

# Products between block-encodings

Dekuan Dong<sup>a</sup>, Yingzhou Li<sup>a,b</sup>, Jungong Xue<sup>a</sup>

<sup>a</sup>*School of Mathematical Science, Fudan University, China*

<sup>b</sup>*Shanghai Key Laboratory for Contemporary Applied Mathematics, China*

---

## Abstract

Block-encoding is a standard framework for embedding matrices into unitary operators in quantum algorithms. Efficient implementation of products between block-encoded matrices is crucial for applications such as Hamiltonian simulation and quantum linear algebra. We present resource-efficient methods for matrix-matrix, Kronecker, and Hadamard products between block-encodings that apply to rectangular matrices of arbitrary dimensions. Our constructions significantly reduce the number of ancilla qubits, achieving exponential qubit savings for sequences of matrix-matrix multiplications, with a moderate increase in gate complexity. These product operations also enable more complex block-encodings, including a compression gadget for time-dependent Hamiltonian simulation and matrices represented as sums of Kronecker products, each with improved resource requirements.

*Keywords:*

Block-encoding, Quantum circuit, Product, Permutation

---

## 1. Introduction

Quantum computing holds the potential to solve certain computational problems far more efficiently than classical algorithms. This promise is especially evident in areas dominated by linear algebraic operations, such as Hamiltonian simulation [1, 2], solving systems of linear equations [3, 4, 5], quantum machine learning [6], and quantum optimization [7, 8]. A central reason for this advantage is that quantum computers can naturally manipulate high-dimensional vector spaces, provided that the underlying linear operator can be accessed efficiently.

The block-encoding framework, formalized in [9], has emerged as a powerful method for embedding matrices into unitary operations, thereby enabling quantum algorithms to operate on non-unitary matrices. The block-encoding technique embeds a matrix  $\mathbf{A} \in \mathbb{C}^{2^n \times 2^n}$  into the top-left block of a larger unitary operator  $\mathbf{U}_{\mathbf{A}}$ , allowing quantum circuits to access and manipulate  $\mathbf{A}$  indirectly while preserving unitary property. Formally, a unitary  $\mathbf{U}_{\mathbf{A}} \in \mathbb{C}^{2^{n+a} \times 2^{n+a}}$  is called an  $(\alpha, a, \varepsilon)$ -block-encoding of  $\mathbf{A}$  if

$$\|\mathbf{A} - \alpha(|0^a\rangle\langle 0^a| \otimes \mathbf{I}) \mathbf{U}_{\mathbf{A}} (|0^a\rangle\langle 0^a| \otimes \mathbf{I})\| \leq \varepsilon,$$

where  $a$  ancilla qubits are initialized and postselected in the  $|0^a\rangle$  state. By representing a matrix as a block of a larger unitary, block-encoding provides a uniform language for quantum linear algebra, streamlining both the analysis and design of algorithms. Consequently, block-encoding has become a cornerstone technique in the development of efficient quantum algorithms. The technique of block-encoding enables efficient implementation of matrix functions [9], quantum linear system solvers [5], and Hamiltonian simulation techniques [2, 10]. Consequently, a growing body of work has focused on constructing block-encodings for individual matrices or matrix classes [11, 12, 13, 14, 15, 16, 17].

In many applications, however, the matrices of interest do not appear in isolation but rather as compositions of simpler matrices. For instance, Dyson-series based Hamiltonian simulation requires ordered products of Hamiltonian terms [2]; quantum differential equation algorithms often involve structured matrices with Kronecker product forms [18, 19, 20]. Thus, the task of constructing block-encodings for compositions of matrices arises frequently and is indispensable for a broad range of quantum algorithms. Several foundational matrix operations have been shown to be efficiently implementable between block-encodings:

- **Matrix-matrix multiplication:** Given an  $(\alpha, a, \varepsilon)$ -block-encoding of  $\mathbf{A}$  and a  $(\beta, b, \delta)$ -block-encoding of  $\mathbf{B}$ , one can construct an  $(\alpha\beta, a + b, \alpha\varepsilon + \beta\delta)$ -block-encoding of  $\mathbf{AB}$  as described in [9].
- **Kronecker product:** If  $\mathbf{A}$  and  $\mathbf{B}$  are block-encoded, their Kronecker product  $\mathbf{A} \otimes \mathbf{B}$  can be block-encoded by taking the tensor product of the two block-encodings and projecting onto the combined ancilla state [21].
- **Matrix Hadamard product:** Since the entries required for the Hadamard product are contained within the Kronecker product, it can be derived by first constructing the Kronecker product and then applying a proper permutation to reposition the relevant elements [22].
- **Linear combinations:** Using the Linear Combination of Unitaries (LCU) technique, weighted sums of block-encoded matrices can be implemented with logarithmic overhead in the number of terms [9].
- **Inversion and matrix functions:** The technique called quantum singular value transformation (QSVT) enables efficient implementation of functions of block-encoded normal matrices, including  $\mathbf{A}^{-1}$  and  $\exp(i\mathbf{A}t)$  [9, 10].

This paper is devoted to the problem of implementing products between block-encodings, including the Kronecker product, the Hadamard product, and the matrix-matrix product, in a qubit- and gate-efficient manner. Our goal is to exploit matrix structures to design product constructions that reduce circuit complexity and ancillary qubit requirements, thereby extending the practicality of block-encoding based algorithms on quantum devices. Our main contributions are as follows:

- We address the more general setting where the two matrices to be multiplied are not necessarily square, and their dimensions are not restricted to powers of two, thereby broadening the applicability of block-encoded matrix operations.
- We propose a new method for implementing matrix-matrix products that significantly reduces the number of required ancilla qubits, at the cost of a modest increase in gate complexity. The qubit savings become particularly significant when computing the product of a sequence of matrices, where the required number of ancilla qubits can be reduced exponentially.
- Compared to the Kronecker product implementation in [21], we replace the SWAP gates with two CNOT gates, achieving a reduction in gate complexity. Extending the same qubit-efficient principle used in our matrix-matrix product construction, we further provide a qubit-efficient method for implementing the Kronecker product.
- We rederive the Hadamard product implementation between block-encodings from [22] and extend it to construct convolution operations and vectorization operators, thereby enriching the toolbox of linear operations that can be efficiently realized within block-encodings framework.
- We apply the idea of qubit-efficient implementation of matrix-matrix products to construct a compression gadget for time-dependent Hamiltonian simulation [2], which has broad applications including non-unitary dynamic simulation [23, 24, 25] and quantum linear system algorithms [26]. Compared to the existing method, our construction is more concise and eliminates redundant control operations.
- By combining the Kronecker product of block-encodings with the LCU framework, matrices represented as sums of Kronecker products can be efficiently block-encoded. The benefit of utilizing such a structure is exemplified by the adjacent matrix of an extended binary tree [11], where our approach naturally yields a more efficient implementation.

The remainder of this paper is organized as follows. In Section 2, we present our constructions for block-encodings of the Kronecker product, the matrix-matrix multiplication, and the Hadamard product. Implementations of convolution operators and vectorization operators are also discussed. In Section 3, we give several applications of matrix products between block-encodings, including the relevance to time-dependent Hamiltonian simulations and the block-encoding of matrices which can be written as a sum of Kronecker products. Finally, Section 4 provides concluding remarks and directions for future research.

## 2. Implementation of products

In this section, we describe the implementation of several types of products between block-encodings. We begin with the Kronecker product, which naturally aligns with the tensor-product structure of quantum computing and can therefore be realized efficiently. Then, we introduce a qubit-efficient method for implementing matrix-matrix products, which substantially reduces the required number of ancilla qubits; the same idea can also be applied to the Kronecker product. Finally, we present an explicit construction of the permutation matrix that extracts the entries of the Hadamard product from the Kronecker product, and we show that this permutation also enables the implementation of the vectorization operator, which maps a matrix to a column vector.

In most of the previous works, the matrix  $\mathbf{A}$  to be block-encoded is assumed to be square, with dimension a power of two. In this paper, we relax both assumptions and adopt the following generalized definition of block-encodings.

**Definition 1.** Let matrix  $\mathbf{A} \in \mathbb{C}^{M \times N}$ . A unitary matrix  $\mathbf{U}_{\mathbf{A}} \in \mathbb{C}^{2^a \times 2^a}$  is an  $(\alpha, a)$ -block-encoding of matrix  $\mathbf{A}$  if

$$\mathbf{U}_{\mathbf{A}} = \begin{bmatrix} \frac{1}{\alpha} \mathbf{A} & * \\ * & * \end{bmatrix},$$

where  $a$  denotes the total number of qubits on which  $\mathbf{U}_{\mathbf{A}}$  acts, rather than the number of ancilla qubits.

### 2.1. Kronecker product

Let  $\mathbf{B} \in \mathbb{C}^{M_b \times N_b}$  and  $\mathbf{C} \in \mathbb{C}^{M_c \times N_c}$ . Suppose the block-encodings of  $\mathbf{B}$  and  $\mathbf{C}$  are, respectively,

$$\mathbf{U}_{\mathbf{B}} = \begin{bmatrix} \mathbf{B} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^b \times 2^b}, \quad \mathbf{U}_{\mathbf{C}} = \begin{bmatrix} \mathbf{C} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^c \times 2^c}$$

Then, the Kronecker product of  $\mathbf{U}_{\mathbf{B}}$  and  $\mathbf{U}_{\mathbf{C}}$  admits

$$\mathbf{U}_{\mathbf{B}} \otimes \mathbf{U}_{\mathbf{C}} = \begin{bmatrix} \mathbf{B} \otimes \begin{bmatrix} \mathbf{C} & * \\ * & * \end{bmatrix} & * \\ * & * \end{bmatrix}, \quad \mathbf{B} \otimes \begin{bmatrix} \mathbf{C} & * \\ * & * \end{bmatrix} = \begin{bmatrix} b_{00}\mathbf{C} & * & b_{01}\mathbf{C} & * & \cdots \\ * & * & * & * & \cdots \\ b_{10}\mathbf{C} & * & b_{11}\mathbf{C} & * & \cdots \\ * & * & * & * & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

where  $b_{ij}$  denotes the  $(i, j)$ -entry of  $\mathbf{B}$ . Thus, while  $\mathbf{U}_{\mathbf{B}} \otimes \mathbf{U}_{\mathbf{C}}$  contains all entries of  $\mathbf{B} \otimes \mathbf{C}$ , these entries are interleaved with undesired star blocks.

Our target is to obtain a block-encoding whose top-left block is exactly  $\mathbf{B} \otimes \mathbf{C}$ , we construct permutation matrices (acting on rows and columns) that reorder the basis so as to gather the  $b_{ij}\mathbf{C}$  blocks contiguously, i.e., find permutation unitaries  $\mathbf{\Pi}_{\text{row}}$  and  $\mathbf{\Pi}_{\text{col}}$  such that

$$\mathbf{\Pi}_{\text{row}}(\mathbf{U}_{\mathbf{B}} \otimes \mathbf{U}_{\mathbf{C}}) \mathbf{\Pi}_{\text{col}}^\dagger = \begin{bmatrix} \mathbf{B} \otimes \mathbf{C} & * \\ * & * \end{bmatrix},$$

where  $\dagger$  denotes the Hermitian conjugate. The required permutations admit an efficient implementation on the qubit registers. To introduce the construction precisely and analyze its gate complexity, we present the following lemmas.

**Lemma 2.** Let  $N = 2^n$  and let  $\mathbf{P} \in \mathbb{C}^{N \times N}$  be a permutation unitary such that

$$\mathbf{P}|i\rangle = |(i+d) \bmod N\rangle, \quad i \in \{0, 1, \dots, N-1\},$$

for a fixed integer  $d$ . Then  $\mathbf{P}$  can be implemented as an addition of the classical constant  $d$  modulo  $N$  on  $n$  qubits:

- using the ripple-carry adder of [27] with  $O(n)$  gates,  $O(n)$  depth, and  $n + 1$  ancilla qubits; or
- using a QFT-based construction [28, 29] with  $O(n^2)$  gates,  $O(n)$  depth, and no ancilla qubits.

**Lemma 3.** *Let*

$$\mathbf{v}_i = \begin{bmatrix} \mathbf{a}_i \\ * \end{bmatrix} \in \mathbb{C}^{2^s}, \quad \mathbf{a}_i \in \mathbb{C}^M, M < 2^s, \quad \forall i = 0, \dots, 2^d - 1,$$

and define  $D = 2^d$ . There exists a quantum circuit  $\Pi$  such that

$$\Pi \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{D-1} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_{D-1} \\ * \\ \vdots \\ * \end{bmatrix}.$$

The gate complexity of  $\Pi$  depends on  $M$ :

- if  $M = 2^t$ ,  $\Pi$  requires  $2(d-1)$  CNOT gates;
- if  $M > 2^{s-1}$ ,  $\Pi$  requires  $O(ds^2)$  one- or two-qubit gates;
- if  $2^{t-1} < M < 2^t$  for some  $t < s$ ,  $\Pi$  requires  $O(dt^2)$  one- or two-qubit gates.

In summary, the gate complexity of  $\Pi$  is bounded by  $O(d(\log M)^2)$ .

PROOF. For each pair  $(\mathbf{v}_i, \mathbf{v}_{i+1})$ , there exists a permutation matrix  $\mathbf{P} \in \mathbb{R}^{2^{s+1} \times 2^{s+1}}$  such that

$$\mathbf{P} \begin{bmatrix} \mathbf{v}_i \\ \mathbf{v}_{i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_i \\ \mathbf{a}_{i+1} \\ * \\ * \end{bmatrix}. \quad (1)$$

That is,  $\mathbf{P}$  extracts the  $M$ -dimensional sub-blocks  $\mathbf{a}_i, \mathbf{a}_{i+1}$  from the vectors  $\mathbf{v}_i, \mathbf{v}_{i+1}$  and moves them to the top of the new block. Applying this transformation pairwise in tensor-product form, we obtain:

$$\mathbf{I}_{2^{d-2}} \otimes \mathbf{P} \begin{bmatrix} \mathbf{a}_0 \\ * \\ \mathbf{a}_1 \\ * \\ \vdots \\ \mathbf{a}_{D-2} \\ * \\ \mathbf{a}_{D-1} \\ * \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ * \\ * \\ \vdots \\ \mathbf{a}_{D-2} \\ \mathbf{a}_{D-1} \\ * \\ * \end{bmatrix} \quad \text{and} \quad \mathbf{I}_{2^{d-3}} \otimes \mathbf{P} \otimes \mathbf{I}_2 \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ * \\ * \\ \vdots \\ \mathbf{a}_{D-2} \\ \mathbf{a}_{D-1} \\ * \\ * \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ * \\ * \\ * \\ * \\ \vdots \end{bmatrix}.$$

Iterating this construction for  $d-1$  layers, we obtain the overall unitary

$$\Pi = \prod_{i=0}^{d-2} \mathbf{I}_{2^{d-2-i}} \otimes \mathbf{P} \otimes \mathbf{I}_{2^i},$$

which rearranges all the  $\mathbf{a}_i$  blocks consecutively at the top of the output vector, yielding the desired form.

It remains to specify the permutation  $\mathbf{P}$  depending on the size  $M$ .

1. Case  $M = 2^t$ : we may set  $\mathbf{P} = \mathbf{T} \otimes I_{2^t}$ , where  $\mathbf{T}$  is a permutation on  $s+1-t$  qubits specified by

$$\begin{aligned}\mathbf{T}|00 \cdots 00\rangle_{s+1-t} &= |00 \cdots 00\rangle_{s+1-t}, \\ \mathbf{T}|10 \cdots 00\rangle_{s+1-t} &= |00 \cdots 01\rangle_{s+1-t}.\end{aligned}$$

Since the operator  $\mathbf{T}$  acts nontrivially only on the first and last qubits of  $|\cdot\rangle_{s+1-t}$ , the construction reduces to finding a two-qubit permutation  $\tilde{\mathbf{T}}$  such that

$$\tilde{\mathbf{T}}|00\rangle = |00\rangle, \quad \text{and} \quad \tilde{\mathbf{T}}|10\rangle = |01\rangle.$$

Therefore,  $\tilde{\mathbf{T}}$  must be of the form

$$\tilde{\mathbf{T}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & * & 0 & * \\ 0 & * & 0 & * \end{bmatrix}.$$

Restricting to the permutation matrices, only two candidates exist:

$$\tilde{\mathbf{T}}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \tilde{\mathbf{T}}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \quad (2)$$

Here,  $\tilde{\mathbf{T}}_1$  is exactly the SWAP gate, requiring 3 CNOT gates. In contrast,  $\tilde{\mathbf{T}}_2$  factorizes as

$$\tilde{\mathbf{T}}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad (3)$$

which is the product of two CNOT gates. Therefore, the better choice is  $\tilde{\mathbf{T}} = \tilde{\mathbf{T}}_2$ , and consequently  $\mathbf{T}$  (and hence  $\mathbf{P}$ ) can be implemented using 2 CNOT gates. Since the overall circuit  $\mathbf{\Pi}$  requires  $(d-1)$  such permutations, the total cost is  $2(d-1)$  CNOT gates.

2. Case  $M > 2^{s-1}$ : let  $N = 2^s - M$ . We choose  $\mathbf{P}$  as

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_M & 0 & 0 & 0 \\ 0 & 0 & \mathbf{I}_M & 0 \\ 0 & 0 & 0 & \mathbf{I}_N \\ 0 & \mathbf{I}_N & 0 & 0 \end{bmatrix} = \mathbf{P}_1 \begin{bmatrix} \mathbf{P}_2 & \\ & \mathbf{I}_{2^s} \end{bmatrix}, \quad (4)$$

where

$$\mathbf{P}_1 := \begin{bmatrix} 0 & \mathbf{I}_M & 0 & 0 \\ 0 & 0 & \mathbf{I}_M & 0 \\ 0 & 0 & 0 & \mathbf{I}_N \\ \mathbf{I}_N & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{2^{s+1} \times 2^{s+1}}, \quad \mathbf{P}_2 := \begin{bmatrix} 0 & \mathbf{I}_N \\ \mathbf{I}_M & 0 \end{bmatrix} \in \mathbb{R}^{2^s \times 2^s}.$$

By construction,  $\mathbf{P}_1$  and  $\mathbf{P}_2$  are cyclic shift operators:

$$\begin{aligned}\mathbf{P}_1|i\rangle &= |(i-N) \bmod 2^{s+1}\rangle, \quad \forall i \in \{0, \dots, 2^{s+1}-1\}, \\ \mathbf{P}_2|j\rangle &= |(j-M) \bmod 2^s\rangle, \quad \forall j \in \{0, \dots, 2^s-1\}.\end{aligned}$$

By Theorem 2, both  $\mathbf{P}_1$  and  $\mathbf{P}_2$  can be implemented using  $O(s^2)$  elementary gates. Since  $\mathbf{P}_2$  is the top-left block of the second matrix in (4), we need the controlled version of  $\mathbf{P}_2$ , which can also be implemented using  $O(s^2)$  elementary gates. Hence  $\mathbf{P}$  also has gate complexity  $O(s^2)$ . Consequently, the full transformation  $\mathbf{\Pi}$  can be implemented with gate complexity  $O(ds^2)$ .

3. Case  $2^{t-1} < M < 2^t$  for some  $t < s$ : we first embed  $\mathbf{a}_i \in \mathbb{C}^M$  into a  $2^t$ -dimensional vector

$$\tilde{\mathbf{a}}_i := \begin{bmatrix} \mathbf{a}_i \\ * \end{bmatrix} \in \mathbb{C}^{2^t},$$

so that  $\tilde{\mathbf{a}}_i$  coincides with the top  $2^t$ -dimensional block of  $\mathbf{v}_i$ . Now consider two consecutive blocks  $\mathbf{v}_i, \mathbf{v}_{i+1}$ . By applying the construction from the case  $M = 2^t$ , we can transform the vector

$$\begin{bmatrix} \mathbf{v}_i \\ \mathbf{v}_{i+1} \end{bmatrix} \mapsto \begin{bmatrix} \tilde{\mathbf{a}}_i \\ \tilde{\mathbf{a}}_{i+1} \\ * \\ * \end{bmatrix}$$

using 2 CNOT gates. Next, we restrict attention to the top two blocks

$$\begin{bmatrix} \tilde{\mathbf{a}}_i \\ \tilde{\mathbf{a}}_{i+1} \end{bmatrix} \in \mathbb{C}^{2^{t+1}}.$$

By applying the construction from the case  $M > 2^{s-1}$ , we can transform it into the desired form using  $O(t^2)$  elementary gates. Therefore, the total gate complexity of implementing  $\mathbf{P}$  is  $O(t^2)$ , and the overall gate complexity of  $\mathbf{\Pi}$  is  $O(dt^2)$ . □

The idea of Theorem 3 naturally extends to block-structured matrices: by applying suitable permutations to both the rows and columns, we can align the desired submatrices into the top-left corner of a larger matrix. The following corollary formalizes this extension.

**Corollary 4.** *Suppose the matrix  $\mathbf{V}_{ij}$  has the form*

$$\mathbf{V}_{ij} = \begin{bmatrix} \mathbf{A}_{ij} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^{s_1} \times 2^{s_2}}, \quad \forall i = 0, \dots, 2^{d_1} - 1, \quad j = 0, \dots, 2^{d_2} - 1,$$

where  $\mathbf{A}_{ij} \in \mathbb{C}^{M_1 \times M_2}$ . Define

$$D_1 = 2^{d_1}, \quad D_2 = 2^{d_2}, \quad \mathbf{V} = \begin{bmatrix} \mathbf{V}_{11} & \cdots & \mathbf{V}_{1,D_2-1} \\ \vdots & & \vdots \\ \mathbf{V}_{D_1-1,1} & \cdots & \mathbf{V}_{D_1-1,D_2-1} \end{bmatrix}.$$

Then, there exist quantum circuits  $\mathbf{\Pi}_1$  and  $\mathbf{\Pi}_2$  such that

$$\mathbf{\Pi}_1 \mathbf{V} \mathbf{\Pi}_2^\dagger = \begin{bmatrix} \mathbf{A}_{11} & \cdots & \mathbf{A}_{1,D_2-1} & * \\ \vdots & & \vdots & * \\ \mathbf{A}_{D_1-1,1} & \cdots & \mathbf{A}_{D_1-1,D_2-1} & * \\ * & \cdots & * & * \end{bmatrix}.$$

In general, the gate complexity of implementing  $\mathbf{\Pi}_\alpha$  is  $O(d_\alpha(\log M_\alpha)^2)$  for  $\alpha \in \{1, 2\}$ . In the special case where  $M_\alpha$  is a power of two, the gate complexity reduces to  $O(d_\alpha)$ .

To obtain the block-encoding

$$\begin{bmatrix} \mathbf{B} \otimes \mathbf{C} & * \\ * & * \end{bmatrix},$$

from  $\mathbf{U}_B \otimes \mathbf{U}_C$ , we first define

$$d_1 = \min\{d \in \mathbb{N} : 2^d \geq M_b\}, \quad \text{and} \quad d_2 = \min\{d \in \mathbb{N} : 2^d \geq N_b\},$$

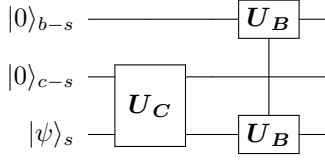


Figure 1: Quantum circuit for standard implementation of matrix-matrix product.

and denote the top-left  $2^{d_1} \times 2^{d_2}$  block of  $\mathbf{U}_B$  as  $\tilde{\mathbf{B}}$ . The desired matrix  $\mathbf{B} \otimes \mathbf{C}$  is the top-left  $M_b M_c \times N_b N_c$  block of

$$\begin{bmatrix} \tilde{\mathbf{B}} \otimes \mathbf{C} & * \\ * & * \end{bmatrix}.$$

Notice that

$$\tilde{b}_{ij} \mathbf{U}_C = \begin{bmatrix} \tilde{b}_{ij} \mathbf{C} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^c \times 2^c}, \quad \forall i = 0, \dots, 2^{d_1-1}, \quad j = 0, \dots, 2^{d_2} - 1,$$

where  $\tilde{b}_{ij}$  denotes the  $(i, j)$ -entry of  $\tilde{\mathbf{B}}$ ,  $\tilde{b}_{ij} \mathbf{C} \in \mathbb{C}^{M_c \times N_c}$ , and thus

$$\mathbf{U}_B \otimes \mathbf{U}_C = \begin{bmatrix} \tilde{\mathbf{B}} \otimes \mathbf{U}_C & * \\ * & * \end{bmatrix}.$$

In general, applying Theorem 4 yields the desired block-encoding with gate complexity

$$O\left(\log M_b (\log M_c)^2 + \log N_b (\log N_c)^2\right),$$

which can be reduced to  $O(\log M_b N_b)$  if  $M_c$  and  $N_c$  are powers of two.

**Remark 1.** The method of [21] implements block-encodings of Kronecker products, but it is restricted to the setting where both  $\mathbf{B}$  and  $\mathbf{C}$  are square matrices with power-of-two dimensions. In its construction [21], the target blocks are moved to the top-left corner using SWAP operators, each of which requires three CNOT gates. We extend this construction to the more general case where  $\mathbf{B}$  and  $\mathbf{C}$  are of arbitrary sizes. When both the row and column dimensions of  $\mathbf{C}$  are powers of two, our circuit retains the same structure as in [21] but substitutes the SWAP operator with the two-CNOT operator  $\tilde{\mathbf{T}}_2$  from eq. (3), thereby reducing the two-qubit gate count.

## 2.2. Matrix-matrix product

Let  $\mathbf{B} \in \mathbb{C}^{M \times K}$ ,  $\mathbf{C} \in \mathbb{C}^{K \times N}$ , and suppose we have block-encodings

$$\mathbf{U}_B = \begin{bmatrix} \mathbf{B} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^b \times 2^b}, \quad \mathbf{U}_C = \begin{bmatrix} \mathbf{C} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^c \times 2^c}.$$

Our goal is to construct a block-encoding of the product  $\mathbf{BC}$ . Even if  $b = c$ , a direct multiplication does not suffice, since

$$\mathbf{U}_B \mathbf{U}_C = \begin{bmatrix} \mathbf{BC} + * & * \\ * & * \end{bmatrix},$$

so the top-left block contains undesired terms. In the special case where  $M = K = N = 2^s$ , the standard construction [9] employs the circuit shown in Figure 1. This construction can be understood in terms of matrix representation: additional ancilla qubits enforce zero-padding in  $\mathbf{U}_B$  and  $\mathbf{U}_C$  so that the unwanted terms  $*$  are eliminated. However, this procedure introduces more zero-padding than necessary, thereby consuming extra qubits and leading to inefficiency.

In the following proposition, we present a qubit-efficient method to construct block-encodings of matrix products. Our approach applies to general rectangular compatible matrices  $\mathbf{B}$  and  $\mathbf{C}$ .

**Proposition 5.** Let  $\mathbf{B} \in \mathbb{C}^{M \times K}$ ,  $\mathbf{C} \in \mathbb{C}^{K \times N}$ , and suppose we are given block-encodings

$$\mathbf{U}_B = \begin{bmatrix} \mathbf{B} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^b \times 2^b}, \quad \mathbf{U}_C = \begin{bmatrix} \mathbf{C} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^c \times 2^c}.$$

Then there exists a quantum circuit that implements the block-encoding of  $\mathbf{BC}$  using  $\max\{b, c\} + 1$  qubits, with an additional gate complexity of  $O(\min\{b^2, c^2\})$  beyond that required for implementing  $\mathbf{U}_B$  and  $\mathbf{U}_C$ .

PROOF. We begin with two simple cases:

- If  $K = 2^b$ , then  $(\mathbf{I}_{2^{c-b}} \otimes \mathbf{U}_B)\mathbf{U}_C$  block-encodes  $\mathbf{BC}$ .
- If  $K = 2^c$ , then  $\mathbf{U}_B(\mathbf{I}_{2^{b-c}} \otimes \mathbf{U}_C)$  block-encodes  $\mathbf{BC}$ .

In both cases, the block-encoding of  $\mathbf{BC}$  requires only  $\max\{b, c\}$  qubits, with no additional gate complexity beyond that of  $\mathbf{U}_B$  and  $\mathbf{U}_C$ .

In the nontrivial case  $K < \min\{2^b, 2^c\}$ , we assume without loss of generality that  $b \geq c$ . (If  $b < c$ , one may instead construct a block-encoding of  $\mathbf{C}^\dagger \mathbf{B}^\dagger$  and then invert the circuit.) We define the auxiliary block-encoding

$$\mathbf{U}_{\tilde{C}} = \begin{bmatrix} \tilde{\mathbf{C}} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^t \times 2^t}, \quad \text{where } \tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} \\ 0 \end{bmatrix} \in \mathbb{C}^{2^b \times N}, \quad t = \max\{b, c\} + 1 = b + 1.$$

Then,

$$(\mathbf{I}_2 \otimes \mathbf{U}_B)\mathbf{U}_{\tilde{C}} = \begin{bmatrix} \mathbf{BC} & * \\ * & * \end{bmatrix},$$

is the desired block-encoding of the product  $\mathbf{BC}$ . Note that the resulting block-encoding costs only one more qubit than  $\mathbf{U}_B$ . It remains to show how to construct the block-encoding  $\mathbf{U}_{\tilde{C}}$ . We begin with the Kronecker product  $\mathbf{I}_{2^{t-c}} \otimes \mathbf{U}_C$ , whose first  $N$  columns take the form

$$\begin{bmatrix} \mathbf{C} \\ * \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} = |0^{t-c}\rangle \otimes \begin{bmatrix} \mathbf{C} \\ * \end{bmatrix} =: \mathbf{V}.$$

Our goal is to rearrange  $\mathbf{V}$  into a block structure where  $\mathbf{C}$  is isolated at the top, separated by at least  $2^b - K$  rows of zeros from the  $*$  terms. More precisely, we seek a  $t$ -qubit permutation  $\mathbf{P}$  such that

$$\mathbf{P}\mathbf{V} = |0\rangle \otimes |0^{t-c-1}\rangle \otimes \begin{bmatrix} \mathbf{C} \\ 0 \end{bmatrix} + |1\rangle \otimes |0^{t-c-1}\rangle \otimes \begin{bmatrix} 0 \\ * \end{bmatrix}. \quad (5)$$

Notice that only the first qubit and the last  $c$  qubits are involved in this transformation; the intermediate  $t - c - 1$  qubits remain untouched. Therefore, it suffices to define a permutation acting on these  $(c + 1)$  qubits. In the computational basis of these qubits, consider

$$\tilde{\mathbf{P}} := \begin{bmatrix} \mathbf{I}_K & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{I}_L \\ 0 & 0 & \mathbf{I}_K & 0 \\ 0 & \mathbf{I}_L & 0 & 0 \end{bmatrix} \in \mathbb{R}^{2^{c+1} \times 2^{c+1}}, \quad L = 2^{c+1} - K.$$

This permutation achieves the transformation

$$|0\rangle \otimes \begin{bmatrix} \mathbf{C} \\ * \end{bmatrix} \xrightarrow{\tilde{\mathbf{P}}} |0\rangle \otimes \begin{bmatrix} \mathbf{C} \\ 0 \end{bmatrix} + |1\rangle \otimes \begin{bmatrix} 0 \\ * \end{bmatrix},$$



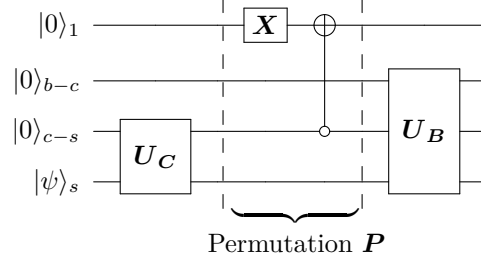


Figure 2: Quantum circuit for qubit-efficient implementation of matrix-matrix product. The subscript of each ket indicates the number of qubits in that ket.

which matches exactly the desired block structure in eq. (5). For implementation,  $\tilde{\mathbf{P}}$  can be factorized as

$$\tilde{\mathbf{P}} = \begin{bmatrix} 0 & 0 & 0 & \mathbf{I}_K \\ \mathbf{I}_L & 0 & 0 & 0 \\ 0 & \mathbf{I}_K & 0 & 0 \\ 0 & 0 & \mathbf{I}_L & 0 \end{bmatrix} \left( \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & \mathbf{I}_L \\ \mathbf{I}_K & 0 \end{bmatrix} \right).$$

Each factor can be implemented using Theorem 2 with gate complexity  $O(c^2)$ . Since  $t - 1 = b$ , it follows that  $\mathbf{P}(\mathbf{I}_{2^{t-c}} \otimes \mathbf{U}_C)$  is a block-encoding of  $\mathbf{U}_{\tilde{C}}$ , and thus

$$(\mathbf{I}_2 \otimes \mathbf{U}_B) \mathbf{P} (\mathbf{I}_{2^{t-c}} \otimes \mathbf{U}_C)$$

provides a block-encoding of  $\mathbf{BC}$ . In summary, this block-encoding requires  $\max\{b, c\} + 1$  qubits and can be implemented with  $O(\min\{b^2, c^2\})$  gate complexity.  $\square$

**Example 1.** Suppose  $M = K = N = 2^s$ . In this case, we can choose

$$\tilde{\mathbf{P}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{I}_{2^{c-s}-1} \\ 0 & 0 & 1 & 0 \\ 0 & \mathbf{I}_{2^{c-s}-1} & 0 & 0 \end{bmatrix} \otimes \mathbf{I}_{2^s},$$

which can be implemented using a single NOT gate and a multi-controlled NOT gate. The quantum circuit for the block-encoding of  $\mathbf{BC}$  is then given in Figure 2. To verify correctness, consider its action on the input state  $|0\rangle_{b-s+1}|\psi\rangle_s$ :

$$\begin{aligned} |0\rangle_1|0\rangle_{b-c}|0\rangle_{c-s}|\psi\rangle_s &\longrightarrow |1\rangle_1|0\rangle_{b-c}(|0\rangle_{c-s}\mathbf{C}|\psi\rangle_s + |g\rangle_c) \\ &\longrightarrow |0\rangle_1|0\rangle_{b-c}|0\rangle_{c-s}\mathbf{C}|\psi\rangle_s + |1\rangle_1|0\rangle_{b-c}|g\rangle_c \\ &\longrightarrow |0\rangle_1|0\rangle_{b-c}|0\rangle_{c-s}\mathbf{BC}|\psi\rangle_s + |0\rangle_1|g'\rangle_b + |1\rangle_1|g''\rangle_b, \end{aligned}$$

where  $|g\rangle_c$  is orthogonal to  $|0\rangle_{c-s} \otimes \mathbf{I}_{2^s}$ , and  $|g'\rangle_b$  is orthogonal to  $|0\rangle_{b-s} \otimes \mathbf{I}_{2^s}$ .

**Remark 2.** The existing method for implementing matrix-matrix product of block-encoded matrices [9] applies only to the case where both matrices are square of the same size, and this dimension must be a power of two. In contrast, our method generalizes to arbitrary matrix sizes without such restrictions. Below, we provide a comparison of the required number of qubits and the gate complexity between the two approaches.

Suppose  $M = K = N = 2^s$  are powers of two. The existing method requires  $b + c - s$  qubits, while our method requires

$$t = \max\{b, c\} + 1$$

qubits. Since  $2^s = K < 2^{\min\{b,c\}}$ , we have

$$b + c - s > b + c - \min\{b, c\} = \max\{b, c\},$$

and thus

$$b + c - s \geq \max\{b, c\} + 1.$$

Therefore, our method always uses fewer qubits. Regarding the gate complexity, the existing method does not incur additional costs beyond the block-encodings  $\mathbf{U}_B$  and  $\mathbf{U}_C$ . In contrast, our method requires  $O(\min\{b^2, c^2\})$  elementary quantum gates in general.

**Remark 3.** A qubit-efficient implementation of the Kronecker product, inspired by our matrix-matrix product construction, is described in [Appendix A](#).

### 2.2.1. Product of a sequence of matrices

The qubit-efficient advantage of our method becomes more pronounced when considering the product of a sequence of matrices. Suppose we are given matrices  $\mathbf{A}_1, \dots, \mathbf{A}_n \in \mathbb{C}^{2^s \times 2^s}$ , and their block-encodings have sizes  $2^{a_1} \times 2^{a_1}, \dots, 2^{a_n} \times 2^{a_n}$ , respectively. Without loss of generality, assume  $s < a_i$  for all  $i$ . Our goal is to construct a block-encoding of the product  $\mathbf{A}_1 \mathbf{A}_2 \cdots \mathbf{A}_n$ .

If we apply the existing method [9] iteratively, the required number of qubits is  $\sum_{i=1}^n a_i - (n-1)s$ . Since  $s < a_i$  for all  $i$ , this number scales as  $\max_{1 \leq i \leq n} a_i + O(n-1)$ . In contrast, our method can reduce the additive overhead from  $O(n-1)$  to  $\lceil \log_2 n \rceil$ . This improvement relies on choosing an optimal multiplication order via dynamic programming. Define  $m_{ij}$  as the minimum number of qubits required to block-encode the product  $\mathbf{A}_i \mathbf{A}_{i+1} \cdots \mathbf{A}_j$ . Then  $m_{ij}$  satisfies the recurrence

$$m_{ij} = \min_{i \leq r < j} \{ \max\{m_{ir}, m_{r+1,j}\} + 1 \}. \quad (6)$$

We now show by induction that

$$m_{ij} \leq \max_{i \leq r \leq j} a_r + \lceil \log_2(j-i+1) \rceil. \quad (7)$$

For  $j = i + 1$ ,

$$m_{i,i+1} = \max\{a_i, a_{i+1}\} + 1,$$

which clearly satisfies eq. (7). Suppose the bound holds for all intervals with length smaller than  $k$ . Then for  $j = i + k$  and  $i \leq r < j$ , we have

$$\begin{aligned} m_{ij} &\leq \max\{m_{ir}, m_{r+1,j}\} + 1 \\ &\leq \max \left\{ \max_{i \leq l \leq r} a_l + \lceil \log_2(r-i+1) \rceil, \max_{r+1 \leq l \leq j} a_l + \lceil \log_2(j-r) \rceil \right\} + 1 \\ &\leq \max_{i \leq l \leq j} a_l + \max\{\lceil \log_2(r-i+1) \rceil, \lceil \log_2(j-r) \rceil\} + 1. \end{aligned}$$

Note that there exists an index  $r^*$  such that the two subintervals  $[i, r^*]$  and  $[r^* + 1, j]$  are approximately balanced in length. In particular, we can choose  $r^*$  so that

$$\max\{r^* - i + 1, j - r^*\} \leq \left\lceil \frac{j - i + 1}{2} \right\rceil.$$

For the choice of  $r^*$ , we obtain

$$\max\{\lceil \log_2(r^* - i + 1) \rceil, \lceil \log_2(j - r^*) \rceil\} + 1 \leq \left\lceil \log_2 \left( \left\lceil \frac{j - i + 1}{2} \right\rceil \right) + 1 \right\rceil.$$

Now we analyze the right-hand side depending on the parity of  $j - i + 1$ :

- If  $j - i + 1$  is even, then

$$\left\lceil \log_2 \left\lceil \frac{j - i + 1}{2} \right\rceil + 1 \right\rceil = \lceil \log_2(j - i + 1) \rceil.$$

- If  $j - i + 1$  is odd, then

$$\left\lceil \log_2 \left\lceil \frac{j - i + 1}{2} \right\rceil + 1 \right\rceil = \lceil \log_2(j - i + 2) \rceil = \lceil \log_2(j - i + 1) \rceil,$$

where the second equality holds due to  $j - i + 1$  is odd and  $j > i + 1$ .

Therefore, in both cases, we have

$$\max \{ \lceil \log_2(r^* - i + 1) \rceil, \lceil \log_2(j - r^*) \rceil \} + 1 \leq \lceil \log_2(j - i + 1) \rceil.$$

Substitute this bound into the recurrence gives

$$m_{ij} \leq \max_{i \leq l \leq j} a_l + \lceil \log_2(j - i + 1) \rceil, \quad \forall j > i + 1.$$

Together with the base case  $j = i + 1$ , this complete the induction for eq. (7). In particular, for the full product we obtain

$$m_{1n} \leq \max_{1 \leq r \leq n} a_r + \lceil \log_2 n \rceil.$$

Moreover, a simple upper bound to the additional gate complexity is

$$O \left( n \left( \log n + \max_i a_i - a \right)^2 \right).$$

Since the baseline cost of implementing  $n$  block-encodings is  $\Omega(n)$ , this additional overhead grows only polylogarithmically and is therefore acceptable.

### 2.3. Hadamard product

In this section, we extend the implementation of the Hadamard product proposed in [22] to matrices of arbitrary dimensions. In addition, we present implementations for convolution operation and vectorization operator, which are closely related to the construction of the Hadamard product.

Let  $\mathbf{B}, \mathbf{C} \in \mathbb{C}^{M \times N}$  be two matrices. Suppose the block-encodings of  $\mathbf{B}$  and  $\mathbf{C}$  are, respectively,

$$\mathbf{U}_{\mathbf{B}} = \begin{bmatrix} \mathbf{B} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^b \times 2^b}, \quad \mathbf{U}_{\mathbf{C}} = \begin{bmatrix} \mathbf{C} & * \\ * & * \end{bmatrix} \in \mathbb{C}^{2^c \times 2^c}$$

Without loss of generality, we assume  $M = 2^l$  and  $N = 2^r$ , since the block-encoding of the Hadamard product for  $\mathbf{B}$  and  $\mathbf{C}$  can be obtained from that of the corresponding top-left submatrices of  $\mathbf{U}_{\mathbf{B}}$  and  $\mathbf{U}_{\mathbf{C}}$  that contain  $\mathbf{B}$  and  $\mathbf{C}$ . Let  $\circ$  denote the Hadamard product, we have

$$\begin{aligned} \mathbf{B} \circ \mathbf{C} &= \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} b_{ij} c_{ij} |i\rangle_l \langle j|_r \\ &= \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |i\rangle_l (\langle i|_b \mathbf{U}_{\mathbf{B}} |j\rangle_b \langle i|_c \mathbf{U}_{\mathbf{C}} |j\rangle_c) \langle j|_r \\ &= \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |i\rangle_l (\langle i|_b \otimes \langle i|_c) (\mathbf{U}_{\mathbf{B}} \otimes \mathbf{U}_{\mathbf{C}}) (|j\rangle_b \otimes |j\rangle_c) \langle j|_r \\ &= \left( \sum_{i=0}^{M-1} |i\rangle_l (\langle i|_b \otimes \langle i|_c) \right) (\mathbf{U}_{\mathbf{B}} \otimes \mathbf{U}_{\mathbf{C}}) \left( \sum_{j=0}^{N-1} (|j\rangle_b \otimes |j\rangle_c) \langle j|_r \right), \end{aligned}$$

where the subscript denotes the number of qubits in that ket (or bra). Define

$$\mathbf{T}_r := \sum_{j=0}^{N-1} (|j\rangle_b \otimes |j\rangle_c) \langle j|_r. \quad (8)$$

To encode  $\mathbf{T}_r$  into quantum circuit, let  $\mathbf{P}_r$  satisfy

$$\mathbf{P}_r |0\rangle_b \otimes |j\rangle_c = |j\rangle_b \otimes |j\rangle_c, \quad \forall j = 0, \dots, 2^r - 1.$$

The operator  $\mathbf{P}_r$  replicates the bit string stored in the register  $|\cdot\rangle_c$  into the register  $|\cdot\rangle_b$ . In general, this operation requires  $c$  CNOT gates. However, since we only need to handle the bit strings smaller than  $2^r - 1$ , it suffices to use  $r$  CNOT gates. Then, we obtain

$$\begin{aligned} \mathbf{P}_r (|0\rangle_{b+c-r} \otimes I_{2^r}) &= \sum_{j=0}^{2^r-1} \mathbf{P}_r (|0\rangle_{b+c-r} \otimes |j\rangle_r \langle j|_r) = \sum_{j=0}^{2^r-1} \mathbf{P}_r (|0\rangle_{b+c-r} \otimes |j\rangle_r) (1 \otimes \langle j|_r) \\ &= \sum_{j=0}^{2^r-1} \mathbf{P}_r (|0\rangle_b \otimes |j\rangle_c) \langle j|_r = \sum_{j=0}^{2^r-1} (|j\rangle_b \otimes |j\rangle_c) \langle j|_r, \end{aligned}$$

where we combine the last  $c - r$  qubits in  $|0\rangle_{b+c-r}$  with  $|j\rangle_r$  to get  $|j\rangle_c$  in the third equality. Since  $N = 2^r$ , the first  $N$  columns of  $\mathbf{P}_r$  coincide with  $\mathbf{T}_r$ . Similarly, the operator

$$\mathbf{T}_l := \sum_{i=0}^{M-1} (|i\rangle_b \otimes |i\rangle_c) \langle i|_l$$

can be encoded as the first  $M$  columns of a permutation matrix  $\mathbf{P}_l$ , which can be implemented using  $l$  CNOT gates. Finally,

$$\begin{aligned} \mathbf{P}_l^\dagger (\mathbf{U}_B \otimes \mathbf{U}_C) \mathbf{P}_r &= \begin{bmatrix} \mathbf{T}_l^\dagger \\ * \end{bmatrix} (\mathbf{U}_B \otimes \mathbf{U}_C) \begin{bmatrix} \mathbf{T}_r & * \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{T}_l^\dagger (\mathbf{U}_B \otimes \mathbf{U}_C) \mathbf{T}_r & * \\ * & * \end{bmatrix} = \begin{bmatrix} \mathbf{B} \circ \mathbf{C} & * \\ * & * \end{bmatrix}. \end{aligned}$$

Hence, the Hadamard product of two block-encodings can be realized using  $l + r = O(\log(MN))$  CNOT gates.

### 2.3.1. Convolution

Let

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \psi_i |i\rangle, \quad |\varphi\rangle = \sum_{i=0}^{2^n-1} \varphi_i |i\rangle.$$

Our goal is to prepare the quantum state encoding their discrete convolution,

$$|\psi * \varphi\rangle = \frac{1}{c} \sum_{i=0}^{2^n-1} (\psi * \varphi)_i |i\rangle,$$

where  $c$  is a normalization factor. Using the standard relation between convolution and Fourier transform  $\mathcal{F}$ , we have

$$\psi * \varphi = \mathcal{F}^{-1} (\mathcal{F}(\psi) \circ \mathcal{F}(\varphi)),$$

where  $\circ$  denotes the element-wise (Hadamard) product. This relation naturally motivates a quantum implementation: one first applies the Quantum Fourier Transform (QFT) to both input states, performs the Hadamard product in the Fourier domain, and finally applies the inverse QFT to obtain the convolution.

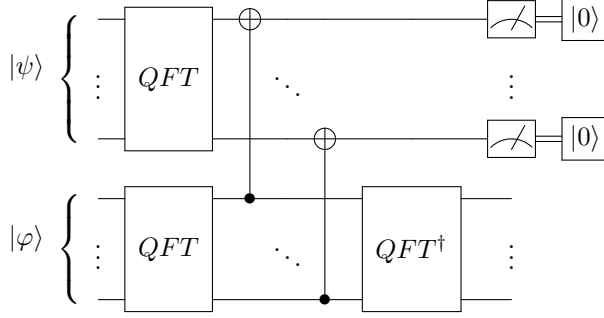


Figure 3: Quantum circuit of convolution. The Hadamard product between two quantum states appears as a special case of the Hadamard product between block-encodings. Here,  $n$  CNOT gates implement  $\mathbf{P}_I$ , while  $\mathbf{P}_r$  reduces to the identity since a quantum state has row dimension one.

The corresponding quantum circuit is shown in Figure 3. The two registers are initialized in  $|\psi\rangle$  and  $|\varphi\rangle$ . Applying the QFT in parallel transforms both registers into the Fourier basis. At this stage, a sequence of  $n$  CNOT gates performs the Hadamard product between the transformed amplitudes. Finally, the inverse QFT is applied to the second register, thereby producing the quantum state corresponding to the convolution in the computational basis. Upon measuring the first register and obtaining  $|0\rangle$ , the second register collapses to the state  $|\psi * \varphi\rangle$ . The overall cost is dominated by the QFT and its inverse, each of which requires  $O(n^2)$  one- and two-qubit gates in the standard implementation. Therefore, the gate complexity of the convolution procedure is  $O(n^2)$ .

In summary, this construction shows how the well-known convolution-Fourier duality can be translated into the quantum setting, leveraging the Hadamard product between block-encodings together with efficient implementations of the QFT.

### 2.3.2. Vectorization

Given a matrix

$$\mathbf{A} = [\mathbf{a}_0 \quad \mathbf{a}_1 \quad \cdots \quad \mathbf{a}_{N-1}] \in \mathbb{C}^{M \times N},$$

the operator  $\text{vec}(\cdot)$  stacks the columns of  $\mathbf{A}$  into a single long vector:

$$\text{vec}(\mathbf{A}) := \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_{N-1} \end{bmatrix} \in \mathbb{C}^{MN}.$$

Our goal is to construct a block-encoding of  $\text{vec}(\mathbf{A})$  given a block-encoding of  $\mathbf{A}$ .

Assume first that  $N = 2^r$ , and let  $\mathbf{U}_\mathbf{A} \in \mathbb{C}^{2^a \times 2^a}$  be a block-encoding of  $\mathbf{A}$ . Consider the stacked action of the first  $N$  columns of  $\mathbf{U}_\mathbf{A}$ ,

$$\begin{bmatrix} \mathbf{U}_\mathbf{A}|0\rangle_a \\ \mathbf{U}_\mathbf{A}|1\rangle_a \\ \vdots \\ \mathbf{U}_\mathbf{A}|N-1\rangle_a \end{bmatrix} = (\mathbf{I}_N \otimes \mathbf{U}_\mathbf{A}) \begin{bmatrix} |0\rangle_a & & & \\ & |1\rangle_a & & \\ & & \ddots & \\ & & & |N-1\rangle_a \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

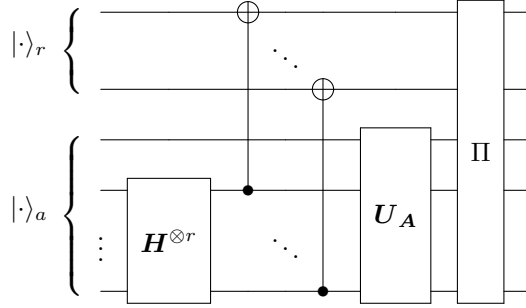


Figure 4: Quantum circuit of vectorization. The Hadamard layer prepares the uniform superposition, the CNOT chain realizes  $T_r$ , and the permutation  $\Pi$  (from Theorem 3) gathers the  $\mathbf{a}_i$  blocks contiguously, resulting in a block-encoding of  $\text{vec}(\mathbf{A})$ .

The block-diagonal operator in the middle can be expressed as

$$\begin{bmatrix} |0\rangle_a & & & \\ & |1\rangle_a & & \\ & & \ddots & \\ & & & |N-1\rangle_a \end{bmatrix} = \sum_{j=0}^{N-1} (|j\rangle_r \langle j|_r) \otimes |j\rangle_a = \sum_{j=0}^{N-1} (|j\rangle_r \otimes |j\rangle_a) \langle j|_r,$$

which has the same form as  $T_r$  introduced in eq. (8). This observation allows us to block-encode it into a simple permutation using  $r$  CNOT gates. The all-ones vector of length  $N$  appearing on the right-hand side is proportional to the first column of  $H^{\otimes r}$ , where  $H$  is the Hadamard gate. Hence, applying  $H^{\otimes r}$  to  $|0\rangle_r$  prepares the required uniform superposition. The resulting stacked vector takes the form

$$\begin{bmatrix} U_A |0\rangle_a \\ U_A |1\rangle_a \\ \vdots \\ U_A |N-1\rangle_a \end{bmatrix} = \begin{bmatrix} \mathbf{a}_0 \\ * \\ \mathbf{a}_1 \\ * \\ \vdots \\ \mathbf{a}_{N-1} \\ * \end{bmatrix},$$

where  $*$  represents additional ancillary components. By applying the permutation described in Theorem 3, these unwanted components can be rearranged so that the  $\mathbf{a}_i$  blocks appear contiguously, giving precisely the block-encoding of  $\text{vec}(\mathbf{A})$ . The full construction is illustrated in Figure 4.

For the case where  $N$  is not a power of two, one can take the smallest  $r$  such that  $2^r > N$ , and thus the same procedure yields the block-encoding of  $\text{vec}(\mathbf{A})$ . The additional gate complexity beyond that of  $U_A$  is dominated by the final permutation  $\Pi$ , which, by Theorem 3, can be performed with gate complexity

$$O(r(\log M)^2) = O(\log N(\log M)^2).$$

If  $M$  happens to be a power of two, the bound improves to  $O(\log N)$ .

### 3. Applications

Matrix products—including the matrix-matrix multiplication, the Kronecker product, and the Hadamard product—are fundamental operations in linear algebra and arises in a variety of contexts. Block-encoding, in turn, is a key primitive in many quantum algorithms, such as Hamiltonian simulation, quantum singular value transformation (QSVT), quantum linear system algorithms (QLSAs), and quantum walks. Naturally, matrix products between block-encodings also occur frequently in quantum computing. In this section, we present several applications of such products.

Bit string	10...01	...	11...11	00...00	00...01	...	10...00
Integer	$-2^{n_b-1} + 1$	...	-1	0	1	...	$2^{n_b-1}$

Table 1: The correspondence between bit strings and integers.

### 3.1. Time-dependent Hamiltonian Simulation

In [2], a low-space-overhead simulation algorithm based on the truncated Dyson series is proposed for time-dependent quantum dynamics. To reduce space requirements, a compression gadget is introduced.

**Lemma 6** (Lemma 13 in [2]). *Let  $\{\mathbf{U}_k : k \in [K]\}$  be a set of  $K$  unitaries that encode matrices  $\mathbf{H}_k \in \mathbb{C}^{2^{n_s} \times 2^{n_s}}$  such that*

$$(\langle 0|_a \otimes \mathbf{I}_s) \mathbf{U}_k (|0\rangle_a \otimes \mathbf{I}_s) = \mathbf{H}_k, \quad \|\mathbf{H}_k\| \leq 1, \quad |0\rangle_a \in \mathbb{C}^{2^{n_a}}.$$

*Then there exists a quantum circuit  $\mathbf{V}$  such that on input states spanned by  $\{|k\rangle_b : k \in \{0, \dots, K\}\}$ ,*

$$(\langle 0|_{ac} \otimes \mathbf{I}_s) \mathbf{V} (|0\rangle_{ac} \otimes \mathbf{I}_s) = |0\rangle \langle 0|_b \otimes \mathbf{I}_s + \sum_{k=1}^K |k\rangle \langle k|_b \otimes \left( \prod_{j=1}^k \mathbf{H}_j \right), \quad |k\rangle_b \in \mathbb{C}^{2^{n_b}}, \quad |0\rangle_c \in \mathbb{C}^{2^{n_c}},$$

*where the number of qubits  $n_b \in O(n_c) = O(\log K)$  and*

$$\prod_{j=1}^k \mathbf{H}_j := \mathbf{H}_k \mathbf{H}_{k-1} \cdots \mathbf{H}_2 \mathbf{H}_1.$$

*The cost of  $\mathbf{V}$  is one query to each controlled-controlled- $\mathbf{U}_k$ , and  $O(K(n_a + \log K))$  additional primitive quantum gates.*

In the truncated Dyson series approach for time-dependent Hamiltonians, one often needs to implement sequential products of many Hamiltonian terms, controlled on an index register that select which terms appear in the series. Theorem 6 from [2] guarantees that these products can be implemented with exponentially fewer ancillary qubits than a naïve implementation, while keeping gate complexity linear in the number of terms. Following the qubit-efficient approach for implementing matrix-matrix products, we provide a simpler construction of  $\mathbf{V}$  that only uses controlled- $\mathbf{U}_k$ . The asymptotic gate complexity remains the same as in the original method, but redundant controls are eliminated, yielding a more concise quantum circuit. Here is our construction.

Let  $\mathbf{T}_{p,q}$ , with  $1 \leq p \leq q \leq K$ , denote a quantum circuit that performs the transformation:

$$\mathbf{T}_{p,q} |0\rangle_c |k\rangle_b |0\rangle_a |\psi\rangle_s = |0\rangle_c |k - q + p - 1\rangle_b |0\rangle_a \left( \prod_{j=p}^{\min\{q, p+k-1\}} \mathbf{H}_j \right) |\psi\rangle_s + |g\rangle, \quad \forall 1 - p \leq k \leq K,$$

where the arithmetic in  $|\cdot\rangle_b$  is modulo  $2^{n_b}$ , and  $|g\rangle$  is orthogonal to the subspace  $|0\rangle_c \otimes \mathbf{I}_b \otimes |0\rangle_a \otimes \mathbf{I}_s$ . To be clarified, in Table 1, we list the correspondence between bit strings and integers in the register  $|\cdot\rangle_b$ . Then, if  $n_b \geq 1 + \log_2 K$ , we have

$$\mathbf{T}_{1,K} |0\rangle_c |k\rangle_b |0\rangle_a |\psi\rangle_s = |0\rangle_c |k - K\rangle_b |0\rangle_a \left( \prod_{j=1}^k \mathbf{H}_j \right) |\psi\rangle_s + |g\rangle, \quad \forall 0 \leq k \leq K,$$

and thus the quantum circuit  $\mathbf{V}$  can be chosen as

$$\text{ADD}_b^K \mathbf{T}_{1,K},$$

where  $\text{ADD}_b^K$  adds the integer  $K$  to the register  $|\cdot\rangle_b$ . The construction of  $\mathbf{T}_{1,K}$  proceeds in three steps:

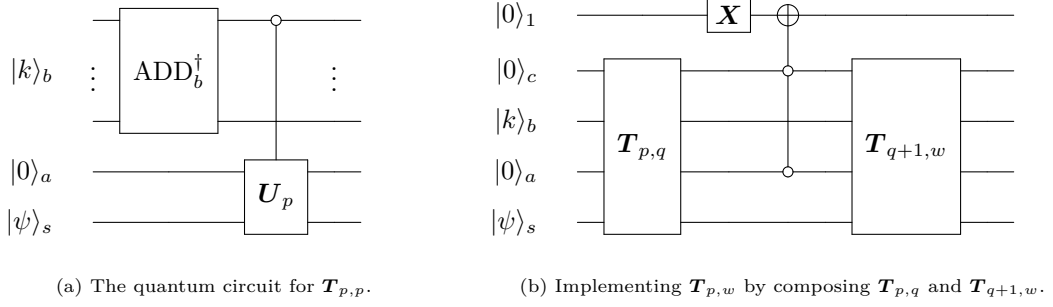


Figure 5: Illustration of the two main steps for constructing  $T_{1,K}$ . (a) base case and (b) recursive composition.

1. **Base case:**  $T_{p,p}$  can be implemented using the circuit shown in Figure 5a. The gate  $\text{ADD}_b^\dagger$  acts as a decrement (minus-one) operator on the register  $|\cdot\rangle_b$ , and can be implemented in two ways according to Theorem 2. For consistency with the analysis in [2], we assume that  $\text{ADD}_b^\dagger$  is implemented using  $O(n_b)$  gates and  $n_b + 1$  ancilla qubits. The operator  $U_p$  is controlled by the most significant qubit in  $|\cdot\rangle_b$ . Specifically, if  $k \leq 0$ , the most significant qubit of  $|k-1\rangle_b$  is 1 and thus the output is

$$|k-1\rangle_b |0\rangle_a |\psi\rangle_s.$$

For  $1 \leq k \leq K \leq 2^{n_b-1}$ , the most significant qubit of  $|k-1\rangle_b$  is 0 and the output is

$$|k-1\rangle_b |0\rangle_a H_p |\psi\rangle_s + |g\rangle.$$

In conclusion, the circuit shown in Figure 5a correctly realizes  $T_{p,p}$ .

2. **Recursive composition:** Given circuits for  $T_{p,q}$  and  $T_{q+1,w}$ , we can construct  $T_{p,w}$  as shown in Figure 5b. For  $1-p \leq k \leq K$ , applying this composition to  $|0\rangle_c |k\rangle_b |0\rangle_a |\psi\rangle_s$  gives

$$\begin{aligned} |0\rangle_c |k\rangle_b |0\rangle_a |\psi\rangle_s &\longrightarrow |0\rangle_1 \left( |0\rangle_c |k-q+p-1\rangle_b |0\rangle_a \left( \prod_{j=p}^{\min\{q,p+k-1\}} H_j \right) |\psi\rangle_s + |g\rangle \right) \\ &\longrightarrow |0\rangle_1 |0\rangle_c |k-q+p-1\rangle_b |0\rangle_a \left( \prod_{j=p}^{\min\{q,p+k-1\}} H_j \right) |\psi\rangle_s + |1\rangle_1 |g\rangle \\ &\longrightarrow |0\rangle_1 \left( |0\rangle_c |k-w+p-1\rangle_b |0\rangle_a \left( \prod_{i=q+1}^{\min\{w,p+k-1\}} H_i \prod_{j=p}^{\min\{q,p+k-1\}} H_j \right) |\psi\rangle_s + |g'\rangle \right) + |1\rangle_1 |g''\rangle \\ &= |0\rangle_1 |0\rangle_c |k-w+p-1\rangle_b |0\rangle_a \left( \prod_{j=p}^{\min\{w,p+k-1\}} H_j \right) |\psi\rangle_s + (|0\rangle_1 |g'\rangle + |1\rangle_1 |g''\rangle), \end{aligned}$$

where  $|0\rangle_1 |g'\rangle + |1\rangle_1 |g''\rangle$  is orthogonal to the subspace  $|0\rangle_1 |0\rangle_c \otimes \mathbf{I}_b \otimes |0\rangle_a \otimes \mathbf{I}_s$ , verifying that the circuit implements  $T_{p,w}$ . This recursive composition requires one additional ancilla qubit and one multi-controlled NOT gate, which can be implemented using  $O(n_a + n_c)$  primitive gates with a few ancillas.

3. **Full construction:** Let  $n_b = \lceil \log_2 K \rceil + 1$ . Starting from  $T_{p,p}$  circuits, we can construct  $T_{1,K}$  using  $\lceil \log_2 K \rceil$  ancilla qubits. Thus,  $n_c$  can be chosen as  $\lceil \log_2 K \rceil$ . The overall cost is one query to each controlled- $U_k$  and  $O(K(\log K + n_a))$  additional primitive quantum gates.

### 3.2. Block-encoding matrices with Kronecker-sum structure

It is often the case that matrices of practical interest exhibit a Kronecker-sum structure, meaning they can be expressed as a sum of Kronecker products. By identifying and exploiting this structure, together



with our method for implementing Kronecker products, the task of block-encoding a complicated matrix can be reduced to block-encoding several simpler matrices. We demonstrate this approach through several representative examples.

### 3.2.1. Generalized Sylvester equation

The generalized Sylvester equation, typically expressed as

$$\sum_{k=1}^m \mathbf{A}_k \mathbf{X} \mathbf{B}_k = \mathbf{C},$$

where  $\mathbf{A}_k, \mathbf{B}_k$  and  $\mathbf{C}$  are known matrices and  $\mathbf{X}$  is unknown, arises naturally in a wide range of scientific and engineering applications [4]. Exploiting the vectorization identity

$$\text{vec}(\mathbf{A} \mathbf{X} \mathbf{B}) = (\mathbf{B}^T \otimes \mathbf{A}) \text{vec}(\mathbf{X}),$$

the equation can be reformulated as the linear system

$$\left( \sum_{k=1}^m \mathbf{B}_k^T \otimes \mathbf{A}_k \right) \text{vec}(\mathbf{X}) = \text{vec}(\mathbf{C}).$$

Given block-encodings of  $\mathbf{A}_k, \mathbf{B}_k$ , and  $\mathbf{C}$ , one can first construct block-encodings of  $\mathbf{B}_k^T \otimes \mathbf{A}_k$ 's, and then apply the LCU method [9] to obtain the block-encoding of

$$\sum_{k=0}^m \mathbf{B}_k^T \otimes \mathbf{A}_k.$$

Further, the block-encoding of  $\text{vec}(\mathbf{C})$  can be constructed directly from the block-encoding of  $\mathbf{C}$ , as described in Section 2.3.2. Equipped with these block-encodings, the generalized Sylvester equation can then be solved on a quantum computer using QLSAs.

### 3.2.2. Discretised differential operators

Differential equations are widely used to model physical processes. As the dimension increases, the size of the discretised matrix grows exponentially. However, many such matrices can be expressed as a sum of Kronecker products, which makes them well-suited for efficient block-encoding.

As a concrete example, consider the problem of pricing multi-asset derivatives by solving the Black-Scholes partial differential equation (PDE) using the finite-difference method. This task suffers from the curse of dimensionality, namely the exponential growth of computational complexity with the number of underlying assets. In [19], the authors present a quantum algorithm for this problem. By discretising the spatial variables in the Black-Scholes PDE, the problem is reformulated as an ODE system:

$$\frac{d}{d\tau} \mathbf{y}(\tau) = \mathbf{F} \mathbf{y}(\tau) + \mathbf{c}(\tau),$$

where  $\mathbf{c}(\tau)$  encodes the inhomogeneous contributions from boundary conditions, and the coefficient matrix  $\mathbf{F}$  admits a Kronecker-sum structure:

$$\begin{aligned} \mathbf{F} &:= \mathbf{F}^{2\text{nd}} + \mathbf{F}^{1\text{st}}, \\ \mathbf{F}^{2\text{nd}} &:= \sum_{i=1}^d \frac{\sigma_i^2}{2h_i^2} \mathbf{I}^{\otimes i-1} \otimes \mathbf{D}^{2\text{nd}} \otimes \mathbf{I}^{\otimes d-i} + \sum_{i=1}^{d-1} \sum_{j=i+1}^d \frac{\sigma_i \sigma_j \rho_{ij}}{4h_i h_j} \mathbf{I}^{\otimes i-1} \otimes \mathbf{D}^{1\text{st}} \otimes \mathbf{I}^{\otimes j-i-1} \otimes \mathbf{D}^{1\text{st}} \otimes \mathbf{I}^{\otimes d-j}, \\ \mathbf{F}^{1\text{st}} &:= \sum_{i=1}^d \frac{1}{2h_i} \left( r - \frac{1}{2} \sigma_i^2 \right) \mathbf{I}^{\otimes i-1} \otimes \mathbf{D}^{1\text{st}} \otimes \mathbf{I}^{\otimes d-i}, \end{aligned}$$

where  $d$  is the number of assets,  $\sigma_i, h_i, \rho_{ij}, r$  are given real parameters,  $\mathbf{I}$  denotes the identity matrix. The one-dimensional finite-difference matrices for first- and second-order derivatives are

$$\mathbf{D}^{\text{1st}} := \begin{pmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & -1 & 0 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 0 & 1 \\ & & & & -1 & 0 \end{pmatrix}, \quad \mathbf{D}^{\text{2nd}} := \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{pmatrix}.$$

To solve such ODEs on a quantum computer, the state-of-the-art approaches rely on efficient block-encodings of the coefficient matrix [30, 24, 31]. The block-encodings of  $\mathbf{D}^{\text{1st}}$  and  $\mathbf{D}^{\text{2nd}}$  can be efficiently implemented by the method of [11]. Once the two block-encodings are available, the block-encoding of  $\mathbf{F}$  follows naturally from its Kronecker-sum structure, combined with the LCU method.

### 3.2.3. Adjacent matrix of extended binary tree.

Consider the adjacent matrix of an extended binary tree, given by

$$\mathbf{A} = \begin{bmatrix} \gamma & \beta & & & & \\ \beta & \alpha & \beta & \beta & & \\ & \beta & \alpha & & \beta & \beta \\ & \beta & & \alpha & & \beta & \beta \\ & & \beta & & \gamma & & \\ & & \beta & & & \gamma & \\ & & & \beta & & & \gamma \\ & & & \beta & & & & \gamma \end{bmatrix}, \quad (9)$$

where  $\alpha, \beta, \gamma \in (0, 1)$ . Block-encodings of such matrices were analyzed in [11]. Here we show that a more efficient block-encoding can be obtained by expressing  $\mathbf{A}$  as a sum of Kronecker products. Specifically, we decompose  $\mathbf{A}$  as

$$\mathbf{A} = \begin{bmatrix} \beta & & & & & & & \\ \beta & & & & & & & \\ & \beta & & & & & & \\ & \beta & & & & & & \\ & & \beta & & & & & \\ & & \beta & & & & & \\ & & & \beta & & & & \\ & & & & \beta & & & \\ & & & & & \beta & & \\ & & & & & & \beta & \\ & & & & & & & \beta \\ \beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} \beta & \beta & & & & & & \\ & \beta & \beta & & & & & \\ & & \beta & \beta & & & & \\ & & & \beta & \beta & & & \\ & & & & \beta & \beta & & \\ & & & & & \beta & \beta & \\ & & & & & & \beta & \beta \\ & & & & & & & \beta & \beta \\ & & & & & & & & \beta & \beta \\ & & & & & & & & & \beta & \beta \\ & & & & & & & & & & \beta & \beta \\ & & & & & & & & & & & \beta & \beta \\ & & & & & & & & & & & & \beta & \beta \\ & & & & & & & & & & & & & \beta & \beta \end{bmatrix} + \begin{bmatrix} \gamma - 2\beta & & & & & & & \\ & \alpha & & & & & & \\ & & \alpha & & & & & \\ & & & \alpha & & & & \\ & & & & \gamma & & & \\ & & & & & \gamma & & \\ & & & & & & \gamma & \\ & & & & & & & \gamma & \\ & & & & & & & & \gamma & \\ & & & & & & & & & \gamma & \\ & & & & & & & & & & \gamma & \\ & & & & & & & & & & & \gamma & \\ & & & & & & & & & & & & \gamma & \\ & & & & & & & & & & & & & \gamma & \end{bmatrix}. \quad (10)$$

The first matrix in Equation (10) can be written as a Kronecker product:

$$\sqrt{2}\beta \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} =: \sqrt{2}\beta \mathbf{B} \otimes \mathbf{C}.$$

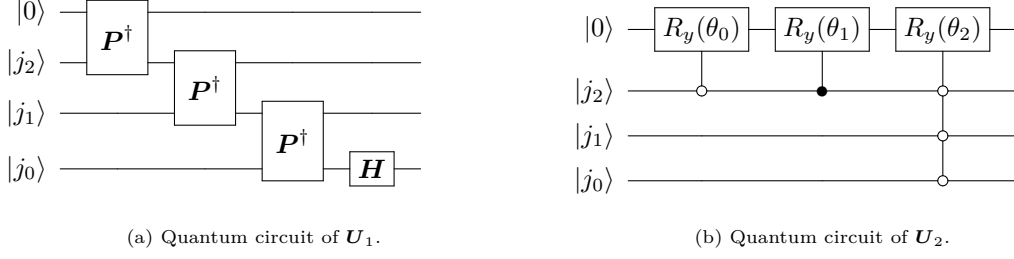


Figure 6: Block-encodings of the three matrices in the decomposition Equation (10). Subfigure (a) shows the block-encoding of the first matrix, whose inverse gives that of the second matrix, while subfigure (b) shows the block-encoding of the third matrix.

By Theorem 1, the block-encodings of  $\mathbf{B}$  and  $\mathbf{C}$  can be chosen as

$$\mathbf{I}_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{and} \quad \mathbf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

To convert  $\mathbf{I}_8 \otimes \mathbf{H}$  into a block-encoding of  $\mathbf{B} \otimes \mathbf{C}$ , we apply Theorem 3 and obtain the permutation matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

which can be implemented with two CNOT gates. Consequently, the unitary

$$\mathbf{U}_1 := (\mathbf{I}_8 \otimes \mathbf{H}) \cdot (\mathbf{I}_4 \otimes \mathbf{P}^\dagger) \cdot (\mathbf{I}_2 \otimes \mathbf{P}^\dagger \otimes \mathbf{I}_2) \cdot (\mathbf{P}^\dagger \otimes \mathbf{I}_4)$$

serves as a block-encoding of  $\mathbf{B} \otimes \mathbf{C}$ , and the corresponding quantum circuit is shown in Figure 6a. To verify, note that  $\mathbf{P}^\dagger|0b\rangle = |b0\rangle$ , for all  $b \in \{0, 1\}$ . Applying the three  $\mathbf{P}^\dagger$  layers followed by  $\mathbf{H}$  gives

$$\begin{aligned} |0j_2j_1j_0\rangle &\longrightarrow |j_20j_1j_0\rangle \longrightarrow |j_2j_10j_0\rangle \longrightarrow |j_2j_1j_00\rangle \\ &\longrightarrow \frac{1}{\sqrt{2}}|j_2j_1j_0\rangle (|0\rangle + |1\rangle). \end{aligned}$$

Taking the overlap with  $|0i_2i_1i_0\rangle$  yields

$$\langle 0i_2i_1i_0 | \mathbf{U}_1 | 0j_2j_1j_0 \rangle = \begin{cases} 0, & j_2 = 1, \\ \frac{1}{\sqrt{2}}, & j_2 = 0 \text{ and } (i_2, i_1) = (j_1, j_0), \\ 0, & \text{otherwise.} \end{cases}$$

Equivalently, with  $j := 4j_2 + 2j_1 + j_0$  and  $i := 4i_2 + 2i_1 + i_0$ , the nonzero entries appear precisely when  $j_2 = 0$  and  $i \in \{2j, 2j + 1\}$ , each with value  $1/\sqrt{2}$ . This exactly reproduces the structure of  $\mathbf{B} \otimes \mathbf{C}$ .

In the same way, a block-encoding of the second matrix in Equation (10) can be obtained either by direct construction or by taking the inverse of  $\mathbf{U}_1$ . Hence, block-encodings of the first two terms in the decomposition are established.

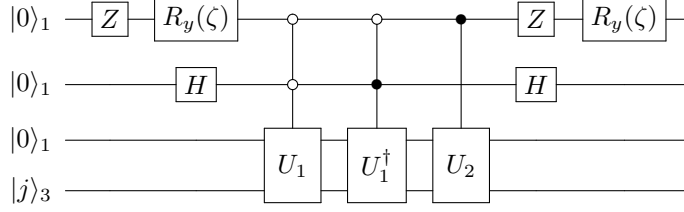


Figure 7: Block-encoding of the adjacent matrix of an extended binary tree.

For the third term in Equation (10), which is diagonal, we employ the block-encoding circuit  $\mathbf{U}_2$  shown in Figure 6b. Define

$$\tilde{\alpha} := \cos \frac{\theta_0}{2}, \quad \tilde{\gamma} := \cos \frac{\theta_1}{2}, \quad \hat{\gamma} := \cos \frac{\theta_0 + \theta_2}{2},$$

then the top-left  $8 \times 8$  block of  $\mathbf{U}_2$  is

$$\text{diag}(\hat{\gamma}, \tilde{\alpha}, \tilde{\alpha}, \tilde{\alpha}, \tilde{\gamma}, \tilde{\gamma}, \tilde{\gamma}, \tilde{\gamma}).$$

The parameters  $\tilde{\alpha}, \tilde{\gamma}, \hat{\gamma}$  will be specified later. Next, we perform LCU on the block-encodings of the three matrices, obtaining

$$x\mathbf{B} \otimes \mathbf{C} + x\mathbf{B}^T \otimes \mathbf{C}^T + (1 - 2x) \text{diag}(\hat{\gamma}, \tilde{\alpha}, \tilde{\alpha}, \tilde{\alpha}, \tilde{\gamma}, \tilde{\gamma}, \tilde{\gamma}, \tilde{\gamma}) = \frac{1}{c}\mathbf{A}.$$

Here, the parameters  $x, c, \hat{\gamma}, \tilde{\alpha}, \tilde{\gamma}$  are chosen such that

$$\begin{cases} \frac{x}{\sqrt{2}} = \frac{\beta}{c}, \\ (1 - 2x)\hat{\gamma} + \sqrt{2}x = \frac{\gamma}{c}, \\ (1 - 2x)\tilde{\gamma} = \frac{\gamma}{c}, \\ (1 - 2x)\tilde{\alpha} = \frac{\alpha}{c}. \end{cases} \implies \begin{cases} x = \frac{\sqrt{2}\beta}{c}, \\ \hat{\gamma} = \frac{\gamma - 2\beta}{c - 2\sqrt{2}\beta}, \\ \tilde{\gamma} = \frac{\gamma}{c - 2\sqrt{2}\beta}, \\ \tilde{\alpha} = \frac{\alpha}{c - 2\sqrt{2}\beta}. \end{cases} \quad (11)$$

To implement this procedure on a quantum computer, the parameters must further satisfy

$$0 \leq x \leq \frac{1}{2}, \quad |\hat{\gamma}|, |\tilde{\alpha}|, |\tilde{\gamma}| \leq 1.$$

By choosing  $c = 2\sqrt{2} + 2$ , these conditions can be fulfilled since  $\alpha, \beta, \gamma \in (0, 1)$ . For explicitly given values of  $\alpha, \beta, \gamma$ , one may select a smaller  $c$ . Setting  $\zeta = 2 \arccos \sqrt{2x}$ , we obtain

$$[R_y(\zeta) \cdot Z] \otimes H = \begin{bmatrix} \sqrt{2x} & \sqrt{1-2x} \\ \sqrt{1-2x} & -\sqrt{2x} \end{bmatrix} \otimes \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} \sqrt{x} & \sqrt{x} & \sqrt{\frac{1}{2}-x} & \sqrt{\frac{1}{2}-x} \\ \sqrt{x} & * & * & * \\ \sqrt{\frac{1}{2}-x} & * & * & * \\ \sqrt{\frac{1}{2}-x} & * & * & * \end{bmatrix}.$$

In summary, the quantum circuit for the block-encoding of  $\frac{1}{c}\mathbf{A}$  is given in Figure 7. This construction naturally generalizes to extended binary trees of greater depth. Suppose the tree has depth  $n$ . Then  $\mathbf{A}$  can be block-encoded using  $O(n^2)$  quantum gates and 3 ancillary qubits. The  $O(n^2)$  complexity arises from the multi-controlled rotation  $R_y(\theta_2)$  required in block-encoding the diagonal part, while the remaining components of the block-encoding can be implemented with only  $O(n)$  gates.

In the special case  $\beta \in (0, 1/3)$  with  $\alpha = 1 - 3\beta$  and  $\gamma = 1 - \beta$ , i.e., when  $\mathbf{A}$  is a symmetric stochastic matrix, setting  $c = 1$  suffices, and the block-encoding of  $\mathbf{A}$  becomes exact. Moreover, in this regime we have

$\hat{\gamma} = \tilde{\alpha}$ , which allows us to eliminate the final multi-controlled rotation  $R_y(\theta_2)$  in Figure 6b. The overall gate complexity of the block-encoding then reduces to  $O(n)$ .

In contrast, the construction of [11] requires 5 ancilla qubits and a more involved circuit to obtain a block-encoding with normalization  $c = 8$ . Our construction uses only 3 ancillas and achieves a strictly better constant: in general we can choose  $c \leq 2\sqrt{2} + 2 < 5$ , and in favorable parameter regimes even  $c = 1$ . This demonstrates the benefit of identifying and exploiting the Kronecker-sum structure.

#### 4. Conclusion

In this work we developed new techniques for implementing matrix products within the block-encoding framework, extending the scope and efficiency of existing constructions. First, we addressed the general case where the matrices are not necessarily square and their dimensions are not restricted to powers of two, thereby broadening the applicability of block-encoded operations. Second, we introduced a qubit-efficient method for realizing matrix-matrix multiplications that significantly reduces ancilla requirements at the cost of a modest increase in gate complexity. This improvement is particularly pronounced in the sequential product setting, where the number of required ancilla qubits decreases exponentially. In addition, we refined the implementation of the Kronecker product compared to prior work [21], replacing SWAP operations with pairs of CNOT gates and thus lowering gate cost. By applying the same qubit-efficient principle from our matrix-matrix multiplication construction, we also developed a qubit-efficient Kronecker product implementation. Moreover, we rederived the Hadamard product between block-encodings and extend its construction to the convolution operation and vectorization operator. Finally, we demonstrated the utility of these basic product operations in applications such as time-dependent Hamiltonian simulation and the block-encoding of the adjacent matrix of an extended binary tree, where our constructions yield concrete resource reductions relative to existing approaches.

Taken together, our results enrich the toolbox for manipulating block-encodings, offering both qubit efficiency and structural flexibility. Looking ahead, it will be of interest to integrate these techniques into practical quantum algorithms—such as Hamiltonian simulation and quantum optimization—and to further explore linear operations and their applications in emerging areas of quantum computational science.

#### Appendix A. Qubit-efficient implementation of Kronecker product

Similarly to the matrix-matrix product, the Kronecker product between block-encoded matrices admits a qubit-efficient implementation.

For simplicity, consider first the case where  $\mathbf{B} \in \mathbb{C}^{2^s \times 2^s}$  and  $\mathbf{C} \in \mathbb{C}^{2^t \times 2^t}$  are square matrices whose dimensions are powers of two, and let their block-encodings be  $\mathbf{U}_\mathbf{B} \in \mathbb{C}^{2^b \times 2^b}$  and  $\mathbf{U}_\mathbf{C} \in \mathbb{C}^{2^c \times 2^c}$ . Without loss of generality, assume that  $\mathbf{U}_\mathbf{C}$  uses more ancilla qubits, i.e.,  $c - t > b - s$ . Noting that

$$\mathbf{B} \otimes \mathbf{C} = (\mathbf{I}_{2^b} \otimes \mathbf{C})(\mathbf{B} \otimes \mathbf{I}_{2^c}),$$

and the block-encodings of  $\mathbf{I}_{2^b} \otimes \mathbf{C}$  and  $\mathbf{B} \otimes \mathbf{I}_{2^c}$  are obtained directly from  $\mathbf{U}_\mathbf{C}$  and  $\mathbf{U}_\mathbf{B}$ , respectively, we may apply the qubit-efficient matrix-matrix product construction to realize the block-encoding of  $\mathbf{B} \otimes \mathbf{C}$ . The resulting quantum circuit is presented in Figure A.8; it implements a block-encoding of  $\mathbf{B} \otimes \mathbf{C}$  while requiring only one additional ancilla qubit beyond those used by  $\mathbf{U}_\mathbf{B}$  and  $\mathbf{U}_\mathbf{C}$ . The effect of this circuit on the input state  $|0\rangle_1|0\rangle_{c-t}|x\rangle_s|y\rangle_t$  is as follows:

$$\begin{aligned} |0\rangle_1|0\rangle_{c-t}|x\rangle_s|y\rangle_t &\longrightarrow |1\rangle_1(|0\rangle_{c-t}\mathbf{B}|x\rangle_b|y\rangle_c + |g\rangle) \\ &\longrightarrow |0\rangle_1|0\rangle_{c-t}\mathbf{B}|x\rangle_b|y\rangle_c + |1\rangle_1|g\rangle \\ &\longrightarrow |0\rangle_1|0\rangle_{c-t}\mathbf{B}|x\rangle_b\mathbf{C}|y\rangle_c + |0\rangle_1|g'\rangle + |1\rangle_1|g''\rangle, \end{aligned}$$

where both of the last two terms are orthogonal to the subspace  $|0\rangle_{c-t+1} \otimes \mathbf{I}_{2^{s+t}}$ . Consequently, the circuit implements a block-encoding of  $\mathbf{B} \otimes \mathbf{C}$  using only one additional ancilla qubit.

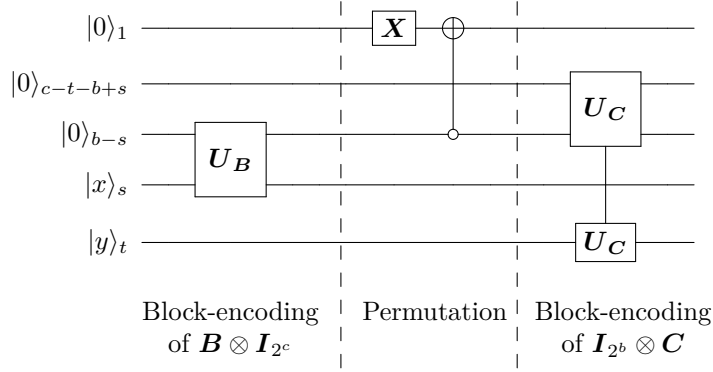


Figure A.8: Quantum circuit for qubit-efficient implementation of the Kronecker product. Two barriers divide the circuit into three phases. The left and right phases implement the block-encodings of  $B \otimes I_{2^c}$  and  $I_{2^b} \otimes C$ , respectively. The middle phase applies the permutation used in Example 1.

For the general case where  $B$  and  $C$  have arbitrary sizes, the same construction applies after embedding  $B$  and  $C$  into larger  $2^s \times 2^s$  and  $2^t \times 2^t$  blocks. The output registers  $|\cdot\rangle_s$  and  $|\cdot\rangle_t$  can then be rearranged as in Section 2.1 so that  $B \otimes C$  occupies the top-left corner of the overall unitary.

## References

- [1] D. W. Berry, A. M. Childs, R. Kothari, [Hamiltonian simulation with nearly optimal dependence on all parameters](#), in: Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS), FOCS '15, IEEE Computer Society, USA, 2015, p. 792–809. [doi:10.1109/FOCS.2015.54](#).  
URL <https://doi.org/10.1109/FOCS.2015.54>
- [2] G. H. Low, N. Wiebe, [Hamiltonian simulation in the interaction picture](#) (2019). [arXiv:1805.00675](#).  
URL <https://arxiv.org/abs/1805.00675>
- [3] A. W. Harrow, A. Hassidim, S. Lloyd, [Quantum algorithm for linear systems of equations](#), Phys. Rev. Lett. 103 (2009) 150502. [doi:10.1103/PhysRevLett.103.150502](#).  
URL <https://link.aps.org/doi/10.1103/PhysRevLett.103.150502>
- [4] V. Simoncini, [Computational methods for linear matrix equations](#), SIAM Review 58 (3) (2016) 377–441. [arXiv:https://doi.org/10.1137/130912839](#), [doi:10.1137/130912839](#).  
URL <https://doi.org/10.1137/130912839>
- [5] P. C. Costa, D. An, Y. R. Sanders, Y. Su, R. Babbush, D. W. Berry, [Optimal scaling quantum linear-systems solver via discrete adiabatic theorem](#), PRX Quantum 3 (2022) 040303. [doi:10.1103/PRXQuantum.3.040303](#).  
URL <https://link.aps.org/doi/10.1103/PRXQuantum.3.040303>
- [6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, [Quantum machine learning](#), Nature 549 (7671) (2017) 195–202. [doi:10.1038/nature23474](#).  
URL <https://doi.org/10.1038/nature23474>
- [7] E. Farhi, J. Goldstone, S. Gutmann, [A quantum approximate optimization algorithm](#) (2014). [arXiv:1411.4028](#).  
URL <https://arxiv.org/abs/1411.4028>

- [8] T. Albash, D. A. Lidar, [Adiabatic quantum computation](#), Rev. Mod. Phys. 90 (2018) 015002. doi: [10.1103/RevModPhys.90.015002](https://doi.org/10.1103/RevModPhys.90.015002).  
URL <https://link.aps.org/doi/10.1103/RevModPhys.90.015002>
- [9] A. Gilyén, Y. Su, G. H. Low, N. Wiebe, [Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics](#), in: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Association for Computing Machinery, New York, NY, USA, 2019, p. 193–204. doi: [10.1145/3313276.3316366](https://doi.org/10.1145/3313276.3316366).  
URL <https://doi.org/10.1145/3313276.3316366>
- [10] G. H. Low, I. L. Chuang, [Optimal hamiltonian simulation by quantum signal processing](#), Phys. Rev. Lett. 118 (2017) 010501. doi: [10.1103/PhysRevLett.118.010501](https://doi.org/10.1103/PhysRevLett.118.010501).  
URL <https://link.aps.org/doi/10.1103/PhysRevLett.118.010501>
- [11] D. Camps, L. Lin, R. Van Beeumen, C. Yang, [Explicit quantum circuits for block encodings of certain sparse matrices](#), SIAM Journal on Matrix Analysis and Applications 45 (1) (2024) 801–827. arXiv: <https://arxiv.org/abs/2204.04812>, doi: [10.1137/22M1484298](https://doi.org/10.1137/22M1484298).  
URL <https://doi.org/10.1137/22M1484298>
- [12] C. Sünderhauf, E. Campbell, J. Camps, [Block-encoding structured matrices for data input in quantum computing](#), Quantum 8 (2024) 1226. doi: [10.22331/q-2024-01-11-1226](https://doi.org/10.22331/q-2024-01-11-1226).  
URL <https://doi.org/10.22331/q-2024-01-11-1226>
- [13] D. Liu, W. Du, L. Lin, J. P. Vary, C. Yang, [An efficient quantum circuit for block encoding a pairing hamiltonian](#), Journal of Computational Science 85 (2025) 102480. doi: <https://doi.org/10.1016/j.jocs.2024.102480>.  
URL <https://www.sciencedirect.com/science/article/pii/S1877750324002734>
- [14] D. Camps, R. Van Beeumen, [Fable: Fast approximate quantum circuits for block-encodings](#), in: 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), 2022, pp. 104–113. doi: [10.1109/QCE53715.2022.00029](https://doi.org/10.1109/QCE53715.2022.00029).
- [15] N. Guseynov, X. Huang, N. Liu, [Efficient explicit gate construction of block-encoding for hamiltonians needed for simulating partial differential equations](#) (2025). arXiv: [2405.12855](https://arxiv.org/abs/2405.12855).  
URL <https://arxiv.org/abs/2405.12855>
- [16] Q. T. Nguyen, B. T. Kiani, S. Lloyd, [Block-encoding dense and full-rank kernels using hierarchical matrices: applications in quantum numerical linear algebra](#), Quantum 6 (2022) 876. doi: [10.22331/q-2022-12-13-876](https://doi.org/10.22331/q-2022-12-13-876).  
URL <https://doi.org/10.22331/q-2022-12-13-876>
- [17] C. Yang, Z. Li, H. Yao, Z. Fan, G. Zhang, J. Liu, [Dictionary-based block encoding of sparse matrices with low subnormalization and circuit depth](#) (2025). arXiv: [2405.18007](https://arxiv.org/abs/2405.18007), doi: <https://doi.org/10.22331/q-2025-07-22-1805>.  
URL <https://arxiv.org/abs/2405.18007>
- [18] S. Jin, N. Liu, Y. Yu, [Quantum circuits for the heat equation with physical boundary conditions via schrödingerization](#), Journal of Computational Physics 538 (2025) 114138. doi: <https://doi.org/10.1016/j.jcp.2025.114138>.  
URL <https://www.sciencedirect.com/science/article/pii/S0021999125004218>
- [19] K. Miyamoto, K. Kubo, [Pricing Multi-Asset Derivatives by Finite-Difference Method on a Quantum Computer](#), IEEE Transactions on Quantum Engineering 3 (01) (2022) 1–25. doi: [10.1109/TQE.2021.3128643](https://doi.org/10.1109/TQE.2021.3128643).  
URL <https://doi.ieeecomputersociety.org/10.1109/TQE.2021.3128643>

- [20] D. Dong, Y. Li, J. Xue, [A quantum algorithm for linear autonomous differential equations via Padé approximation](#), Quantum 9 (2025) 1770. doi:[10.22331/q-2025-06-17-1770](https://doi.org/10.22331/q-2025-06-17-1770).  
URL <https://doi.org/10.22331/q-2025-06-17-1770>
- [21] D. Camps, R. Van Beeumen, [Approximate quantum circuit synthesis using block encodings](#), Phys. Rev. A 102 (2020) 052411. doi:[10.1103/PhysRevA.102.052411](https://doi.org/10.1103/PhysRevA.102.052411).  
URL <https://link.aps.org/doi/10.1103/PhysRevA.102.052411>
- [22] N. Guo, Z. Yu, M. Choi, A. Agrawal, K. Nakaji, A. Aspuru-Guzik, P. Rebentrost, [Quantum linear algebra is all you need for transformer architectures](#) (2024). arXiv:[2402.16714](https://arxiv.org/abs/2402.16714).  
URL <https://arxiv.org/abs/2402.16714>
- [23] S. Jin, X. Li, N. Liu, Y. Yu, [Quantum simulation for partial differential equations with physical boundary or interface conditions](#), Journal of Computational Physics 498 (2024) 112707. doi:<https://doi.org/10.1016/j.jcp.2023.112707>.  
URL <https://www.sciencedirect.com/science/article/pii/S0021999123008021>
- [24] D. An, A. M. Childs, L. Lin, [Quantum algorithm for linear non-unitary dynamics with near-optimal dependence on all parameters](#) (2023). arXiv:[2312.03916](https://arxiv.org/abs/2312.03916).  
URL <https://arxiv.org/abs/2312.03916>
- [25] D. W. Berry, P. C. S. Costa, [Quantum algorithm for time-dependent differential equations using Dyson series](#), Quantum 8 (2024) 1369. doi:[10.22331/q-2024-06-13-1369](https://doi.org/10.22331/q-2024-06-13-1369).  
URL <https://doi.org/10.22331/q-2024-06-13-1369>
- [26] D. An, L. Lin, [Quantum linear system solver based on time-optimal adiabatic quantum computing and quantum approximate optimization algorithm](#), ACM Transactions on Quantum Computing 3 (2) (Mar. 2022). doi:[10.1145/3498331](https://doi.org/10.1145/3498331).  
URL <https://doi.org/10.1145/3498331>
- [27] S. A. Cuccaro, T. G. Draper, S. A. Kutin, D. P. Moulton, [A new quantum ripple-carry addition circuit](#) (2004). arXiv:[quant-ph/0410184](https://arxiv.org/abs/quant-ph/0410184).  
URL <https://arxiv.org/abs/quant-ph/0410184>
- [28] T. G. Draper, [Addition on a quantum computer](#) (2000). arXiv:[quant-ph/0008033](https://arxiv.org/abs/quant-ph/0008033).  
URL <https://arxiv.org/abs/quant-ph/0008033>
- [29] S. Beauregard, [Circuit for shor's algorithm using  \$2n+3\$  qubits](#) (2003). arXiv:[quant-ph/0205095](https://arxiv.org/abs/quant-ph/0205095).  
URL <https://arxiv.org/abs/quant-ph/0205095>
- [30] D. W. Berry, A. M. Childs, A. Ostrander, G. Wang, [Quantum algorithm for linear differential equations with exponentially improved dependence on precision](#), Communications in Mathematical Physics 356 (2017) 1057–1081. arXiv:<https://doi.org/10.1007/s00220-017-3002-y>, doi:[10.1007/s00220-017-3002-y](https://doi.org/10.1007/s00220-017-3002-y).  
URL <https://doi.org/10.1007/s00220-017-3002-y>
- [31] S. Jin, N. Liu, C. Ma, [On schrödingerization based quantum algorithms for linear dynamical systems with inhomogeneous terms](#) (2024). arXiv:[2402.14696](https://arxiv.org/abs/2402.14696).  
URL <https://arxiv.org/abs/2402.14696>