



This is a repository copy of *Block-encoding structured matrices for data input in quantum computing*.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/230540/>

Version: Accepted Version

Article:

Sünderhauf, C., Campbell, E. and Camps, J. (2024) Block-encoding structured matrices for data input in quantum computing. *Quantum*, 8. p. 1226. ISSN: 2521-327X

<https://doi.org/10.22331/q-2024-01-11-1226>

Reuse

This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here:

<https://creativecommons.org/licenses/>

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Block-encoding structured matrices for data input in quantum computing

Christoph Sünderhauf¹, Earl Campbell^{1,2}, and Joan Camps¹

¹Riverlane, St. Andrews House, 59 St. Andrews Street, Cambridge CB2 3BZ, United Kingdom

²Dept. of Physics and Astronomy, University of Sheffield, Sheffield S3 7RH, United Kingdom

February 21st, 2023. Accepted: January 5th, 2024

The cost of data input can dominate the run-time of quantum algorithms. Here, we consider data input of arithmetically structured matrices via *block encoding* circuits, the input model for the quantum singular value transform and related algorithms. We demonstrate how to construct block encoding circuits based on an arithmetic description of the sparsity and pattern of repeated values of a matrix. We present schemes yielding different subnormalisations of the block encoding; a comparison shows that the best choice depends on the specific matrix. The resulting circuits reduce flag qubit number according to sparsity, and data loading cost according to repeated values, leading to an exponential improvement for certain matrices. We give examples of applying our block encoding schemes to a few families of matrices, including Toeplitz and tridiagonal matrices.

Contents

1	Introduction	2
2	Block-encoding schemes	3
2.1	Base scheme	4
2.1.1	Simplified introductory case	5
2.1.2	Full block encoding	6
2.2	Preamplified scheme	8
2.3	PREP/UNPREP scheme	10
2.4	Hermitian block encoding for symmetric matrices	11
3	Example block encodings	12
3.1	Checkerboard matrix	12
3.2	Toeplitz matrix	14
3.3	Tridiagonal matrix	15
3.4	Extended binary tree matrix	17
3.5	2-dimensional Laplacian	18
4	Conclusions and Outlook	21
	Bibliography	22
A	Implementation of data loading step	25
A.1	Multiplexed rotations	25
A.2	State preparation	26
A.3	Data loading oracle in Gilyén et. al.'s sparse scheme	26
B	Singular value amplification	27

Christoph Sünderhauf: christoph.sunderhauf@riverlane.com

1 Introduction

The advent and astonishing increase in computational power of classical computing has truly revolutionised the world and ushered in the age of information. Yet, there are computational problems that are and will stay out of reach of classical computation due to their exponential complexity. Quantum computing [1] offers the dazzling prospect to provide a speed up and move select problems from the intractable to the tractable. Demonstrations of first quantum computers [2–4] up to a few hundred qubits provide a first step towards realising such a computational advantage. Apart from vast increases in the number and quality of qubits on the hardware level, improvements and development of new quantum algorithms are also dearly called for. Currently, the number of quantum algorithms and application use cases known to provide an exponential advantage are rather limited [5, 6].

The inception of the quantum singular value transform (QSVT) [7] has recently led to a new perspective on quantum algorithms. In what has been termed the “grand unification of quantum algorithms” [8], many previously known quantum algorithms have been reformulated within the framework of QSVT, including matrix inversion, phase estimation, Hamiltonian simulation, and amplitude amplification. The core of the QSVT algorithm is a polynomial transformation of an input matrix’s singular values. Different choices of polynomials and input matrices give rise to these various applications. In order to run a quantum algorithm in a real-world setting, data input (into the quantum computer) and data output (readout) are important steps and can severely limit any speed up provided by the quantum algorithm itself [9]. In this article, we will study how to input structured data efficiently and provide a scheme that facilitates the construction of explicit quantum circuits for input of structured data matrices, demonstrated by several examples.

In QSVT based algorithms, the input model for matrices of data is that of a *block encoding* [7]. Generally, an input matrix of data A could be non-unitary and there is no quantum circuit that could implement the operator A directly. Instead, in a block encoding, A is embedded as a block inside a larger unitary U :

$$U = \begin{pmatrix} A/\alpha & \text{junk}_i \\ \text{junk}_{ii} & \text{junk}_{iii} \end{pmatrix}. \quad (1)$$

Inside the larger unitary, the matrix A is scaled down by the subnormalisation α . The precise values in the junk blocks are inconsequential (but must be consistent throughout one QSVT circuit). Subnormalisation and junk serve two purposes: First of all, they can be necessary to ensure that an embedding of A into a unitary exists. However, the existence of an embedding (or even numerical knowledge of possible junk values) is not sufficient to input the data matrix A . Rather, U must be implemented as a quantum circuit and expressed in an elementary quantum gate set in order to run it on a quantum computer. Finding such a quantum circuit will typically further increase the subnormalisation α and the dimension of the junk blocks. In a quantum circuit, the block encoding can be written as follows:

$$\begin{array}{c} \text{flag qubits: } |0\rangle \\ |j\rangle \end{array} \xrightarrow[N]{U} \begin{array}{c} |0\rangle \\ |i\rangle \end{array} = A_{ij}/\alpha. \quad (2)$$

The top qubit register consists of the flag qubits, its dimension depends on the dimensions of the junk blocks in (1). The block of the unitary containing A/α is selected by initialising the flag qubits as $|0\rangle$ and postselecting them as $|0\rangle$. The probability of measuring the correct $|0\rangle$ outcome on all flag qubits is related to the subnormalisation; a smaller subnormalisation α is better. A smaller subnormalisation typically also reduces the circuit length of a QSVT transforming the matrix, as lower resolution and lower polynomial degree are required. The bottom register has the same dimension N as the matrix A . The matrix elements A_{ij}/α can then be recovered as the amplitudes on the bottom register, as indicated in the circuit diagram.

Since block encodings are fundamental to QSVT, they have been studied previously. Quantum circuits implementing block encodings for arbitrary dense matrices have been worked out [10, 11] but are exponentially expensive: For a $2^n \times 2^n$ matrix on n qubits, they require $O(2^n)$ T gates.¹ However,

¹The authors propose using a qRAM [12, 13] to arrange the exponential number of T gates in a linear *depth* circuit, which however leads to an unpractical large exponential number of ancilla qubits.

the scope of these works did not include optimisations for matrices that are sparse or structured (have repeated values). Other work shows quantum circuit constructions for sparse matrices based on black-box oracles [7]. There, the implementation of the oracles is not discussed, and complexity of the block encoding circuits is analysed in terms of black box usages. More specialised schemes for, eg. density operators, POVM operators, Gram matrices [7], or kernel matrices [14] have also been discussed. Explicit circuits for certain sparse, structured matrices are shown in [15] for a specific value of N .

To gain a computational advantage for some problems, the exponential cost for arbitrary matrices [10, 11] must be reduced. In this work, we consider matrices that are sparse and/or structured (having repeated elements); with sufficient structure, we can construct exponentially more efficient circuits. We provide several variations of a block-encoding scheme based on oracles, each with a different subnormalisation. It depends on the matrix which variation performs best. The schemes fully take into account sparsity, reducing the number of flag qubits beyond schemes in [7, 10, 11]. Moreover, we explain how to construct circuit implementations of the required oracles given a family of matrix structures for increasing N , and provide several explicit examples.

Block encodings also appear beyond QSVT. In fact, there is a calculus allowing block-encoded matrices to be summed and multiplied [7]. Many modern quantum algorithms for chemistry are based on phase estimation via qubitisation [16–20], an application of quantum walks [21, 22]. They are based on block-encoding the Hamiltonian. Successive articles go into great detail on how to construct the block encoding, and advances mainly stem from lower subnormalisation and shorter circuits of the block encoding. For chemistry applications, the block encodings are constructed as a linear combination of unitaries (LCU) with so-called PREPARE and UNPREPARE oracles. This viewpoint is quite distinct from other block-encoding schemes in the literature mentioned in the previous paragraphs; in section 2.3, we connect these two viewpoints, which can lead to an improved subnormalisation.

The quantum circuits involved are too long to directly run with noisy qubits provided by current hardware devices. Despite efforts to run QSVT primitives on noisy qubits [23], error correction and fault-tolerant quantum computation remains essential for longer circuits. In quantum error correction, an error correcting code is applied to several noisy physical qubits, yielding one or more logical qubits [24]. The circuit is run at the level of logical qubits. Not all gates are equal in error correction codes, rather, in the popular surface codes [25], so-called T gates and Toffoli gates (which can be reexpressed with T gates) are much more expensive than Clifford gates [26–30]. Consequently, we will focus on T gate count in assessing the cost of the block encoding.

Section 2 explains and compares the proposed block encoding schemes. Their costs are summarised in Table 1, and an adaptation to yield Hermitian block encodings for symmetric matrices is considered in section 2.4. Examples of using the scheme to construct block encodings of specific matrix families are provided in section 3, for example, our Toeplitz matrix encoding provides an exponential advantage to arbitrary matrix encoding. Finally, we draw some conclusions and give an outlook in section 4. It is our hope that the schemes presented in this work will in the future enable the construction of efficient block encoding circuits for further families of matrices from a wide range of application areas. The appendices give more detail on the circuit implementations of some parts of the block encoding: Appendix A focuses on the data loading oracles. Appendix B explains singular value amplification, a technique that can be employed to improve the subnormalisation, and provides some new analysis of its efficacy.

2 Block-encoding schemes

Throughout, we consider real matrices, including negative values. The schemes presented are adapted to structured matrices; that is, possibly sparse matrices with repeated data values and arithmetic descriptions of the structure available. This will become clear when constructing the circuits.

We take the following product as a total figure of merit for a block encoding:

$$(T\text{-gate count}) \cdot \text{subnormalisation}. \quad (3)$$

A lower subnormalisation increases the probability to measure $|0\rangle$ for the flag qubits, making it easier

Structured matrices (this work)	Data loading cost	Subnormalisation	Flag qubits
Base (sec. 2.1)	D	$\sqrt{S_c S_r} \ A\ _{\max}$	$2 + \log_2 S$
Preamplified (sec. 2.2)	$D \cdot \text{amp}$	$\sqrt{2} \mu_p(A)$	$5 + \log_2 S$
PREP (for $D \leq S$) (sec. 2.3)	$2D$	$\sqrt{\sum_d A_d ^{2p} \sum_d A_d ^{2-2p}}$	$1 + \log_2 S$
↳ with optimal $p = 1/2$	D	$\frac{\sqrt{S_c S_r}}{D} \sum_d A_d $	$1 + \log_2 S$
Sparse access matrices (Gilyén et. al. [7])			
Base ([7], Lemma 48)	blackbox	$\sqrt{S_c S_r} \ A\ _{\max}$	$3 + \log_2 N$
Preamplified ([7], Lemma 49)	blackbox \cdot amp	$\sqrt{2} \mu_p(A)$	$8 + \log_2 N$
Explicit matrices (D. Camps et. al. [15])			
Banded circulant ([15], sec. 4.1)	$D = S$	$S \ A\ _{\max}$	$1 + \log_2 S$
Quantum data structure (Chakraborty et. al. [11], Clader et. al. [10])			
Base ([11], Lemma 6.2)	$N^2 + N$	$\ A\ _F$	$1 + \log_2 N$
p -norm ([11], Lemma 6.1)	$2N^2$	$\mu_p(A)$	$2 + \log_2 N$

$$\mu_p(A) = \sqrt{\max_i \sum_j |A_{ij}|^{2p} \max_j \sum_i |A_{ij}|^{2-2p}} \quad \text{amp} \approx 3 \left(\frac{\gamma_c}{\delta} \log \frac{\gamma_c}{\varepsilon} + \frac{\gamma_r}{\delta} \log \frac{\gamma_r}{\varepsilon} \right)$$

$$\gamma_c = \sqrt{\frac{S_c \|A\|_{\max}^{2p}}{\sqrt{2} \max_i \sum_j |A_{ij}|^{2p}}} \quad \gamma_r = \sqrt{\frac{S_r \|A\|_{\max}^{2-2p}}{\sqrt{2} \max_j \sum_i |A_{ij}|^{2-2p}}}$$

Table 1: Comparison of block-encoding schemes for a matrix A . N dimension of the matrix. D number of distinct data values A_d . M maximum multiplicity of each value. S_c, S_r maximum column and row sparsities (no. non-zero elements per column/row). The maximum sparsity $S = \max(S_c, S_r)$ or multiplicity M are padded to ensure $SN = DM$, if necessary. In the comparison we’ve added further factors to Gilyén et. al.’s results to remove the $|A_{ij}| \leq 1$ assumption. Block encoding from a quantum data structure is also discussed in [10]. Lower is better for all of data loading cost, subnormalisation, and flag qubit number. D data loading incurs $O(D)$ T-gate count, or $O(\sqrt{D})$ if a large number $O(\sqrt{D})$ of possibly dirty ancillas are available (see appendix A).

to extract the matrix. Lower subnormalisation typically also leads to shorter circuits in QSVT [7, 8] or qubitisation [17] algorithms that use the block encoding. Taking the product (3) as the figure of merit is motivated as it is approximately constant under singular value amplification (see appendix B)—which can reduce the subnormalisation of a block encoding by inversely increasing its circuit length (up to a logarithmic factor). For simplicity, instead of considering T -gate count as a measure for circuit cost as in (3), we focus on “data loading cost”, i.e. the number of data values loaded, which generically dominates the cost. As we will see, we expect the other circuit parts to be implementable with $O(\text{polylog } N)$ T gates. Of course, for low data loading cost, these other circuit parts could become dominant. Therefore we focus on the figure of merit

$$(\text{data loading cost}) \cdot \text{subnormalisation}. \quad (4)$$

The next subsections first describe our base block encoding scheme (section 2.1), followed by variants termed preamplified (section 2.2) and PREP/UNPREP (section 2.3) that for some applications will improve the figure of merit (4).

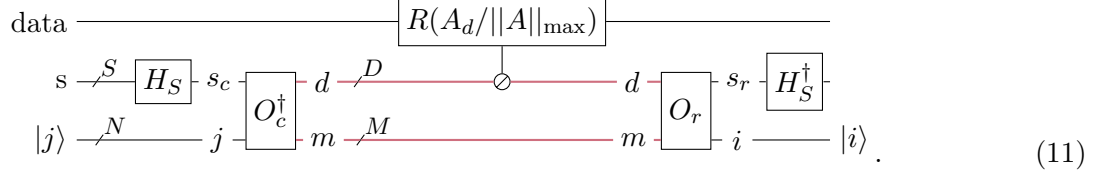
2.1 Base scheme

Let us introduce notation. Let N be the dimension of the matrix A , and D be the number of distinct data items A_d ($0 \leq d < D$) in the matrix (apart from zeros following the sparsity pattern). Crucially, each data value may appear multiple times in the matrix. The multiplicity M is the maximum multiplicity of any of the D data items. Finally, we have column S_c and row S_r sparsities, the maximal number of non-zero elements per column or row (according to the sparsity pattern, if any) and maximum sparsity $S = \max(S_c, S_r)$. The smaller S , the more sparse the matrix.

The oracles are supplemented by H_S to construct the block encoding. The gate H_S , sometimes called diffusion operator, creates an equal superposition state

$$H_S |0\rangle = \frac{1}{\sqrt{S}} \sum_{s=0}^{S-1} |s\rangle. \quad (10)$$

If S is a power of 2, H_S is simply a string of Hadamard gates. In other cases, it can for example be constructed by amplitude amplification that uses an ancilla (see [17]). Putting these together, the block encoding is



Crucially, the pink qubit registers in the middle can have dimensions D and M distinct from the black qubit registers of dimension S and N . Yet, thanks to the equality $NS = MD$, the oracles have same total input and output dimensions. To recover the matrix from the block encoding, the flag qubits (i.e. the data qubit and the s register) must be initialised and postselected as $|0\rangle$. The values $\frac{A_{ij}}{S||A||_{\max}}$ are then recovered when initialising the bottom register with $|j\rangle$ and postselecting/measuring an $|i\rangle$. This can be most easily seen by inserting a resolution of the identity $\mathbb{I} = \sum_d \sum_m |d\rangle\langle d| \otimes |m\rangle\langle m|$ into the middle of the circuit:

$$\begin{aligned} & \langle 0|^{\text{data}} \langle 0|^s \langle i| H_S^\dagger O_r O_{\text{data}} \left(\sum_{d=0}^{D-1} \sum_{m=0}^{M-1} |d\rangle\langle d| \otimes |m\rangle\langle m| \right) O_c^\dagger H_S |0\rangle^{\text{data}} |0\rangle^s |j\rangle \\ &= \sum_{d=0}^{D-1} \sum_{m=0}^{M-1} \frac{A_d}{||A||_{\max}} \langle 0|^s \langle i| H_S^\dagger O_r |d\rangle\langle m| \langle d|\langle m| O_c^\dagger H_S |0\rangle^s |j\rangle \\ &= \sum_{d=0}^{D-1} \sum_{m=0}^{M-1} \frac{A_d}{||A||_{\max}} \frac{1}{\sqrt{S}} \delta_{i=i(d,m)} \frac{1}{\sqrt{S}} \delta_{j=j(d,m)} = \frac{A_{ij}}{S||A||_{\max}}. \end{aligned} \quad (12)$$

The factor of S comes from the H_S gates. Thus, the circuit is a block encoding of A with subnormalisation $S||A||_{\max}$ and $1 + \log_2 S$ flag qubits of total dimension $2S$. The matrix is implemented exactly, apart from any finite accuracy in the multiplexed rotations (the data loading).

In applications of the block encoding scheme, we expect the matrix structure (the sparsity pattern and pattern of repeated values) to be given, from which O_c and O_r can be inferred. Different instances of the problem only require replacement of the data loading oracle, a straightforward automatable task.

2.1.2 Full block encoding

A general structured matrix may not fulfill the strict requirements of the simplified case in the preceding section 2.1.1. Here, we consider the general case, where each of the D data items may have a distinct multiplicity (with maximum M), and each of the N rows and columns may have distinct sparsities (with maximum S_c for columns and S_r for rows, and $S = \max(S_c, S_r)$). Therefore, a priori, $MD = NS$ may not hold. We pad M and/or S , increasing their size with further dummy index range, until

$$MD = NS \quad (13)$$

is fulfilled and the same construction with oracles O_c and O_r is possible. The action of these oracles on the padded dummy indices is insignificant (they will be flagged and deleted by an out-of-range oracle introduced in the next paragraph), as long as unitarity is ensured. Without loss of generality, we assume the resulting qubit register dimensions are powers of 2; otherwise they can trivially be embedded in a larger register made up of an integer number of qubits.

As in the simplified case, the matrix is labelled from the three perspectives (d, m) , (i, s_r) , and (j, s_c) . Now however, because of the padding, not all labels (d, m) with $0 \leq d < D, 0 \leq m < M$ are in-range and are mapped by O_r and O_c to row and column indices i, j matching the matrix pattern. These are flagged by a new oracle, which we call O_{rg} , the out-of-range oracle:

$$O_{rg} |d\rangle |m\rangle |0\rangle = \begin{cases} |d\rangle |m\rangle |0\rangle & \text{if } A_{i(d,m),j(d,m)} = A_d \\ |d\rangle |m\rangle |1\rangle & \text{if } A_{i(d,m),j(d,m)} = 0. \end{cases} \quad (14)$$

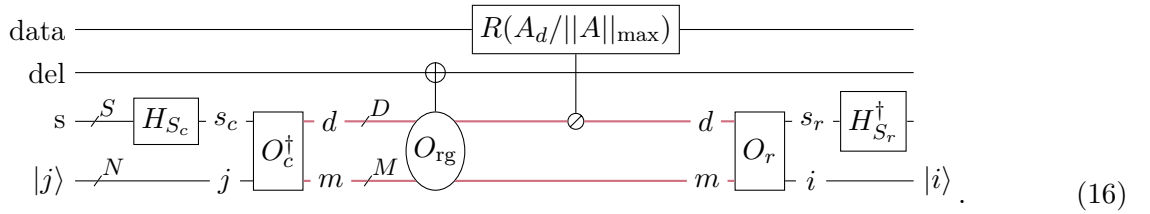
The oracle is controlled on the D, M registers and flips a “delete” flag qubit whenever (d, m) is out-of-range, which we draw as



Reminiscent of the usual circular notation for controls, the circle/oval serves as the control for the O_{rg} -controlled NOT; the d and m registers are not modified. The out-of-range oracle can be implemented with quantum arithmetics, possibly using ancilla qubits, and have a low $O(\text{polylog } N)$ T -gate count. Like for the O_c and O_r oracles, the arithmetic expressions underlying O_{rg} follow from the structure of the matrix and the chosen labelling in terms of (d, m) .

The oracles O_c and O_{data} may map invalid (d, m) to any values, they just need to be unitary. For valid (d, m) , the specific values of the corresponding s_r and s_c are irrelevant, as long as they are within range $0 \leq s_r < S_r$ and $0 \leq s_c < S_c$. Example constructions are shown in section 3.

The full block encoding circuit is:



It has one flag qubit (the del qubit) more than the simplified scheme, giving $2 + \log_2 S$ flag qubits. The subnormalisation now takes into account possibly unequal $S_c \neq S_r \neq S$ and is

$$\sqrt{S_c S_r} ||A||_{\max}. \quad (17)$$

The data loading oracle must load D data items. Because the other oracles can be implemented with arithmetics in $O(\text{polylog } N)$ T -count [31, 32], we take the number of data items to load as a metric for the cost of the block encoding circuit. Note that if D is small and constant as the size N grows, eventually the cost of the other oracles will surpass the data loading.

The cost figures (data loading cost, subnormalisation, flag qubit dimension) are summarised in table 1. As a reference, we also include the cost figures from encoding schemes found in the literature. Gilyén et. al. present a block encoding scheme for sparse access matrices ([7] Lemma 48). In contrast to our scheme, where the data loading oracle is multiplexed only on the D distinct indices, in their scheme the data loading oracle is multiplexed on the N row and N column indices i and j . The inner workings of the oracle are not specified. See appendix A.3 for a possible efficient implementation. The subnormalisation is the same as in our case. While they have $3 + \log_2 N$ flag qubits, our encoding takes into account the sparsity and requires only $2 + \log_2 S$ flag qubits.

Chakraborty et. al. present a block encoding scheme for a matrix from a quantum data structure ([11], Lemma 6.2). A similar construction is also discussed by Clader et. al. [10]. Regardless of the sparsity, the data loading cost is always $N^2 + N$ items. The works presuppose that this data loading can be performed efficiently with a quantum data structure like qRAM [12, 13]. It promises a logarithmic T -gate depth. However, this is achieved by a large parallel execution of T gates on a large number of ancillary qubits. The total number of T gates is not reduced compared to other approaches (QROM [17], select-swap [33]). In the comparison table 1 we remain agnostic about the type of data loading

(qRAM, QROM, ...) and record the number of items to be loaded. When $D \leq N$, we find a lower figure of merit (4), (data loading cost) · subnormalisation, for our block encoding scheme:

$$(N^2 + N)\|A\|_F \geq N^2\|A\|_{\max} \geq D\sqrt{S_c S_r}\|A\|_{\max}. \quad (18)$$

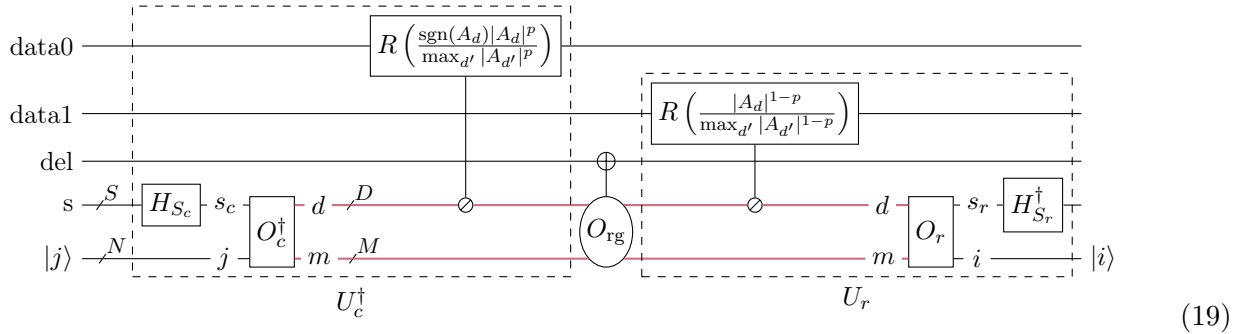
However, for a matrix with more distinct data entries than N , the scheme by Chakraborty may or may not be advantageous to the base scheme presented above, depending on the matrix norms.

D. Camps et al. [15] show an explicit example construction for a circulant matrix. Similarly to our scheme, it requires only D data loading even though values appear multiple times, and a flag qubit number of $\approx \log_2 S$. However, the scheme is only suitable for matrices with a very particular structure: Each value must appear exactly once in each column. Our block-encoding encompasses this case and goes beyond, covering other types of structured matrices.

2.2 Preamplified scheme

In this section, we will show how the block encoding from the base scheme can, in some cases, be improved by a method called preamplification. The subnormalisation of a block encoding can be reduced by performing singular value amplification (see app. B) [7, 11], resulting in an ε -approximate block encoding with subnormalisation reduced by an amplification factor γ . However, it requires $O(\frac{\gamma}{\delta} \log \frac{\gamma}{\varepsilon})$ applications of the original block encoding (δ is related to a bound on the matrix's singular values). Simply performing singular value amplification on the full block encoding therefore does not improve the figure of merit (4), (data loading cost) · subnormalisation; in fact, because of the factors related to the accuracy of the approximate result, it gets worse. Preamplification, presented by Gilyén et. al. [7], is based on two individual singular value amplifications of two separate circuit parts. As we will see, it is also applicable to our block encoding and can improve its figure of merit (4) in some cases, intuitively when the matrix has values of strongly varying magnitude.

The starting point for preamplification is to split the data loading oracle into two parts according to a choice of $0 \leq p \leq 1$. The block encoding circuit (16) becomes:



The two data qubits and data loading oracles combine to give the same matrix as the original block encoding. In fact, also the subnormalisation

$$\left(\sqrt{S_c} \max_d |A_d|^p\right) \left(\sqrt{S_r} \max_d |A_d|^{1-p}\right) = \sqrt{S_c S_r} \max_d |A_d| = \sqrt{S_c S_r} \|A\|_{\max} \quad (20)$$

has not changed. The idea of preamplification is that, as we will see momentarily, the two circuit parts U_c^\dagger and U_r are block-encodings in their own right, and can be singular value amplified individually by amplification factors γ_c and γ_r . While the data loading cost increases by a factor of $O(\gamma_c + \gamma_r)$ (dropping logarithmic factors), the total subnormalisation of the matrix reduces multiplicatively by $\gamma_c \gamma_r$. Depending on the factors hidden in the big- O notation, preamplification can thereby reduce the figure of merit (data loading cost) · subnormalisation.

Let us discuss U_c^\dagger , where U_r is similar. The unitary U_c^\dagger is a slightly more general block encoding, where the flag qubits on the left in circuit (19) are data0 through s, and on the right only data0 through del. Omitting data1 and del that do not participate in U_c^\dagger , the encoded non-square matrix

can be deduced by inserting resolutions of identity $\sum_d |d\rangle\langle d|, \sum_m |m\rangle\langle m|, \sum_j |j\rangle\langle j|$:

$$\langle 0_{\text{data}0} | U_c^\dagger | 0_{\text{data}0} \rangle | 0_s \rangle = \sum_{j=0}^{N-1} \sum_{d=0}^{D-1} \sum_{m=0}^{M-1} |d\rangle |m\rangle \langle d| \langle m| \frac{\text{sgn}(A_d) |A_d|^p}{\max_{d'} |A_{d'}|^p} O_c^\dagger H_{S_c} | 0_s \rangle |j\rangle \langle j| \quad (21)$$

$$= \sum_{j=0}^{N-1} \sum_{d=0}^{D-1} \sum_{m=0}^{M-1} \sum_{s_c=0}^{S_c-1} \frac{1}{\sqrt{S_c}} |d\rangle |m\rangle \frac{\text{sgn}(A_d) |A_d|^p}{\max_{d'} |A_{d'}|^p} \underbrace{\langle d| \langle m| O_c^\dagger |s_c\rangle |j\rangle \langle j|}_{\text{fixes } (d, m) \text{ corresponding to } (j, s_c)} \quad (22)$$

$$= \sum_{j=0}^{N-1} \sum_{s_c=0}^{S_c-1} \underbrace{|d\rangle |m\rangle}_{|x_j\rangle} \frac{\text{sgn}(A_d) |A_d|^p}{\sqrt{S_c} \max_{d'} |A_{d'}|^p} \langle j| \quad (23)$$

with $d = d(j, s_c), m = m(j, s_c)$ according to O_c^\dagger . This matrix is already written in the form of a singular value decomposition because $\{|x_j\rangle\}$ and $\{|j\rangle\}$ are orthogonal systems. The singular values follow as their normalisation

$$\zeta_j = \sqrt{\langle x_j | x_j \rangle} = \sqrt{\frac{\sum_{i=0}^{N-1} |A_{ij}|^{2p}}{S_c \max_d |A_d|^{2p}}}. \quad (24)$$

Following [7], we choose the amplification factor γ_c as follows (see appendix B for details):

$$\gamma_c = \frac{1}{\sqrt[4]{2}} \frac{1}{\max_j \zeta_j} = \max_d |A_d|^p \sqrt{\frac{S_c}{\sqrt{2} \max_j \sum_i |A_{ij}|^{2p}}}. \quad (25)$$

Replacing U_c^\dagger and U_r by their amplification circuits results in a total subnormalisation divided by $\gamma_c \gamma_r$, such that the new subnormalisation after preamplification is

$$\frac{\sqrt{S_c} \max_d |A_d|^p \cdot \sqrt{S_r} \max_d |A_d|^{1-p}}{\gamma_c \gamma_r} = \sqrt{2 \max_j \sum_i |A_{ij}|^{2p} \max_i \sum_j |A_{ij}|^{2-2p}}, \quad (26)$$

which we record in table 1. Two flag qubits are needed in addition to (19) to perform the two amplifications. The data loading cost will be

$$\approx D \cdot 3 \left(\frac{\gamma_c}{\delta} \log \frac{\gamma_c}{\varepsilon} + \frac{\gamma_r}{\delta} \log \frac{\gamma_r}{\varepsilon} \right), \quad (27)$$

see appendix B where we have determined the prefactor 3. The largest possible choice of δ is $\delta = 1 - 1/\sqrt[4]{2} \approx 0.16$, see (98) in the appendix B. The accuracy ε determines the accuracy of the amplifications of U_c^\dagger and U_r . Hence, the accuracy of the full matrix is bounded by $(2\varepsilon + \varepsilon^2) \cdot \text{subnormalisation}$.

Preamplification was introduced in the scheme by Gilyén et. al. [7], where it works the same way. Like in the comparison of the base block encoding, our scheme requires a lower number of flag qubits for sparse matrices ($S < N$).

While the p -norm encoding from the quantum data structure scheme ([11]) does not employ singular value amplification, it is similar in spirit because it splits the matrix element into powers by p and $1 - p$. The subnormalisation is the same as in preamplification, except for the $\sqrt{2}$ factor. To compare the data loading cost, bound the amplification factors by $\gamma_c \leq \sqrt{S_c/\sqrt{2}}, \gamma_r \leq \sqrt{S_r/\sqrt{2}}$. Then the preamplified data loading cost (dropping logarithms) obeys $D \cdot O(\gamma_c + \gamma_r) \leq O(D\sqrt{S})$, which is to be contrasted with $2N^2$ from p -norm encoding. If the matrix has $D = N^2$ different values, p -norm encoding is clearly favourable. Our scheme does well if the matrix is structured with a lower D .

Preamplification has a significant overhead in data loading cost, due to the prefactor, δ and ε in (27). In order to still achieve a benefit over the base block encoding scheme, the amplification factors γ_c (25) and γ_r must be large. Intuitively, this requires matrices with values of strongly varying magnitude. Whether the base or preamplified schemes yield a better block encoding w.r.t. (data loading cost) \cdot subnormalisation depends on the specific matrix. The optimal choice of p in the preamplification scheme depends on the specific matrix.

2.3 PREP/UNPREP scheme

When the matrix structure has repeated data such that $D \leq S_c, S_r$, often, the row/column oracle and the multiplexed rotations commute. Intuitively, this is the case when each data values appears in all (or most) of the rows and columns and the structure of the matrix is such that the columns and rows are (mostly) permutations of each other. Then, assuming commutativity, one can use a PREP/UNPREP scheme to reduce the subnormalisation of the base block encoding. PREP and UNPREP operators previously appeared in quantum algorithms for chemistry, when block-encoding Hamiltonians [17]. We show how such operators can be used in our block encoding scheme for more general structured matrices.

The starting point is splitting the multiplexed rotations into two parts (19), just like in preamplification. By assumption, the row/column oracles commute with the rotations, hence we can use the following identity:

$$\begin{array}{c}
|0\rangle \text{ --- } \boxed{R\left(\frac{\text{sgn}(A_d)|A_d|^p}{\max_{d'}|A_{d'}|^p}\right)} \text{ --- } |0\rangle \\
|0\rangle \xrightarrow{\mathcal{D}} \boxed{H_D} \text{ --- } \bigcirc \text{ --- }
\end{array}
= \underbrace{\left(\frac{\sqrt{\sum_d |A_d|^{2p}}}{\sqrt{D} \max_d |A_d|^p}\right)}_{:=X} \cdot \text{PREP } |0\rangle \quad (28)$$

with a prepare operator acting as

$$\text{PREP} |0\rangle = \frac{1}{\sqrt{\sum_d |A_d|^{2p}}} \sum_d \text{sgn}(A_d) |A_d|^p |d\rangle \quad (29)$$

and, because the left side is not a normalised quantum state, a scaling factor X . While $\text{PREP}|0\rangle$ is of course a normalised state, the left hand side of (28) is not, due to the postselection of the flag qubit as $|0\rangle$. The quotient factor X in (28) is smaller than one and results in a lower subnormalisation after application of the identity. A state preparation operator for a D -dimensional state can be implemented in various ways with no more than D data loading cost, see appendix A.2 for details. We use a similar operator on the right side of the circuit,

$$\langle 0 | \text{UNPREP} = \frac{1}{\sqrt{\sum_d |A_d|^{2-2p}}} \sum_d |A_d|^{1-p} \langle d |. \quad (30)$$

The total block encoding circuit is

[illegible]

with subnormalisation

$$\alpha_p = \sqrt{\frac{S_c}{D}} \sqrt{\sum_d |A_d|^{2p}} \sqrt{\frac{S_r}{D}} \sqrt{\sum_d |A_d|^{2-2p}} = \frac{\sqrt{S_c S_r}}{D} \sqrt{\sum_d |A_d|^{2p} \sum_d |A_d|^{2-2p}}. \quad (32)$$

Contrary to preamplification, there is no singular value amplification that will necessitate repeated data loading. Instead, data will just need to be loaded twice (i.e. $2D$ data loading cost), for PREP and for UNPREP.

An application of Callebaut’s inequality (a refinement of the Cauchy-Schwarz inequality) [34] shows that the subnormalisation α_p improves as $p \rightarrow 1/2$:

$$\frac{\sqrt{S_c S_r}}{D} \sum_d |A_d| = \alpha_{1/2} \leq \alpha_p \leq \alpha_q \text{ for } 1/2 \leq p \leq q. \quad (33)$$

There is a single optimal choice $p = 1/2$ of the parameter for all matrices, contrary to preamplification. In fact, for $p = 1/2$, one can choose $\text{UNPREP} = \text{PREP}^\dagger$ up to $\text{sgn}(A_d)$, and for certain circuit

implementations of PREP one can use measurement based uncomputation to significantly reduce the number of Toffoli gates in UNPREP [18]. Effectively, we take this into account by recording reduced data loading D for the optimal $p = 1/2$ in table 1.

Compared to our base scheme, the subnormalisation is always reduced:

$$\alpha_p = \sqrt{S_c S_r} \sqrt{\frac{\sum_d |A_d|^{2p}}{D}} \sqrt{\frac{\sum_d |A_d|^{2-2p}}{D}} \leq \sqrt{S_c S_r} \|A\|_{\max}^p \|A\|_{\max}^{1-p} = \sqrt{S_c S_r} \|A\|_{\max}. \quad (34)$$

The reduction is stronger when the matrix values have large varying magnitude than if they are all of similar size. Data loading cost is only increased by a factor of 2 compared to the base scheme; except for the optimal $p = 1/2$, when it is the same. Whenever possible (i.e. $D \leq S_c, S_r$ and the row and column oracles commute with the data loading), one should therefore choose PREP/UNPREP with $p = 1/2$ instead of the base scheme. It depends on the specific matrix including its values whether the preamplified or PREP/UNPREP scheme have a better figure of merit (4).

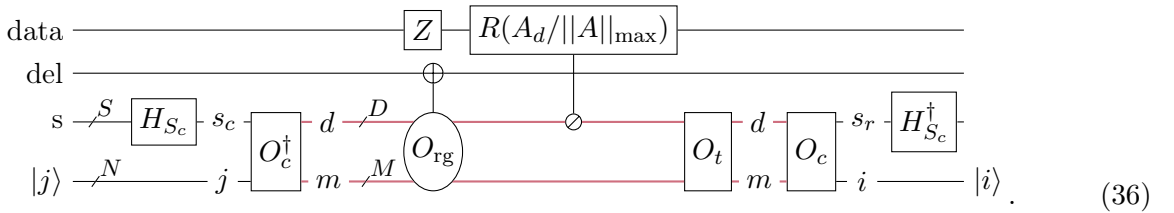
2.4 Hermitian block encoding for symmetric matrices

When the matrix A is symmetric, it can be desirable to construct a Hermitian block encoding. That simplifies quantum walks [21, 22] and phase estimation via qubitisation [16]. A priori, the block encodings constructed above may not be Hermitian, even if the matrix is symmetric.

In this section, we show how all our block encodings become Hermitian with only slight modifications when the oracles are constructed in a particular way. The starting point is, like in all our constructions, a labelling (d, m) of the matrix elements, and a column oracle $O_c : (d, m) \mapsto (j, s_c)$. Because the matrix is symmetric, elements related by transposition ($i \leftrightarrow j$) have the same d . Thus, there is an oracle O_t that maps $(d, m) \mapsto (d, m')$, the element related to (d, m) by transposition. On the diagonal of the matrix, $m' = m$. When the arithmetic expression is cast into a quantum circuit, it becomes unitary and Hermitian, because transposition is an involution $O_t^2 = \mathbb{I}$. As usual, the action of O_t on out-of-range (d, m) does not matter (apart from ensuring its unitarity and Hermiticity) because of O_{rg} flipping the delete qubit. A row oracle can then be constructed from the column oracle as

$$O_r = O_c O_t, \quad (35)$$

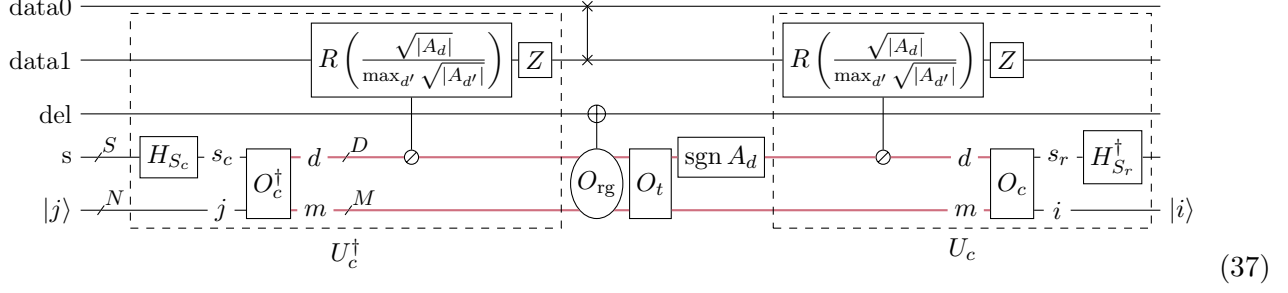
keeping in mind that $S_c = S_r$. We will show examples of constructions of Hermitian block encodings in section 3. A further Z gate must be added to the data qubit to make the base block encoding scheme Hermitian:



The circuit is Hermitian, because the left and right parts are Hermitian conjugates, and the middle part consists of three commuting Hermitian operators (the O_{rg} -controlled NOT, O_t , and the X -axis rotations (7) preceded by Z). The Z does not affect the encoded matrix because it has no effect when the flag qubit is initialised as $|0\rangle$. Yet, it is needed to make the full block encoding unitary Hermitian. This Hermitian counterpart of the base block encoding scheme has the same data loading cost, subnormalisation, and flag qubit number.

The preamplified scheme can also be made Hermitian by using the O_r oracle (35) constructed

above. Using $p = 1/2$, the circuit (19) can be adapted to:



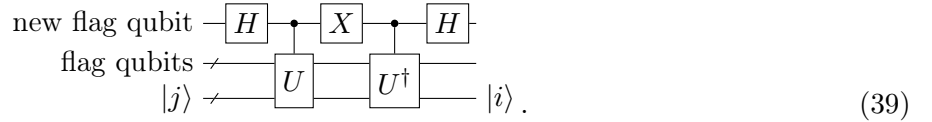
By removing the sign of A_d from the first multiplexed rotations in (19), the two subcircuits to be amplified can be made Hermitian conjugates. The same will hold for their amplification circuits. The correct sign is obtained with the gate denoted by $\text{sgn } A_d$. It can be seen as a new data loading oracle loading and applying the sign. It can be implemented with multicontrolled Z gates flipping the signs for those $|d\rangle$ with $\text{sgn } A_d = -1$. All circuit elements in the middle are Hermitian and commute. Compared to the preamplification scheme, this Hermitian counterpart has the same number of flag qubits, the same subnormalisation, but data loading increased by D , due to the separate sign oracle.

The PREP/UNPREP scheme is already Hermitian as-is for $p = 1/2$, provided $O_r = O_c O_t$ is used for the row oracle and matching PREP and UNPREP operators are used: While equations (29) and (30) only specify the operators' actions on $|0\rangle$, the identity

$$\text{PREP} = O_{\text{sgn}} \text{UNPREP}^\dagger \quad (38)$$

must hold as an operator identity to make the full circuit Hermitian. The oracle O_{sgn} flips the signs to match $\text{sgn}(A_d)$. Because O_{sgn} is Hermitian, the full PREP/UNPREP block encoding (31) is Hermitian. In practice, O_{sgn} is integrated into PREP, and the Hermitian counterpart of the PREP/UNPREP scheme has the same costs.

A simple way to make any block encoding U Hermitian is the following circuit [16]:



It increases the number of flag qubits by 1 and increases the gate complexity / data loading cost by a factor of two. Additional cost is incurred for controlling U and U^\dagger .

A Hermitian block encoding scheme in terms of certain black-box oracles is suggested in [15]. It cannot deal with negative matrix values and has $\log_2 N + 2$ flag qubits. It also duplicates data loading cost, in contrast to our Hermitian counterpart of the base block encoding.

3 Example block encodings

3.1 Checkerboard matrix

We will first consider a matrix with a checkerboard pattern as a nice example demonstrating how our block encoding scheme provides for repeated values, and application of PREP/UNPREP. The $N \times N$ checkerboard matrix (take N a power of 2) only has $D = 2$ data values, the sparsities are $S_c = S_r = S = N$ (it is dense), and $M = N^2/2$ multiplicity of each of the two values. It clearly has an arithmetic structure, which can be translated into row and column oracles. The starting point is a labelling of the matrix elements by (d, m) , $0 \leq d < D$, $0 \leq m < M$. We choose a labelling where m increases reading the matrix row by row from left to right. Specifically, for $N = 4$, we have the labelling:

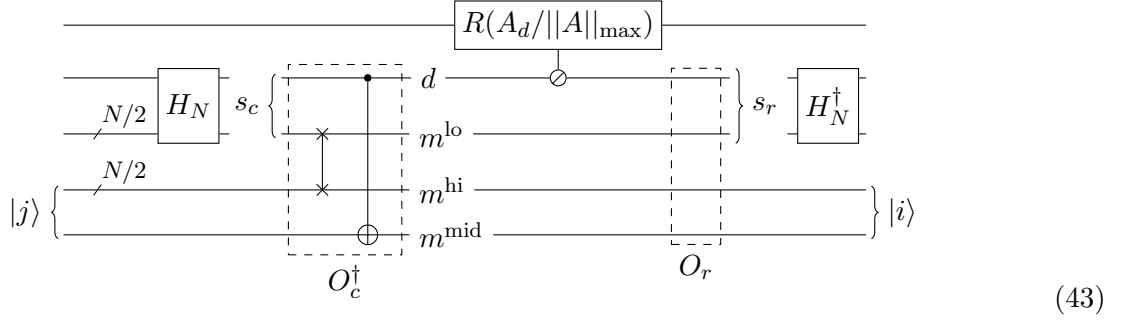
$$\begin{pmatrix} (d=0, m=0) & (d=1, m=0) & (d=0, m=1) & (d=1, m=1) \\ (d=1, m=2) & (d=0, m=2) & (d=1, m=3) & (d=0, m=3) \\ (d=0, m=4) & (d=1, m=4) & (d=0, m=5) & (d=1, m=5) \\ (d=1, m=6) & (d=0, m=6) & (d=1, m=7) & (d=0, m=7) \end{pmatrix} \quad (40)$$

For simplicity of presentation we split $m = Nm^{\text{hi}} + (N/2)m^{\text{mid}} + m^{\text{lo}}$ into its high $\log(N/2)$ bits, mid bit, and low $\log(N/2)$ bits. The row and column indices i and j can be computed from (d, m) as follows:

$$i(d, m) = \left\lfloor \frac{m}{N/2} \right\rfloor = 2m^{\text{hi}} + m^{\text{mid}} \quad (41)$$

$$j(d, m) = 2(m \bmod (N/2)) + (d + i \bmod 2) = 2m^{\text{lo}} + (d + m^{\text{mid}} \bmod 2). \quad (42)$$

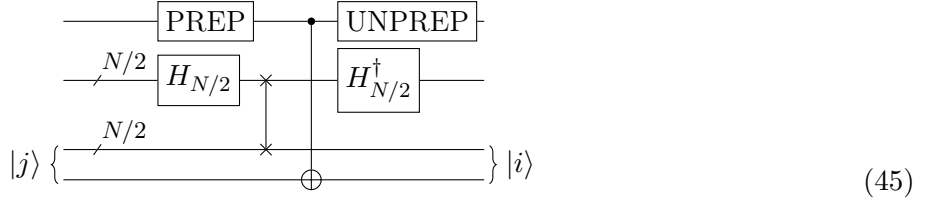
The specific values of s_c and s_r are not important for the block encoding. Oracles for these expressions are very simple, O_r can even be implemented as the identity. The full block encoding circuit is:



The block encoding has data loading cost $D = 2$. Note that the oracles are very cheap, in fact, zero T gate count. It turns out that this block encoding is already Hermitian. The subnormalisation is

$$\sqrt{S_r S_c} \|A\|_{\max} = N \max(|A_0|, |A_1|) \quad (44)$$

The subnormalisation can be improved by going to PREP/UNPREP, since $D \leq S$ and the oracles commute with the multiplexed rotations. The corresponding block encoding circuit is:



with

$$\text{PREP} |0\rangle = \frac{1}{\sqrt{|A_0| + |A_1|}} \left(\sqrt{A_0} |0\rangle + \sqrt{A_1} |1\rangle \right). \quad (46)$$

The subnormalisation follows as

$$\frac{\sqrt{S_c S_r}}{D} \sum_d |A_d| = N \frac{|A_0| + |A_1|}{2}. \quad (47)$$

When the matrix elements have varying magnitude, this presents an improvement over the subnormalisation (44) of the base scheme.

The checkerboard matrix serves as a simple pedagogical example. Because of its low rank and known eigenvalues, it is unlikely to be used in an actual quantum algorithm. Alternatively to our schemes, a block encoding could also be constructed by appreciating the factorisation

$$A = (|+\rangle\langle+|)^{\otimes \log_2 N-1} \otimes (A_0 \mathbb{I} + A_1 X) \quad (48)$$

of the matrix, leading to a circuit with the same ancilla, T count and subnormalisation as our schemes.

Our scheme can adapt to changes in the structure of the matrix. For example, to demonstrate the out-of-range O_{rg} , consider a checkerboard matrix with top left and bottom right entries replaced by zero. The row and column oracles are as above; the additional O_{rg} oracle deletes out-of-range labels

(d, m) . These are $(d = 0, m = 0)$ and $(d = 0, m = N^2/2 - 1)$ for top-left and bottom-right entries. A circuit implementation of the oracle is

$$(49)$$

and can be inserted in both the base circuit (43) and the PREP/UNPREP circuit (45).

3.2 Toeplitz matrix

Consider a Toeplitz matrix with D diagonals, i.e. D values, offset from the main diagonal by k :

$$\begin{pmatrix} A_k & A_{k-1} & \cdots & A_0 & & & \\ A_{k+1} & A_k & A_{k-1} & \cdots & A_0 & & \\ \vdots & A_{k+1} & A_k & A_{k-1} & \cdots & A_0 & \\ A_{D-1} & \vdots & A_{k+1} & A_k & A_{k-1} & \cdots & A_0 \\ & A_{D-1} & \vdots & A_{k+1} & A_k & A_{k-1} & \cdots & A_0 \\ & & A_{D-1} & \vdots & A_{k+1} & A_k & A_{k-1} & \cdots \\ & & & A_{D-1} & \vdots & A_{k+1} & A_k & A_{k-1} \end{pmatrix} \quad (50)$$

For an $N \times N$ matrix with $N \geq D$ it follows $S_c = S_r = S = D$, $M = N$. This already fulfills $NS_c = NS_r = MD$ such that no padding is necessary, and we assume N and D to be powers of 2 for simplicity.

Different choices can be made for the labelling in terms of (d, m) (distinct values and their repetitions). For the distinct values, we choose d as above. For m , we will simply choose the column index. Note that for example, $(d = 0, m = 0)$ is out-of-range: In general, there is no A_0 in column 0. Arithmetically, the mapping to row and column indices is then:

$$i(d, m) = d - k + m, \quad j(d, m) = m \quad (51)$$

and out-of-range (d, m) pairs are those where an overflow or underflow in the calculation of i occurs, that is when

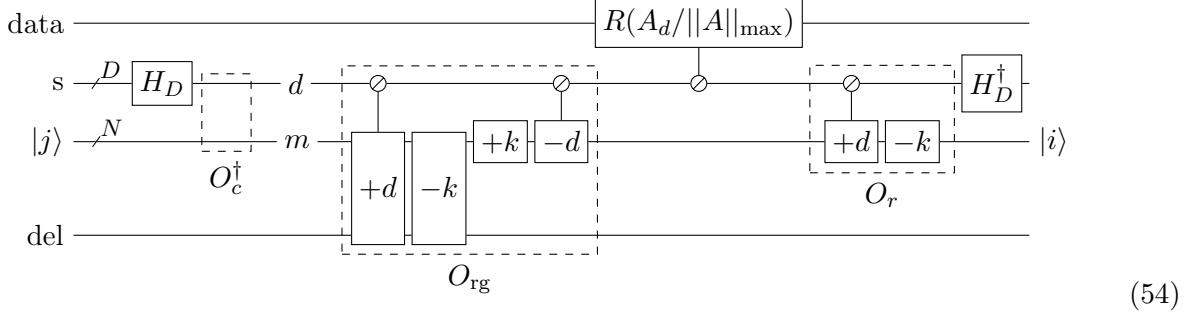
$$d - k + m < 0 \text{ or } d - k + m \geq N. \quad (52)$$

The oracles can be constructed with in-place additions, such that the block-encoding circuit is as follows. The last qubit is an ancilla qubit serving as the overflow (as the next higher bit) for the modular additions. It is not a flag qubit because it is uncomputed back to $|0\rangle$. (Note the additions can be performed without further ancillas if only few qubits are available [35–37].)

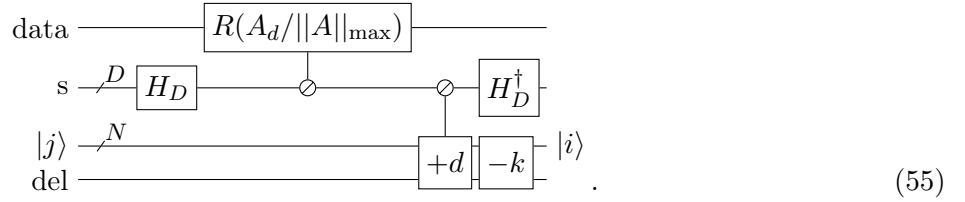
$$(53)$$

If we were encoding a banded circulant matrix, the equations (51) would have been taken mod N . There would be no out-of-range pairs (d, m) and no O_{rg} oracle, del qubit, or overflow ancilla qubit. The above circuit would then be a block encoding with $\log_2 D + 1 = \log_2 S + 1$ flag qubits, subnormalisation $S\|A\|_{\text{max}}$, and D data loading. That is the same as the block encoding discussed in [15].

Yet for the Toeplitz matrix, the circuit can be simplified to be more compact. The overflow qubit can be used directly as the del flag qubit; then only the modular part of the addition/subtraction must be uncomputed:



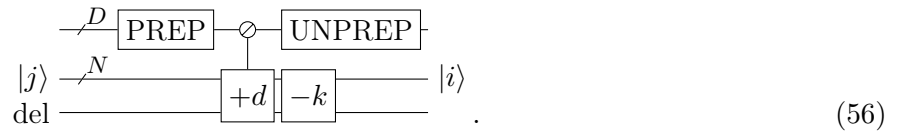
Then, the O_{rg} and O_r oracles can be merged, resulting in the simpler circuit



The circuit can also be interpreted as block-encoding a circulant matrix, and the del qubit selecting the top left block: Every Toeplitz matrix can be embedded in a larger circulant matrix. Such an observation was already used for HHL [38]. Further, the Fourier transformation of a circulant matrix is diagonal. Conceivably, one could construct a block encoding of the diagonal Fourier transformation. That approach is equivalent to compiling the additions in the above circuit with QFT based adders [35].

The arithmetics in the oracles only have $\propto \log N$ Toffoli gates. The flag qubit number is $2 + \log_2 D$ and we have D data loading. The Gilyén et. al. method would have more flag qubits, $3 + \log_2 N$.

Note that this matrix has $D \leq S$, and the row and column oracles commute with the data loading oracle. Hence, application of PREP/UNPREP can reduce the subnormalisation, resulting in the circuit



3.3 Tridiagonal matrix

We consider a matrix that is tridiagonal and symmetric, where entries along a diagonal are all different. For an $N \times N$ matrix (we consider N a power of 2), we have $D = 2N - 1$, $M = 2$, $S_c = S_r = S = 3$. A possible labelling in terms of (d, m) is

$$\begin{pmatrix} (d=0, m=0) & (d=1, m=0) & & & \\ (d=1, m=1) & (d=2, m=0) & (d=3, m=0) & & \\ & (d=3, m=1) & (d=4, m=0) & \ddots & \\ & & \ddots & \ddots & (d=2N-3, m=0) \\ & & & (d=2N-3, m=1) & (d=2N-2, m=0) \end{pmatrix}. \quad (57)$$

Since $SN = 3N \neq DM = 4N - 4$, we must pad S and/or D . The equality can be fulfilled by padding to $D = 2N$ and $S = 4$.

Arithmetic expressions for row and column indices can be written as

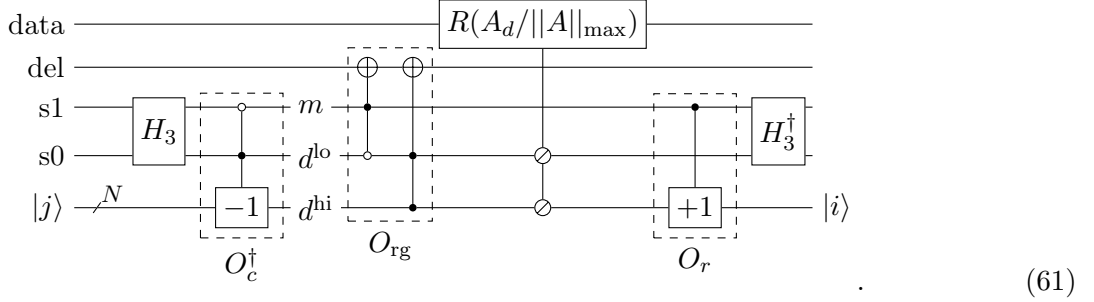
$$i(d, m) = \left\lfloor \frac{d}{2} \right\rfloor + m = d^{\text{hi}} + m \quad (58)$$

$$j(d, m) = \left\lfloor \frac{d}{2} \right\rfloor + \{d \bmod 2 \text{ if } m = 0\} = d^{\text{hi}} + \{d^{\text{lo}} \text{ if } m = 0\} \quad (59)$$

when splitting $d = 2d^{\text{hi}} + d^{\text{lo}}$ into its high $\log_2 N$ bits and lowest bit. Pairs (d, m) out-of-range are those with

$$(d^{\text{lo}} = 0 \text{ and } m = 1) \text{ or } d = 2N - 1. \quad (60)$$

Translating these expressions to quantum circuits results in the block encoding circuit



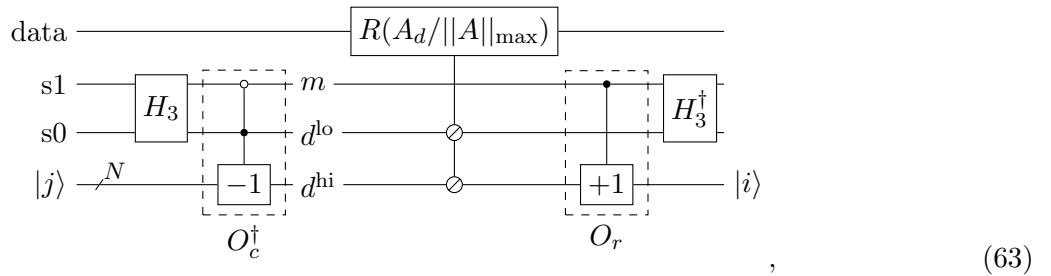
The equal superposition prepared by H_3 must be

$$H_3 |0\rangle^{s_1} |0\rangle^{s_0} = (|0\rangle^{s_1} |0\rangle^{s_0} + |0\rangle^{s_1} |1\rangle^{s_0} + |1\rangle^{s_1} |1\rangle^{s_0})/\sqrt{3}, \quad (62)$$

since those are the values of s_c and s_r that the oracles map valid (d, m) pairs to. Note that in this example, $S > D$ and the PREP/UNPREP method cannot be applied. Indeed, the data loading does not commute with the oracles.

Note the Toffoli count of the arithmetic oracles is logarithmic in N , and the addition and subtraction can be implemented without further ancilla qubits. The Toffoli cost is dominated by data loading of $D = 2N$ items. There are only 4 flag qubits. In the Gilyén et. al. scheme, there would be $3 + \log_2 N$ flag qubits.

The circuit can be simplified further by appreciating that the first CNOT in O_{rg} will never be triggered (it commutes with the column and row oracles and is annihilated by the H_3 of eq. (62)) and can be removed. Further, instead of the second multicontrolled NOT in O_{rg} , one could explicitly load a zero value for $d = 2N - 1$ with the multiplexed rotations. Then the block encoding does not need a delete flag qubit or O_{rg} oracle:



with $A_{2N-1} = 0$.

The above block encoding circuit is not Hermitian. We will demonstrate how to use our scheme to construct a Hermitian block encoding. The O_t oracle has the following action:

$$O_t(d, m) = \begin{cases} (d, m) & \text{for } d \text{ even} \\ (d, \text{NOT } m) & \text{for } d \text{ odd} \end{cases} \quad (64)$$

and can easily be implemented in a quantum circuit:



The oracle column oracle O_c is the same as above, and the row oracle is now constructed as $O_r = O_c O_t$. Putting this into the Hermitian version of the base block encoding circuit (36) gives

$$(66)$$

3.4 Extended binary tree matrix

The symmetric adjacency matrix of a balanced binary tree was considered in [15]. While the authors considered a non-Hermitian block encoding for $N = 8$, we will demonstrate how our scheme can be used to generate a Hermitian block encoding for N an arbitrary power of 2. For $N = 8$, the binary tree and its adjacency matrix are

$$\begin{pmatrix} A_0 & A_2 & & & & & & \\ A_2 & A_1 & A_2 & A_2 & & & & \\ & A_2 & A_1 & & A_2 & A_2 & & \\ & A_2 & & A_1 & & & A_2 & A_2 \\ & & A_2 & & A_0 & & & \\ & & & A_2 & & A_0 & & \\ & & & & A_2 & & A_0 & \\ & & & & & A_2 & & A_0 \end{pmatrix}. \quad (67)$$

The root and leaf nodes have weight A_0 , all other nodes weight A_1 , and the edges weight A_2 . The labelling of d is already exemplified in above matrix. One can see that $D = 3, M = 2N - 2, S = S_r = S_c = 4$. Hence, M is padded to $2N$, and S is padded to 6. Moreover, in this example D and the padded S are not powers of two, such that they are embedded in 2 and 3 qubits, respectively.

The labelling of m can be chosen as

$$\begin{pmatrix} 0 & 1 & & & & & & \\ 9 & 1 & 2 & 3 & & & & \\ & 10 & 2 & & 4 & 5 & & \\ & 11 & & 3 & & & 6 & 7 \\ & & 12 & & 4 & & & \\ & & 13 & & & 5 & & \\ & & & 14 & & & 6 & \\ & & & 15 & & & & 7 \end{pmatrix}, \quad (68)$$

which generalises to the following relations:

$$i(d, m) = \begin{cases} m & \text{for } d = 0, 1 \\ \lfloor \frac{m}{2} \rfloor & \text{for } d = 2 \text{ and } m < N \\ m - N & \text{for } d = 2 \text{ and } m > N \end{cases} \quad (69)$$

$$m'(d, m) = \begin{cases} m & \text{for } d = 0, 1 \\ m + N & \text{for } d = 2 \text{ and } m < N \\ m - N & \text{for } d = 2 \text{ and } m > N \end{cases} \quad (70)$$

with out-of-range (d, m) those with $(d = 0, m \geq N)$ or $(d = 0, 1 \leq m < N/2)$ or $(d = 1, m = 0)$ or $(d = 1, m \geq N/2)$ or $(d = 2, m = 0, N)$ or $d = 3$.

The oracle O_t can be easily implemented by flipping one qubit because we are assuming that N is

a power of 2.

$$\begin{array}{c}
 d^{\text{hi}} \\
 d^{\text{lo}} \\
 m^{\text{hi}} \\
 m^{\text{mid}} \\
 m^{\text{lo}} \quad N/2
 \end{array}
 \xrightarrow{O_t}
 \begin{array}{c}
 -d^{\text{hi}} \\
 -d^{\text{lo}} \\
 -m^{\text{hi}} \\
 -m^{\text{mid}} \\
 -m^{\text{lo}}
 \end{array}
 \quad (71)$$

When implementing the oracle O_r , we must make sure to only get $S_r = 3$ values of s_r for valid (d, m) .

$$\begin{array}{c}
 d^{\text{hi}} \\
 d^{\text{lo}} \\
 m^{\text{hi}} \\
 m^{\text{mid}} \\
 m^{\text{lo}} \quad N/2
 \end{array}
 \xrightarrow{O_r}
 \begin{array}{c}
 -d^{\text{hi}} \\
 -d^{\text{lo}} \\
 -m^{\text{hi}} \\
 -m^{\text{mid}} \\
 -m^{\text{lo}}
 \end{array}
 \quad (72)$$

The first X together with the two cnots ensure that d^{lo} is set to zero for all entries on the diagonal. As indicated in the circuit, that qubit is then $|0\rangle$ for all valid inputs (d, m) . It is then used as a control qubit for a rotation implementing $\lfloor m/2 \rfloor$ from (69). The rotation consists of swaps circularly rotating the qubits down. Finally, the cnot and ccnot ensure that $0 \leq s_r < S_r = 4$, i.e. $s_r^{\text{hi}} = 0$ for all valid inputs. Specifically, the enumeration by s_r within rows implemented by the circuit is

$$\begin{pmatrix}
 0 & 3 & & & & \\
 1 & 0 & 2 & 3 & & \\
 & 1 & 0 & & 2 & 3 \\
 & 1 & & 0 & & 2 & 3 \\
 & & 1 & & 0 & & \\
 & & 1 & & & 0 & \\
 & & & 1 & & & 0 \\
 & & & 1 & & & 0
 \end{pmatrix}. \quad (73)$$

The out-of-range oracle O_{rg} is:

$$\begin{array}{c}
 \text{del} \\
 d^{\text{hi}} \\
 d^{\text{lo}} \\
 m^{\text{hi}} \\
 m^{\text{mid}} \\
 m^{\text{lo}} \quad N/2
 \end{array}
 \xrightarrow{O_{\text{rg}}}
 \begin{array}{c}
 -\text{del} \\
 -d^{\text{hi}} \\
 -d^{\text{lo}} \\
 -m^{\text{hi}} \\
 -m^{\text{mid}} \\
 -m^{\text{lo}}
 \end{array}
 \quad (74)$$

From these oracles, a Hermitian base and preamplified block encoding can be constructed with $S_r = 4$.

One cannot use the PREP/UNPREP scheme with above oracles as the multiplexed rotations do not commute with O_r . This has to do with the fact that in (31), $H_{S_r/D} = H_{4/3}$ does not exist. One can artificially increase S_r to 6 and then construct a PREP/UNPREP block encoding circuit. It would have subnormalisation $2(|A_0| + |A_1| + |A_2|)$.

3.5 2-dimensional Laplacian

Here, we construct an example block-encoding of a 2-dimensional discrete Laplacian operator on a grid. Using finite differences, the one-dimensional Laplacian operator can be expressed as

$$\Delta f(x_i) = \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1}))}{(\Delta x)^2}. \quad (75)$$

When the discrete values $f(x_i)$ are understood as the components of a vector, the Laplacian corresponds to a Toeplitz matrix with $-2/(\Delta x)^2$ on the diagonal, and $+1/(\Delta x)^2$ on the first two offdiagonals. The

block encoding scheme in 3.2 can be used. Now, let us consider a two-dimensional regular grid of size $N_x \times N_y$, with dimensions powers of 2 for simplicity as usual. The finite difference Laplacian becomes

$$\Delta f(x_a, y_b) = \frac{f(x_{a-1}, y_b) - 2f(x_a, y_b) + f(x_{a+1}, y_b)}{(\Delta x)^2} + \frac{f(x_a, y_{b-1}) - 2f(x_a, y_b) + f(x_a, y_{b+1})}{(\Delta y)^2}. \quad (76)$$

A standard encoding of the values on the grid into an $N = N_x N_y$ -dimensional vector is row by row:

$$f_{a+bN_x} = f(x_a, y_b), \quad a \in 0, \dots, N_x - 1, \quad b \in 0, \dots, N_y - 1 \quad (77)$$

Then, the Laplacian matrix A is defined by

$$A_{a_1+b_1N_x, a_2+b_2N_x} = \begin{cases} A_0 := -2(1/(\Delta x)^2 + 1/(\Delta y)^2) & \text{for } a_1 = a_2, \quad b_1 = b_2 \\ A_1 := 1/(\Delta x)^2 & \text{for } |a_1 - a_2| = 1, \quad b_1 = b_2 \\ A_2 := 1/(\Delta y)^2 & \text{for } |b_1 - b_2| = 1, \quad a_1 = a_2 \\ 0 & \text{else} \end{cases} \quad (78)$$

We have 3 different values, which is padded to $D = 4$. Visually, for $N_x = 4, N_y = 4$, the matrix looks like

$$\begin{pmatrix} \begin{array}{cccc|cccc} A_0 & A_1 & & & & & & \\ A_1 & A_0 & A_1 & & & & & \\ & A_1 & A_0 & A_1 & & & & \\ & & A_1 & A_0 & & & & \\ & & & & A_2 & & & \\ A_2 & & & & A_0 & A_1 & & A_2 \\ & A_2 & & & A_1 & A_0 & A_1 & \\ & & A_2 & & A_1 & A_0 & A_1 & \\ & & & A_2 & & A_1 & A_0 & \\ & & & & A_2 & & & A_2 \\ & & & & & A_0 & A_1 & \\ & & & & & A_1 & A_0 & A_1 \\ & & & & & & A_1 & A_0 & A_1 \\ & & & & & & & A_2 \\ & & & & & & & A_0 & A_1 \\ & & & & & & & A_1 & A_0 & A_1 \\ & & & & & & & & A_1 & A_0 \end{array} \end{pmatrix} \quad (79)$$

where there are N_y dashed rectangles of dimension $N_x \times N_x$ each. We have maximum sparsity 5 padded to $S = 8$, and maximum multiplicity $2N - 2$ padded to $M = 2N$, such that $NS = DM = 8N$. For the labelling by m , we separate the high bit of m^{hi} from the low $\log_2 N$ bits m^{lo} . We choose $m^{\text{hi}} = 0$ for the lower left triangular matrix (including the diagonal), and $m^{\text{hi}} = 1$ for the upper triangular matrix. We choose m^{lo} to be the row index (in the lower triangular matrix) or the column index (in the upper triangular matrix).

The Hermitian block encoding scheme for symmetric matrices (section 2.4) requires a transposition oracle $O_t(d, m) = (d, m')$ that gives the corresponding label of the transposed element. We have

$$O_t(d, m^{\text{hi}}, m^{\text{lo}}) = \begin{cases} (d, m^{\text{hi}}, m^{\text{lo}}) & \text{for } d = 0 \\ (d, 1 - m^{\text{hi}}, m^{\text{lo}}) & \text{for } d = 1 \text{ or } d = 2 \end{cases} \quad (80)$$

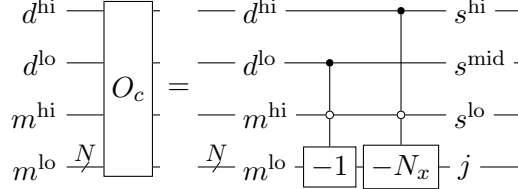
which in quantum circuit form is

$$\begin{array}{c} d^{\text{hi}} \\ d^{\text{lo}} \\ m^{\text{hi}} \\ m^{\text{lo}} \end{array} \xrightarrow{N} \boxed{O_t} = \begin{array}{c} \text{---} \bullet \text{---} \\ \text{---} \bullet \text{---} \\ \oplus \oplus \\ \text{---} \end{array} \quad (81)$$

Next, we need a column oracle $O_c : (d, m) \rightarrow j$ that gives the column index.

$$O_c(d, m^{\text{hi}}, m^{\text{lo}}) = \begin{cases} m^{\text{lo}} & \text{for } d = 0 \\ m^{\text{lo}} & \text{for } m^{\text{hi}} = 1 \\ m^{\text{lo}} - 1 & \text{for } m^{\text{hi}} = 0 \text{ and } d = 1 \\ m^{\text{lo}} - N_x & \text{for } m^{\text{hi}} = 0 \text{ and } d = 2 \end{cases} \quad (82)$$

In quantum circuit form this is



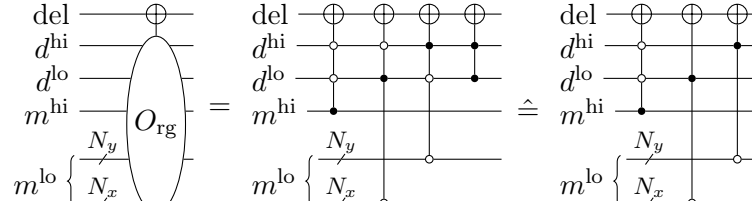
$$= \quad (83)$$

and the $|s\rangle$ states required for H_S are $H_S = \frac{1}{\sqrt{5}}(|000\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle)$ (reading s^{hi} to s^{lo} left to right).

The out-of-range indices (d, m) are the following:

$$(d, m^{\text{hi}}, m^{\text{lo}}) \text{ out-of-range for } \begin{cases} d = 0, m^{\text{hi}} = 1 \\ d = 1, (m^{\text{lo}} \bmod N_x) = 0 \\ d = 2, m^{\text{lo}} < N_x \\ d = 3 \end{cases} \quad (84)$$

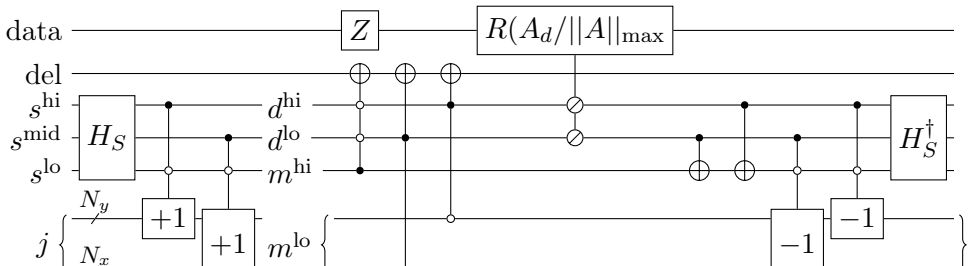
To write this in quantum circuit form, we split the m^{lo} register into two registers of $\log_2 N_y$ and $\log_2 N_x$ bits; this respects the structure in (77), and makes it easy to implement the conditions $(m^{\text{lo}} \bmod N_x) = 0$ and $m^{\text{lo}} < N_x$. (Splitting j into those two registers also simplifies the $-N_x$ in O_c , which is just a -1 on the $\log_2 N_y$ -bit register.) The out-of-range oracle is



$$= \quad (85)$$

When putting together the circuit, we can use the simplified form on the right because certain controls are never triggered.

The full Hermitian block encoding circuit (36) for these oracles is then:



$$= \quad (86)$$

The subnormalisation is

$$\alpha = 5 \max(|A_0|, |A_1|, |A_2|) = 5|A_0| = 10 \left(\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} \right). \quad (87)$$

Since the data loading and row/column oracles commute, we can use the PREP/UNPREP scheme to improve the subnormalisation to (in the symmetric $p = 1/2$ case)

$$|A_0| + |A_1| + |A_2| = 3 \left(\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} \right). \quad (88)$$

Block encodings of a 2-dimensional Laplacian for use in quantum algorithms were already considered in [39], using an approach that constructs an approximate block encoding. Here, in contrast, the block encoding is exact (up to finite accuracy in the data loading oracle). Moreover, the block encoding constructed with our scheme requires $O(\log N + \log N_y)$ gates, which comes from the additions in (86). Whereas, the approximate block encoding in [39] appears to have an exponentially worse scaling. Our method does well, because treating it solely as a sparse [7] or dense matrix [10, 11] does not harness the repeated elements, and it is not of the class considered in [15].

4 Conclusions and Outlook

In this work, we have presented a number of schemes to block encode structured matrices (base, preamplified, PREP/UNPREP schemes, and Hermitian extensions). Such a block encoding is necessary to use a matrix in QSVT and related quantum algorithms. All the schemes are based on a labelling of the matrix structure in terms of (d, m) , where d labels distinct non-zero values and m distinguishes different elements with the same value. Arithmetic quantum circuits relating this labelling to the column and row indices constitute the core of the quantum circuits. Section 2 shows our circuit constructions based on these arithmetic oracles along with a data loading oracle.

All schemes fully incorporate the sparsity of the matrix, reflected in the flag qubit number, as well as repeated values: The data loading oracle is controlled on d , such that the data loading cost corresponds to the number of distinct values; no value is loaded twice, even if it appears in the structured matrix multiple times. The schemes differ in the subnormalisation achieved; our detailed analysis is summarised in table 1. Which scheme performs best depends on the specific matrix in question. Further, our block encodings can be adapted to be Hermitian in the case of symmetric matrices (section 2.4) without extra data loading cost or subnormalisation.

In section 3 we have provided examples showing our block-encoded schemes in action for several families of structured matrices (Toeplitz matrix, tridiagonal matrix, extended binary tree matrix, 2d Laplacian matrix). Together with the information provided in the appendix on data loading, the full circuits can be elaborated. We hope that, beyond the examples considered here, our schemes will prove useful for constructing block encodings for a large variety of matrix families from different application areas.

A theoretical bound for the subnormalisation is the spectral norm $\|A\|_{op}$, due to the requirement that the block encoding be unitary. None of the schemes achieve this for an arbitrary matrix, so improvements may still be possible. While this article focuses on real matrices, an extension to complex valued matrices is straightforward: One could sum block encodings of a matrices real and imaginary parts, or adapt the multiplexed rotations to yield complex amplitudes of the $|0\rangle$ state. We have assumed a structure in the pattern and position of matrix elements, but not in the values. If the matrix possesses further structure, we expect more efficient circuits can be found. For example, if the values depend arithmetically on any of the indices i, j, d, m , data loading may not be necessary. Further, the block encodings constructed are exact, up to finite accuracy in the data loading (see appendix A) and, in the preamplification scheme, the accuracy of singular value amplification (see appendix B). Possibly, approximate block encodings [39, 14] could be implemented with more efficient circuits.

Acknowledgements

We thank Leigh Lapworth of Rolls-Royce plc for inspiration of matrix structures. Work partially funded by the UK's Commercialising Quantum Technologies Programme (Grant reference 10004857).

Bibliography

- [1] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge ; New York, 10th anniversary ed edition, 2010. ISBN 978-1-107-00217-3.
- [2] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerín, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779), October 2019. ISSN 1476-4687. DOI: [10.1038/s41586-019-1666-5](https://doi.org/10.1038/s41586-019-1666-5). URL <https://www.nature.com/articles/s41586-019-1666-5>.
- [3] IBM. IBM Unveils Breakthrough 127-Qubit Quantum Processor, 2021. URL <https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor>.
- [4] Yulin Wu, Wan-Su Bao, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong, Cheng Guo, Chu Guo, Shaojun Guo, Lianchen Han, Linyin Hong, He-Liang Huang, Yong-Heng Huo, Liping Li, Na Li, Shaowei Li, Yuan Li, Futian Liang, Chun Lin, Jin Lin, Haoran Qian, Dan Qiao, Hao Rong, Hong Su, Lihua Sun, Liangyuan Wang, Shiyu Wang, Dachao Wu, Yu Xu, Kai Yan, Weifeng Yang, Yang Yang, Yangsen Ye, Jianghan Yin, Chong Ying, Jiale Yu, Chen Zha, Cha Zhang, Haibin Zhang, Kaili Zhang, Yiming Zhang, Han Zhao, Youwei Zhao, Liang Zhou, Qingling Zhu, Chao-Yang Lu, Cheng-Zhi Peng, Xiaobo Zhu, and Jian-Wei Pan. Strong quantum computational advantage using a superconducting quantum processor. *Physical Review Letters*, 127(18):180501, October 2021. ISSN 0031-9007, 1079-7114. DOI: [10.1103/PhysRevLett.127.180501](https://doi.org/10.1103/PhysRevLett.127.180501). URL <http://arxiv.org/abs/2106.14734>. arXiv:2106.14734 [quant-ph].
- [5] Scott Aaronson. How Much Structure Is Needed for Huge Quantum Speedups?, September 2022. URL <http://arxiv.org/abs/2209.06930>. arXiv:2209.06930 [quant-ph].
- [6] Seunghoon Lee, Joonho Lee, Huanchen Zhai, Yu Tong, Alexander M. Dalzell, Ashutosh Kumar, Phillip Helms, Johnnie Gray, Zhi-Hao Cui, Wenyuan Liu, Michael Kastoryano, Ryan Babbush, John Preskill, David R. Reichman, Earl T. Campbell, Edward F. Valeev, Lin Lin, and Garnet Kin-Lic Chan. Is there evidence for exponential quantum advantage in quantum chemistry?, November 2022. URL <http://arxiv.org/abs/2208.02199>. arXiv:2208.02199 [physics, physics:quant-ph].
- [7] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 193–204, June 2019. DOI: [10.1145/3313276.3316366](https://doi.org/10.1145/3313276.3316366). URL <http://arxiv.org/abs/1806.01838>. arXiv: 1806.01838.
- [8] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. Grand Unification of Quantum Algorithms. *PRX Quantum*, 2(4):040203, December 2021. DOI: [10.1103/PRXQuantum.2.040203](https://doi.org/10.1103/PRXQuantum.2.040203). URL <https://link.aps.org/doi/10.1103/PRXQuantum.2.040203>. Publisher: American Physical Society.
- [9] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4), April 2015. ISSN 1745-2481. DOI: [10.1038/nphys3272](https://doi.org/10.1038/nphys3272). URL <https://www.nature.com/articles/nphys3272>.
- [10] B. David Clader, Alexander M. Dalzell, Nikitas Stamatopoulos, Grant Salton, Mario Berta, and William J. Zeng. Quantum Resources Required to Block-Encode a Matrix of Classical Data. *arXiv*, June 2022. URL <http://arxiv.org/abs/2206.03505>. arXiv:2206.03505 [quant-ph].

- [11] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation. *arXiv:1804.01973 [quant-ph]*, page 14 pages, 2019. DOI: [10.4230/LIPIcs.ICALP.2019.33](https://doi.org/10.4230/LIPIcs.ICALP.2019.33). URL <http://arxiv.org/abs/1804.01973>. arXiv: 1804.01973.
- [12] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16):160501, April 2008. ISSN 0031-9007, 1079-7114. DOI: [10.1103/PhysRevLett.100.160501](https://doi.org/10.1103/PhysRevLett.100.160501). URL <http://arxiv.org/abs/0708.1879>. arXiv:0708.1879 [quant-ph].
- [13] Connor T. Hann, Gideon Lee, S. M. Girvin, and Liang Jiang. Resilience of quantum random access memory to generic noise. *PRX Quantum*, 2(2):020311, April 2021. ISSN 2691-3399. DOI: [10.1103/PRXQuantum.2.020311](https://doi.org/10.1103/PRXQuantum.2.020311). URL <http://arxiv.org/abs/2012.05340>. arXiv:2012.05340 [quant-ph].
- [14] Quynh T. Nguyen, Bobak T. Kiani, and Seth Lloyd. Block-encoding dense and full-rank kernels using hierarchical matrices: applications in quantum numerical linear algebra. *Quantum*, 6: 876, December 2022. DOI: [10.22331/q-2022-12-13-876](https://doi.org/10.22331/q-2022-12-13-876). URL <https://quantum-journal.org/papers/q-2022-12-13-876/>. Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften.
- [15] Daan Camps, Lin Lin, Roel Van Beeumen, and Chao Yang. Explicit Quantum Circuits for Block Encodings of Certain Sparse Matrices. *arXiv:2203.10236 [quant-ph]*, March 2022. URL <http://arxiv.org/abs/2203.10236>. arXiv: 2203.10236.
- [16] Guang Hao Low and Isaac L. Chuang. Hamiltonian Simulation by Qubitization. *Quantum*, 3:163, July 2019. ISSN 2521-327X. DOI: [10.22331/q-2019-07-12-163](https://doi.org/10.22331/q-2019-07-12-163). URL <http://arxiv.org/abs/1610.06546>. arXiv: 1610.06546.
- [17] Ryan Babbush, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity. *Physical Review X*, 8(4):041015, October 2018. DOI: [10.1103/PhysRevX.8.041015](https://doi.org/10.1103/PhysRevX.8.041015). URL <https://link.aps.org/doi/10.1103/PhysRevX.8.041015>. Publisher: American Physical Society.
- [18] Dominic W. Berry, Craig Gidney, Mario Motta, Jarrod R. McClean, and Ryan Babbush. Qubitization of Arbitrary Basis Quantum Chemistry Leveraging Sparsity and Low Rank Factorization. *Quantum*, 3:208, December 2019. ISSN 2521-327X. DOI: [10.22331/q-2019-12-02-208](https://doi.org/10.22331/q-2019-12-02-208). URL <http://arxiv.org/abs/1902.02134>. arXiv:1902.02134 [physics, physics:quant-ph].
- [19] Joonho Lee, Dominic W. Berry, Craig Gidney, William J. Huggins, Jarrod R. McClean, Nathan Wiebe, and Ryan Babbush. Even more efficient quantum computations of chemistry through tensor hypercontraction. *PRX Quantum*, 2(3):030305, July 2021. ISSN 2691-3399. DOI: [10.1103/PRXQuantum.2.030305](https://doi.org/10.1103/PRXQuantum.2.030305). URL <http://arxiv.org/abs/2011.03494>. arXiv: 2011.03494.
- [20] Aleksei V. Ivanov, Christoph Sünderhauf, Nicole Holzmann, Tom Ellaby, Rachel N. Kerber, Glenn Jones, and Joan Camps. Quantum Computation for Periodic Solids in Second Quantization, October 2022. URL <http://arxiv.org/abs/2210.02403>. arXiv:2210.02403 [quant-ph].
- [21] M. Szegedy. Quantum speed-up of Markov chain based algorithms. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–41, October 2004. DOI: [10.1109/FOCS.2004.53](https://doi.org/10.1109/FOCS.2004.53). ISSN: 0272-5428.
- [22] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 792–809, October 2015. DOI: [10.1109/FOCS.2015.54](https://doi.org/10.1109/FOCS.2015.54). URL <http://arxiv.org/abs/1501.01715>. arXiv:1501.01715 [quant-ph].
- [23] Yuta Kikuchi, Conor Mc Keever, Luuk Coopmans, Michael Lubasch, and Marcello Benedetti. Realization of quantum signal processing on a noisy quantum computer. *npj Quantum Information*, 9(1), September 2023. ISSN 2056-6387. DOI: [10.1038/s41534-023-00762-0](https://doi.org/10.1038/s41534-023-00762-0). URL <http://dx.doi.org/10.1038/s41534-023-00762-0>.
- [24] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52(4):R2493–R2496, October 1995. ISSN 1050-2947, 1094-1622. DOI: [10.1103/PhysRevA.52.R2493](https://doi.org/10.1103/PhysRevA.52.R2493). URL <https://link.aps.org/doi/10.1103/PhysRevA.52.R2493>.
- [25] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface

- codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, September 2012. DOI: [10.1103/PhysRevA.86.032324](https://doi.org/10.1103/PhysRevA.86.032324). URL <https://link.aps.org/doi/10.1103/PhysRevA.86.032324>. Publisher: American Physical Society.
- [26] Sergei Bravyi and Alexei Kitaev. Universal Quantum Computation with ideal Clifford gates and noisy ancillas. *arXiv:quant-ph/0403025*, December 2004. DOI: [10.1103/PhysRevA.71.022316](https://doi.org/10.1103/PhysRevA.71.022316). URL <http://arxiv.org/abs/quant-ph/0403025>. arXiv: quant-ph/0403025.
- [27] Joe O’Gorman and Earl T. Campbell. Quantum computation with realistic magic state factories. *Physical Review A*, 95(3):032338, March 2017. ISSN 2469-9926, 2469-9934. DOI: [10.1103/PhysRevA.95.032338](https://doi.org/10.1103/PhysRevA.95.032338). URL <http://arxiv.org/abs/1605.07197>. arXiv:1605.07197 [quant-ph].
- [28] Earl T. Campbell, Barbara M. Terhal, and Christophe Vuillot. Roads towards fault-tolerant universal quantum computation. *Nature*, 549(7671):172–179, September 2017. ISSN 0028-0836, 1476-4687. DOI: [10.1038/nature23460](https://doi.org/10.1038/nature23460). URL <http://arxiv.org/abs/1612.07330>. arXiv: 1612.07330.
- [29] Austin G. Fowler and Craig Gidney. Low overhead quantum computation using lattice surgery. *arXiv:1808.06709 [quant-ph]*, August 2019. URL <http://arxiv.org/abs/1808.06709>. arXiv: 1808.06709.
- [30] Nick S. Blunt, Joan Camps, Ophelia Crawford, Róbert Izsák, Sebastian Leontica, Arjun Mirani, Alexandra E. Moylett, Sam A. Scivier, Christoph Sünderhauf, Patrick Schopf, Jacob M. Taylor, and Nicole Holzmann. Perspective on the Current State-of-the-Art of Quantum Computing for Drug Discovery Applications. *Journal of Chemical Theory and Computation*, 18(12):7001–7023, December 2022. ISSN 1549-9618. DOI: [10.1021/acs.jctc.2c00574](https://doi.org/10.1021/acs.jctc.2c00574). URL <https://doi.org/10.1021/acs.jctc.2c00574>. Publisher: American Chemical Society.
- [31] Craig Gidney. Halving the cost of quantum addition. *Quantum*, 2:74, June 2018. DOI: [10.22331/q-2018-06-18-74](https://doi.org/10.22331/q-2018-06-18-74). URL <https://quantum-journal.org/papers/q-2018-06-18-74/>. Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften.
- [32] Yuval R. Sanders, Dominic W. Berry, Pedro C.S. Costa, Louis W. Tessler, Nathan Wiebe, Craig Gidney, Hartmut Neven, and Ryan Babbush. Compilation of Fault-Tolerant Quantum Heuristics for Combinatorial Optimization. *PRX Quantum*, 1(2):020312, November 2020. DOI: [10.1103/PRXQuantum.1.020312](https://doi.org/10.1103/PRXQuantum.1.020312). URL <https://link.aps.org/doi/10.1103/PRXQuantum.1.020312>. Publisher: American Physical Society.
- [33] Guang Hao Low, Vadym Kliuchnikov, and Luke Schaeffer. Trading T-gates for dirty qubits in state preparation and unitary synthesis, December 2018. URL <http://arxiv.org/abs/1812.00954>. arXiv:1812.00954 [quant-ph] type: article.
- [34] D.K. Callebaut. Generalization of the cauchy-schwarz inequality. *Journal of Mathematical Analysis and Applications*, 12(3):491–494, 1965. ISSN 0022-247X. DOI: [https://doi.org/10.1016/0022-247X\(65\)90016-8](https://doi.org/10.1016/0022-247X(65)90016-8). URL <https://www.sciencedirect.com/science/article/pii/0022247X65900168>.
- [35] Thomas G. Draper. Addition on a Quantum Computer. *arXiv:quant-ph/0008033*, August 2000. URL <http://arxiv.org/abs/quant-ph/0008033>. arXiv: quant-ph/0008033.
- [36] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv:quant-ph/0410184*, October 2004. URL <http://arxiv.org/abs/quant-ph/0410184>. arXiv: quant-ph/0410184.
- [37] Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin. Quantum arithmetic with the Quantum Fourier Transform. *Quantum Information Processing*, 16(6):152, June 2017. ISSN 1570-0755, 1573-1332. DOI: [10.1007/s11128-017-1603-1](https://doi.org/10.1007/s11128-017-1603-1). URL <http://arxiv.org/abs/1411.5949>. arXiv:1411.5949 [quant-ph].
- [38] A. Mahasinghe and J. B. Wang. Efficient quantum circuits for Toeplitz and Hankel matrices. *Journal of Physics A: Mathematical and Theoretical*, 49(27):275301, July 2016. ISSN 1751-8113, 1751-8121. DOI: [10.1088/1751-8113/49/27/275301](https://doi.org/10.1088/1751-8113/49/27/275301). URL <http://arxiv.org/abs/1605.07710>. arXiv:1605.07710 [quant-ph].
- [39] Daan Camps and Roel Van Beeumen. FABLE: Fast Approximate Quantum Circuits for Block-Encodings. April 2022. URL <http://arxiv.org/abs/2205.00081>. arXiv:2205.00081 [quant-ph].
- [40] Mikko Mottonen, Juha J. Vartiainen, Ville Bergholm, and Martti M. Salomaa. Quantum Circuits for General Multiqubit Gates. *Physical Review Letters*, 93(13):130502, September 2004.

- ISSN 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.93.130502. URL <http://arxiv.org/abs/quant-ph/0404089>. arXiv:quant-ph/0404089.
- [41] Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov. Synthesis of Quantum Logic Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, June 2006. ISSN 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2005.855930. URL <http://arxiv.org/abs/quant-ph/0406176>. arXiv:quant-ph/0406176.
- [42] Neil J. Ross and Peter Selinger. Optimal ancilla-free Clifford+T approximation of z-rotations, June 2016. URL <http://arxiv.org/abs/1403.2975>. arXiv:1403.2975 [quant-ph].
- [43] Vera von Burg, Guang Hao Low, Thomas Häner, Damian S. Steiger, Markus Reiher, Martin Roetteler, and Matthias Troyer. Quantum computing enhanced computational catalysis. *Physical Review Research*, 3(3), July 2021. ISSN 2643-1564. DOI: 10.1103/PhysRevResearch.3.033055. URL <http://arxiv.org/abs/2007.14460>. arXiv:2007.14460 [physics, physics:quant-ph].
- [44] Guang Hao Low. Halving the cost of quantum multiplexed rotations. *arXiv:2110.13439 [quant-ph]*, October 2021. URL <http://arxiv.org/abs/2110.13439>. arXiv: 2110.13439.
- [45] Guang Hao Low and Isaac L. Chuang. Hamiltonian Simulation by Uniform Spectral Amplification, July 2017. URL <http://arxiv.org/abs/1707.05391>. arXiv:1707.05391 [quant-ph].
- [46] Yulong Dong, Xiang Meng, K. Birgitta Whaley, and Lin Lin. Efficient phase-factor evaluation in quantum signal processing. *arXiv:2002.11649 [physics, physics:quant-ph]*, July 2021. DOI: 10.1103/PhysRevA.103.042419. URL <http://arxiv.org/abs/2002.11649>. arXiv: 2002.11649.

A Implementation of data loading step

An important component in all of the block encodings is data loading, in which the values of matrix elements are loaded into the matrix structure supplied by the other, arithmetic, oracles. In fact, we use the data loading cost (number of data items) as a stand-in for the complexity of a block encoding circuit. In this appendix, we will give details on the data loading step.

A.1 Multiplexed rotations

In the base and preamplified schemes (sections 2.1 and 2.2), data loading is accomplished with a data loading oracle O_{data} implementing multiplexed rotations (7):

$$O_{\text{data}} = \sum_{d=0}^{D-1} R_X(2 \arccos A_d / \|A\|_{\max}) \otimes |d\rangle\langle d|. \quad (89)$$

It has been shown [40, 41] how such rotations multiplexed on D values can be decomposed into D uncontrolled rotation and D CNOT gates (for D a power of 2), without use of any ancillas. The T gate cost is then that of synthesizing D arbitrary rotations into Clifford and T gates up to the required accuracy ε . Such synthesis requires $O(\log(1/\varepsilon))$ T gates [42], such that the total T gate count is the product

$$D \cdot O(\log(1/\varepsilon)). \quad (90)$$

The multiplicative factor $O(\log(1/\varepsilon))$ can be changed to additive, reducing overall T count. To this end, the (approximate) bitvalues of all of the required angles are first loaded into ancilla qubits, and rotations then performed according to the loaded bitvalues. Accordingly, this implementation of O_{data} consists of 3 steps [43, 44, 33]:

$$|0\rangle |d\rangle |0\rangle^{\otimes b} \xrightarrow{(1)} |0\rangle |d\rangle |\alpha_d\rangle \xrightarrow{(2)} (R_X(\alpha_d) |0\rangle) |d\rangle |\alpha_d\rangle \xrightarrow{(3)} (R_X(\alpha_d) |0\rangle) |d\rangle |0\rangle^{\otimes b}, \quad (91)$$

with $\alpha_d = 2 \arccos A_d / \|A\|_{\max}$.

Step (1) loads a binary representation of α_d into the b -qubit ancilla register ($b \sim \log_2 1/\varepsilon$). Using additional $\lceil \log_2 D \rceil$ ancilla qubits, this can be done with $D - 2$ Toffoli gates with the QROM presented in [17]. Note that for QROM, D needn't be a power of 2. If further $\sim \sqrt{D}$ ancilla qubits are available, the select-swap technique [33] (also dubbed QROAM [18]) can be used to lower Toffoli count to $\sim \sqrt{D}$.

Next, step (2) rotates the data qubit according to the bitvalue saved in the ancilla register. This can be done with the phase-gradient technique [43, 44, 33], in which the b -bit angle register is added into a reusable phase gradient state. This addition requires $b - 1$ Toffolis. Alternatively, step (2) can be implemented with b rotations controlled on each of the qubits in $|\alpha_d\rangle$ in turn [33].

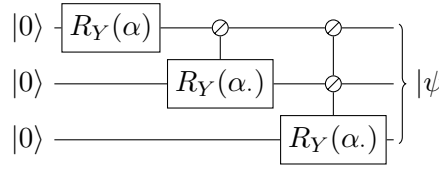
Finally, in step (3), the data lookup must be uncomputed to return the ancillas to $|0\rangle^{\otimes b}$. In some cases, the Toffoli cost of uncomputing the lookup can be reduced compared to step (1) by using measurement based uncomputation [17].

The cost of the three-step procedure is asymptotically reduced from (90) because steps (1), (2), and (3) are in sequence and the term related to the accuracy of the rotations is added rather than multiplied to the term D related to the data look-up. In either case, the T cost scales with D as $O(D)$ when using QROM; the number of data items to load is a sensible stand-in for circuit length (T count) while staying agnostic about the exact procedure.

A.2 State preparation

In the PREP/UNPREP scheme (section 2.3), the data loading oracle O_{data} and diffusion operator H_S are merged into a state preparation operator, reducing the subnormalisation. The data loading is not performed with multiplexed rotations O_{data} on a data flag qubit like in the base and preamplified scheme. Instead, data is loaded as the amplitudes of a state that is prepared with a PREP operator.

An arbitrary quantum state $|\psi\rangle = \text{PREP}|0\rangle^{\otimes \log_2 D}$ can be implemented by a sequence of multiplexed rotations, whose angles α have been precomputed from ψ 's amplitudes [41, 33], like in this example with $D = 8$:



$$(92)$$

These multiplexed rotations can be implemented as in appendix A.1. Alternative approaches to preparing $|\psi\rangle$ include coherent alias sampling [17] and prerotation [10].

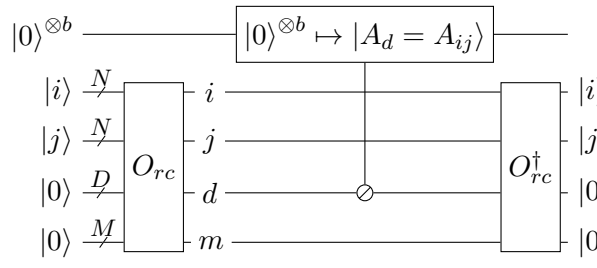
A.3 Data loading oracle in Gilyén et. al.'s sparse scheme

Lemma 48 in [7] shows a construction of a block encoding in terms of black-box oracles. One of the black boxes is the data loading oracle called O_A . Given the row and column indices i and j , it loads the b -bit bitstring A_{ij} of the corresponding matrix entry:

$$O_A |i\rangle |j\rangle |0\rangle^{\otimes b} = |i\rangle |j\rangle |A_{ij}\rangle. \quad (93)$$

The implementation of this oracle is not discussed.

From a first cursory look at the oracle, one might conclude that repeated entries must be loaded separately. However, given a structured matrix as considered in this paper, it can be implemented with only D data loading and the usual $O(\text{polylog } N)$ arithmetic overheads:



$$(94)$$

The arithmetic oracle O_{rc} computes (d, m) into ancilla qubits from (i, j) . Possibly, some of the arithmetics could be done in-place, and the i and j registers reused for d and m .

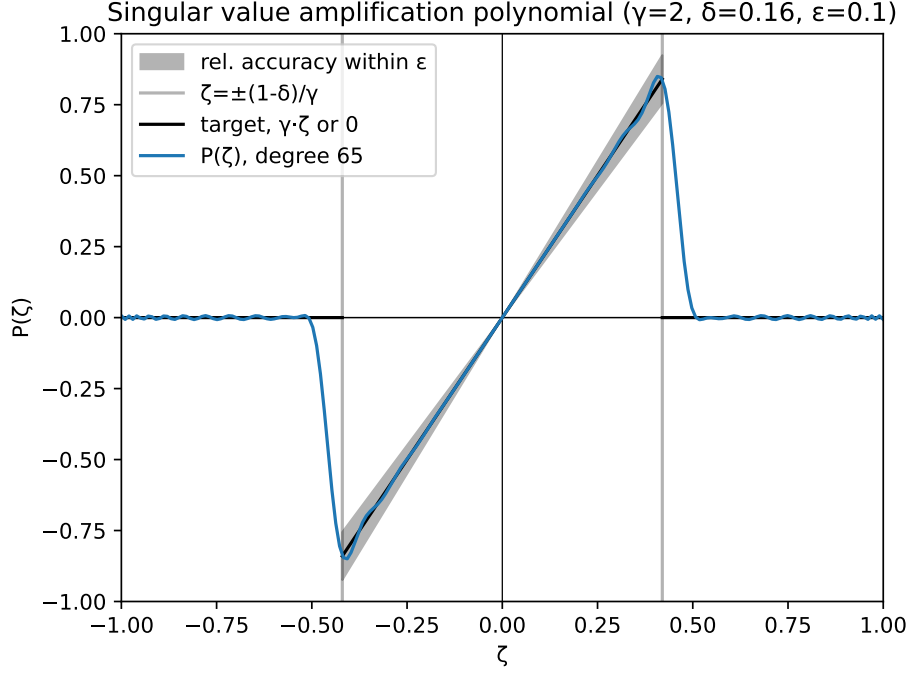


Figure 1: Singular value amplification. The target function for an amplification by γ is $\gamma\zeta$ (black line), and 0 outside of the validity region. For QSVT, it has to be approximated by a polynomial (blue) of sufficient accuracy. The region of accuracy (grey shaded) is determined by the parameters δ and ε . Asymptotically, the degree was shown to be $O(\gamma/\delta \log(\gamma/\varepsilon))$ [45]. We truncate a Chebyshev expansion of an analytic approximation to the target function to find the polynomial's degree.

B Singular value amplification

Singular value amplification [7, 45] allows to reduce the subnormalisation of a block encoding by increasing the circuit length. This is achieved by performing a QSVT that multiplies the singular values by an amplification factor γ . While the resulting block encoding (with one flag qubit more) will have a subnormalisation reduced by a factor of γ , the circuit length will have roughly increased by a factor of γ , up to logarithmic factors and constants. This reciprocal behaviour substantiates the figure of merit

$$\text{circuit cost} \cdot \text{subnormalisation}, \quad (95)$$

of which eq. (4) is a proxy, used to assess a block encoding.

As a crucial ingredient to the preamplified scheme (section 2.2), the complexity of singular value amplification including constant factors determines which block encoding scheme performs best for specific data. While previous work has determined its big- O scaling [7, 45], this appendix will shed light on the constant factors.

Theorem (Uniform singular value amplification (adapted from Theorem 20, [7])). *Let U be a block encoding, $\gamma > 1$, and $0 < \delta, \varepsilon < 1/2$, whose block-encoded matrix A has singular values $\zeta_i \in [0, (1 - \delta)/\gamma]$. Then there is an efficiently computable polynomial of degree*

$$O\left(\frac{\gamma}{\delta} \log \frac{\gamma}{\varepsilon}\right) \quad (96)$$

whose application to U by QSVT results in a block encoding \tilde{U} encoding \tilde{A} , an ε -approximation of γA . In particular, each amplified singular value $\tilde{\zeta}_i$ of \tilde{A} has relative accuracy ε : $|\tilde{\zeta}_i/(\gamma\zeta_i) - 1| \leq \varepsilon$.

Note that singular value amplification can be applied to general projected matrices beyond quadratic A flagged by $|0\rangle$ s. As such, it encompasses uniform amplitude amplification.

Figure 1 shows an example of a target function for amplification, along with a polynomial approximation for the QSVT. The parameter ε controls the relative accuracy of the singular value amplification, while δ controls the applicable range. The requirement $\zeta_i \leq 1/\gamma$ is natural because the singular values $\tilde{\zeta}_i \approx \gamma\zeta_i$ of the block-encoding \tilde{U} must be bounded by one. To ensure the polynomial approximation can be bounded by one across the entire range $[-1, +1]$ (a requirement for QSVT), the permissible range of singular values of A must be slightly lowered by δ . In principle, a trade-off between γ and δ is possible. For preamplification, Gilyén et. al. [7] choose an amplification factor of

$$\gamma = \frac{1}{\sqrt[4]{2}} \frac{1}{\zeta_{\max}} \approx 0.84 \frac{1}{\zeta_{\max}} \quad (\text{with } \zeta_{\max} = \max_i \zeta_i), \quad (97)$$

which leads to

$$\delta = 1 - \gamma\zeta_{\max} = 1 - \frac{1}{\sqrt[4]{2}} \approx 0.16. \quad (98)$$

The starting point for the polynomial $P(\zeta)$ is a sufficiently good analytic approximation of the rectangle function on the domain $[-(1 - \delta)/\gamma, (1 - \delta)/\gamma]$ based on error functions, see [7, 45]. Its expansion in Chebyshev polynomials is truncated such that an absolute accuracy of ε is achieved. The product with $\gamma\zeta$ then gives the desired polynomial approximation. Yet, the authors of [45] “emphasize that our proposed sequence of polynomial transformations serve primarily to prove their asymptotic scaling.” They suggest to obtain the constant factors by a direct Chebyshev truncation of the entire functions.

Here, we therefore numerically perform Chebyshev truncations of the entire approximation to the rectangle function². The optimal degree satisfying the required accuracy is found with a binary search for various parameter combinations of the amplification factor γ , the accuracy ε , and δ . Fig. 2 shows our results along with a fit to the big- O result (96). This study confirms the scaling behaviour (96) and determines a surprisingly small scaling factor of ≈ 3 . We thus find singular value amplification requires approximately

$$3 \frac{\gamma}{\delta} \log \frac{\gamma}{\varepsilon} \quad (99)$$

repetitions of the block encoding, where \log is a natural logarithm.

Note that this analysis follows the prescription from [45], where the polynomial is constructed from an approximate rectangle function multiplied by $\gamma\zeta$. Actually, the behaviour of the polynomial outside of the accuracy region does not matter, as long as it stays bounded by ± 1 . In particular, in Fig. 1 it does not need to decay to zero rapidly around $\zeta = \pm(1 - \delta)/\gamma$. Perhaps a lower degree polynomial exists that does not follow the analytic construction based on the rectangle function. Improved degrees to the analytic construction have already been achieved for other target functions (like for matrix inversion) in [46]. The authors used a numerical optimisation based approach based on the Remez method to directly find Chebyshev approximations.

²The Chebyshev truncation was performed with numpy’s `np.polynomial.Chebyshev.interpolate()` function, and the accuracy of the truncation checked on a grid of $10^3\gamma/(1 - \delta)$ points spanning $[-1, +1]$. Calculations were carried out with the default 64bit floating point accuracy.

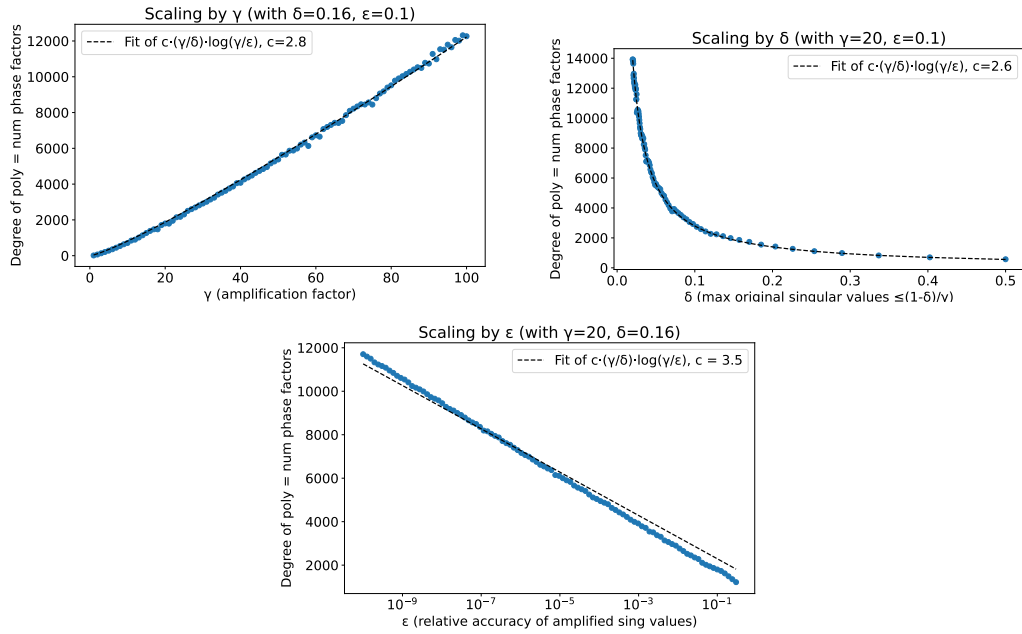


Figure 2: Degree of the polynomial for singular value amplification. Asymptotically, the degree was shown to be $O(\gamma/\delta \log(\gamma/\epsilon))$ [45]. We truncate a Chebyshev expansion of an analytic approximation to the target function to find the polynomials' exact degrees. By varying the amplification factor γ , the accuracy ϵ , and δ , we aim to find the constant factor in the asymptotic complexity. From the fits shown in the plots, we conclude the degree is approximately $3 \gamma/\delta \log(\gamma/\epsilon)$. Log refers to the natural logarithm.