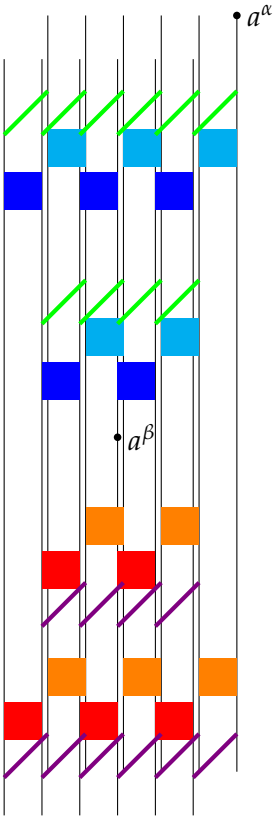


Figure 3.12.: 2D grid of alternating level dual unitaries, darkblue y=0, light blue y=1, green connects y-levels

The question is whether such a grid be simplified in the same way as the one-dimensional case can. As pictured in figures 3.13 and 3.14 it can be simplified indeed. Again, between the initial setup and step 1 the unitarity of all gates was used, and between step 1 and step 2 the dual unitarity of the gates was used. Why do the allowed simplifications lead to a similar result as for the one-dimensional case? First, for all y-layers the structure of the grid is identical to the one-dimensional case. By setting up the grid in this way, all simplification steps work as before. Also, on each level the potentially allowed area follows the light cone restriction. Putting them together leads to a light cone encompassing all y-layers.

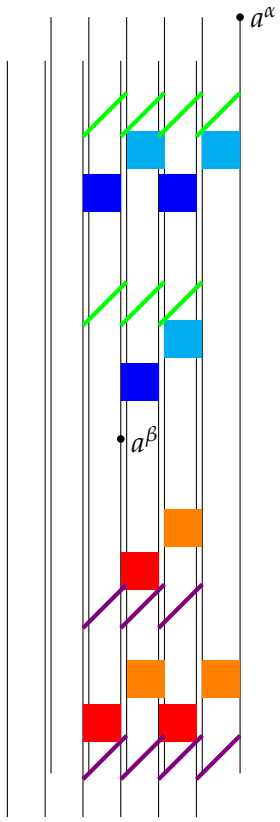


Figure 3.13.: 2D grid of alternating level dual unitaries, simplification step 1

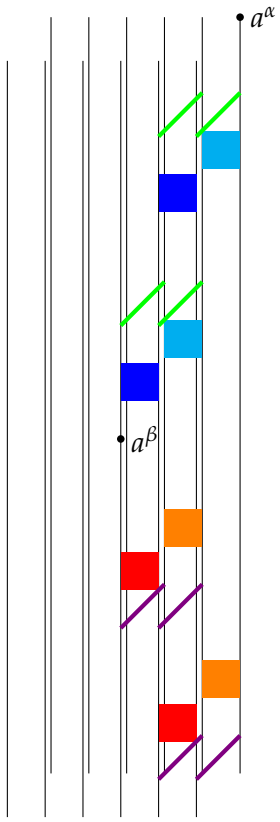


Figure 3.14.: 2D grid of alternating level dual unitaries, simplification step 2

Changes to the analytical solution

To compute the analytical solution for this new scenario, one needs to look at the exact setup of the remaining circuit after simplification. It can be seen that - ignoring the gates of the other layers - each y-layer looks exactly as the one-dimensional equivalent figure 3.4. So each y-layer can be described by the one-dimensional solution, if the other layers and the connecting layer were to be ignored. Unfortunately, the multi-gate equation 3.1 can not be used to compute the analytical solution, since the y-connecting layer does not have the same structure as each y-layer and in particular can not be reduced to a chain of M_v. The main difference is, that on each y-layer and in the wanted structure of figure 3.4, each gate shares a qubit with the next inner gate and shares the other qubit with the next outer gate. In contrast, the gates of the y-connecting layer are always between the same qubit on different y-levels. For example, for a gate on $q_{a,0}$ and $q_{a+1,0}$ and its outer neighbor gate on qubits $q_{a+1,0}$ and $q_{a+2,0}$ and the same two gates on level $y = 1$, the connecting layer consists of a gate between $q_{a,0}$ and $q_{a,1}$, a gate between $q_{a+1,0}$ and $q_{a+1,1}$, another gate between $q_{a+1,0}$ and $q_{a+1,1}$ and a gate between $q_{a+2,0}$ and $q_{a+2,1}$. It can be seen that this set of gates does not show the same overlap or shares the same structure as desired. Thus, the simple analytical solution can unfortunately no longer be used for this case, except for the special case where all used gates commutate and thus each y-level can be evaluated separately and afterwards the connecting layer can be evaluated.

4. Implementation

This chapter analyzes the result of implementing the benchmark scenario on a quantum circuit simulator. This benchmark compares the results computed by equations 2.18 and 3.1 with the result computed by building the corresponding circuit in IBM's qiskit [16]. This result is obtained by getting the matrix representation A of the circuit and then computing equation 4.1.

$$res = \frac{1}{2^{2L}} tr(A) \tag{4.1}$$

The source code of the implementation can be found in the corresponding GitLab repository [18]. The comparative implementation is done in Phython using IBM's qiskit library [16]. An implementation in Julia might be faster, however no direct qiskit package is currently available for Julia. All executions of the program and all time measurements were done on a laptop with Ryzen 9 5900HS CPU. This is a 35W laptop CPU with 8 cores and 16 threads released in 2021. The laptop has 16 GB RAM.
The main focus of this analysis is to find out whether the translation of the original setup to a quantum circuit setup works well and whether equation 2.18 is a good candidate for such a benchmark. The main difference between those two scenarios is that the original grid is assumed to be of infinite width and also implements periodic boundary conditions in temporal direction. It is easy to see that this can not directly be translated onto any quantum computer or simulator, since they only have a finite number of qubits available.

4.1. Boundary conditions

The aforementioned problems have to be accounted for when implementing those circuits. The straightforward solution to simulate a circuit of infinite width is imposing periodic boundary conditions on the grid in horizontal direction. This is done by simply adding another dual unitary gate between qubit $n-1$ and qubit 0 (for a grid of n qubits numbered from 0 to $n-1$). Of course, it has to be ensured that these added gates not present in the idealized original grid do not interfere with the computation results.
The evaluation of the trace in equation 2.16, which is the equivalent to temporal

periodic boundary conditions, is unproblematic on a simulator, since it returns a matrix representation of the whole quantum circuit anyway. Then the trace of this matrix can be computed. As discussed in chapter 5 it is a problem to do this on a real quantum computer, however.

4.1.1. Placement limits for a_y^β and a_x^α

As mentioned above, it has to be ensured that the newly added gates connecting q_{n-1} with q_0 don't interfere with the actual computation. So the remaining circuit after simplification as pictured in figure 2.7 must not interact with these added gates. This means that neither a^α nor a^β are allowed to be on the two outer qubits in each direction, meaning q_0, q_1, q_{n-2} and q_{n-1}. This of course limits the usable locations for a^α and a^β rather harshly, especially since on a computer with 16 GB RAM 8 qubits is the maximum possible circuit size in qiskit before the matrix of the circuit becomes too large to store in memory (caused by the exponential growth of the matrix representation of size $2^n \times 2^n$). However, it is the most feasible solution on a quantum circuit simulator. On a real quantum computer it might be possible to exploit the topology of the physical qubits to get periodic boundary conditions.

4.1.2. Maximum t for a given circuit width

Another problem that arises with horizontal periodic boundary conditions is that with increasing t the width of the light cone where non-zero results are possible increases. If it increases too much, the periodic boundaries might create unwanted overlap between the right part and the left part of the light cone, which of course would not happen in a circuit of unlimited width. Therefore t has to be chosen in a way that prevents this. In principle, the width of the light cone at t has to be smaller than the number of qubits (excluding q_0 and q_{n-1}). The width of the light cone for any t is $2t$, since each row in the diagrammatic representation of the correlation function represents a half time step. So to be safe from interactions introduced by periodic boundary conditions, equation 4.2 (or, alternatively equation 4.3 in relation to L) has to be fulfilled.

$$n_{qubits} - 2 \geq 2t \tag{4.2}$$

$$L - 1 \geq t \tag{4.3}$$

4.2. Single gate

First the exact same scenario as in [2] is implemented. This scenario consists of a quantum circuit of $2L$ qubits, controlled by parameter L. All dual unitary gates are

identical, and there a two subscenarios. One uses the SWAP gate, since it is numerically easy and has no risk of escalating floating point inaccuracies, the next uses a randomly generated dual unitary. The program utilizes multithreading with 10 threads.

4.2.1. SWAP

This section shows some results for a circuit of SWAP gates. The tables contain information about the parameter L defining the width of the circuit, the number of computed results, the number of total errors (which are all differences between the result from the quantum circuit as defined in 4.1 and the analytical solution given in equation 2.18), the number of so-called boundary errors, where either a^α or a^β are on one of the outer qubits or on the direct neighbor of one of the outer qubits, and the average time for a single evaluation. The difference between the total number of errors and the number of boundary errors can be explained by the errors introduced by accidentally overlapping the ligthcone with itself because of the periodic boundary conditions. The average evaluation time is obtained by evaluating a batch of 10 results with randomly chosen parameters α, β, x, y, t, taking the average of this batch and repeating this multiple times, afterwards the average over all batches is computed. Note that the evaluation for time measurement takes place single-threaded, so in practice 10 comparisons are made in the given timeframe.
An example circuit is pictured in figure 4.1.

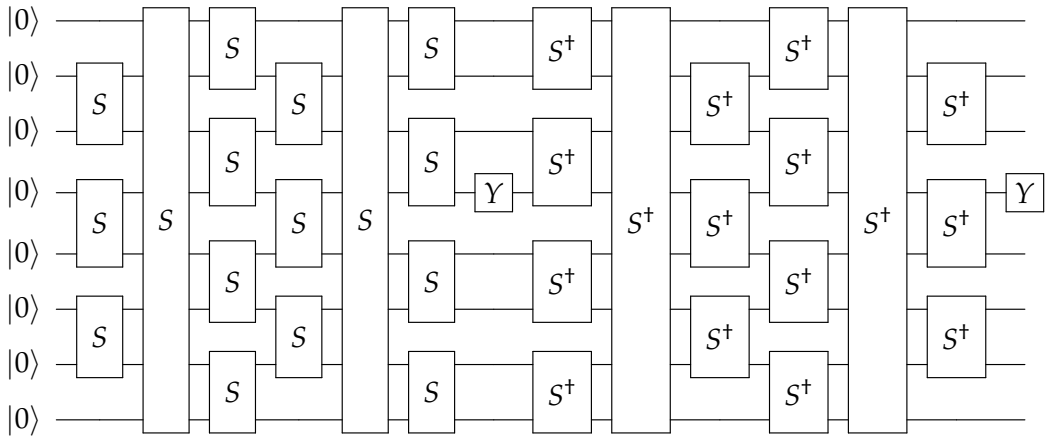


Figure 4.1.: Quantum circuit of the correlation function for $\alpha = \beta = 2$, $x = y = 0$, $t = 2$, S is used as shorthand notation for $SWAP$

For the smallest ranges of t and x and y, analyzed in table 4.2, it can be seen that all

results can be obtained without errors beginning at $L = 4$. For results from larger domains (table 4.1 and 4.3) this circuit size is not enough for error free results. However, for the medium case where $x, y \in -1.5, ..., 2$, but $t <= 2$, the step to $L = 6$ is enough to eliminate all those discrepancies. Also, it can be clearly seen that the execution time for a single comparison increases exponentially with increasing L.

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
4	3136	48	30	0.03492
6	3136	12	0	12.15818

Table 4.1.: Results for $U = SWAP$, $x \in -1.5, ..., 2$, $y \in -1.5, ..., 2$, $t_{max} = 4$

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
2	512	18	15	0.00154
4	512	0	0	0.01853

Table 4.2.: Results for $U = SWAP$, $x \in -1, .., 1$, $y \in -1, .., 1$, $t_{max} = 2$

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
4	1568	12	12	0.01744
6	1568	0	0	6.17833

Table 4.3.: Results for $U = SWAP$, $x \in -1.5, .., 2$, $y \in -1.5, .., 2$, $t_{max} = 2$

4.2.2. Random

Now the same ranges and number of qubits are used, but the SWAP gate is replaced with the randomly generated gate U_{rdm}. The exact definition of the used gate can be found in the appendix (equations A.1 to A.17). It was generated within the program using equation 2.7.
The results show the same behavior as for the SWAP gate scenario, however the general number of errors is higher. This can be explained by escalating floating point errors, since for randomly generated dual unitaries every single matrix entry utilizes the full available floating point precision to ensure the result fulfills the unitary constraints in both directions. The sheer amount of arithmetic operations on circuits built entirely from those gates amplifies any small floating point error and results in noticeable differences between the analytical solution and the solution obtained from the circuit. However, in most of the cases where these errors occur, both solutions are still somewhat

close, but not close enough to be assumed identical.

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
4	3136	765	423	0.03070
6	3136	171	0	11.06941

Table 4.4.: Results for $U = rdm$, $x \in -1.5, ..., 2$, $y \in -1.5, ..., 2$, $t_{max} = 4$

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
2	512	180	153	0.00157
4	512	18	0	0.01784
6	512	18	0	5.85110

Table 4.5.: Results for $U = rdm$, $x \in -1, .., 1$, $y \in -1, .., 1$, $t_{max} = 2$

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
4	1568	108	54	0.01819
6	1568	72	0	5.51481

Table 4.6.: Results for $U = rdm$, $x \in -1.5, .., 2$, $y \in -1.5, .., 2$, $t_{max} = 2$

4.3. Multi-gate

The next scenario includes multiple dual unitaries. It would be possible to do this with a totally random pattern, however for ease of implementation, here a certain number of dual unitaries are chosen or generated and then applied repetitively. The first sub-scenario uses the following array of numerically easy dual unitaries for $L = 2$: SWAP, iSWAP, $SWAP_{neg}$, $fSim_{Syc}$(the fSim gate implemented in the Sycamore architecture). For larger L, this array is repeated accordingly, so that the size of the array is at least $2L$. The definitions of the negative SWAP gate and the Sycamore-fSim gate are given in equations 4.4 and 4.5.

$$\mathrm{SWAP}_{neg} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.4}$$

$$\text{fSim}_{Syc} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & -i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

The second subscenario uses a mixture of randomly generated gates and the iSWAP gate. The array [iSWAP, U_{rdm}] is repeated as often as necessary to reach a length of at least $2L$.
The third subscenario uses an array of different randomly generated dual unitaries enumerated as U_0 to U_L and repeated once to reach length $2L$. The exact definition of those gates can be found in the appendix (equations A.18 to A.119). Again multithreading with ten threads is utilized.

4.3.1. Non-random gates

As mentioned before, the used gates for this scenario are SWAP, iSWAP, $SWAP_{neg}$, $fSim_{Syc}$, repeated to a length of $2L$. The behavior for this multi-gate scenario is very similar to that of the single-gate SWAP scenario. Therefore it can be concluded that circuits of multiple dual unitaries can indeed be described by equation 3.1. The extension of grids of a single dual unitary to circuits of multiple dual unitary gates is therefore indeed possible.

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
4	3136	30	16	0.03215
6	3136	4	0	11.59519

Table 4.7.: Results for array of safe gates, $x \in -1.5, ..., 2$, $y \in -1.5, ..., 2$, $t_{max} = 4$

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
2	512	14	13	0.00181
4	512	0	0	0.01696

Table 4.8.: Results for array of safe gates, $x \in -1, .., 1$, $y \in -1, .., 1$, $t_{max} = 2$

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
4	1568	4	4	0.01826
6	1568	0	0	5.69236

Table 4.9.: Results for array of safe gates, $x \in -1.5, .., 2$, $y \in -1.5, .., 2$, $t_{max} = 2$

4.3.2. Mixture of iSWAP and U_{rdm}

In this scenario the two gates iSWAP and U_{rdm} (defined in the appendix in equations A.1 to A.17) are repeated to a length of $2L$. As for the single gate scenario, the introduction of randomly generated gates introduces some floating point error again. However, since half of the gates are not randomly generated, the effect is not as large as for a grid consisting completely of randomly generated gates. Otherwise the results are in the expected range.

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
4	3136	627	351	0.03592
6	3136	103	0	10.67562

Table 4.10.: Results for mixed array of iSWAP and U_{rdm}, $x \in -1.5, ..., 2$, $y \in -1.5, ..., 2$, $t_{max} = 4$

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
2	512	134	119	0.00158
4	512	8	0	0.01787
6	512	8	0	5.25262

Table 4.11.: Results for mixed array of iSWAP and U_{rdm}, $x \in -1, .., 1$, $y \in -1, .., 1$, $t_{max} = 2$

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
4	1568	42	18	0.01926
6	1568	30	0	6.01287

Table 4.12.: Results for mixed array of iSWAP and U_{rdm}, $x \in -1.5, .., 2$, $y \in -1.5, .., 2$, $t_{max} = 2$

4.3.3. Random

In comparison to the previous multi-gate scenarios, the number of errors increases again because of the higher amount of random gates. Disregarding these errors, even for grids of multiple random gates the results behave as expected and as long as the circuit is embedded in a large enough grid, the solution is described exactly by the corresponding equation.

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
4	3136	765	423	0.03301
6	3136	171	0	11.79374

Table 4.13.: Results for array of random gates, $x \in -1.5, ..., 2$, $y \in -1.5, ..., 2$, $t_{max} = 4$

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
2	512	180	153	0.00163
4	512	18	0	0.01684
6	512	18	0	5.79567

Table 4.14.: Results for array of random gates, $x \in -1, .., 1$, $y \in -1, .., 1$, $t_{max} = 2$

L	No. of results	No. of errors (total)	No. of boundary errors	Avg. execution time [s]
4	1568	108	54	0.01817
6	1568	72	0	5.52640

Table 4.15.: Results for array of random gates, $x \in -1.5, .., 2$, $y \in -1.5, .., 2$, $t_{max} = 2$

4.3.4. Interpretation of results

In conclusion for all examined scenarios in both the single gate setup and the multi-gate setup the results fit very well to the analytical solution and thus the step from hypothetical infinite grids to a quantum simulator can be done successfully. It can be seen that it is important to chose the domain of the inputs in a way that keeps a_y^β and a_x^α out of the two outer qubits in both directions and also respects equation 4.3, so that there are no unwanted interactions of the light cone with itself. If this is done, the only remaining source of errors are the floating point inaccuracies introduced by executing multiple arithmetic operations on grids with randomly generated gates. Unfortunately it is not possible to round the entries of the generated matrices to a precision not utilizing the whole available floating point precision, since then they will no longer be recognized as unitary by qiskit and can thus not be applied in a quantum circuit. A possible way to prevent these errors is to run the comparative algorithm on an already simplified version of the circuit, since this drastically reduces the amount of floating point operations because the simplified circuit consists of drastically fewer quantum gates.

5. Adapting the scenario to run on a real quantum computer

In chapter 4 it was shown that circuits of dual unitaries work fine as a benchmark when using a quantum circuit simulator. This chapter discusses which additional adaptations to such an implementation have to be added if this benchmark is run on a quantum computer instead of a simulator. This again regards mostly the periodic boundary conditions. Also one has to consider how any arbitrary dual unitary can be built efficiently from the basic gates available on the respective quantum computer.

5.1. Spatial periodic boundary conditions

Firstly the periodic boundary conditions in spatial directions are revisited. The main problem lies in the fact that while gates connecting far removed qubits (like q_0 and q_{n-1} for this specific case) can be easily simulated, connecting non-neighboring qubits is not trivial on a physical quantum computer. Of course there are different qubit topologies and thus different definitions of neighboring qubits, however connecting the qubits farthest away from each other will pose problematic in general.

5.1.1. Singular value decomposition

One option is to decompose the violating two-qubit gate into single-qubit gates. If this is done by simply using generating single-qubit gates there might be a lot of single-qubit gates required, however. Therefore an approach limiting the number of gates required for such a decomposition is needed. This can be done by singular value decomposition.
In general, singular value decomposition is a form of matrix decomposition in accordance with equation 5.1 [7].

$$M = U\Sigma V^\dagger \tag{5.1}$$

Σ is a diagonal matrix whose entries are called the singular values of M. All singular values are non-negative [7]. The matrices U and $V^\dagger$ are both unitary.
Tensor network representation of singular value decomposition is depicted in figure

5.1.
For quantum circuits (and more specifically for a two-qubit-gate), this can be done in

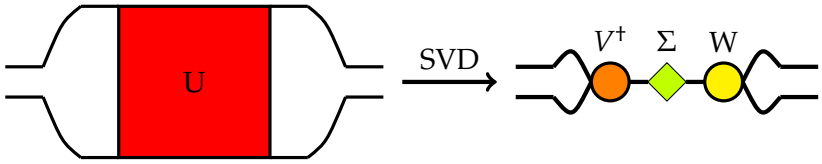


Figure 5.1.: Tensor network representation of singular value decomposition

the following way: First, reshape U into a $2 \times 2 \times 2 \times 2$-tensor. Then interchange the second and the third dimension. Afterwards, reshape this back into a 4×4 matrix. This matrix is now decomposed into $U\Sigma V^\dagger$ by standard singular value decomposition. Then, count the number of singular values in Σ. This is equivalent to the number of non-zeros in Σ. This number is called n_{nz} hereafter. All three matrices U, Σ, $V^\dagger$ are truncated in accordance with n_{nz} now. U keeps its first dimension, the second is trucated to n_{nz}. For $V^\dagger$, the second dimension stays the same and the first is truncated to n_{nz}. Σ is truncated into a $n_{nz} \times n_{nz}$ matrix. Now, U is reshaped into a $2 \times 2 \times n_{nz}$-tensor and $V^\dagger$ is reshaped into a $n_{nz} \times 2 \times 2$-tensor. The resulting U and $V^\dagger$ are then separated into n_{nz} single-qubit gates. For example, for U the first gate is retrieved by setting the third index to 1, to 2, etc. For $V^\dagger$ this is done on the first index. This leads to n_{nz} u_i and v_i. The original gate can be reconstructed as the sum over the kronecker product of the u_i and v_i where each summand is weighted with the respective singular value σ_i (see equation 5.2).

$$M = \sum_{j=0}^{n_{nz}} \sigma_j u_j \otimes v_j \tag{5.2}$$

However, the downside of this approach is that while fast and compact, the resulting gates u_i and v_i are not unitary in general. Therefore, using SVD to circumvent gates connecting far removed qubits is only helpful in a classical simulator. It can not be done directly on a quantum computer. Potentially, it might be possible to use a hybrid approach and apply SVD outside the quantum computer and then feed the results back into it. Since the gates in question occur twice for each timestep, once in the "normal" part and once in the "daggered" part, this unfortunately might not be very feasible. Therefore, another solution for the problem caused by periodic boundaries is needed.

5.1.2. Utilizing 2D qubit setup

The other option is to exploit the topology of the quantum computer. This could potentially be done on a 2D grid of qubits by building a ring of qubits and thus making

the additional border gates unnecessary. Instead, this would implement periodic boundary conditions directly. An example of such a structure is pictured in figure 5.2.

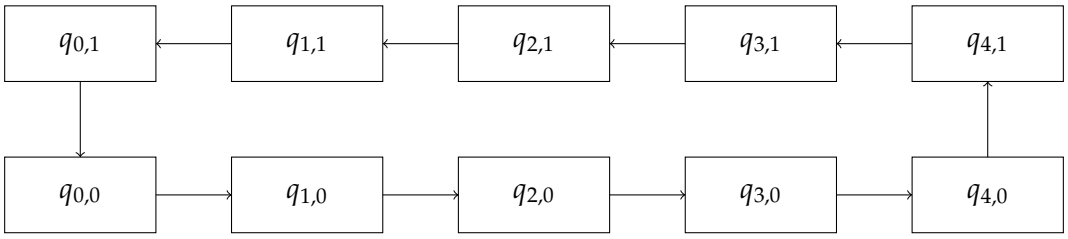


Figure 5.2.: Example of qubit topology and interconnectivity for a 10-qubit ring

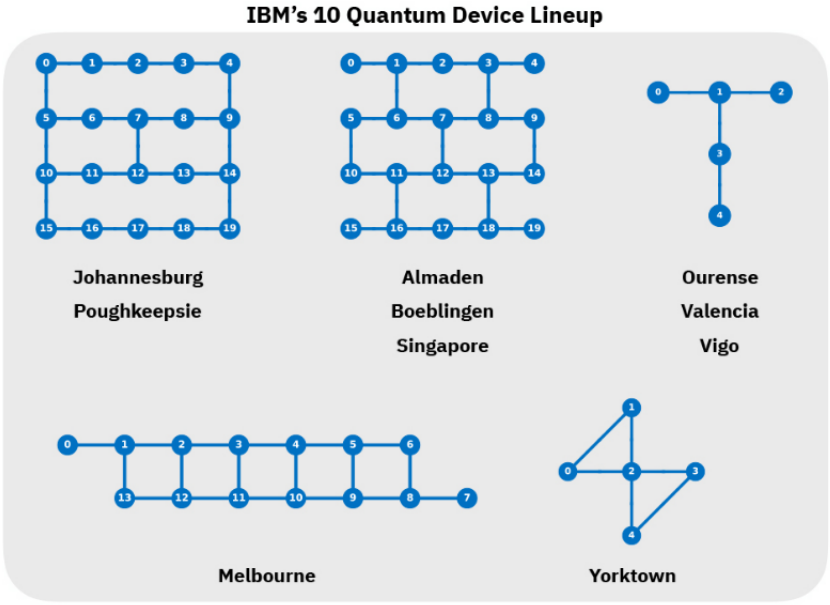


Figure 5.3.: Some qubit topologies of IBM quantum computers, from [10]

The smaller available quantum computers reachable via IBM Q unfortunately have such an interconnectivity that no ring of qubits can be found [13] [11]. This includes the five-qubit computers Lima, Quito, Belem, Athens, Santiago and Manila. The newer five-qubit quantum computer Yorktown has a ring, however only of size 3 qubits, which is too small to simulate a meaningful dual unitary circuit. Also one of the connections would go against its preferred direction. This direction results from the interaction

between qubits being stronger if the qubit with higher frequency is chosen as control (see [22]). So by default CNOT gates and thus any two-qubit gates are applied in this direction on IBM's quantum computers. The 16-qubit quantum computer Melbourne (pictured in figure 5.3 among some other qubit topologies of quantum computers at IBM) contains a ring of the 12 qubits, but this goes against the prefered direction for two-qubit gates for some of the qubits again. However, this is not a big problem, since the control and target qubits of any CNOT gate can be easily inverted by applying Hadamard gates on both qubits before and after the CNOT in question.
The qubit grid of Google's sycamore architecture[1] is a lot more interconnected than the previously listed grids, since most qubits are coupled with 4 neighbors each, so in principle larger rings of qubits are possible (see figure 5.4 for the grid layout).

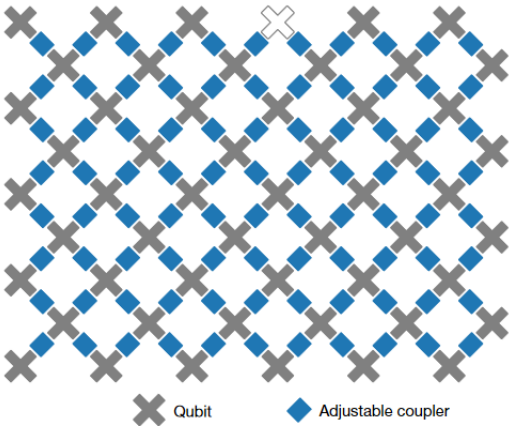


Figure 5.4.: Interconnectivity of Google's Sycamore architecture, from [1] (p.506)

Figure 5.5 shows a potential ordering of qubits that leads to a circle of 46 qubits. However, different coupler activation patterns than the ones published in [1] need to be used to realize this circle. Figure 5.6 shows the original coupler activation patterns.

All operations not induced by periodic boundary conditions are strictly between neighboring qubits for the scenario of these dual unitary circuits. This connectivity will be available in most current and future quantum computers and thus it does not further restrict the requirements for connectivity.

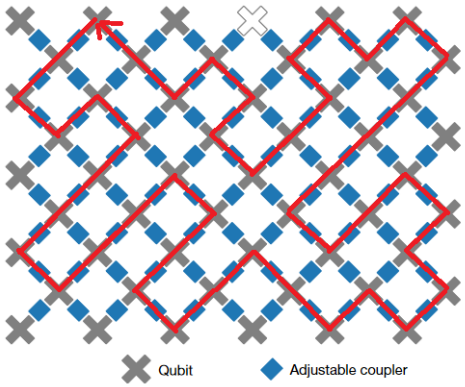


Figure 5.5.: Path of 46 qubits forming a circle on Google's sycamore architecture

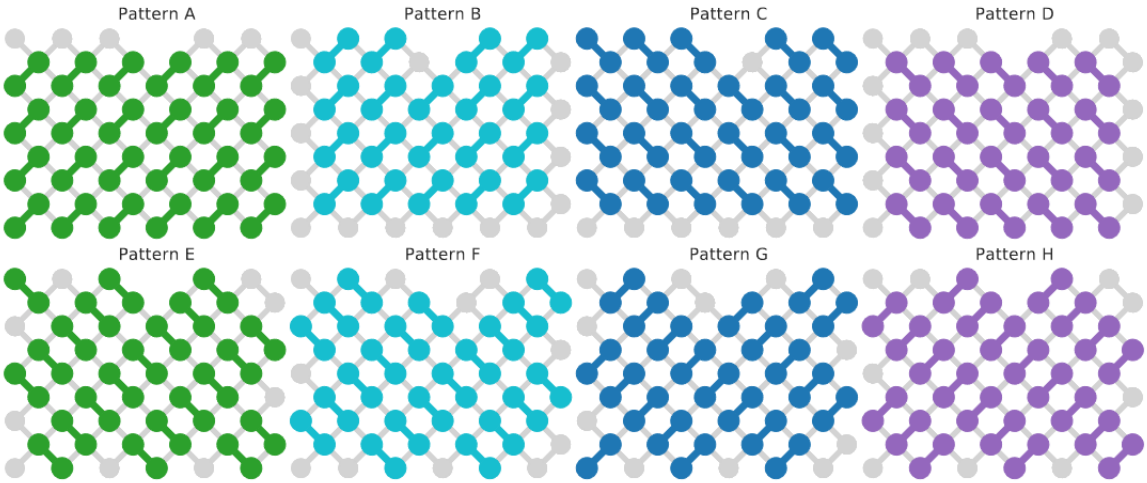


Figure 5.6.: Coupler activation patterns of Google's sycamore, sequences used by Google were either ABCDCDAB or EFGH, from supplemental material of [1] (p.28)

5.2. Temporal periodic boundary conditions

Another problem that arises when switching from a simulator to a real quantum computer is the existence of temporal periodic boundary conditions in the grid setup. They are present in the form of a trace over the whole circuit. While taking the trace of a circuit is fairly trivial on a quantum computer simulator, which simply gives a matrix representation of the whole circuit that can be used to obtain said trace, this is

not so easy on a quantum computer, where measurements are the only possible way to interact with the circuit.

5.2.1. Computing the trace of a quantum circuit on a quantum computer

Naive

The naive way to compute the trace on a real quantum computer is to count how often each input state is mapped to itself by the circuit and average this over all input states of the computational basis [5]. For a n-qubit system there are 2^n computational basis states and the overlap between input and output state needs to be averaged over a sufficient number of tries. Therefore, getting the trace this way is highly computationally expensive.

Other approaches

The trace can also be computed by summing all eigenvalues of a matrix [24]. In principle, the phase estimation algorithm can be used to estimate eigenvalues of a gate U (see [15]). However, since the wanted trace is taken of the gate representing the whole quantum circuit, the circuit for phase estimation of this gate would also be of large size, since multiple controlled-U gates of this gate representing the whole circuit are part of the phase estimation circuit. Therefore, this approach is not feasible for dual unitary circuits using many qubits.
In paper [5] another way to compute the trace of a unitary operator is referenced. It utilizes a so-called DQC1 circuit. To estimate the trace of a gate of n qubits, this DQC1-circuit has $n + 1$ qubits, which initally are in a certain state. For this circuit the expectation value for the Pauli gates X and Y is computed by repeated sampling. This can then be used to estimate the trace. In the same paper another approach is introduced that can be interpreted as simultaneously solving two polynomials, however the authors assume that this approach is still computationally expensive [5].
Therefore, it seems like the needed evaluation of the trace in the equation for the correlation function is one major hindrance to use these circuits as a benchmark for real quantum computers.

5.3. Building the gates

The other part that has to be adapted for running such circuits on a real quantum computer are the gates. On a simulator, any unitary can be defined as a gate and then added to a circuit. In contrast, quantum computers supply a certain set of basic gates

and all other gate have to be built from this set. This set differs from quantum computer to quantum computer. To elaborate how general dual unitaries can be built from more basic gates, the parametrization from equation 2.7 has to be compared against the set of available gates of the respective quantum computer.

5.3.1. IBM Q

First the quantum computers used by IBM Q are analyzed. The set of basic gates varies between quantum computers in the set of quantum computers made available by IBM. For example, the 16-qubit Melbourne computer has the following set of basis gates: CX, I, RZ, $\sqrt{X}$, X. The newer computers also share this set. The matrix form of these gates is given in eqations 5.3, 5.4, 5.5 and 5.6.

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{5.3}$$

$$RZ(\lambda) = \begin{pmatrix} e^{-i\frac{\lambda}{2}} & 0 \\ 0 & e^{i\frac{\lambda}{2}} \end{pmatrix} \tag{5.4}$$

$$\sqrt{X} = \frac{1}{2}\begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix} \tag{5.5}$$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{5.6}$$

The CX gate definition given here differs from the standard definition of the CNOT gate (equation 5.7) because of the little endian qubit ordering in qiskit's enumerating convention. The CNOT gate is simply the CX gate given here with flipped control and target.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{5.7}$$

As a first task, it is checked how the SWAP gate (in its role as the simplest two-qubit dual unitary gate) can be built from those basis gates. In [23] a decomposition for the SWAP gate is given (see equation 5.8).

$$SWAP = CNOT \cdot CX \cdot CNOT \tag{5.8}$$

So the remaining problem is finding a way to get the CNOT gate from the given basis gates. Since CNOT is simply a version of CX with flipped control and target, it is already available in the provided basis gates. Therefore, the SWAP gate can indeed be easily built from IBM's basic gates.
For the general dual unitary, $V(J)$ is looked at (equation 2.9). Unfortunately, the parametrization given in equation 2.7 can not be used to help with this, since it does not only contain basic Pauli gates, but also the exponentiation of the result. There exists at least a decomposition of the form

$$V(J) = K_1 A K_2 \tag{5.9}$$

where $K_1, K_2 \in SU(2) \otimes SU(2) \otimes U(1)$ and $A \in exp(\mathfrak{h})$, as described in detail in [14]. The matrices from $SU(2)$ can again be decomposed using the Pauli gates, which can be derived from the basic gate set. It is unknown whether there exists a more efficient decomposition for a general dual unitary.

5.3.2. Sycamore

Second, Google's Sycamore architecture is considered. The supplementary material to article [1] mentions that they chose to implement their fSim gate with parameters $\theta = 90°$ and $\phi = 30°$, which results in an iSWAP gate with inverse sign at the i-positions. The provided single-qubit gates are $X^{\frac{1}{2}}$, $Y^{\frac{1}{2}}$ and $W^{\frac{1}{2}}$, listed in equations 5.10, 5.11 and 5.12.

$$X^{\frac{1}{2}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} \tag{5.10}$$

$$Y^{\frac{1}{2}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \tag{5.11}$$

$$W^{\frac{1}{2}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -\sqrt{i} \\ \sqrt{-i} & 1 \end{pmatrix} \tag{5.12}$$

The chosen fSim gate implemented on Sycamore is a dual unitary itself. Also, it is possible to build a SWAP gate from iSWAP gates and vice versa (see [19] for more details on this). General dual unitaries can be built from the basic gates as described for IBM's architecture, however there is currently no efficient general parametrization of this.

5.4. Benchmark scenario

In summary, using a dual unitary circuit as a benchmark on a real quantum computer is possible with some hindrances. The feasibility of this relies heavily on the architecture of the quantum computer in question, especially on the qubit topology and also on the used number of qubits, since the trace evaluation scales with the number of qubits of the dual unitary circuit. If all previously described problems - and especially the problem of trace evaluation on a quantum computer, since it seems to be the harder problem - can be solved, it would be a pretty straightforward way to benchmark the results computed by a quantum computer against an analytical solution. All that has to be done is comparing those results against the results computed with equation 2.18. The main benefit of using this equation is that it is very easy to compute, and no numerically expensive simulation of quantum circuits is necessary.

6. Conclusion

In conclusion, dual unitary circuits hold a lot of potential, especially for using them as a benchmark on quantum computers. In principle, they are a subset of non-trivial circuits for which an analytical solution also exists and this analytical solution is easy to compute. Also, there is no restriction that limits which patterns of dual unitaries can be used for this, as long as the very basic grid structure of a non-Hermitian conjugated part and the Hermitian conjugate part remains. The whole range of circuits made of a single dual unitary gate up to circuits where all gates on the non-conjugated part are chosen at random (but are still dual unitary) is possible to solve this way.
The implementation in qiskit and the comparison of these results obtained via a quantum circuit simulator against the analytical solution shows that as long as the additional limits introduced by the necessity for periodic boundary conditions are observed, the results align very well with the analytical solution. However, for circuits of randomly generated dual unitaries some numerical errors are encountered. The most likely explanation for these discrepancies are escalating floating point inaccuracies. Since most of the original circuits vanish after some simplifications, one option to prevent such errors is to give the simplified circuit as input to the quantum computer or simulator, since this decreases the amount of floating point operations substantially.

Some other questions remain open to be resolved in the future. This includes a more detailed look into using higher dimensional circuits of dual unitaries, simulating these higher dimensional circuits in qiskit or comparable simulators, finding a way to explicitly generate ternary unitaries, perhaps by taking inspiration from the solution structure of 16×16 sudokus. Also, the problem of having to compute the trace of the circuit on a quantum computer is not solved satisfactorily. Even more efficient approaches, as briefly mentioned in the previous chapter, are still computationally expensive and require a lot of sampling and averaging, which leads to concerns about accumulating measurement errors overshadowing the results.
Nevertheless, the aforementioned potential of dual unitary circuits as benchmarks for quantum computers could be demonstrated in this thesis.

A. Appendix

A.1. List of randomly generated dual unitaries used in the implementation

A.1.1. Single gate

The following randomly generated dual unitary gate U_{rdm} was used for the computations in chapter 4 in the section about circuits of a single gate:

$$U_{rdm} = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{pmatrix} \tag{A.1}$$

$$x_{1,1} = -0.023454084870896008 + 0.7788720714292787i \tag{A.2}$$

$$x_{1,2} = -0.09384302015176321 + 0.22012750666447922i \tag{A.3}$$

$$x_{1,3} = -0.15724920203301676 + 0.5245138066957771i \tag{A.4}$$

$$x_{1,4} = -0.0914986456089649 + 0.16532248342563388i \tag{A.5}$$

$$x_{2,1} = -0.3587712324755185 + 0.30007778805380975i \tag{A.6}$$

$$x_{2,2} = -0.33792807155645793 + 0.050859737982743014i \tag{A.7}$$

$$x_{2,3} = 0.5840786163253436 - 0.31295040647966865i \tag{A.8}$$

$$x_{2,4} = 0.46742355369397126 + 0.08296905362225407i \tag{A.9}$$

$$x_{3,l} = -0.1159670856244002 + 0.18634757635594076i \tag{A.10}$$

$$x_{3,2} = 0.5566667476743986 - 0.4908878222313485i \tag{A.11}$$

$$x_{3,3} = -0.17130675864191208 + 0.12428341634132042i \tag{A.12}$$

$$x_{3,4} = 0.5736897999321653 - 0.16451484485159656i \tag{A.13}$$

$$x_{4,1} = -0.12337247048635595 - 0.33264121133618935i \tag{A.14}$$

$$x_{4,2} = -0.12451916918585376 + 0.5095110679943781i \tag{A.15}$$

$$x_{4,3} = -0.022196095144636524 + 0.4645289316152753i \tag{A.16}$$

$$x_{4,4} = 0.3964459833419752 - 0.4749455689740611i \tag{A.17}$$

A.1.2. Multi-gate

In the scenario where all gates are randomly generated dual unitaries, the following six dual unitaries T, U, V, W, Y, Z were used. The matrix entries of the respective matrix are listed directly below the matrix definition.

$$T = \begin{pmatrix} t_{1,1} & t_{1,2} & t_{1,3} & t_{1,4} \\ t_{2,1} & t_{2,2} & t_{2,3} & t_{2,4} \\ t_{3,1} & t_{3,2} & t_{3,3} & t_{3,4} \\ t_{4,1} & t_{4,2} & t_{4,3} & t_{4,4} \end{pmatrix} \tag{A.18}$$

$$t_{1,1} = 0.3561409644834417 + 0.6089775134498268i \tag{A.19}$$

$$t_{1,2} = -0.0007630951213893622 + 0.21490339073865591i \tag{A.20}$$

$$t_{1,3} = 0.4132039123904312 + 0.4415045097530581i \tag{A.21}$$

$$t_{1,4} = -0.024916031191228163 + 0.2997356133441025i \tag{A.22}$$

$$t_{2,1} = -0.5400112771078476 + 0.3408936887429157i \tag{A.23}$$

$$t_{2,2} = -0.21896425715465248 + 0.01897154512542756 4i \tag{A.24}$$

$$t_{2,3} = 0.4587920456979941 - 0.4813652389779524i \tag{A.25}$$

$$t_{2,4} = 0.31661566989523 + 0.0377615044802032i \tag{A.26}$$

$$t_{3,l} = -0.07899116319996342 + 0.18984184140210125i \tag{A.27}$$

$$t_{3,2} = 0.4846236660745784 - 0.4403504087462805i \tag{A.28}$$

$$t_{3,3} = -0.23856100609559433 + 0.19353401236541679i \tag{A.29}$$

$$t_{3,4} = 0.6477023764292524 + 0.12274671149605428i \tag{A.30}$$

$$t_{4,1} = -0.21182005722279013 - 0.08568576222675939i \tag{A.31}$$

$$t_{4,2} = 0.4031899263198415 + 0.5605176072263478i \tag{A.32}$$

$$t_{4,3} = 0.17821065925027896 + 0.25691993600902724i \tag{A.33}$$

$$t_{4,4} = 0.17829905234736845 - 0.5843720254734419i \tag{A.34}$$

$$U = \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ u_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} \\ u_{3,1} & u_{3,2} & u_{3,3} & u_{3,4} \\ u_{4,1} & u_{4,2} & u_{4,3} & u_{4,4} \end{pmatrix} \tag{A.35}$$

$$u_{1,1} = 0.0586329362126316 + 0.41344412068998737i \tag{A.36}$$

$$u_{1,2} = 0.05543056532051108 + 0.6014782487208153i \tag{A.37}$$

$$u_{1,3} = -0.12724784300895148 + 0.40516458620543644i \tag{A.38}$$

$$u_{1,4} = -0.052286462558836205 + 0.5269661060918325i \tag{A.39}$$

$$u_{2,1} = -0.3947259973940406 + 0.04276082612148424i \tag{A.40}$$

$$u_{2,2} = -0.5458141804544264 + 0.07229997059134555i \tag{A.41}$$

$$u_{2,3} = 0.4452040884215723 + 0.1528006854483037i \tag{A.42}$$

$$u_{2,4} = 0.5627604303158787 + 0.03112011645181481i \tag{A.43}$$

$$u_{3,l} = -0.5158368803653615 + 0.3257505571414839i \tag{A.44}$$

$$u_{3,2} = 0.37563672763265693 - 0.21047896658560653i \tag{A.45}$$

$$u_{3,3} = -0.40927602472165847 + 0.32491199632746137i \tag{A.46}$$

$$u_{3,4} = 0.25808506935767783 - 0.32048707939730037i \tag{A.47}$$

$$u_{4,1} = -0.3423312323066661 - 0.42260768018695666i \tag{A.48}$$

$$u_{4,2} = 0.2503750487292193 + 0.28968767986172383i \tag{A.49}$$

$$u_{4,3} = 0.4094907756672496 + 0.3966579755756422i \tag{A.50}$$

$$u_{4,4} = -0.421030494450919 - 0.23519893057293786i \tag{A.51}$$

$$V = \begin{pmatrix} v_{1,1} & v_{1,2} & v_{1,3} & v_{1,4} \\ v_{2,1} & v_{2,2} & v_{2,3} & v_{2,4} \\ v_{3,1} & v_{3,2} & v_{3,3} & v_{3,4} \\ v_{4,1} & v_{4,2} & v_{4,3} & v_{4,4} \end{pmatrix} \tag{A.52}$$

$$v_{1,1} = -0.45449845871732775 + 0.5847833987875758i \tag{A.53}$$

$$v_{1,2} = 0.20680039369974448 + 0.4688174970352018i \tag{A.54}$$

$$v_{1,3} = -0.28429203976346523 + 0.030346862632222277i \tag{A.55}$$

$$v_{1,4} = -0.2557494855862689 + 0.20433449009692845i \tag{A.56}$$

$$v_{2,1} = -0.17055589463140947 + 0.14463701219958738i \tag{A.57}$$

$$v_{2,2} = -0.37126041716281055 - 0.03255462059148287i \tag{A.58}$$

$$v_{2,3} = 0.5759674372607979 - 0.07111466310237391i \tag{A.59}$$

$$v_{2,4} = 0.1603693389640409 + 0.6697631012992561i \tag{A.60}$$

$$v_{3,l} = -0.4497647073146577 + 0.24850636644174412i \tag{A.61}$$

$$v_{3,2} = -0.011756219103471799 - 0.738426745224926i \tag{A.62}$$

$$v_{3,3} = -0.29728805987525386 - 0.13150184382442404i \tag{A.63}$$

$$v_{3,4} = 0.2888279298208557 - 0.03807267554004662i \tag{A.64}$$

$$v_{4,1} = -0.2803791008130013 - 0.24247515904491163i \tag{A.65}$$

$$v_{4,2} = 0.017204112280422512 + 0.22987257934576463i \tag{A.66}$$

$$v_{4,3} = -0.09358655897416812 + 0.6833958958527133i \tag{A.67}$$

$$v_{4,4} = 0.5774050505035198 + 0.016461653108807373i \tag{A.68}$$

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} \end{pmatrix} \tag{A.69}$$

$$w_{1,1} = 0.5842049533480377 + 0.5157123793473193i \tag{A.70}$$

$$w_{1,2} = 0.5307741973699516 - 0.18351850125180388i \tag{A.71}$$

$$w_{1,3} = -0.014206313793684339 + 0.2307120399831316i \tag{A.72}$$

$$w_{1,4} = 0.13995692494295914 + 0.06578160678979074i \tag{A.73}$$

$$w_{2,1} = -0.22354839866080792 + 0.0611315737171647i \tag{A.74}$$

$$w_{2,2} = -0.0143075142589267 + 0.15306653639153306i \tag{A.75}$$

$$w_{2,3} = 0.6532376778482297 + 0.4873455985993129i \tag{A.76}$$

$$w_{2,4} = 0.3829336703292088 - 0.33435248244541454i \tag{A.77}$$

$$w_{3,l} = -0.5570311017295327 - 0.06923722731015652i \tag{A.78}$$

$$w_{3,2} = 0.4654061716006296 - 0.6262745193720829i \tag{A.79}$$

$$w_{3,3} = -0.08666788923109307 - 0.12933168429805608i \tag{A.80}$$

$$w_{3,4} = 0.2221046651624284 - 0.05031292088158369i \tag{A.81}$$

$$w_{4,1} = 0.09309922195150165 - 0.12365052104623028i \tag{A.82}$$

$$w_{4,2} = 0.12432191264675849 + 0.1915385742730914i \tag{A.83}$$

$$w_{4,3} = -0.5067382851104615 + 0.03637639119964395i \tag{A.84}$$

$$w_{4,4} = 0.11497442479685027 - 0.8078205681508337i \tag{A.85}$$

$$Y = \begin{pmatrix} y_{1,1} & y_{1,2} & y_{1,3} & y_{1,4} \\ y_{2,1} & y_{2,2} & y_{2,3} & y_{2,4} \\ y_{3,1} & y_{3,2} & y_{3,3} & y_{3,4} \\ y_{4,1} & y_{4,2} & y_{4,3} & y_{4,4} \end{pmatrix} \tag{A.86}$$

$$y_{1,1} = 0.011650777573248838 + 0.5689324165282631i \tag{A.87}$$

$$y_{1,2} = -0.1651106608505127 + 0.394358096815135i \tag{A.88}$$

$$y_{1,3} = -0.14328481645170854 + 0.5778881431874396i \tag{A.89}$$

$$y_{1,4} = -0.1451477566866447 + 0.3432888142973919i \tag{A.90}$$

$$y_{2,1} = -0.5533363505713 - 0.21434382555766587i \tag{A.91}$$

$$y_{2,2} = -0.31923428118677705 - 0.19840717540452774i \tag{A.92}$$

$$y_{2,3} = 0.48755740801495073 + 0.36360746930297627i \tag{A.93}$$

$$y_{2,4} = 0.3245574013566732 + 0.17702979813107633i \tag{A.94}$$

$$y_{3,l} = -0.4116505809113227 + 0.09953500387454192i \tag{A.95}$$

$$y_{3,2} = 0.6094709103313195 + 0.12020685255412131i \tag{A.96}$$

$$y_{3,3} = -0.3715671837714263 + 0.06534011268937051i \tag{A.97}$$

$$y_{3,4} = 0.5397444370522958 - 0.03281073889783334i \tag{A.98}$$

$$y_{4,1} = -0.009452848324740748 - 0.3802670091982799i \tag{A.99}$$

$$y_{4,2} = -0.06493004058640667 + 0.5346249521874213i \tag{A.100}$$

$$y_{4,3} = 0.00474377450909555 + 0.3650179882086899i \tag{A.101}$$

$$y_{4,4} = -0.06352327625034399 - 0.6541957306218833i \tag{A.102}$$

$$Z = \begin{pmatrix} z_{1,1} & z_{1,2} & z_{1,3} & z_{1,4} \\ z_{2,1} & z_{2,2} & z_{2,3} & z_{2,4} \\ z_{3,1} & z_{3,2} & z_{3,3} & z_{3,4} \\ z_{4,1} & z_{4,2} & z_{4,3} & z_{4,4} \end{pmatrix} \tag{A.103}$$

$$z_{1,1} = 0.5712204418920757 + 0.20611901975380598i \tag{A.104}$$

$$z_{1,2} = 0.14935106212147137 + 0.21100784361191555i \tag{A.105}$$

$$z_{1,3} = 0.49128471171193416 + 0.4685128232242606i \tag{A.106}$$

$$z_{1,4} = -0.013801443883276782 + 0.32146025150410346i \tag{A.107}$$

$$z_{2,1} = -0.4838905246393483 + 0.4898484937307974i \tag{A.108}$$

$$z_{2,2} = -0.2973726692341131 + 0.04312789798140472i \tag{A.109}$$

$$z_{2,3} = 0.5534674397821115 - 0.2196979885126474i \tag{A.110}$$

$$z_{2,4} = 0.24749132780507124 + 0.14057924088578821i \tag{A.111}$$

$$z_{3,l} = -0.0933107693092414 + 0.24143246987157174i \tag{A.112}$$

$$z_{3,2} = 0.5004613878203114 - 0.339543054900186i \tag{A.113}$$

$$z_{3,3} = -0.25636729997466323 + 0.19400127138562323i \tag{A.114}$$

$$z_{3,4} = 0.6667513483201828 + 0.1390479403311287i \tag{A.115}$$

$$z_{4,1} = -0.20439325873504594 - 0.21988048667532267i \tag{A.116}$$

$$z_{4,2} = 0.06860053563763147 + 0.687329840663105i \tag{A.117}$$

$$z_{4,3} = 0.0538407672321104 + 0.2797894555650855i \tag{A.118}$$

$$z_{4,4} = 0.38319877429182103 - 0.452465861394039i \tag{A.119}$$

List of Figures

2.1. Tensor network visualization of gate U, adapted from [2] 2
2.2. Visualization of unitarity in t and x direction, adapted from [2] 3
2.3. Grid representation of $tr\left(\mathbb{U}^t\right)$, adapted from [2] 8
2.4. Tensor network representation of equation 2.16 (without prefactor $\frac{1}{d^{2L}}$), adapted from [2] 8
2.5. Grid representation of M_+ (missing a prefactor of $\frac{1}{d}$), adapted from [2] 9
2.6. Grid representation of M_- (missing a prefactor of $\frac{1}{d}$), adapted from [2] 9
2.7. Grid representation of the simplified correlation function (here $C_+^{\alpha\beta}$), adapted from [2] 10

3.1. Initial grid of multiple dual unitaries ($L = 4$, $t = 1$) 12
3.2. Initial grid of multiple dual unitaries 13
3.3. Grid of multiple dual unitaries after first simplifications 14
3.4. Grid of multiple dual unitaries after simplifcations 14
3.5. Initial setup for grid simplification example 15
3.6. Step 1 of grid simplification example 15
3.7. Step 2 of grid simplification example 16
3.8. Step 3 of grid simplification example 16
3.9. Example of a one-dimensional dual unitary with legs a, b, c, d 18
3.10. Example of a two-dimensional gate with legs a, b, c, d, e, f, g, h 18
3.11. Quantum circuit to build the self-ternary four-qubit gate 23
3.12. 2D grid of alternating level dual unitaries, darkblue y=0, light blue y=1, green connects y-levels 24
3.13. 2D grid of alternating level dual unitaries, simplification step 1 25
3.14. 2D grid of alternating level dual unitaries, simplification step 2 26

4.1. Quantum circuit of the correlation function for $\alpha = \beta = 2$, $x = y = 0$, $t = 2$, S is used as shorthand notation for $SWAP$ 30

5.1. Tensor network representation of singular value decomposition 37
5.2. Example of qubit topology and interconnectivity for a 10-qubit ring . . 38

5.3. Some qubit topologies of IBM quantum computers, from [10] 38
5.4. Interconnectivity of Google's Sycamore architecture, from [1] (p.506) . . 39
5.5. Path of 46 qubits forming a circle on Google's sycamore architecture . . 40
5.6. Coupler activation patterns of Google's sycamore, sequences used by Google were either ABCDCDAB or EFGH, from supplemental material of [1] (p.28) 40

List of Tables

2.1. List of notable two-qubit gates and their dual unitarity 5
2.2. Classification of permutations of the 4×4 identity matrix regarding their dual unitarity 6

4.1. Results for $U = SWAP$, $x \in -1.5, ..., 2$, $y \in -1.5, ..., 2$, $t_{max} = 4$ 31
4.2. Results for $U = SWAP$, $x \in -1, .., 1$, $y \in -1, .., 1$, $t_{max} = 2$ 31
4.3. Results for $U = SWAP$, $x \in -1.5, .., 2$, $y \in -1.5, .., 2$, $t_{max} = 2$ 31
4.4. Results for $U = rdm$, $x \in -1.5, ..., 2$, $y \in -1.5, ..., 2$, $t_{max} = 4$ 32
4.5. Results for $U = rdm$, $x \in -1, .., 1$, $y \in -1, .., 1$, $t_{max} = 2$ 32
4.6. Results for $U = rdm$, $x \in -1.5, .., 2$, $y \in -1.5, .., 2$, $t_{max} = 2$ 32
4.7. Results for array of safe gates, $x \in -1.5, ..., 2$, $y \in -1.5, ..., 2$, $t_{max} = 4$ . . 33
4.8. Results for array of safe gates, $x \in -1, .., 1$, $y \in -1, .., 1$, $t_{max} = 2$ 33
4.9. Results for array of safe gates, $x \in -1.5, .., 2$, $y \in -1.5, .., 2$, $t_{max} = 2$ . . . 33
4.10. Results for mixed array of iSWAP and U_{rdm}, $x \in -1.5, ..., 2$, $y \in -1.5, ..., 2$, $t_{max} = 4$ 34
4.11. Results for mixed array of iSWAP and U_{rdm}, $x \in -1, .., 1$, $y \in -1, .., 1$, $t_{max} = 2$ 34
4.12. Results for mixed array of iSWAP and U_{rdm}, $x \in -1.5, .., 2$, $y \in -1.5, .., 2$, $t_{max} = 2$ 34
4.13. Results for array of random gates, $x \in -1.5, ..., 2$, $y \in -1.5, ..., 2$, $t_{max} = 4$ 35
4.14. Results for array of random gates, $x \in -1, .., 1$, $y \in -1, .., 1$, $t_{max} = 2$ . . 35
4.15. Results for array of random gates, $x \in -1.5, .., 2$, $y \in -1.5, .., 2$, $t_{max} = 2$ 35

Bibliography

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, and et al. "Quantum supremacy using a programmable superconducting processor." In: *Nature* 574.7779 (Oct. 2019), pp. 505–510. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1666-5.

[2] B. Bertini, P. Kos, and T. Prosen. "Exact Correlation Functions for Dual-Unitary Lattice Models in 1+1 Dimensions." In: *Physical Review Letters* 123.21 (Nov. 2019). ISSN: 1079-7114. DOI: 10.1103/physrevlett.123.210601.

[3] G. Dahl. "Permutation matrices related to Sudoku." In: *Linear Algebra and its Applications* 430.8 (2009), pp. 2457–2463. ISSN: 0024-3795. DOI: https://doi.org/10.1016/j.laa.2008.12.023.

[4] A. Daskin, A. Grama, and S. Kais. "A universal quantum circuit scheme for finding complex eigenvalues." In: *Quantum Information Processing* 13.2 (Oct. 2013), pp. 333–353. ISSN: 1573-1332. DOI: 10.1007/s11128-013-0654-1.

[5] A. Datta, S. T. Flammia, and C. M. Caves. "Entanglement and the power of one qubit." In: *Physical Review A* 72.4 (Oct. 2005). ISSN: 1094-1622. DOI: 10.1103/physreva.72.042316.

[6] B. Foxen, C. Neill, A. Dunsworth, P. Roushan, B. Chiaro, A. Megrant, J. Kelly, Z. Chen, K. Satzinger, R. Barends, and et al. "Demonstrating a Continuous Set of Two-qubit Gates for Near-term Quantum Algorithms." In: *Physical Review Letters* 125.12 (Sept. 2020). ISSN: 1079-7114. DOI: 10.1103/physrevlett.125.120504.

[7] G. Golub and W. Kahan. "Calculating the Singular Values and Pseudo-Inverse of a Matrix." In: *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis* 2.2 (Jan. 1965), pp. 205–224. DOI: 10.1137/0702016.

[8] D. Goyeneche, D. Alsina, J. I. Latorre, A. Riera, and K. Życzkowski. "Absolutely maximally entangled states, combinatorial designs, and multiunitary matrices." In: *Physical Review A* 92.3 (Sept. 2015). ISSN: 1094-1622. DOI: 10.1103/physreva.92.032316.

[9] Q. Huang and C. B. Mendl. *Efficient quantum circuit simulation using a multi-qubit Bloch vector representation of density matrices*. 2021. arXiv: 2103.13962 [quant-ph].

[10] IBM. *Quantum computation center opens*. https://www.ibm.com/blogs/research/2019/09/quantum-computation-center/. Jan. 2020.

[11] *IBM Quantum System Availability*. https://quantum-computing.ibm.com/services?services=systems. Accessed: 2021-06-29.

[12] W. R. Inc. *Mathematica, Version 12.3.1*. Champaign, IL, 2021.

[13] A. J., A. Adedoyin, J. Ambrosiano, P. Anisimov, A. Bärtschi, W. Casper, G. Chennupati, C. Coffrin, H. Djidjev, D. Gunter, S. Karra, N. Lemons, S. Lin, A. Malyzhenkov, D. Mascarenas, S. Mniszewski, B. Nadiga, D. O'Malley, D. Oyen, S. Pakin, L. Prasad, R. Roberts, P. Romero, N. Santhi, N. Sinitsyn, P. J. Swart, J. G. Wendelberger, B. Yoon, R. Zamora, W. Zhu, S. Eidenbenz, P. J. Coles, M. Vuffray, and A. Y. Lokhov. *Quantum Algorithm Implementations for Beginners*. 2020. arXiv: 1804.03719 [cs.ET].

[14] N. Khaneja and S. J. Glaser. "Cartan decomposition of SU(2n) and control of spin systems." In: *Chemical Physics* 267.1 (2001), pp. 11–23. ISSN: 0301-0104. DOI: https://doi.org/10.1016/S0301-0104(01)00318-4.

[15] A. Kitaev. "Quantum measurements and the Abelian Stabilizer Problem." In: (Nov. 1995).

[16] D. C. McKay, T. Alexander, L. Bello, M. J. Biercuk, L. Bishop, J. Chen, J. M. Chow, A. D. Córcoles, D. Egger, S. Filipp, J. Gomez, M. Hush, A. Javadi-Abhari, D. Moreda, P. Nation, B. Paulovicks, E. Winston, C. J. Wood, J. Wootton, and J. M. Gambetta. *Qiskit Backend Specifications for OpenQASM and OpenPulse Experiments*. 2018. arXiv: 1809.03452 [quant-ph].

[17] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[18] L. Scheller. *Dual unitaries*. https://gitlab.lrz.de/ga96sik/dual-unitaries. Final commit of implementation: b09b821862522b6010e3aa940c19dc1ab44a1ca9. 2021.

[19] N. Schuch and J. Siewert. "Natural two-qubit gate for quantum computation using theXYinteraction." In: *Physical Review A* 67.3 (Mar. 2003). ISSN: 1094-1622. DOI: 10.1103/physreva.67.032301.

[20] Y. Sun, J.-Y. Zhang, M. S. Byrd, and L.-A. Wu. *Adiabatic Quantum Simulation Using Trotterization*. 2018. arXiv: 1805.11568 [quant-ph].

[21] R. Suzuki, K. Mitarai, and K. Fujii. *Computational power of one- and two-dimensional dual-unitary quantum circuits*. 2021. arXiv: 2103.09211 [quant-ph].

[22] I. Q. Team. *Backend information*. https://github.com/Qiskit/ibmq-device-information. Accessed: 2021-07-05. 2021.

[23] F. Vatan and C. Williams. "Optimal quantum circuits for general two-qubit gates." In: *Physical Review A* 69.3 (Mar. 2004). ISSN: 1094-1622. DOI: 10.1103/physreva.69.032315.

[24] L. Zhao, Z. Zhao, P. Rebentrost, and J. Fitzsimons. *Compiling basic linear algebra subroutines for quantum computers*. 2019. arXiv: 1902.10394 [quant-ph].