

An explicit ancilla-free quantum circuit construction for Gaussian convolutions

Oliver Hüttenhofer, Jakob S. Kottmann

Angaben zur Veröffentlichung / Publication details:

Hüttenhofer, Oliver, and Jakob S. Kottmann. 2025. "An explicit ancilla-free quantum circuit construction for Gaussian convolutions." In *INFORMATIK 2025: The Wide Open - Offenheit von Source bis Science*, 16.-19. September, 2025, Potsdam, Griebnitzsee, edited by Ulrike Lucke, Stefan Stieglitz, Falk Uebernickel, Anna-Lena Lamprecht, and Maike Klein, 1729–38. Bonn: Gesellschaft für Informatik e.V. https://doi.org/10.18420/inf2025_155.

Nutzungsbedingungen / Terms of use:

CC BY-SA 4.0



An explicit ancilla-free quantum circuit construction for Gaussian convolutions

Oliver Hüttenhofer ¹ and Jakob S. Kottmann ¹

Abstract: We show an explicit ancilla-free quantum circuit construction implementing Gaussian convolutions. This is based on a block-encoding scheme developed by Sünderhauf et al. (2024), but decomposes the high-level overview into an explicit quantum circuit consisting of single-qubit rotations, CNOTs and Toffoli gates. We construct the circuit in such a way that it requires no additional ancillary qubits and enables efficient implementation of linear combinations of Gaussian convolutions. Because Gaussians are separable, we can extend the circuit to high-dimensional Gaussian convolutions, and convolution kernels which can be approximated as linear combinations of Gaussians.

Keywords: Gaussian convolution, ancilla-free quantum circuit, multi-dimensional convolution

1 Introduction

This paper considers the task of constructing a quantum circuit that implements a Gaussian convolution, which on a basis state $|j\rangle$ acts like

$$U|j\rangle = z_j \cdot \sum_{i=0}^{2^n-1} e^{-(i-j-b)^2/c} |i\rangle$$

for some normalization factor z_j . Despite having direct applications, such as in (image) smoothing operations, Gaussian convolutions are a useful primitive for various types of operators of interest in numerical analysis [BM05]. In addition they are separable, so multi-dimensional convolutions can be implemented by running multiple one-dimensional convolutions on different qubit registers in parallel and therefore keeping the circuit depth almost constant with respect to the dimensionality.

It is known how to implement convolutions in quantum circuits, and we base this construction on a block-encoding developed by Sünderhauf et al. [SCC24, equation (56)], however this gives only a high-level circuit with operations like addition that can't be directly run on a quantum computer without further decomposition. Here we show such a decomposition into single-qubit rotations, CNOTs and Toffoli gates in a way that avoids using dedicated ancilla qubits and makes it easy to implement linear combinations. Alongside this manuscript we provide an operational implementation in TEQUILA, though it is in some places less optimized than the decompositions discussed here [Hü24; Ko21]. In all circuit diagrams,

¹ University of Augsburg, Institute of Computer Science, Universitätsstr. 2, 86135 Augsburg, Germany,
oliver.huettenhofer@uni-a.de,  <https://orcid.org/0009-0006-4996-9404>;
jakob.kottmann@uni-a.de,  <https://orcid.org/0000-0002-4156-2048>

if multiple qubits represent a binary number, the top qubit is the least significant, and the bottom qubit the most significant.

2 One-dimensional Gaussian convolution

The high-level block-encoding [SCC24, equation (56)] is shown in Fig. 1, with the squared control indicating a multiplexed operation, i.e. a different operation for each control value, which in this case simply means that for the control value s , the constant s is added. The $|0\rangle\langle 0|$ projectors indicate a postselection onto the $|0\rangle$ state, i.e. the circuit is repeated until the qubits are measured in the state $|0\rangle$. Intuitively, the prepare and unprepare operations load the convolution kernel, the multiplexed addition applies the convolution to the state, and the constant addition is used to add an offset.

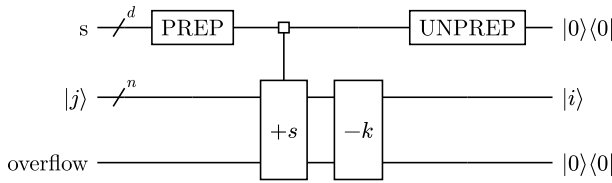


Fig. 1: Block encoding circuit for convolutions/Toeplitz matrices [SCC24, equation (56)]

The prepare and unprepare operations can be decomposed as described in [SBM06, chapter IV]. For a real-valued convolution kernel, no Z-rotations are necessary, so the result is a triangle of multiplexed Y-rotations as shown in Fig. 2 which can then be decomposed into single-qubit rotations and CNOTs, as shown in Fig. 3, with round gates indicating rotations [SBM06, chapter III]. The unprepare operation is then simply the inverse of the prepare operation (but this would not be the case if any of the values weren't real). The number of gates required here is exponential in the number of qubits, so if the error is tolerable, it is useful to limit the range of the convolution kernel and set it to 0 outside that range, leading to smaller (un)prepare operations. This exponential number could be improved with the Kitaev-Webb algorithm [KW09] to achieve polynomial scaling. However, the arithmetic required in this algorithm introduces such large constant factors that it only makes sense for approximately $n > 16$ [Ba21], so it isn't used here, but shows that this exponential scaling is no fundamental issue.

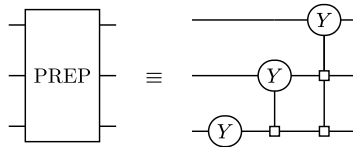


Fig. 2: Decomposition of a real-valued prepare operation into a series of multiplexed Y-rotations

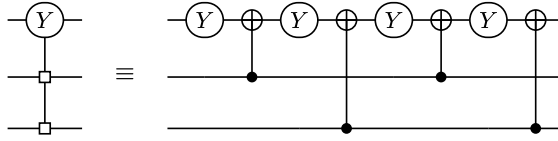


Fig. 3: Decomposition of a multiplexed Y-rotation into single-qubit Y-rotations and CNOTs

The addition of the s register, which contains the convolution kernel, to the state register can be implemented using a circuit described in [TTK10, chapter 2], shown in Fig. 4.

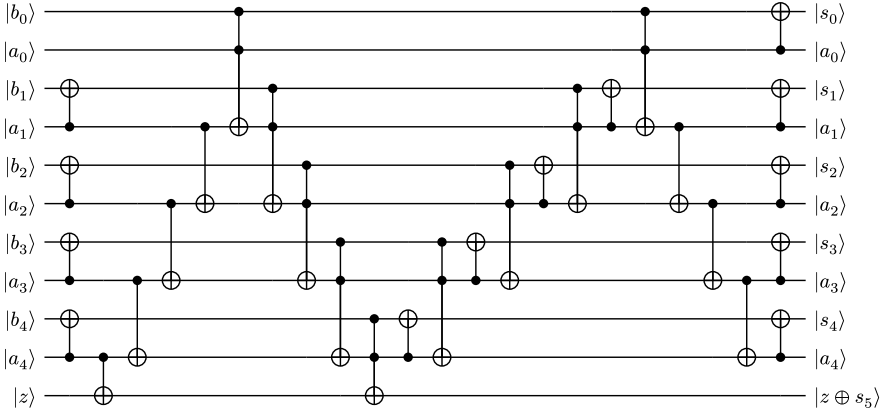


Fig. 4: The addition circuit [TTK10, Figure 2]

However, this requires both registers to be of the same size, but we want to allow a convolution kernel that is smaller than the state to which it is applied. To make this possible, we replace the CNOT and Toffoli gate acting on the overflow qubit $|z\rangle$ with two controlled increment gates. We also add CNOTs that invert the second increment, which makes the two gates cancel out if both are run, the same way that the CNOT and Toffoli would, while having no effect if only one of the increments is active. This construction for differently sized registers is shown in Fig. 5.

To implement the increment gate, we follow [Gi15b], but instead of the subtraction widget shown there, we use the inverse of the addition circuit mentioned earlier. The first construction is an increment gate using $n - 1$ borrowed ancilla qubits $|g_i\rangle$, i.e. qubits which can be in any state and have to be returned to this state. They are subtracted from the value v twice, but one of the subtractions is surrounded by NOTs as shown in Fig. 6.

An additional NOT on the high bit of v is necessary because there is one less ancilla qubit than value qubits, so the sign of the ancilla bits will act like a normal binary digit when added, which is counteracted by flipping the high bit. Since inverting all bits of a two's

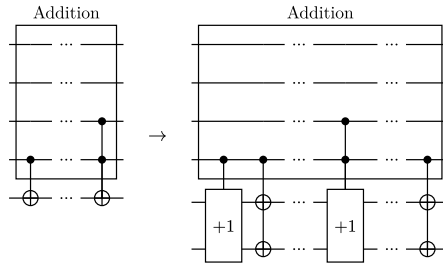
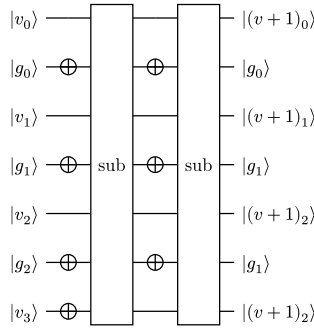


Fig. 5: Extending the addition circuit from Fig. 4 to differently sized registers

Fig. 6: Constructing n -qubit increment gates requiring $n - 1$ borrowed ancilla qubits

complement number g leads to the number $\bar{g} = -g - 1$, the result of this operation is an increment:

$$v - g - \bar{g} = v - g - (-g - 1) = v + 1$$

Then the number of required ancillae can be reduced to one by splitting the increment into smaller increments of half the size as shown in Fig. 7, such that the other half can be used for the borrowed qubits.

Multi-controlled NOTs with n controls can be implemented using $n - 2$ borrowed ancillae [Gi15a], for which there are already enough unused qubits in Fig. 7. Controlled increments can then be implemented by noting that an increment is equivalent to a triangle of controlled NOTs [Gi15b], so a controlled increment is equivalent to a larger increment followed by a smaller decrement as shown in Fig. 8.

Now this construction can be used for the addition in Fig. 4 and requires only one borrowed ancilla. We assume the registers which the addition acts on have at least two qubits each, which means the least significant qubit is not affected by the increments and can be used as the ancilla. This is not a significant restriction, as a convolution kernel on only one qubit

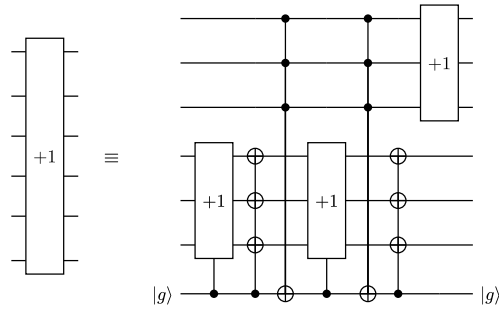


Fig. 7: Reducing the number of required ancilla qubits for an increment gate from $n - 1$ to 1 [Gi15b]

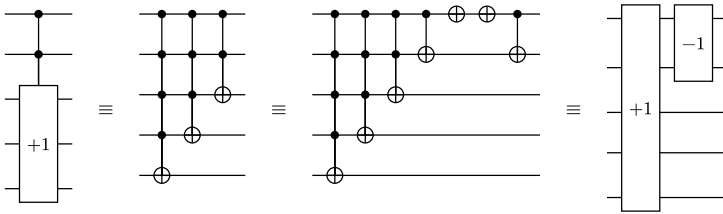


Fig. 8: Creating a controlled increment from a larger increment and a decrement

could only have two values which likely isn't very useful, but even then a single borrowed ancilla could be found elsewhere, e.g. on the register used for the linear combination (u in Fig. 11).

The construction might seem circular because the addition circuit contains an increment gate, which then contains an addition circuit, but this is not an issue because the increment is only required when the register sizes of the addition differ by more than one qubit, and this is not the case for the subtraction block in Fig. 6.

The last operation required for the block-encoding is the addition of a constant. There we follow [HRS17, chapter 2], which introduces a carry gate to determine if a carry occurs when adding a constant. Fig. 9 shows how this carry gate might look, although the exact form depends on the constant. Fig. 10 illustrates how it is used to split a constant addition into two smaller operations, which works because the only interaction between those qubits happens when such a carry occurs. This is then applied recursively until each of the operations acts only on a single qubit and is implemented by either a NOT gate or the identity.

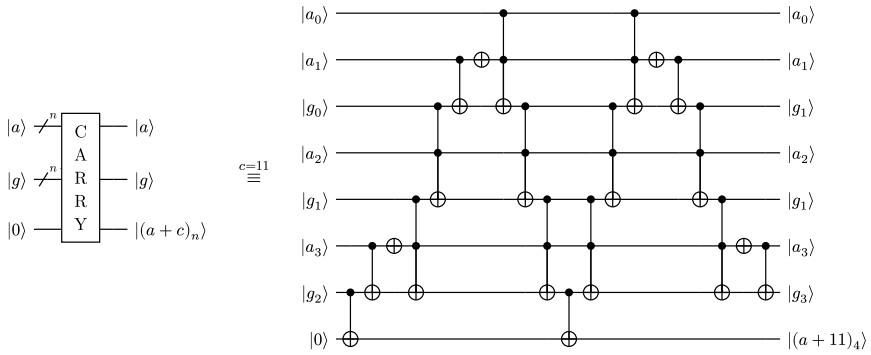


Fig. 9: The carry circuit to determine if a carry occurs during constant addition [HRS17, Fig. 3]

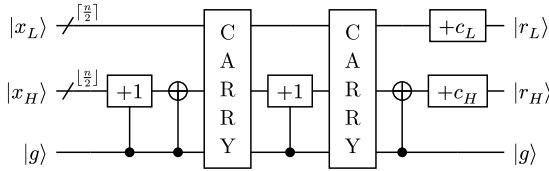


Fig. 10: Splitting a larger constant addition into two smaller constant additions using the carry gate from Fig. 9 [HRS17, Fig. 5]

3 Linear combination of multi-dimensional Gaussian convolution

As mentioned, extending this circuit to multiple dimensions works by simply running the one-dimensional circuit on multiple registers. To build linear combinations, we start with the general circuit structure of linear combinations of unitaries (LCU) which gives us the circuit shown in Fig. 11 for a two-dimensional convolution that combines four Gaussians.

Each of the “gauss conv”-blocks contains the circuit from Fig. 1 with added controls. We then use the fact that circuits with different controls commute, so we can rearrange the blocks and first execute all prepare operations, next the additions, and then all unprepare operations. Then the additions are the same for all Gaussians, so multiple controlled operations can be replaced by a single uncontrolled one. The same applies for constant additions if they add the same offset, which for example is the case when approximating isotropic functions. The prepare and unprepare operations differ, but since there is one for each configuration of the controls and they are decomposed into multiplexed Y-rotations which are equivalent to Y-rotations multiplexed on more qubits. Last, we can reduce the number of qubits at the cost of a higher circuit depth by reusing the s register and overflow qubit and handling the dimensions sequentially. This series of transformations leads to the circuit shown in Fig. 12.

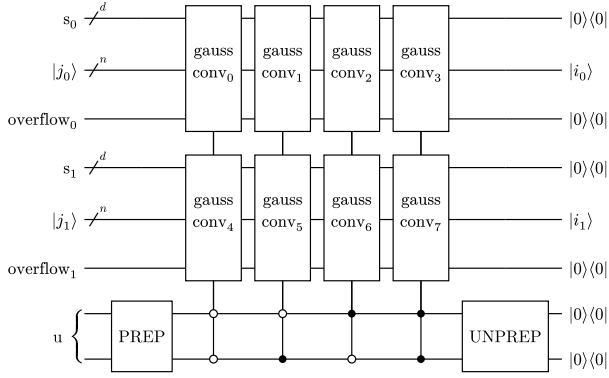


Fig. 11: Extending the circuit to linear combinations of multi-dimensional convolutions

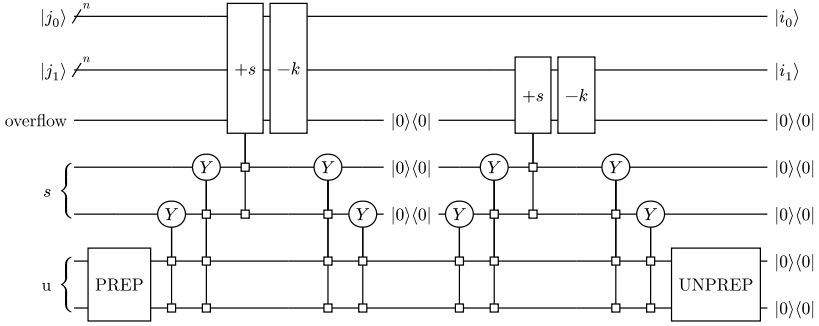


Fig. 12: An optimized version of Fig. 11. The larger addition operation does not act on $|j_1\rangle$.

3.1 A Hands-On Example

As a concrete example, consider the convolution

$$U|j_1, j_2\rangle = z_{j_1, j_2} \cdot \sum_{i_1, i_2=0}^{15} \left(2 \cdot e^{-|(i_1, i_2)^T - (j_1, j_2)^T|^2/4} - e^{-|(i_1, i_2)^T - (j_1, j_2)^T - (5, 2)^T|^2/2} \right) |i_1, i_2\rangle$$

on a 16×16 grid. The result is a linear combination of two two-dimensional Gaussian convolutions and the output for an example input wavefunction is illustrated in Fig. 13, which was computed with TEQUILA [Ko21] (v.1.9.9dev) similar as illustrated in [Hü24]. The convolutions are approximated with kernels of size 7×7 , so any interaction extending beyond this is assumed to be 0 and ignored. This reduces the required number of qubits and the size of the circuits for the kernel state preparations.

This operation can be implemented by a circuit with the structure shown in Fig. 12, except that all offsets are powers of two, so increment gates can be used instead of the more general

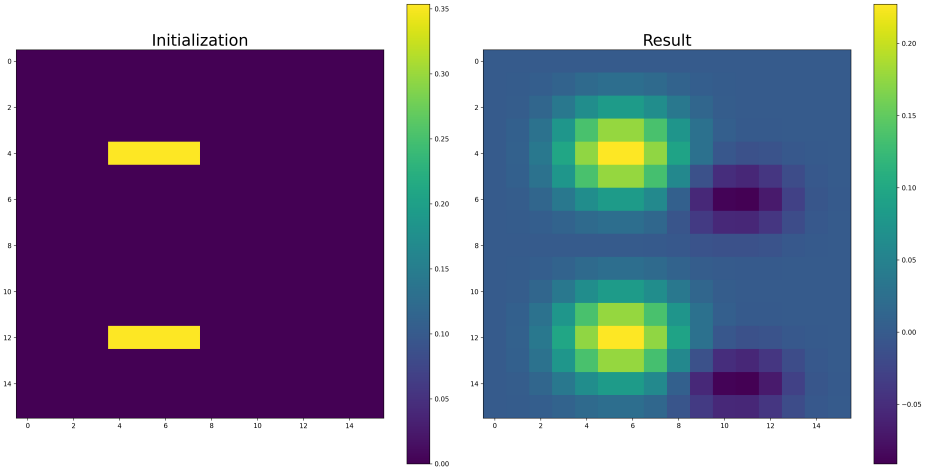


Fig. 13: Output of a linear combination of two two-dimensional Gaussian convolutions when applying it to the wavefunction shown on the left.

constant addition circuit. This is convenient because it is simple to add controls to increment gates (though it is also possible for constant addition by adding controls to the CNOTs at the bottom in Fig. 9). One thing to note is that the block-encoding scales the Gaussians by the inverse of their 1-norm

$$\left| \sum_{i_1, i_2 = -2^{k-1}}^{2^k-1} e^{-|(i_1, i_2)^T|^2/c} \right|^{-1}$$

where k is the number of qubits used by the convolution kernel, which is 3 in this example, so the scaling is $\approx 12.28^{-1}$ for the first Gaussian and $\approx 6.28^{-1}$ for the second Gaussian. The difference comes from the parameter c and must be taken into account when calculating the factors for the LCU.

The circuit then consists of the following components:

- A prepare operation on the u register that consists of one qubit (since there are only two Gaussians), which is implemented using one Y- and one Z-rotation.
- A prepare operation on the s register, multiplexed on the u register, which prepares the two Gaussian convolution kernels. This gets compiled into a triangle of multiplexed Y-rotations, and is then decomposed into single qubit Y-rotations and CNOTs.
- An addition circuit from the convolution kernel to the horizontal dimension of the state register to apply the convolution. It does not need to be controlled because the operation is the same for both Gaussians.

- The constant addition for the first Gaussian in the horizontal direction, which needs to subtract 4 to counteract the offset from the state preparation. This is implemented as a decrement, and can be controlled by adding the u register as a low bit to the decrement, meaning that it only has an effect on the other qubits if u is in the state $|0\rangle$. Then a NOT gate is added on the u qubit to counteract the bitflip caused by the decrement.
- The offset for the second Gaussian in the horizontal direction, which is implemented by adding 1 using an increment. This can again be controlled by extending the increment to the u register, so the increment will only have an effect if it is in the state $|1\rangle$, and then again a NOT gate.
- An unprepare operation multiplexed on u as the counterpart to the multiplexed prepare operation mentioned above.
- A mid-circuit postselection of the s register and overflow qubits in the state $|0\rangle$.
- Another series of prepare, addition, constant addition and unprepare operations similar to above, but now acting on the vertical dimension of the state register to make the full convolution two-dimensional.
- An unprepare operation on the u register which is the counterpart to the prepare operation of the first step.
- A postselection of the s , overflow and u registers in the state $|0\rangle$.

The final circuit uses 13 qubits ($2 \cdot 4$ for the state, 3 for the convolution kernel, 1 for the overflow qubit and 1 for u register), contains 112 NOT gates, 250 CNOTs, 113 Toffoli gates, 58 Y-rotations, 1 Z-rotation and has a depth of 384. The parts of the circuit implementing the convolutions in different dimensions can be parallelized by using separate qubits for the LCU controls and convolution kernel registers per dimension, which reduces the depth to 208, at the cost of increasing the qubit count to 18. This parallelization also moves the measurements and postselection to the end and avoids the need for mid-circuit measurements.

4 Conclusion

We have described how to implement multi-dimensional Gaussian convolutions in a quantum circuit using only elementary gates, combined with a concrete example. Borrowed ancillae are used in multiple places to avoid the need for dedicated ancilla qubits and to reduce the total number of qubits required. With the addition circuit on n qubits requiring $O(n)$ and the constant addition $O(n \log n)$ gates, the main bottleneck is the state preparation circuit, which has a gate count exponential in the number of qubits. This could be solved using a more sophisticated method like the Kitaev-Webb algorithm to use the structure of a Gaussian to achieve polynomial scaling, but due to large constant factors, this is only advantageous at higher qubit counts.

Acknowledgements

This work has been funded by the Federal Ministry of Research, Technology and Space (BMFTR) of Germany (Quantum Technologies/HoliQC2).

References

- [Ba21] Bauer, C. W. et al.: Practical considerations for the preparation of multivariate Gaussian states on quantum computers, 2021, arXiv: 2109.10918 [quant-ph].
- [BM05] Beylkin, G.; Mohlenkamp, M. J.: Algorithms for Numerical Analysis in High Dimensions. *SIAM Journal on Scientific Computing* 26 (6), pp. 2133–2159, 2005.
- [Gi15a] Gidney, C.: Constructing Large Controlled Nots, 2015, <https://algassert.com/circuits/2015/06/05/Constructing-Large-Controlled-Nots.html>, accessed: 05/02/2025.
- [Gi15b] Gidney, C.: Constructing Large Increment Gates, 2015, <https://algassert.com/circuits/2015/06/12/Constructing-Large-Increment-Gates.html>, accessed: 05/02/2025.
- [HRS17] Häner, T.; Roetteler, M.; Svore, K. M.: Factoring using $2n+2$ qubits with Toffoli based modular multiplication, 2017, arXiv: 1611.07995 [quant-ph].
- [Hü24] Hüttenhofer, O.: Building a Quantum Circuit for Gaussian Convolutions, source-code for implementations described here, 2024, https://tequilahub.github.io/tequila-tutorials/research/gaussian_convolution/, accessed: 05/02/2025.
- [Ko21] Kottmann, J. S. et al.: Tequila: A platform for rapid development of quantum algorithms. *Quantum Science and Technology* 6 (2), v.1.9.9dev source-code: <https://github.com/tequilahub/tequila@devel>, p. 024009, 2021.
- [KW09] Kitaev, A.; Webb, W. A.: Wavefunction preparation and resampling using a quantum computer, 2009, arXiv: 0801.0342 [quant-ph].
- [SBM06] Shende, V.; Bullock, S.; Markov, I.: Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25 (6), pp. 1000–1010, 2006.
- [SCC24] Sünderhauf, C.; Campbell, E.; Camps, J.: Block-encoding structured matrices for data input in quantum computing. *Quantum* 8, p. 1226, 2024.
- [TTK10] Takahashi, Y.; Tani, S.; Kunihiro, N.: Quantum addition circuits and unbounded fan-out. *Quantum Info. Comput.* 10 (9), pp. 872–890, 2010.