Dictionary-based Block Encoding of Sparse Matrices with Low Subnormalization and Circuit Depth

Chunlin Yang [*1], Zexian Li [*2], Hongmei Yao[1], Zhaobing Fan[1], Guofeng Zhang[2], and Jianshe Liu[3]

[1]School of Mathematical and Sciences, Harbin Engineering University, China
[2]Department of Applied Mathematics, The Hong Kong Polytechnic University, China
[3]College of Underwater Acoustic Engineering, Harbin Engineering University, China

Block encoding severs as an important data input model in quantum algorithms, enabling quantum computers to simulate non-unitary operators effectively. In this paper, we propose an efficient block-encoding protocol for sparse matrices based on a novel data structure, called the dictionary data structure, which classifies all non-zero elements according to their values and indices. Non-zero elements with the same values, lacking common column and row indices, belong to the same classification in our block-encoding protocol's dictionary. When compiled into the {U(2), CNOT} gate set, the protocol queries a $2^n \times 2^n$ sparse matrix with s non-zero elements at a circuit depth of $\mathcal{O}(\log(ns))$, utilizing $\mathcal{O}(n^2 s)$ ancillary qubits. This offers an exponential improvement in circuit depth relative to the number of system qubits, compared to existing methods [1, 2] with a circuit depth of $\mathcal{O}(n)$. Moreover, in our protocol, the subnormalization, a scaled factor that influences the measurement probability of ancillary qubits, is minimized to $\sum_{l=0}^{s_0} |A_l|$, where s_0 denotes the number of classifications in the dictionary and A_l represents the value of the l-th classification. Furthermore, we show that our protocol connects to linear combinations of unitaries (LCU) and the sparse access input model (SAIM). To demonstrate the practical utility of our approach, we provide several applications, including Laplacian matrices in graph problems and discrete differential operators.

Contents

Hongmei Yao: hongmeiyao@163.com

*These authors contributed equally to this work

1 Introduction

Quantum algorithms have demonstrated unprecedented potential in solving classically intractable problems, exemplified by landmark algorithms such as the Deutsch-Jozsa algorithm [3], Shor's algorithm [4], and the HHL algorithm [5]. A critical enabler of these advancements lies in efficient data input models, which bridge classical information with quantum processing. Commonly used models include block encoding [6], SAIM [5–13], and LCU [14, 15]. Recent studies have shown that many quantum algorithms can be unified and reconstructed within the framework of quantum singular value transformation (QSVT) [6, 16]. These algorithms rely on the block encoding, enabling the embedding of arbitrary matrices into unitary operators,

$$U_A = \begin{pmatrix} A/\alpha & * \\ * & * \end{pmatrix},$$

where $*$ denotes a matrix block yet to be determined. The factor α, called *subnormalization*, scales A to ensure $\|A/\alpha\|_{\text{sp}} \leq 1$, since the singular values of any matrix block of a unitary must not be larger than 1, where $\|\cdot\|_{\text{sp}}$ denotes the spectral norm.

Given the important role of block encoding in quantum computation, substantial research efforts have been dedicated to optimizing its implementation. Special matrices including density operators, the hierarchical matrices, pseudo-differential operators, and pairing Hamiltonians, have been studied in [17–21]. Generally, for dense matrices, Kerenidis and Prakash [22], and Chakraborty et al. [12] showed how to implement block encoding efficiently of matrices stored in quantum data structures. Based on the quantum random access memory query model, Clader et al. [1] developed several block-encoding methods for a dense matrix of classical data. Using single- and two-qubit gates, a method, called FABLE [23, 24], was proposed to generate approximate quantum circuits for block-encoding matrices in a fast manner, while Li et al. [25] improved it by fast computational methods for block encoding classical matrices with few ancillary qubits, low computing time, and low quantum gate counts. For sparse matrices, Gilyén et al. [6] introduced a foundational block encoding, which was subsequently improved by using preamplification. If a sparse matrix has a well-defined structure, Camps et al. [26] presented a scheme to efficiently block encode it. Furthermore, if a sparse matrix has an arithmetical structure, Sünderhauf et al. [27] developed a novel block encoding scheme and provided two improved schemes using preamplification and state preparation to reduce *subnormalization*.

The structural characteristics of matrices significantly influence the implementation and resource requirements of algorithms, which in turn affect their efficiency. Sparse structured matrices, in particular, stand out due to their substantial number of zero elements and the regular distribution of non-zero elements. They have widely used applications, including fluid mechanics [28, 29], image processing [30, 31], and network analysis [32, 33]. Thanks to the characteristics, it is possible to store only non-zero elements to compress storage.

Despite the advantages of sparse structured matrices, existing block encoding schemes face certain limitations when dealing with them. Block encoding in [26] requires that every data value should appear in all columns, while the PREP/UNPREP scheme in [27] requires $\lceil \log D \rceil \leq \lceil \log S_c \rceil, \lceil \log S_r \rceil$, where D is the number of distinct data values, S_c, S_r are the maximum column and row sparsities, respectively. These constraints limit the applicability and flexibility of block encoding schemes. To address these issues, this paper proposes a block encoding scheme of sparse matrices based on a dictionary data structure. Crucially, the block-encoding protocol optimizes the trade-off between *subnormalization* and *circuit depth*. This enhancement allows for more efficient and flexible handling of sparse structured matrices in quantum algorithms, advancing the practical application of quantum computing.

The complexity of a block encoding scheme is related to the design of the scheme and the properties of the encoded matrix. To assess the cost of a block encoding, a *figure of merit* [27] is defined as

$$(T\text{-}gate\ count) \cdot subnormalization.$$

The *subnormalization* is a factor to scale the matrix. A lower *subnormalization* can increase the probability to measure $|0\rangle$ for ancillary qubits and lead to shorter circuits of algorithms within the framework of block encoding. The *circuit depth* can reflect the time consumption of implementing quantum circuits. In this article, inspired by above *figure of merit*, we introduce the definition of *time metric* for block-encoding protocols,

$$time\ metric = circuit\ depth \times subnormalization, \tag{1.1}$$

as the quantitative indicator of the time complexity, where the *circuit depth* is the maximum number of layers of element gates in a quantum circuit. A lower *circuit depth* can enhance the feasibility and efficiency of quantum algorithms. In algorithms within the framework of QSVT, the time complexity is commonly characterized by the query complexity of block encoding, which can be directly translated into *circuit depth*. This correspondence establishes that the *time metric* defined in Equation (1.1) provides a unified framework to analytically quantify the time complexity of quantum algorithms within the framework of QSVT.

Our main contribution in this paper is summarized as follows.

- **Dictionary Data Structure:** We introduce a new data structure for sparse structured matrices, namely, dictionary data structure, which is shown in Table 3. It depends on the classification of all triplets of non-zero matrix elements. Within this framework, we can unify several existing block encoding schemes of sparse matrices, details in Appendix A.1.

- **Dictionary-based block encoding of sparse matrices:** Based on the dictionary data structure in Table 3, we provide a dictionary-based block encoding of sparse matrices in Theorem 3.1 with the *subnormalization* $\sum_{l=0}^{s_0-1} |A_l|$, where s_0 is the number of data items. For non-negative symmetric matrices, we extend the Hermitian version in Theorem 3.3 based on the dictionary data structure in Table 5. We also demonstrate that our dictionary-based block encoding of sparse matrices is a linear combination of unitaries.

- **Circuit Depth Optimization:** We analyze the *circuit depth* of the dictionary-based block encoding of sparse matrices, achieving a scaling of $\mathcal{O}\big(\log(ns)\big)$ (Theorem 4.5), where n denotes the number of qubits encoding the matrix, s is the number of non-zero elements in the matrix. We also analyze the *circuit depth* of the PREP/UNPREP block encoding in [27] as well as the block encoding using controlled state preparation in [1]. These results are summarized in Table 1.

This paper is organized as follows. Section 2 establishes foundational notations and conventions. In Section 3, we develop a dictionary data structure framework and propose a dictionary-based block encoding of sparse matrices with reduced *subnormalization*, subsequently extending it to the Hermitian version for non-negative symmetric matrices. We further establish connections between this block encoding scheme and LCU, while rigorously analyzing its logarithmic *circuit depth* implementation. Comparative benchmarks are provided against the PREP/UNPREP protocol in [27] and controlled state preparation methods in [1]. Section 5 demonstrates practical implementations for the signless Laplacian matrices in graphs and matrices of discrete differential

Block Encoding	Circuit Depth		Ancillary qubits	Subnormalization
PREP/UNPREP [27]	Lemma B.1	$\mathcal{O}\left(n2^{n/2}\right)$	$\Omega(4^n/n)$	$\frac{\sqrt{S_c S_r}}{D}\sum_{d=0}^{D-1}\lvert A_d\rvert$
[1, 2]	Lemma B.2	$\mathcal{O}\left(n\right)$	$\mathcal{O}(2^{2n})$	$\lVert A\rVert_F$
Dictionary-based block encoding (**This work**)	Theorem 4.5	$\mathcal{O}\left(\log(ns)\right)$	$\mathcal{O}(n^2 s)$	$\sum_{l=0}^{s_0-1}\lvert A_l\rvert$

Table 1: *Circuit depth* and *subnormalization* of block-encoding protocols for encoding a sparse structured matrix $A \in \mathbb{C}^{2^n\times 2^n}$ with s non-zero elements. The matrix A can be encoded by a dictionary data structure specified in Table 4 with s_0 data items under our block encoding (or D data items under PREP/UNPREP block encoding [27]). Here, S_c and S_r denote column and row sparsity, respectively. The number of data items is always not large than the number of non-zero elements in a matrix, that is, $s_0 \leq s$. For matrices with repeated elements, our approach achieves superior *subnormalization* compared to the low-depth block-encoding protocols proposed in [1, 2]. This advantage is particularly evident in the *time metric* defined in Equation (1.1), where our protocols offer an exponential improvement.

operators, verifying the feasibility of the proposed block encoding of sparse matrices with Python code (https://github.com/ChunlinYangHEU/DIBLE). Conclusions and future directions are presented in Section 6. Appendix A details existing dictionary-based block-encoding protocols of sparse matrices, with *circuit depth* comparisons in Appendix B. The application to generalized eigenvalue problems (GEPs) in ocean acoustics appears in Appendix C.

2 Notations and Conventions

In this section, we introduce the notations and conventions used throughout this paper.

Let $[m,n] = \{m, m+1, \cdots, n\}$, with the convention that $[n] = [1,n]$. $|0\rangle$ denotes the vector $e_0 = (1,0)^T$, and $|1\rangle$ denotes the vector $e_1 = (0,1)^T$. The m-fold tensor product $|0\rangle \otimes \cdots \otimes |0\rangle$ is compactly denoted $|0\rangle^{\otimes m}$. We denote the $N \times N$ identity matrix by I_N. For the special case of 2×2 identity matrices (common in quantum computing), we simply write $I \equiv I_2$. The row and column indices of a matrix always start from 0 in this paper. The j-th column of I_N is denoted by $|j\rangle$, where $j \in [0, N-1]$. For a nonnegative integer $j \in [0, 2^n - 1]$, it has a binary representation

$$j = [j_{n-1}\cdots j_1 j_0] = j_{n-1}\times 2^{n-1} + \cdots + j_1 \times 2^1 + j_0 \times 2^0,$$

where $j_k \in \{0,1\}$, $k \in [0, n-1]$. For a 1-bit binary number a, $\overline{a}$ denotes the NOT operation of a. We denote modulo 2 addition/subtraction as $\oplus$ using an exclusive OR (XOR) operation on the corresponding binary digits of each operand. For the n-bit binary operation, it has

$$\mathrm{XOR}(j,0) = j \oplus 0 = j, \quad \mathrm{XOR}(j,j) = j \oplus j = 0, \quad \forall j \in \mathbb{Z}.$$

The state of a qubit is a superposition of $|0\rangle$ and $|1\rangle$, and a nonnegative integer $j = [j_{n-1}\cdots j_1 j_0]$ can be prepared as a set of quantum states $|j\rangle \equiv |j_{n-1}\rangle \otimes \cdots |j_1\rangle \otimes |j_0\rangle$. For a complex $c = |c|e^{\imath\theta}$, its square root is uniquely defined as $\sqrt{c} = \sqrt{|c|}e^{\imath\theta/2}$.

The letters H, X, Y, and Z are used to represent the Hadamard, Pauli-X, Pauli-Y, and Pauli-Z matrices, respectively, which are defined as follows,

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \quad X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, \quad Y = \begin{pmatrix}0 & -\imath\\ \imath & 0\end{pmatrix}, \quad Z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}.$$

In diagrams of quantum circuits, we follow standard convention in Table 2, which introduces some basic controlled quantum gates used in this paper. The controlled gate is an important quantum gate that uses one or more qubits to control operation on other qubits. A horizontal line represents a single qubit, and a register consisting of multiple qubits is represented by adding a short slash at the beginning of a horizontal line. The rectangular box represents a single- or multi-qubit gate, and the circle represents the control set. For simplicity, if a Pauli-X gate is controlled by one qubit, it is called a controlled X (CNOT) gate.

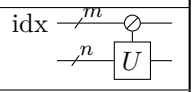
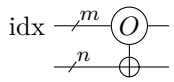
Notation	Mathematical form	Circuit
0-CNOT	$\|0\rangle\langle 0\| \otimes X + \|1\rangle\langle 1\| \otimes I = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	
1-CNOT	$\|1\rangle\langle 1\| \otimes X + \|0\rangle\langle 0\| \otimes I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	
SWAP	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	
	$2n$-qubit SWAP gate is an operator that swaps the qubits of the two n-qubit registers pairwise.	
Multiplexor operation	$U = \sum_{l=0}^{2^m-1} \|l\rangle\langle l\|_{\text{idx}} \otimes U_l$, where U_l are n-qubit unitaries controlled by m-qubit register idx.	
Multiplexed X gate (O-control X gate)	$\sum_{l\in\{O\}} \|l\rangle\langle l\|_{\text{idx}} \bigotimes_{k=1}^{n} [X]_{l,k} + \sum_{l'\notin\{O\}} \|l'\rangle\langle l'\|_{\text{idx}} \otimes I_{2^n}$, where $[X]_{l,k}$ is a Pauli-X or identity operator acting on the kth qubit.	

Table 2: Notations, mathematical forms, and circuits of basic controlled quantum gates.

3 Block Encoding

In this section, we introduce the dictionary data structure for block encoding of sparse structured matrices that unifies multiple approaches. Using this data structure, we present a dictionary-based block encoding scheme of sparse matrices with low *subnormalization*. We further extend this framework to its Hermitian counterpart for non-negative symmetric matrices and analyze the *circuit depth* required for implementation.

3.1 Dictionary Data Structure

A dictionary data structure is a collection of key-value pairs. It allows for the storage and retrieval of data based on a unique key, which is used to access the associated value. A sparse structured matrix is a type of matrix that combines two properties: sparsity (many zero elements) and structured non-zero elements (repetition and predictable arrangement). This type of matrix can be represented by a dictionary data structure, shown in Table 3. The key is a non-negative integer. The associated value is a set consisting of triplets,

$$(\textit{element value}, \textit{row index}, \textit{column index}).$$

For triplets in the l-th key-value pair, they share the same value A_l, while the row and column indices (i, j) belong to the coupled index set $(S_r(l), S_c(l))$. For different key-value pairs, the element values of triplets can be the same.

Keys	Values
0	$\{(a_{ij}, i, j) : a_{ij} = A_0, (i, j) \in (S_r(0), S_c(0))\}$
1	$\{(a_{ij}, i, j) : a_{ij} = A_1, (i, j) \in (S_r(1), S_c(1))\}$
2	$\{(a_{ij}, i, j) : a_{ij} = A_2, (i, j) \in (S_r(2), S_c(2))\}$
$\vdots$	$\vdots$

Table 3: The dictionary data structure of a sparse structured matrix.

The dictionary data structure of a sparse structured matrix is not unique. In the dictionary data structure framework, we define the following terminology.

- A *data item* refers to a key-value pair;
- The key l is called the *data index* and the associated value $\{(a_{ij}, i, j) : a_{ij} = A_l, i \in S_r(l), j \in S_c(l)\}$ is called the *data set*;
- The non-zero value A_l is called the *data value*;
- The index set $S_r(l)$ and $S_c(l)$ are called the *row index set* and *column index set*, respectively. They can be characterized by functions, which are collectively referred to as *data functions*.

Our dictionary framework provides a unified approach to block encoding of sparse matrices, encompassing several existing protocols [6, 24, 26, 27] (see Appendix A.1 for detailed comparisons). Inspired by [26], in this work, we define the following dictionary data structure, which is shown in Table 4. The row and column index sets can be characterized by the following data function:

(1) The row indices $i = c_l(j)$, where $c_l(j)$ is an injective data function that maps column indices j to the corresponding row indices i (unlike the bijective function $c(l, j)$ used in [26]), $l \in [0, s_0 - 1]$;

(2) The column indices $j \in S_c(l)$, where $S_c(l)$ denotes the set that includes all column indices of the l-th data value.

Keys	**Values**
0	$\{(a_{ij}, i, j) : a_{ij} = A_0, (i, j) \in (c_0(j), S_c(0))\}$
1	$\{(a_{ij}, i, j) : a_{ij} = A_1, (i, j) \in (c_1(j), S_c(1))\}$
2	$\{(a_{ij}, i, j) : a_{ij} = A_2, (i, j) \in (c_2(j), S_c(2))\}$
$\vdots$	$\vdots$

Table 4: Dictionary data structure in this article.

In contrast to the block encoding protocol in [26] that mandates the presence of every data value across all columns, our block encoding based on the dictionary data structure in Table 4 eliminates such column-wise distribution constraints, thereby enabling more flexible handling of a wider variety of sparse matrices.

Concrete examples of matrix representations using our dictionary data structure are presented in Section 5, demonstrating its application to various sparse matrix types.

3.2 Dictionary-based Block Encoding of Sparse Matrices

In this section, we introduce the block encoding of sparse matrices and its Hermitian version based on the dictionary data structure in Tables 4 and 5, respectively. The connection between the dictionary-based block encoding of sparse matrices and LCU is explored.

The design rationale behind this encoding strategy is inspired by [26] and [27]. It is an improvement of the block encoding scheme in [26] by removing the limitation that every data value should appear in all columns and using the state preparation technique proposed in [27] to reduce its *subnormalization*.

3.2.1 Dictionary-based Block Encoding of Sparse Matrices and LCU

Based on the dictionary data structure defined in Table 4, our block encoding implementation consists of two quantum operations:

- State Preparation: The oracles PREP and UNPREP perform the amplitude encoding of the data values $\{A_l\}_{l=0}^{s_0-1}$ stored in the dictionary;

- Index Mapping: The oracle O_c implements the injective function $c_l : j \mapsto c_l(j)$ for $l \in [0, s_0 - 1]$, $j \in S_c(l)$.

Leveraging these oracles, we formally present the dictionary-based block encoding scheme of sparse matrices in Theorem 3.1.

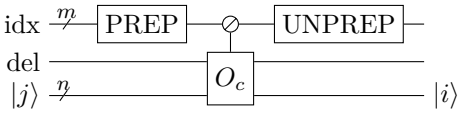


Figure 1: Basic framework of block encoding of sparse matrices with the dictionary data structure in Table 4.

Theorem 3.1. *Let $A \in \mathbb{C}^{2^n \times 2^n}$ be a matrix that can be represented by a dictionary data structure with s_0 data items as stated in Table 4, and $m = \lceil \log_2 s_0 \rceil$. If there exists a column index oracle O_c such that*

$$O_c |l\rangle_{\text{idx}} |0\rangle_{\text{del}} |j\rangle = \begin{cases} |l\rangle_{\text{idx}} |0\rangle_{\text{del}} |c_l(j)\rangle, & \text{if } l \in [0, s_0 - 1] \text{ and } j \in S_c(l), \\ |l\rangle_{\text{idx}} |1\rangle_{\text{del}} |j\rangle, & \text{if } l \in [s_0, 2^m - 1] \text{ or } j \notin S_c(l), \end{cases} \tag{3.1}$$

and two state preparation oracles PREP, UNPREP *such that*

$$\text{PREP} |0\rangle_{\text{idx}}^{\otimes m} = \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|}} \left(\sum_{l=0}^{s_0-1} \sqrt{A_l} |l\rangle_{\text{idx}} + \sum_{l=s_0}^{2^m-1} 0 |l\rangle_{\text{idx}} \right), \tag{3.2}$$

$$\text{UNPREP}^\dagger |0\rangle_{\text{idx}}^{\otimes m} = \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} |A_l|}} \left(\sum_{l=0}^{s_0-1} \sqrt{A_l}^* |l\rangle_{\text{idx}} + \sum_{l=s_0}^{2^m-1} 0 |l\rangle_{\text{idx}} \right), \tag{3.3}$$

where $\sqrt{A_l}^*$ denotes the complex conjugate of $\sqrt{A_l}$, then the unitary

$$U_A = (\text{UNPREP} \otimes I_{2^{n+1}}) O_c (\text{PREP} \otimes I_{2^{n+1}}),$$

as shown in Figure 1, can block encode A with the subnormalization $\alpha = \sum_{l=0}^{s_0-1} |A_l|$.

Proof. To recover the matrix from the block encoding, the index register and the del register must be initialized and postselected as $|0\rangle$. The values $\frac{1}{\sum_{l=0}^{s_0-1} |A_l|} a_{ij}$ are then recovered when initializing the bottom register with $|j\rangle$ and postselecting/measuring an $|i\rangle$ as

$$\begin{aligned}
&\langle 0|_{\text{idx}}^{\otimes m} \langle 0|_{\text{del}} \langle i| (\text{UNPREP} \otimes I_{2^{n+1}}) O_c (\text{PREP} \otimes I_{2^{n+1}}) |0\rangle_{\text{idx}}^{\otimes m} |0\rangle_{\text{del}} |j\rangle \\
=&\frac{1}{\sqrt{\sum_{l'=0}^{s_0-1} |A_{l'}| \sum_{l=0}^{s_0-1} |A_l|}} \sum_{l',l=0}^{s_0-1} \langle l'|_{\text{idx}} \langle 0|_{\text{del}} \langle i| \sqrt{A_{l'} A_l} O_c |l\rangle_{\text{idx}} |0\rangle_{\text{del}} |j\rangle \\
=&\frac{1}{\sqrt{\sum_{l'=0}^{s_0-1} |A_{l'}| \sum_{l=0}^{s_0-1} |A_l|}} \sum_{l',l=0}^{s_0-1} \langle l'|_{\text{idx}} \langle i| \sqrt{A_{l'} A_l} \delta_{l,[0,s_0-1]} \delta_{j,S_c(l)} |l\rangle_{\text{idx}} |c_l(j)\rangle \\
=&\frac{1}{\sqrt{\sum_{l'=0}^{s_0-1} |A_{l'}| \sum_{l=0}^{s_0-1} |A_l|}} \sum_{l',l=0}^{s_0-1} \sqrt{A_{l'} A_l} \delta_{l,[0,s_0-1]} \delta_{j,S_c(l)} \delta_{l',l} \delta_{i,c_l(j)} \\
=&\frac{1}{\sum_{l=0}^{s_0-1} |A_l|} a_{ij},
\end{aligned}$$

where a_{ij} refers to the l-th data value in the dictionary as $\{(a_{ij}, i, j) : a_{ij} = A_l, (i, j) \in (c_l(j), S_c(l))\}$, and $\delta_{i,S}$ is the indicate function as

$$\delta_{i,S} = \begin{cases} 1, & \text{if } i \in S \text{ or } i = S, \\ 0, & \text{otherwise.} \end{cases}$$

□

Remark 1. *We will present a low-depth circuit implementation of the aforementioned block encoding of sparse matrices in Section 4, achieving the subnormalization as* $\sum_l |A_l|$*. Notably, while the theoretical lower bound for subnormalization is given by the spectral norm* $\|A\|_{\mathrm{sp}}$*, the construction of a corresponding low-depth block encoding protocol remains an open challenge in this field.*

Another general approach for encoding sparse Hamiltonians is LCU [14, 15], which constructs quantum circuits in the form,

$$U = \sum_{l=0}^{L-1} c_l U_l, \tag{3.4}$$

where $c_l > 0$ are positive coefficients satisfying $\sum_{l=0}^{L-1} c_l = 1$, U_l are n-qubit unitary operators implementable with polynomial-size circuits, $L = \mathcal{O}(\mathrm{poly}(n))$ scales polynomially with system size n. For practical implementation with low *circuit depth*, it is commonly assumed that the unitaries U_l are self-inverse operators ($U_l^2 = I$) [34].

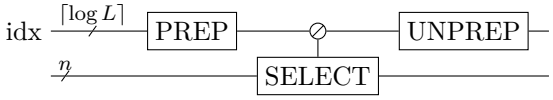


Figure 2: Quantum circuit of LCU [34].

The quantum circuit for LCU is constructed in Figure 2. Its oracles PREP and UNPREP are used to prepare quantum states, that is,

$$\mathrm{PREP}\,|0\rangle^{\otimes\lceil\log L\rceil} = \mathrm{UNPREP}^\dagger\,|0\rangle^{\otimes\lceil\log L\rceil} = \frac{1}{\sqrt{\sum_l c_l}}\sum_l \sqrt{c_l}\,|l\rangle\,,$$

and the SELECT is the Hamiltonian selection oracle [34] defined as

$$\mathrm{SELECT} = \sum_l |l\rangle\langle l| \otimes U_l,$$

where U_l are n-qubit unitaries.

The following corollary establishes the connection between the dictionary-based block encoding of sparse matrices and LCU.

Corollary 3.2. *Let* $A \in \mathbb{C}^{2^n\times 2^n}$ *be a matrix that can be represented by a dictionary data structure with* s_0 *data items as stated in Table 4, and* $m = \lceil\log_2 s_0\rceil$*. Then the dictionary-based block encoding of* A *represents a linear combination of* $(n+1)$*-qubit unitaries.*

Proof. The dictionary-based block encoding of sparse matrices consists of three oracles: O_c, PREP, and UNPREP.

- The oracle O_c in Equation (3.1) can also be expressed as

$$O_{\mathrm{c}} = \sum_{l=0}^{2^m-1} |l\rangle\langle l|_{\mathrm{idx}} \otimes U_l^{(X)},$$

where $U_l^{(X)}$ are $(n+1)$-qubit unitaries as

$$U_l^{(X)} = \bigotimes_{k=0}^{n-1}\left(\left[\mathrm{XOR}\left([c_l(j)]_k, [j]_k\right)\right]X + \overline{\left[\mathrm{XOR}\left([c_l(j)]_k, [j]_k\right)\right]}I\right),$$

satisfying

$$U_l^{(X)}\,|0\rangle_{\mathrm{del}}\,|j\rangle = \begin{cases} |0\rangle_{\mathrm{del}}\,|c_l(j)\rangle\,, & \text{if } l\in[0, s_0-1] \text{ and } j\in S_c(l),\\ |1\rangle_{\mathrm{del}}\,|j\rangle\,, & \text{if } l\in[s_0, 2^m-1] \text{ or } j\notin S_c(l),\end{cases}$$

XOR denotes the Exclusive Or operation of two 1-bit binary operation. Therefore, the oracle O_{c} forms an $(n+1)$-qubit Hamiltonian simulation selection operator.

- The oracles PREP and UNPREP in Equation (3.2) and (3.3) encode the quantum states $\frac{1}{\sqrt{\sum_l |A_l|}} \sum_l \sqrt{A_l} |l\rangle$ and $\frac{1}{\sqrt{\sum_l |A_l|}} \sum_l \sqrt{A_l}^* |l\rangle$, respectively.

These oracles in the dictionary-based block encoding of sparse matrices shown in Figure 3 correspond to the oracles with the same notations of LCU as shown in Figure 2. Above all, the dictionary-based block encoding of $A \in \mathbb{C}^{2^n \times 2^n}$ with s_0 data items is a linear combination of $(n+1)$-qubit unitaries.

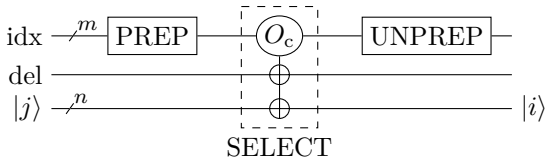


Figure 3: Dictionary-based block encoding of n-qubit sparse matrices is a case of linear combination of $(n+1)$-qubit unitaries, where $O_c = \sum_l |l\rangle\langle l| \otimes U_l^{(X)}$ forms a Hamiltonian selection oracle.

□

3.2.2 Dictionary-based Hermitian Block Encoding of Non-negative Symmetric Matrices

Given that the data functions $c_l(j)$ are injective in j for all $l \in [0, s_0 - 1]$, it is logical that we can construct the corresponding functions $c_j(l)$ for $j \in [0, 2^n - 1]$, which are injective in l. These functions map the data indices $l \in [0, s_0 - 1]$ to the row indices, establishing the dictionary data structure presented in Table 5.

Keys	**Values**
0	$\{(a_{ij}, i, j) : a_{ij} = A_0, (i,j) \in (c_j(0), S_c(0))\}$
1	$\{(a_{ij}, i, j) : a_{ij} = A_1, (i,j) \in (c_j(1), S_c(1))\}$
2	$\{(a_{ij}, i, j) : a_{ij} = A_2, (i,j) \in (c_j(2), S_c(2))\}$
⋮	⋮

Table 5: Dictionary data structure for the Hermitian block encoding of non-negative symmetric matrices, where the data function $i = c_j(l)$ is different from the data function $i = c_l(j)$ stated in Table 4.

Therefore, we propose the following theorem for the dictionary-based Hermitian block encoding.

Theorem 3.3. *Let $A \in \mathbb{R}^{2^n \times 2^n}$ be a non-negative symmetric matrix that can be represented by a dictionary data structure with s_0 data items as stated in Table 5, and $m = \lceil \log_2 s_0 \rceil$, where $s_0 \leq 2^n$. If there exists a column index oracle O_c such that*

$$O_c |0\rangle_{\text{idx}}^{\otimes(n-m)} |l\rangle_{\text{idx}} |0\rangle_{\text{del}} |j\rangle = \begin{cases} |c_j(l)\rangle_{\text{idx}} |0\rangle_{\text{del}} |j\rangle, & \text{if } j \in S_c(l) \text{ and } l \in [0, s_0 - 1], \\ |0\rangle_{\text{idx}}^{\otimes(n-m)} |l\rangle_{\text{idx}} |1\rangle_{\text{del}} |j\rangle, & \text{if } j \notin S_c(l) \text{ or } l \in [s_0, 2^m - 1], \end{cases} \tag{3.5}$$

and a state preparation oracle PREP *such that*

$$\text{PREP} |0\rangle_{\text{idx}}^{\otimes m} = \frac{1}{\sqrt{\sum_{l=0}^{s_0-1} A_l}} \left(\sum_{l=0}^{s_0-1} \sqrt{A_l} |l\rangle_{\text{idx}} + \sum_{l=s_0}^{2^m-1} 0 |l\rangle_{\text{idx}} \right),$$

then U_A represented by the circuit shown in Figure 4 is a Hermitian block encoding of A with the subnormalization $\sum_{l=0}^{s_0-1} A_l$.

Proof. Performing U_A on $|0\rangle_{\text{idx}}^{\otimes n} |0\rangle_{\text{del1}} |0\rangle_{\text{del0}} |j\rangle$ and postselecting/measuing it with $\langle 0|_{\text{idx}}^{\otimes n} \langle 0|_{\text{del1}} \langle 0|_{\text{del0}} \langle i|$,

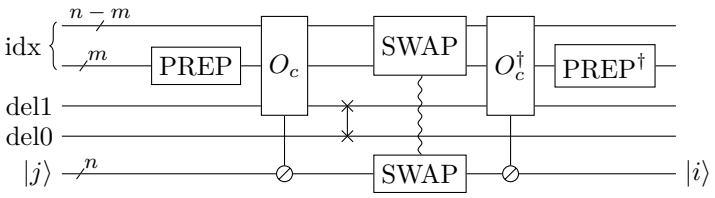


Figure 4: Basic framework of Hermitian block encoding of non-negative symmetric matrices with the dictionary data structure in Table 5. A wavy line connecting two boxes indicates that the operator acts on the registers associated with those two boxes.

we obtain

$$
\begin{aligned}
&\langle 0|_{\text{idx}}^{\otimes n} \langle 0|_{\text{del1}} \langle 0|_{\text{del0}} \langle i| U_A |0\rangle_{\text{idx}}^{\otimes n} |0\rangle_{\text{del1}} |0\rangle_{\text{del0}} |j\rangle \\
=&\frac{1}{\sum_{l=0}^{s_0-1} A_l} \sum_{l,l'=0}^{2^m-1} \langle 0|_{\text{idx}}^{\otimes(n-m)} \langle l'|_{\text{idx}} \langle 0|_{\text{del1}} \langle 0|_{\text{del0}} \langle i| \sqrt{A_{l'}A_l} O_c^\dagger \text{SWAP} O_c |0\rangle_{\text{idx}}^{\otimes(n-m)} |l\rangle_{\text{idx}} |0\rangle_{\text{del1}} |0\rangle_{\text{del0}} |j\rangle \\
=&\frac{1}{\sum_{l=0}^{s_0-1} A_l} \sum_{l,l'=0}^{2^m-1} \delta_{i,S_c(l')}\delta_{l',[0,s_0-1]} \langle c_i(l')|_{\text{idx}} \langle i| \sqrt{A_{l'}A_l}\delta_{j,S_c(l)}\delta_{l,[0,s_0-1]} |j\rangle_{\text{idx}} |c_j(l)\rangle \\
=&\frac{1}{\sum_{l=0}^{s_0-1} A_l} \sum_{l',l=0}^{2^m-1} \sqrt{A_{l'}A_l}(\delta_{i,S_c(l')}\delta_{l',[0,s_0-1]}\delta_{j,c_i(l')})(\delta_{j,S_c(l)}\delta_{l,[0,s_0-1]}\delta_{i,c_j(l)}) \\
=&\frac{1}{\sum_{l=0}^{s_0-1} A_l} \sqrt{a_{ji}a_{ij}} \\
=&\frac{1}{\sum_{l=0}^{s_0-1} A_l} a_{ij},
\end{aligned}
$$

where a_{ij} refers to the l-th data value in the dictionary as $\{(a_{ij}, i, j) : a_{ij} = A_l, (i, j) \in (c_j(l), S_c(l))\}$. □

Note that the oracle O_c in Equation (3.5) is different from that in Equation (3.1). The former is controlled by the register idx and evolves the register $|j\rangle$, whereas the latter is controlled by the register $|j\rangle$ and evolves the register idx. These two oracles O_c are both reasonable because the functions $c_l(j)$ and $c_j(l)$ are separately injective in both j and l.

4 Low Time Metric Implementation

The *time metric* of a block-encoding is defined in Equation (1.1) as

$$\textit{time metric} = \textit{circuit depth} \times \textit{subnormalization}.$$

There is a trade-off between the *circuit depth*, ancillary qubits, and *subnormalization* in the dictionary-based block encoding of sparse matrices. To simplify the statement of implementation and proof, we make the following assumptions on the sparse matrix $A \in \mathbb{C}^{2^n \times 2^n}$,

(A1) Suppose that the encoding dictionary (as specified in Table 4) has s_0 data items, where $\lceil \log s_0 \rceil = \mathcal{O}(n)$;

(A2) Suppose that the matrix A has a total of s non-zero elements.

4.1 Low Circuit Depth Implementations of Oracles

In this section, we analyze the oracle implementation of our block encoding using the $\{\text{U}(2), \text{CNOT}\}$ gate set. The construction depends on efficient implementations of several key oracles, including the column oracle O_c and the state preparation oracles PREP and UNPERP.

4.1.1 Implementation of O_c

The column oracle O_c can be efficiently implemented by adapting the low-depth circuit design of the SAIM [2], an advanced framework for representing general sparse matrices, and the LCU [35, 36]. This implementation leverages the sparse Boolean memory architecture [2, 37] to query the positions of non-zero elements, which serves as the foundation for the low-depth circuit realization.

Definition 4.1 (Sparse Boolean Memory (SBM) [37])**.** *For an n-index, $\tilde{n}$-word Boolean function $f : \{0,1\}^n \to \{0,1\}^{\tilde{n}}$, let $\mathcal{S}_f = \{k : f(k) \neq 0\cdots0\}$ be a set that contains all input indices with non-zero output. We say that f is s-sparse if $\mathcal{S}_f$ has no more than s elements. Its corresponding sparse Boolean function selector satisfies $select(f)\,|k\rangle\,|z\rangle = |k\rangle\,|z \oplus f(x)\rangle$, where $\oplus$ represents bitwise XOR. Let $[f(k)]_l$ be the l-th digit of $f(k)$, $select(f)$ can also be expressed as*

$$select(f) \equiv \sum_{k=0}^{2^n-1} |k\rangle\langle k| \bigotimes_{l=0}^{\tilde{n}-1} \left([f(k)]_l X + \overline{[f(k)]_l} I \right). \tag{4.1}$$

Lemma 4.2 (Circuit Depth of SBM [37])**.** *Given an arbitrary n-index, $\tilde{n}$-word, s-sparse Boolean function f, $select(f)$ in Equation (4.1) can be realized with circuit depth $\mathcal{O}(\log(ns\tilde{n}))$ and $\mathcal{O}(ns\tilde{n})$ ancillary qubits using only single- and two-qubit gates.*

Based on SBM, the column oracle O_c in Equation (3.1) can then be implemented in the following five stages.

- Stage 1. Perform a Pauli-X on the del register.

$$|0\rangle_{\text{del}} \xrightarrow{X} |1\rangle_{\text{del}}. \tag{4.2}$$

- Stage 2. Perform a ranging SBM O_{c_1}.

  We introduce a $(\lceil \log s_0 \rceil + n)$-index, 1-word, s_{c_1}-sparse Boolean function $f_{c_1}(l,j) : \{0,1\}^{\lceil \log s_0 \rceil + n} \mapsto \{0,1\}$ defined as

$$f_{c_1}(l,j) = \begin{cases} 1, & \text{if } l \in [0, s_0 - 1] \text{ and } j \in S_c(l), \\ 0, & \text{if } l \in \left[s_0, 2^{\lceil \log s_0 \rceil} - 1\right] \text{ or } j \notin S_c(l). \end{cases}$$

  The sparsity of f_{c_1} is given by

$$s_{c_1} = |\{(l,j) : f_{c_1}(l,j) \neq 0\}| = |\{(l,j) : l \in [0, s_0 - 1] \text{ and } j \in S_c(l)\}| = s,$$

  where the last equality holds since the number of non-zero element's indices (l,j) is equal to the count of non-zero elements in the matrix. Therefore, there exists a SBM O_{c_1} such that

$$O_{c_1} = \text{select}(f_{c_1}) = \sum_{l,j=0}^{2^{\lceil \log s_0 \rceil}-1, 2^n-1} |l\rangle\langle l|_{\text{idx}} \otimes \left(f_{c_1}(l,j) X + \overline{f_{c_1}(l,j)} I \right)_{\text{del}} \otimes |j\rangle\langle j|. \tag{4.3}$$

- Stage 3. Perform a mapping SBM O_{c_2}.

  We introduce a $(\lceil \log s_0 \rceil + n + 1)$-index, n-word, s_{c_2}-sparse Boolean function $f_{c_2}(l,j) : \{0,1\}^{\lceil \log s_0 \rceil + n + 1} \mapsto \{0,1\}^n$ as

$$f_{c_2}(l,j,\text{del}) = \begin{cases} c_l(j), & \text{if del} = 0, \\ 0, & \text{if del} = 1, \end{cases}$$

  where the sparsity of f_{c_2} is given by $s_{c_2} = |\{(l,j) : f_{c_2}(l,j) \neq 0\}| = s_{c_1} = s$. Furthermore,

there exists a SBM O_{c_2} such that

$$\begin{aligned} O_{c_2} =& \text{select}(f_{c_2}) \\ =& \sum_{l=0,j=0}^{2^{\lceil \log s_0 \rceil}-1,2^n-1} |l\rangle\langle l|_{\text{idx}} \otimes |0\rangle\langle 0|_{\text{del}} \otimes |j\rangle\langle j| \otimes \bigotimes_{k=0}^{n-1} \left([f_{c_2}(l,j,0)]_k X + \overline{[f_{c_2}(l,j,0)]}_k I \right) \\ &+ \sum_{l=0,j=0}^{2^{\lceil \log s_0 \rceil}-1,2^n-1} |l\rangle\langle l|_{\text{idx}} \otimes |1\rangle\langle 1|_{\text{del}} \otimes |j\rangle\langle j| \otimes \bigotimes_{k=0}^{n-1} \left([f_{c_2}(l,j,1)]_k X + \overline{[f_{c_2}(l,j,1)]}_k I \right). \end{aligned} \tag{4.4}$$

- Stage 4. Perform an uncompute SBM O_{c_3}.

  Due to the dictionary requirement of Table 4, $c_l(j)$ is an injective function. Given l and j, there exists a $(\lceil \log s_0 \rceil + n + 1)$-index, n-word, s_{c_3}-sparse Boolean function f_{c_3} such that

$$f_{c_3}(l, c_l(j), \text{del}) = \begin{cases} j, & \text{if del} = 0, \\ 0, & \text{if del} = 1, \end{cases}$$

  where the sparsity of f_{c_3} is given by $s_{c_3} = |\{(l,j) : f_{c_3}(l,j,\text{del}) \neq 0\}| = s_{c_1} = s$. Similarly, there exists a SBM O_{c_3} such that

$$\begin{aligned} O_{c_3} =& \text{select}(\hat{U}_{c_3}) \\ =& \sum_{l=0,j=0}^{2^{\lceil \log(s_0) \rceil}-1,2^n-1} |l\rangle\langle l|_{\text{idx}} \otimes |0\rangle\langle 0|_{\text{del}} \otimes \left[\bigotimes_{k=0}^{n-1} \left([f_{c_3}(l, c_l(j), 0)]_k X + \overline{[f_{c_3}(l, c_l(j), 0)]}_k I \right) \right] \\ & \otimes |c_l(j)\rangle\langle c_l(j)|, \\ &+ \sum_{l=0,j=0}^{2^{\lceil \log(s_0) \rceil}-1,2^n-1} |l\rangle\langle l|_{\text{idx}} \otimes |1\rangle\langle 1|_{\text{del}} \otimes \left[\bigotimes_{k=0}^{n-1} \left([f_{c_3}(l, 0, 1)]_k X + \overline{[f_{c_3}(l, 0, 1)]}_k I \right) \right] \\ & \otimes |0\rangle\langle 0|^{\otimes n}. \end{aligned} \tag{4.5}$$

- Stage 5. Perform a 0-controlled $2n$-qubit SWAP gate O_{SWAP} gate.

$$|0\rangle_{\text{del}} |0\rangle^{\otimes n} |c_l(j)\rangle \xrightarrow{O_{\text{SWAP}}} |0\rangle_{\text{del}} |c_l(j)\rangle |0\rangle^{\otimes n}, \quad |1\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n} \xrightarrow{O_{\text{SWAP}}} |1\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n}. \tag{4.6}$$

Above all, the oracle O_c can be represented by

$$O_c = O_{\text{SWAP}} O_{c_3} O_{c_2} O_{c_1} X.$$

The implementation of oracle O_c leads to the following process,

$$|l\rangle_{\text{idx}} |0\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n} \xrightarrow{X} |l\rangle_{\text{idx}} |1\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n} \xrightarrow{O_{c_1}} |l\rangle_{\text{idx}} |1 \oplus f_{c_1}(l,j)\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n},$$

if $l \in [0, s_0 - 1]$ and $j \in S_c(l)$, we have

$$\begin{aligned} & |l\rangle_{\text{idx}} |1 \oplus f_{c_1}(l,j)\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n} = |l\rangle_{\text{idx}} |0\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n} \\ \xrightarrow{O_{c_2}} & |l\rangle_{\text{idx}} |0\rangle_{\text{del}} |j\rangle |0 \oplus f_{c_2}(l,j,0)\rangle = |l\rangle_{\text{idx}} |0\rangle_{\text{del}} |j\rangle |c_l(j)\rangle \\ \xrightarrow{O_{c_3}} & |l\rangle_{\text{idx}} |0\rangle_{\text{del}} |j \oplus f_{c_3}(l, c_l(j), 0)\rangle |c_l(j)\rangle = |l\rangle_{\text{idx}} |0\rangle_{\text{del}} |j \oplus j\rangle |c_l(j)\rangle \\ \xrightarrow{O_{\text{SWAP}}} & |l\rangle_{\text{idx}} |0\rangle_{\text{del}} |c_l(j)\rangle |0\rangle^{\otimes n}; \end{aligned}$$

if $l \in [s_0, 2^m - 1]$ or $j \notin S_c(l)$, we have

$$\begin{aligned} & |l\rangle_{\text{idx}} |1 \oplus f_{c_1}(l,j)\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n} = |l\rangle_{\text{idx}} |1\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n} \\ \xrightarrow{O_{c_2}} & |l\rangle_{\text{idx}} |1\rangle_{\text{del}} |j\rangle |0 \oplus 0\rangle = |l\rangle_{\text{idx}} |1\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n} \\ \xrightarrow{O_{c_3}} & |l\rangle_{\text{idx}} |1\rangle_{\text{del}} |j \oplus 0\rangle |0\rangle^{\otimes n} = |l\rangle_{\text{idx}} |1\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n} \\ \xrightarrow{O_{\text{SWAP}}} & |l\rangle_{\text{idx}} |1\rangle_{\text{del}} |j\rangle |0\rangle^{\otimes n}. \end{aligned}$$

That is,

$$|l\rangle_{\text{idx}}|0\rangle_{\text{del}}|j\rangle|0\rangle^{\otimes n} \xrightarrow{O_c} \begin{cases} |l\rangle_{\text{idx}}|0\rangle_{\text{del}}|c_l(j)\rangle|0\rangle^{\otimes n}, & \text{if } l \in [0, s_0-1] \text{ and } j \in S_c(l), \\ |l\rangle_{\text{idx}}|1\rangle_{\text{del}}|j\rangle|0\rangle^{\otimes n}, & \text{if } l \in [s_0, 2^m-1] \text{ or } j \notin S_c(l). \end{cases}$$

4.1.2 Implementation of PREP and UNPREP

There are two oracles PREP and UNPREP for state preparation. The task of quantum state preparation is to prepare an n-qubit quantum state $|\psi\rangle$ from an initial product state $|0\rangle^{\otimes n}$ using single- and two-qubit gates. A general quantum state can be expressed as

$$|\psi\rangle = \sum_{k=0}^{N-1} \psi_k |k\rangle,$$

where $N = 2^n$, $\psi_k \in \mathbb{C}$, $\sum_{k=0}^{N-1} |\psi_k|^2 = 1$.

Lemma 4.3 (State Preparation [37])**.** *With only single- and two-qubit gates, an arbitrary n-qubit quantum state can be deterministically prepared with circuit depth $\mathcal{O}(n)$ and $\mathcal{O}(2^n)$ ancillary qubits.*

Lemma 4.4 (Sparse State Preparation [37])**.** *With only single- and two-qubit gates, arbitrary n-qubit, d-sparse ($d \geq 2$) quantum states can be deterministically prepared with circuit depth $\Theta(\log(nd))$ and $\mathcal{O}(nd \log d)$ ancillary qubits.*

4.2 Proof of Circuit Depth

The following theorem demonstrates the low *circuit depth* for implementing our block encoding protocol of sparse matrices based on the dictionary data structure specified in Table 4.

Theorem 4.5 (Circuit depth of dictionary-based block encoding of sparse matrices)**.** *Given a sparse matrix $A \in \mathbb{C}^{2^n \times 2^n}$, if Assumptions (A1) and (A2) hold, then there exists a dictionary-based block encoding of A, which can be implemented with circuit depth $\mathcal{O}(\log(ns))$ and $\mathcal{O}(n^2 s)$ ancillary qubits, using only single- and two-qubit gates.*

Proof. The dictionary-based block encoding of sparse matrices in Theorem 3.1 comprises three oracles: O_c, PREP, and UNPREP.

- The oracle O_c in Equation (3.1) can be implemented by five stages, which consists of
  - a Pauli-X gate in Equation (4.2);
  - $(\lceil \log s_0 \rceil + n)$-index, 1-word, s-sparse ranging SBM O_{c_1} in Equation (4.3);
  - $(\lceil \log s_0 \rceil + n + 1)$-index, n-word, s-sparse mapping SBM O_{c_2} in Equation (4.4);
  - $(\lceil \log s_0 \rceil + n + 1)$-index, n-word, s-sparse decomputing SBM O_{c_3} in Equation (4.5);
  - Controlled $2n$-qubit SWAP gate O_{SWAP} in Equation (4.6).

  By Lemma 4.2, the oracles O_{c_1}, O_{c_2} and O_{c_3} can be realized with *circuit depth*

  $$\mathcal{O}\left(\log\left((\lceil \log s_0 \rceil + n + 1)ns\right)\right)$$

  and

  $$\mathcal{O}\left((\lceil \log s_0 \rceil + n + 1)ns\right)$$

  ancillary qubits. Besides, the Pauli-X gate and the controlled $2n$-qubit SWAP gate can be implemented with *circuit depth* $\mathcal{O}(1)$ without ancillary qubits.

- The oracles PREP in Equation (3.2) and UNPREP in Equation (3.3) prepare two $\lceil \log_2 s_0 \rceil$-qubit quantum states. By Lemma 4.3, these two oracles can be realized with *circuit depth* $\mathcal{O}(\lceil \log_2 s_0 \rceil)$ and $\mathcal{O}\left(2^{\lceil \log_2 s_0 \rceil}\right)$ ancillary qubits.

Therefore, the dictionary-based block encoding of sparse matrices can be implemented with *circuit depth*

$$\mathcal{O}\left(\log\left((\lceil\log s_0\rceil + n + 1)ns\right)\right) + \mathcal{O}\left(\lceil\log_2 s_0\rceil\right) = \mathcal{O}\left(\log_2(ns)\right)$$

and

$$\mathcal{O}\left((\lceil\log s_0\rceil + n + 1)ns\right) + \mathcal{O}\left(2^{\lceil\log_2 s_0\rceil}\right) = \mathcal{O}\left(n^2 s\right)$$

ancillary qubits. □

Remark 2. *The task of block-encoding a $2^n \times 2^n$ sparse matrix with s nonzero elements is analogous to prepare a $2n$-qubit s-sparse quantum state under the framework of third quantization [38]. While the optimal circuit depth for measurement-free sparse state preparation using single- and two-qubit gates is known to be $\Theta(\log(ns))$ [37], the corresponding lower bound for measurement-free sparse matrix block encoding remains an open problem in quantum computation.*

In Appendix B, we analyze the *circuit depth* of existing block-encoding protocols. Our dictionary-based framework outperforms these state-of-the-art methods, particularly for matrices with repeated elements, achieving low *subnormalization* (Theorem 3.1) and low *circuit depth* (Theorem 4.5). These advancements and their associated trade-offs are quantitatively summarized in Table 1.

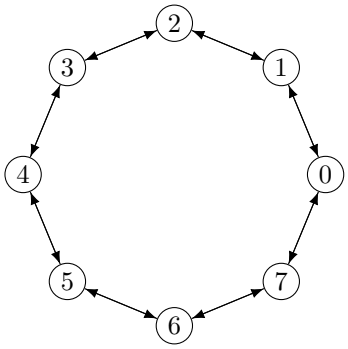
5 Applications

In this section, we demonstrate the practical utility of our dictionary data structure through two key applications: (i) signless Laplacian matrices of graphs and (ii) discrete differential operators. While the low-depth circuit implementation proposed in this work currently requires substantial ancillary qubit resources—rendering full-scale quantum circuit simulations impractical—we provide open-source Python code (https://github.com/ChunlinYangHEU/DIBLE) to validate the *subnormalization* properties of our dictionary-based block encoding protocol of sparse matrices.

5.1 Signless Laplacian Matrix of Graph

A weighted directed graph G consists of a vertex set $V = [0, n-1]$, a directed edge set $E = \{e = (i,j) : i,j \in V\}$ and a weight function $W = \{w(i) : i \in V\} \cup \{w(i,j) : (i,j) \in E\}$, where $w(i)$ is the weight of vertex $i \in V$ and $w(i,j)$ is the weight of directed edge $(i,j) \in E$. It is associated with a signless Laplacian matrix Q_G [39], where $(Q_G)_{ij} = w(i,j)(i \neq j)$ and $(Q_G)_{ii} = w(i)$.

Consider a weighted directed cyclic graph in Figure 5 with vertex weights $w(i) = \alpha_1$, and edge weights $w(i, \mathrm{mod}(i-1, 8)) = \alpha_2$ (clockwise), $w(i, \mathrm{mod}(i+1, 8)) = \alpha_3$ (counterclockwise), where $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{R}$ are non-zero and $i \in [0, 7]$.



$$Q_G = \begin{bmatrix} \alpha_1 & \alpha_3 & & & & & & \alpha_2 \\ \alpha_2 & \alpha_1 & \alpha_3 & & & & & \\ & \alpha_2 & \alpha_1 & \alpha_3 & & & & \\ & & \alpha_2 & \alpha_1 & \alpha_3 & & & \\ & & & \alpha_2 & \alpha_1 & \alpha_3 & & \\ & & & & \alpha_2 & \alpha_1 & \alpha_3 & \\ & & & & & \alpha_2 & \alpha_1 & \alpha_3 \\ \alpha_3 & & & & & & \alpha_2 & \alpha_1 \end{bmatrix}$$

Figure 5: A weighted directed cyclic graph G and its signless Laplacian matrix Q_G.

For the signless Laplacian matrix Q_G, its dictionary data structure is expressed in Table 6.

There are 3 data items in that dictionary (Table 6), and the number of non-zero elements is proportional to the dimension of Q_G, that is, $s_0 = 3$ and $s = 3 \times 2^n = 24$. The *subnormalization*

Keys	Values
0	$\{(a_{ij}, i, j) : a_{ij} = \alpha_1, i = j, j \in [0, 7]\}$
1	$\{(a_{ij}, i, j) : a_{ij} = \alpha_2, (i, j) \in (\mathrm{mod}(j + 1, 8), [0, 7])\}$
2	$\{(a_{ij}, i, j) : a_{ij} = \alpha_3, (i, j) \in (\mathrm{mod}(j - 1, 8), [0, 7])\}$

Table 6: The dictionary data structure of signless Laplacian matrix Q_G.

of the dictionary-based block encoding is the sum of modules of data values, that is, $\alpha = |\alpha_1| + |\alpha_2| + |\alpha_3|$.

Specifically, if we consider a weighted k-regular undirected graph G with n vertices, where for each vertex, it has a weight of w_0 and its k adjacent edges have weights of $w_1, \ldots, w_k$, respectively. The corresponding signless Laplacian matrix Q_G has exactly $k + 1$ non-zero elements in each row and column, $\alpha_0, \alpha_1, \ldots, \alpha_k$, where α_0 lies on the diagonal. This signless Laplacian can be encoded in an n-qubit system using a dictionary with parameters $s_0 = k + 1$ and $s = (k + 1)2^n$. The *subnormalization* for the dictionary-based sparse block-encoding is given by $\sum_{i=0}^{k} |\alpha_i|$. For $n \geq \log(k + 1)$, that *subnormalization* is less than the Frobenius norm $\|Q_G\|_F$ by the Cauchy-Schwarz inequality as

$$\|Q_G\|_F = \sqrt{2^n \sum_{i=1}^{k+1} |\alpha_i|^2} \geq \sqrt{(k+1) \sum_{i=1}^{k+1} |\alpha_i|^2} \geq \sum_{i=1}^{k+1} |\alpha_i| = \alpha.$$

5.2 Two-Dimensional Discrete Laplacian

Consider the finite difference two-dimensional Laplacian which has been discussed in [27]. It is discretized by a regular grid of size $N_x \times N_y$ as

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

To encode the values of the grid, the grid is reshaped as an $N_x N_y$-dimensional vector,

$$f_{a+bN_x} = f(x_a, y_b),\ a \in [0, N_x - 1],\ b \in [0, N_y - 1].$$

Then, all non-zero elements of Laplacian matrix A are given by

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

For $N_x = 4$, $N_y = 4$, the matrix is in form of

$$
\begin{pmatrix}
\boxed{\begin{matrix} A_0 & A_1 & & \\ A_1 & A_0 & A_1 & \\ & A_1 & A_0 & A_1 \\ & & A_1 & A_0 \end{matrix}} & \begin{matrix} A_2 & & & \\ & A_2 & & \\ & & A_2 & \\ & & & A_2 \end{matrix} & & \\
\begin{matrix} A_2 & & & \\ & A_2 & & \\ & & A_2 & \\ & & & A_2 \end{matrix} & \boxed{\begin{matrix} A_0 & A_1 & & \\ A_1 & A_0 & A_1 & \\ & A_1 & A_0 & A_1 \\ & & A_1 & A_0 \end{matrix}} & \begin{matrix} A_2 & & & \\ & A_2 & & \\ & & A_2 & \\ & & & A_2 \end{matrix} & \\
& \begin{matrix} A_2 & & & \\ & A_2 & & \\ & & A_2 & \\ & & & A_2 \end{matrix} & \boxed{\begin{matrix} A_0 & A_1 & & \\ A_1 & A_0 & A_1 & \\ & A_1 & A_0 & A_1 \\ & & A_1 & A_0 \end{matrix}} & \begin{matrix} A_2 & & & \\ & A_2 & & \\ & & A_2 & \\ & & & A_2 \end{matrix} \\
& & \begin{matrix} A_2 & & & \\ & A_2 & & \\ & & A_2 & \\ & & & A_2 \end{matrix} & \boxed{\begin{matrix} A_0 & A_1 & & \\ A_1 & A_0 & A_1 & \\ & A_1 & A_0 & A_1 \\ & & A_1 & A_0 \end{matrix}}
\end{pmatrix}
\tag{5.1}
$$

This matrix can be encoded using a dictionary as stated in Table 7, where the encoding parameters $s_0 = 5$, $s = 64$. The matrix can be encoded using the dictionary-based block encoding with *subnormalization* $\alpha = |A_0| + 2(|A_1| + |A_2|)$.

Keys	Values
0	$\{(a_{ij}, i, j) : a_{ij} = A_0, (i, j) \in (j, [0, 15])\}$
1	$\{(a_{ij}, i, j) : a_{ij} = A_1, (i, j) \in (j-1, \{j \in [0, 15] : \mathrm{mod}(j, 4) \neq 0\})\}$
2	$\{(a_{ij}, i, j) : a_{ij} = A_1, (i, j) \in (j+1, \{j \in [0, 15] : \mathrm{mod}(j, 4) \neq 3\})\}$
3	$\{(a_{ij}, i, j) : a_{ij} = A_2, (i, j) \in (j-4, [4, 15])\}$
4	$\{(a_{ij}, i, j) : a_{ij} = A_2, (i, j) \in (j+4, [0, 11])\}$

Table 7: The dictionary data structure of two-dimensional discrete Laplacian A in Equation (5.1).

5.3 Matrices in Ocean Acoustic GEPs

Ocean acoustics plays a crucial role in understanding underwater sound propagation. The problem of solving the acoustic field essentially reduces to solving the following differential equations,

$$
\frac{d^2\varphi_m(z)}{dz^2} + \left(\frac{\omega^2}{c^2(z)} - k_m^2\right)\varphi_m(z) = \mathbf{0},
$$

$$
\begin{cases}
\frac{d\xi_{m,1}(z)}{dz} = -\xi_{m,2}(z) + \frac{1}{\mu(z)}\xi_{m,4}(z), \\
\frac{d\xi_{m,2}(z)}{dz} = \frac{\lambda(z)k_m^2}{\lambda(z)+2\mu(z)}\xi_{m,1}(z) + \frac{1}{\lambda(z)+2\mu(z)}\xi_{m,4}(z), \\
\frac{d\xi_{m,3}(z)}{dz} = \left(\frac{4\mu(z)(\lambda(z)+\mu(z))k_m^2}{\lambda(z)+2\mu(z)} - \rho(z)\omega^2\right)\xi_{m,1}(z) - \frac{\lambda(z)}{\lambda(z)+2\mu(z)}\xi_{m,4}(z), \\
\frac{d\xi_{m,4}(z)}{dz} = -\rho(z)\omega^2\xi_{m,2}(z) + k_m^2\xi_{m,3}(z).
\end{cases}
$$

which respectively describe the propagation of sound pressure modes $\varphi_m(z)$ in seawater and the propagation of stress-displacement modes $\xi_m(z) = (\xi_{m,1}(z), \xi_{m,2}(z), \xi_{m,3}(z), \xi_{m,4}(z))^{\mathrm{T}}$ in ice. The quantities to be determined include the sound pressure modes, the stress-displacement modes, and the corresponding wave numbers k_m.

Using the finite difference method, it can be further transformed into the following generalized eigenvalue problem [28, 29],

$$AV = BV\Sigma. \tag{5.2}$$

The details of this problem and the general structure of A and B are shown in Appendix C. The dimensions of the matrices A and B are $4N_1 + N_2 + 5$, where N_1 and N_2 represent the numbers of stratified grid layers for ice and seawater, respectively. These two matrices can be encoded in a system with $n = \lceil \log(4N_1 + N_2 + 5) \rceil$ qubits. Based on their structure, their dictionary data structures are constructed and shown in Tables 9 and 10 of Appendix C, where the dictionaries satisfy that the data functions $c_l(j)$ are all injective.

For matrix A in Equation (5.2), there are 21 data items in the encoding dictionary of A and the non-zero count is associated with N_1 and N_2. Thus, the encoding parameters of A are $s_0 = 19$ and $s = 5 + 24N_1 + 3N_2$. The sum of the modules of all data values serves as the *subnormalization* in its block encoding, that is,

$$|a_0| + 2(|a_1| + |a_2| + |a_3| + |a_4| + |a_5| + |a_6|) + |a_7| + |a_8| + |a_9| + |a_{10}| + |a_{11}| + |a_{12}|\,.$$

For the matrix B in Equation (5.2), the number of data items is $s_0 = 9$, and the non-zero elements count is $s = 3 + 6N_1 + N_2$. Besides, its *subnormalization* of the block encoding derived from the associated dictionary is

$$2(|b_0| + |b_1| + |b_2|) + |b_3| + |b_4| + |b_5|\,.$$

6 Conclusion and Outlook

In this work, we have advanced the implementation of sparse matrix block encodings through optimized *circuit depth* architectures.

We have proposed a unified dictionary data structure (Table 3) that generalizes multiple block-encoding protocols of sparse matrices [6, 24, 26, 27]. Our proposed dictionary-based block-encoding framework of sparse matrices was constructed using the dictionary data function specified in Table 4. The dictionary-based block encoding for complex matrices (Theorem 3.1) and its Hermitian extension for non-negative symmetric matrices (Theorem 3.3) generalized the results in [26], achieving improved *subnormalization*. Furthermore, we have demonstrated that the dictionary-based block encoding of sparse matrices can be viewed as a special case of linear combination of unitaries.

Based on the sparse Boolean memory architecture, we have established fundamental bounds on quantum *circuit depth* requirements for the implementation of the dictionary-based block encoding of sparse matrices (Theorems 4.5). Our theoretical analysis revealed a doubly logarithmic scaling relationship: the required *circuit depth* demonstrates $\mathcal{O}(\log n)$ dependence on the Hilbert space dimension $N = 2^n$, while maintaining $\mathcal{O}(\log s)$ scaling with respect to the number of non-zero elements s. To facilitate the rigorous comparison across different block-encoding protocols, we propose a novel time-efficiency metric that combines *circuit depth* with *subnormalization*. For matrices with repeated elements, our approach achieved superior *subnormalization* and *circuit depth* compared to existing sparse block-encoding protocols (Table 1). We have demonstrated the effectiveness of our method through several applications, including graph problems, two-dimensional discrete Laplacian operators, and ocean acoustic GEPs.

While our block-encoding scheme achieved an efficient *time metric*, two fundamental questions remain open: (1) the optimality of our *circuit depth* $\mathcal{O}(\log(ns))$, and (2) the gap between our *subnormalization* $\alpha = \sum_l |A_l|$ and the theoretical minimum $\|A\|_{\mathrm{sp}}$. These observations reveal promising directions for future optimization.

Acknowledgments

This work is supported by the Stable Supporting Fund of Acoustic Science and Technology Laboratory (Grant No. JCKYS2024604SSJS001), the Fundamental Research Funds for the Central Universities (Grant No. 3072024XX2401), the Hong Kong Research Grants Council (RGC) (Grant No. 15213924), and the National Natural Science Foundation of China (Grant No. 62173288).

References

[1] 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. IEEE Transactions on Quantum Engineering, 3:1–23, 2022. DOI: 10.1109/TQE.2022.3231194.

[2] Xiaoming Zhang and Xiao Yuan. Circuit complexity of quantum access models for encoding classical data. npj Quantum Information, page 42, 2024. DOI: 10.1038/s41534-024-00835-8.

[3] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences, 439(1907):553–558, 1992. DOI: 10.1098/rspa.1992.0167.

[4] P. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In Proceedings 35th Annual Symposium on Foundations of Computer Science, pages 124–134, 1994. DOI: 10.1109/SFCS.1994.365700.

[5] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. Physical review letters, 103(15):150502, 2009. DOI: 10.1103/PhysRevLett.103.150502.

[6] András Gilyén, Yuan Su, Guang Hao Low, and Nathan 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, page 193–204, 2019. DOI: 10.1145/3313276.3316366.

[7] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. Communications in Mathematical Physics, 2007. DOI: 10.1007/s00220-006-0150-x.

[8] Andrew M. Childs and Robin Kothari. Simulating sparse hamiltonians with star decompositions. In Theory of Quantum Computation, Communication, and Cryptography, pages 94–103, 2011. DOI: 10.1007/978-3-642-18073-6˙8.

[9] Andrew M. Childs. On the relationship between continuous- and discrete-time quantum walk. Communications in Mathematical Physics, pages 581–603, 2010. DOI: 10.1007/s00220-009-0930-1.

[10] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Exponential improvement in precision for simulating sparse hamiltonians. In Proceedings of the forty-sixth annual ACM symposium on Theory of computing, 2014. DOI: 10.1145/2591796.2591854.

[11] Andrew M. Childs, Robin Kothari, and Rolando D. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. SIAM Journal on Computing, 46(6):1920–1950, 2017. DOI: 10.1137/16M1087072.

[12] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: Improved regression techniques via faster hamiltonian simulation. In 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019), volume 132, pages 33:1–33:14, 2019. DOI: 10.4230/LIPIcs.ICALP.2019.33.

[13] Ryan Babbush, Dominic W. Berry, Robin Kothari, Rolando D. Somma, and Nathan Wiebe. Exponential quantum speedup in simulating coupled classical oscillators. Physical Review X, 13:041041, 2023. DOI: 10.1103/PhysRevX.13.041041.

[14] Long Gui-Lu. General quantum interference principle and duality computer. Communications in Theoretical Physics, 45(5):825, 2006. DOI: 10.1088/0253-6102/45/5/013.

[15] Andrew M. Childs and Nathan Wiebe. Hamiltonian simulation using linear combinations of unitary operations. Quantum Information and Computation, (11–12):901–924, 2012. DOI: 10.26421/QIC12.11-12-1.

[16] John M Martyn, Zane M Rossi, Andrew K Tan, and Isaac L Chuang. Grand unification of quantum algorithms. PRX quantum, 2(4):040203, 2021. DOI: 10.1103/PRXQuantum.2.040203.

[17] Guang Hao Low and Isaac L Chuang. Hamiltonian simulation by qubitization. Quantum, 3: 163, 2019. DOI: 10.22331/q-2019-07-12-163.

[18] Joran van Apeldoorn and András Gilyén. Improvements in Quantum SDP-Solving with Applications. In 46th International Colloquium on Automata, Languages, and Programming

(ICALP 2019), volume 132, pages 99:1–99:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. DOI: 10.4230/LIPIcs.ICALP.2019.99.

[19] 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, 2022. DOI: 10.22331/q-2022-12-13-876.

[20] Haoya Li, Hongkang Ni, and Lexing Ying. On efficient quantum block encoding of pseudo-differential operators. Quantum, 7:1031, 2023. DOI: 10.22331/q-2023-06-02-1031.

[21] Diyi Liu, Weijie Du, Lin Lin, James P. Vary, and Chao Yang. An efficient quantum circuit for block encoding a pairing hamiltonian. Journal of Computational Science, 85:102480, 2025. DOI: 10.1016/j.jocs.2024.102480.

[22] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. Physical Review A, 101(2):022316, 2020. DOI: 10.1103/PhysRevA.101.022316.

[23] Daan Camps and Roel Van Beeumen. Fable: Fast approximate quantum circuits for block-encodings. 2022 IEEE International Conference on Quantum Computing and Engineering, pages 104–113, 2022. DOI: 10.1109/QCE53715.2022.00029.

[24] Parker Kuklinski and Benjamin Rempfer. S-fable and ls-fable: Fast approximate block-encoding algorithms for unstructured sparse matrices, 2024. URL https://arxiv.org/abs/2401.04234.

[25] Zexian Li, Xiao-Ming Zhang, Chunlin Yang, and Guofeng Zhang. Binary tree block encoding of classical matrix, 2025. URL https://arxiv.org/abs/2504.05624.

[26] Daan Camps, Lin Lin, Roel Van Beeumen, and Chao Yang. Explicit quantum circuits for block encodings of certain sparse matrices. SIAM Journal on Matrix Analysis and Applications, 45 (1):801–827, 2024. DOI: 10.1137/22M1484298.

[27] Christoph Sünderhauf, Earl Campbell, and Joan Camps. Block-encoding structured matrices for data input in quantum computing. Quantum, 8:1226, 2024. DOI: 10.22331/q-2024-01-11-1226.

[28] Keiiti Aki and Paul G Richards. Quantitative Seismology, Second Edition. University Science Books, 2002. DOI: 10.1007/978-1-4419-8678-8.

[29] Finn B Jensen, William A Kuperman, Michael B Porter, Henrik Schmidt, and Alexandra Tolstoy. Computational ocean acoustics. Springer New York, 2011. DOI: 10.1007/978-1-4419-8678-8.

[30] Jun Xiao, Rui Zhao, and Kin-Man Lam. Bayesian sparse hierarchical model for image denoising. Signal Processing: Image Communication, 96:116299, 2021. DOI: 10.1016/j.image.2021.116299.

[31] Weimin Yuan, Yuanyuan Wang, Ruirui Fan, Yuxuan Zhang, Guangmei Wei, Cai Meng, and Xiangzhi Bai. Simultaneous image denoising and completion through convolutional sparse representation and nonlocal self-similarity. Computer Vision and Image Understanding, 249: 104216, 2024. DOI: 10.1016/j.cviu.2024.104216.

[32] A. Abusalah, O. Saad, J. Mahseredjian, U. Karaagac, and I. Kocar. Accelerated sparse matrix-based computation of electromagnetic transients. IEEE Open Access Journal of Power and Energy, 7:13–21, 2020. DOI: 10.1109/OAJPE.2019.2952776.

[33] Zhaoli Shen, Guoliang Han, Yutong Liu, Bruno Carpentieri, Chun Wen, and Jianjun Wang. Weak dangling block reordering and multi-step block compression for efficiently computing and updating pagerank solutions. Journal of Computational and Applied Mathematics, 458: 116332, 2025. DOI: 10.1016/j.cam.2024.116332.

[34] 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, 8:041015, 2018. DOI: 10.1103/PhysRevX.8.041015.

[35] Dominic W. Berry and Andrew M. Childs. Black-box hamiltonian simulation and unitary implementation. Quantum Info. Comput., 12(1–2):29–62, January 2012. ISSN 1533-7146. DOI: 10.26421/QIC12.1-2.

[36] Ryan Babbush, Dominic W. Berry, Robin Kothari, Rolando D. Somma, and Nathan Wiebe. Exponential quantum speedup in simulating coupled classical oscillators. Physical Review X, 13:041041, 2023. DOI: 10.1103/PhysRevX.13.041041.

[37] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. Quantum state preparation with optimal circuit depth: Implementations and applications. Physical review letters, 129:230504, 2022. DOI: 10.1103/PhysRevLett.129.230504.

[38] Tomaž Prosen. Third quantization: a general method to solve master equations for quadratic open fermi systems. New Journal of Physics, 10(4):043026, 2008. DOI: 10.1088/1367-2630/10/4/043026.

[39] Dragoš Cvetković, Peter Rowlinson, and Slobodan Simić. An Introduction to the Theory of Graph Spectra. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511801518.

[40] Lin Lin. Lecture notes on quantum algorithms for scientific computation. arXiv preprint arXiv:2201.08309, 2022. DOI: 10.48550/arXiv.2201.08309.

[41] Pei Yuan and Shengyu Zhang. Optimal (controlled) quantum state preparation and improved unitary synthesis by quantum circuits with any number of ancillary qubits. Quantum, 7:956, 2023. DOI: 10.22331/q-2023-03-20-956.

A Block Encoding and Dictionary Data Structure

Block encoding is a technique that represents a non-unitary operator A as a subsystem of a larger unitary operator U_A.

Definition A.1 (Block Encoding [6])**.** *Suppose that A is an n-qubit operator, $N = 2^n, \alpha, \epsilon \in \mathbb{R}_+$ and $a \in \mathbb{N}$, then we say that the $(n+a)$-qubit unitary U_A is an (α, a, ϵ) block encoding of A, if*

$$\left\| A - \alpha \left(\langle 0|^{\otimes a} \otimes I_N \right) U_A \left(|0\rangle^{\otimes a} \otimes I_N \right) \right\| \leq \epsilon.$$

A Hermitian block encoding is a specialized form of block encoding whose U_A is not only unitary but also Hermitian. Consequently, the encoded matrix A must also be Hermitian.

Definition A.2 (Hermitian Block Encoding([40]))**.** *Let U_A be an (α, a, ϵ) block encoding of a Hermitian matrix A. If U_A is also Hermitian, then it is called an (α, a, ϵ) Hermitian block encoding of A. When $\epsilon = 0$, it is called an (α, a) Hermitian block encoding. The set of all (α, a, ϵ) Hermitian block encodings of A is denoted by $\mathrm{HBE}_{\alpha,a}(A, \epsilon)$, and $\mathrm{HBE}_{\alpha,a}(A) = \mathrm{HBE}_{\alpha,a}(A, 0)$.*

A.1 Several Block Encoding Protocols of Sparse Matrices within the Framework of Dictionary Data Structure

Different dictionary data structures result in different block-encoding protocols. Within the unified framework proposed in section 3.1, we can summarize several block-encoding protocols of sparse matrices in [6, 24, 26, 27]. Given triplets (a_{ij}, i, j) that encode a sparse matrix, these protocols satisfy the following row and column rules, respectively.

1. A sparse matrix can be characterized by two data functions: $j = r(i, k)$ locating the k-th non-zero element in the i-th row ($k \in [0, S_r - 1]$) and $i = c(l, j)$ finding the l-th non-zero element in the j-th column ($l \in [0, S_c - 1]$), where S_r/S_c denotes the row/column sparsities. Based on this, the triplets (a_{ij}, i, j) form the dictionary data structure of [6].

2. The positions of elements in a sparse matrix can be fully characterized by a vectorized index mapping $\alpha = iN + j$, where N is the dimension of the matrix. All triplets (a_{ij}, i, j) with the vectorized indices α form the fundamental data items of the dictionary data structure of [24], enabling efficient storage and retrieval.

3. The positions of non-zero elements in a sparse matrix can be characterized by a bijective function $i = c(l, j)$ mapping (l, j) pairs to row indices i, where j denotes the column index and l enumerates non-zero elements in every column. All non-zero elements labeled by l should share both the same value A_l and the same function $c(l, j)$. The corresponding triplets form a data item, which constitutes the dictionary data structure of [26]. It compactly represents the sparse matrix through its non-zero pattern and value distribution.

4. The row and column index sets of a sparse matrix are characterized by two data functions: $(j, s_c) = c(d, m)$ for column indices and column sparsity indices, and $(i, s_r) = r(d, m)$ for row indices and row sparsity indices, where $d \in [0, D-1]$ is the data index, D is the number of distinct non-zero values A_d, $m \in [0, M-1]$ is the multiplicity index, M is the maximum multiplicity of any of D data items. All non-zero elements sharing the same value A_d under identical functions $c(d, m)$ and $r(d, m)$ form equivalence classes that constitute the dictionary data structure of [27], providing an efficient representation of sparse matrices through their value distributions and index patterns.

B Circuit Depth Comparison

To demonstrate the circuit-depth efficiency of our block-encoding framework, we conduct a rigorous comparative analysis in this section. Specifically:

- In Lemma B.1, we analyze the low *circuit depth* implementation of PREP/UNPREP block encoding for sparse matrices [27], which currently achieves the lowest known *subnormalization* factor for sparse matrices.
- In Lemma B.2, we analyze the sparse implementation for the low-circuit-depth block-encoding protocol [1, 2], which currently represents the most depth-efficient approach for general matrices.

These carefully selected benchmarks provide a comprehensive basis for evaluating our improvements in circuit-depth complexity.

Lemma B.1. *Given a matrix $A \in \mathbb{C}^{2^n \times 2^n}$ with D distinct non-zero elements, row sparsity S_r and column sparsity S_c. If $\lceil \log D \rceil \leq \lceil \log S_c \rceil, \lceil \log S_r \rceil$, then the* PREP/UNPREP *block encoding in [27] can be implemented with circuit depth $\mathcal{O}\left(n 2^{n/2}\right)$ and n_{anc} ancillary qubits using only single- and two-qubit gates, where $n_{\text{anc}} \geq \Omega\left(4^n/n\right)$.*

The proof is given in Appendix B.1.

Lemma B.2. *Given a matrix $A \in \mathbb{C}^{2^n \times 2^n}$ with s non-zero elements, where $\lceil \log s \rceil = \mathcal{O}(n)$. Then there exists a controlled state preparation oracle U_L and a state preparation oracle U_R such that*

$$U_R |0\rangle^{\otimes n} |j\rangle = |A_j\rangle |j\rangle, \quad U_L |0\rangle^{\otimes n} = |A\rangle, \tag{B.1}$$

where $|A_j\rangle = \sum_{k=0}^{2^n-1} \frac{a_{jk}}{\|A_{j,\cdot}\|_F} |k\rangle$, $|A\rangle = \sum_{j=0} \frac{\|A_{j,\cdot}\|_F}{\|A\|_F} |j\rangle$, $A_{j,\cdot}$ denotes the j-th row of A. Then $U_A = U_R^\dagger SWAP(U_L \otimes I_{2^n})$ is a block-encoding of A with subnormalization $\|A\|_F$, as stated in [1], and it can be implemented with circuit depth $\mathcal{O}(n)$ and $\mathcal{O}\left(2^{2n}\right)$ ancillary qubits, using only single- and two-qubit gates.

The *circuit depth* in these two protocols [27, 1] (Lemma B.1 and Lemma B.2) primarily arises from their dependence on two computationally costly operations: unitary synthesis procedures and controlled state preparation routines [41]. Both approaches inherently require considerable *circuit depth*, a factor that proves especially critical in the context of sparse matrix encodings.

B.1 Proof of Lemma B.1

As stated in Appendix A.1, the PREP/UNPREP block encoding scheme in [27] can be described by a dictionary data structure, which is constructed in Table 8. The PREP/UNPREP block encoding scheme relies on the equality $\lceil \log_2 D \rceil + \lceil \log_2 M \rceil = \lceil \log_2 N \rceil + \lceil \log_2 S \rceil$ and the assumption $\lceil \log_2 D \rceil \leq \lceil \log_2 S_c \rceil, \lceil \log_2 S_r \rceil$, where D represents the number of distinct data values, M denotes the maximum multiplicity of data values, $S = \max\{S_c, S_r\}$ captures the matrix sparsity (with S_c and S_r being column and row sparsities, respectively), and N is the dimension of the matrix.

These ensure the versatility in their implementation with the quantum circuit in Figure 6. The oracles O_c and O_r are used to implement the mapping of data functions $c(d, m)$ and $r(d, m)$, respectively, while the oracle O_{rg} identifies the defining domain of these two functions. The oracles PREP and UNPREP are used to prepare the states encoding all distinct data values. The specific statements are as follows.

Keys	Values
0	$\{(a_{i,j}, i, j) : a_{ij} = A_0, (i,j) \in (S_r(0), S_c(0)), (i, s_r) = r(0,m), (j, s_c) = c(0,m)\}$
1	$\{(a_{i,j}, i, j) : a_{ij} = A_1, (i,j) \in (S_r(1), S_c(1)), (i, s_r) = r(1,m), (j, s_c) = c(1,m)\}$
2	$\{(a_{i,j}, i, j) : a_{ij} = A_2, (i,j) \in (S_r(2), S_c(2)), (i, s_r) = r(2,m), (j, s_c) = c(2,m)\}$
$\vdots$	$\vdots$

Table 8: A dictionary data structure for implementing the PREP/UNPREP block-encoding scheme in [27].

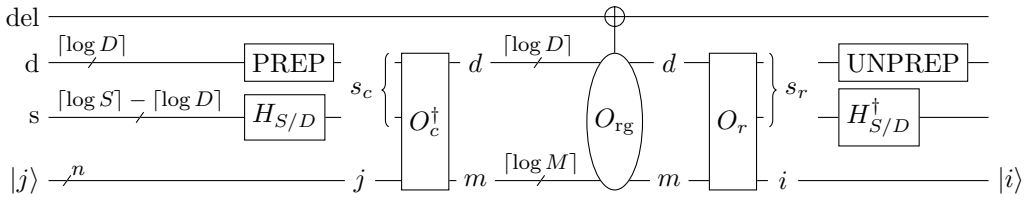


Figure 6: Basic framework of PREP/UNPREP block encoding in [27] with dictionary data structure in Table 8, where $H_{S/D} = H^{\otimes(\lceil \log S \rceil - \lceil \log D \rceil)}$.

- The out-of-range oracle $\tilde{O}_{\mathrm{rg}}$ satisfies

$$\tilde{O}_{\mathrm{rg}} |d\rangle |m\rangle |0\rangle_{\mathrm{del}} = \begin{cases} |d\rangle |m\rangle |0\rangle_{\mathrm{del}}, & \text{if } A_{i(d,m),j(d,m)} = A_d, \\ |d\rangle |m\rangle |1\rangle_{\mathrm{del}}, & \text{if } A_{i(d,m),j(d,m)} = 0, \end{cases} \tag{B.2}$$

  where $d \in [0, D-1]$, $m \in [0, M-1]$.

- The column and row oracles $\tilde{O}_c, \tilde{O}_r$ satisfy

$$\tilde{O}_{\mathrm{c}} |d\rangle |m\rangle = |j\rangle |s_c\rangle, \quad \tilde{O}_{\mathrm{r}} |d\rangle |m\rangle = |i\rangle |s_r\rangle, \tag{B.3}$$

  where $d \in [0, D-1]$, $m \in [0, M-1]$, $i, j \in [0, N-1]$, $s_c \in [0, S_c - 1]$, $s_r \in [0, S_r - 1]$.

- The state preparation oracles PREP and UNPREP satisfy

$$\mathrm{PREP} |0\rangle^{\otimes \lceil \log_2 D \rceil} = \frac{1}{\sqrt{\sum_{d=0}^{D-1} |A_d|}} \sum_{d=0}^{D-1} \mathrm{sgn}(A_d) \sqrt{|A_d|} |d\rangle,$$

$$\langle 0|^{\otimes \lceil \log_2 D \rceil} \mathrm{UNPREP} = \frac{1}{\sqrt{\sum_{d=0}^{D-1} |A_d|}} \sum_{d=0}^{D-1} \sqrt{|A_d|} \langle d|.$$

Noting that the column and row oracles $\tilde{O}_c, \tilde{O}_r$ are unitary synthesis, which takes relatively high *circuit depth* using the state-of-the-art techniques [41], compared to the column oracles O_c (3.1) in this article.

Lemma B.3 (Unitary Synthesis [41])**.** *For any $m \geq 0$, any n-qubit unitary can be implemented by a quantum circuit with m ancillary qubits, using single-qubit gates and CNOT gates, of depth $\mathcal{O}\left(\frac{n^{1/2} 2^{3n/2}}{m^{1/2}}\right)$ when $\Omega(2^n/2) \leq \mathcal{O}(4^n/n)$. In particular, the depth is $\mathcal{O}\left(n 2^{n/2}\right)$ when $m \geq \Omega(4^n/n)$.*

Proof of Lemma B.1. The PREP/UNPREP block encoding scheme in [27] comprises five oracles: $\tilde{O}_{\mathrm{rg}}$, $\tilde{O}_c$, $\tilde{O}_r$, PREP, and UNPREP.

- The oracle $\tilde{O}_{\mathrm{rg}}$ can be implemented by SBM. We introduce a $(\lceil \log_2 D \rceil + \lceil \log_2 M \rceil)$-index, 1-word, $\tilde{s}_{\mathrm{rg}}$-sparse Boolean function $\tilde{f}_{\mathrm{rg}}$ such that

$$\tilde{f}_{\mathrm{rg}}(d, m) = \begin{cases} 0 & \text{if } A_{i(d,m),j(d,m)} = A_d, \\ 1 & \text{if } A_{i(d,m),j(d,m)} = 0, \end{cases}$$

where the sparsity is given by

$$\tilde{s}_{\mathrm{rg}} = \left|\left\{(d,m) : \tilde{f}_{\mathrm{rg}}(d,m) = 1\right\}\right| = \left|\left\{(d,m) : A_{i(d,m),j(d,m)} = 0\right\}\right| = \tilde{s}_1 < 4^n,$$

$\tilde{s}_1$ is the number of invalid (d,m) and always less than the total number of elements of the matrix. Then, there exits a SBM such that

$$\mathrm{select}(\tilde{f}_{\mathrm{rg}}) = \sum_{d,m=0}^{2^{\lceil \log_2 D\rceil}-1, 2^{\lceil \log_2 M\rceil}-1} |d,m\rangle\langle d,m| \otimes \left(\tilde{f}_{\mathrm{rg}}(d,m)X + \overline{\tilde{f}_{\mathrm{rg}}}(d,m)I\right).$$

By Lemma 4.2, it can be implemented with *circuit depth* of $\mathcal{O}\left(\log_2\left(\left(\lceil\log_2 D\rceil + \lceil\log_2 M\rceil\right)\tilde{s}\right)\right)$ and ancillary qubits of $\mathcal{O}\left(\left(\lceil\log_2 D\rceil + \lceil\log_2 M\rceil\right)\tilde{s}\right)$.

- The oracles $\tilde{O}_c$ and $\tilde{O}_r$ are two $(\lceil\log_2 D\rceil + \lceil\log_2 M\rceil)$-qubit unitary synthesis. By Lemma B.3, they can be implemented with *circuit depth*

$$\mathcal{O}\left(\left(\lceil\log_2 D\rceil + \lceil\log_2 M\rceil\right) 2^{\frac{\lceil\log_2 D\rceil + \lceil\log_2 M\rceil}{2}}\right) = \mathcal{O}\left(n2^{n/2}\right)$$

and n_{anc} ancillary qubits, where

$$n_{\mathrm{anc}} \geq \Omega\left(4^{\lceil\log_2 D\rceil + \lceil\log_2 M\rceil} / \left(\lceil\log_2 D\rceil + \lceil\log_2 M\rceil\right)\right) = \Omega\left(4^n/n\right).$$

- The oracles PREP and UNPREP prepare two $\lceil log_2 D\rceil$-qubit states. Using the state preparation in Lemma 4.3, they can be implemented with *circuit depth* $\mathcal{O}\left(\lceil log_2 D\rceil\right)$ and $\mathcal{O}\left(2^{\lceil log_2 D\rceil}\right)$ ancillary qubits using only single- and two-qubit gates.

Since $\lceil\log_2 D\rceil + \lceil\log_2 M\rceil = \lceil\log_2 N\rceil + \lceil\log_2 S\rceil = \mathcal{O}(n)$, we have $\lceil\log_2 D\rceil = \mathcal{O}(n)$ and $\lceil\log_2 M\rceil = \mathcal{O}(n)$. Therefore, its PREP/UNPREP block encoding can be implemented with *circuit depth*

$$\mathcal{O}\left(\log_2\left(\left(\lceil\log_2 D\rceil + \lceil\log_2 M\rceil\right)\tilde{s}_1\right)\right) + \mathcal{O}\left(n2^{n/2}\right) + \mathcal{O}\left(\lceil log_2 D\rceil\right) = \mathcal{O}\left(n2^{n/2}\right)$$

and

$$\mathcal{O}\left(\left(\lceil\log_2 D\rceil + \lceil\log_2 M\rceil\right)\tilde{s}_1\right) + n_{\mathrm{anc}} + \mathcal{O}\left(2^{\lceil log_2 D\rceil}\right) = n_{\mathrm{anc}} \geq \Omega\left(4^n/n\right)$$

ancillary qubits using only single- and two-qubit gates. □

B.2 Proof of Lemma B.2

Lemma B.4 (Controlled Quantum State Preparation [41])**.** *For any integers $k, m > 0$, $n > 0$ and any quantum state $\left\{|\psi_j\rangle : j \in \{0,1\}^k\right\}$, the following controlled quantum state preparation*

$$|j\rangle|0\rangle^{\otimes n} \to |j\rangle|\psi_j\rangle, \ \forall j \in \{0,1\}^k$$

can be implemented by a quantum circuit consisting of single-qubit and CNOT gates of depth $\mathcal{O}\left(n + k + \frac{2^{n+k}}{n+k+m}\right)$ and size $\mathcal{O}\left(2^{n+k}\right)$ with ancillary qubits. These bounds are optimal for any $k, m > 0$.

Proof of Lemma B.2. We only need to demonstrate the *circuit depth*. This block encoding comprises three oracles: U_R, U_L, and SWAP.

- U_R is can represented as

$$|0\rangle^{\otimes n}|j\rangle \xrightarrow{U_R} |A_j\rangle|j\rangle, \ j \in \{0,1\}^n$$

which can be implemented by controlled-state preparation with *circuit depth* $\mathcal{O}(n)$ and $\mathcal{O}\left(2^{2n}\right)$ ancillary qubits by Lemma B.4;

- U_L is an n-qubit s-sparse state-preparation, which can be implemented with *circuit depth* $\mathcal{O}(\log(ns))$ and $\mathcal{O}(ns\log s)$ ancillary qubits by Lemma 4.4;

- $2n$-qubit SWAP gate can be implemented with *circuit depth* $\mathcal{O}(1)$ without ancillary qubits.

Above all, U_A can be implemented with *circuit depth*

$$\mathcal{O}(n) + \Theta(\log(ns)) = \mathcal{O}(n)$$

and

$$\mathcal{O}\left(2^{2n}\right) + \mathcal{O}(ns \log s)) = \mathcal{O}\left(2^{2n}\right)$$

ancillary qubits. □

C Generalized Eigenvalue Problems in Ocean Acoustic Modeling

In the field of ocean acoustics, the propagation of acoustic waves through seawater and ice is a complex process crucial for obtaining underwater information. This information is essential for various applications, including underwater communication, navigation, and environmental monitoring. Therefore, the study of underwater acoustic propagation is a core component of research in underwater acoustic information.

Ocean acoustics encompasses several subfields, including shallow sea acoustics, deep sea acoustics, and polar acoustics. In polar acoustics, the interaction of acoustic waves with ice and seawater presents distinct challenges. Typically, sound pressure fields are measured in seawater, whereas displacement fields are measured in ice [29, 28].

Utilizing the normal-mode method, on one hand, the sound pressure field $p(r,z)$ can be computed from the sound pressure modes $\varphi_m(z)$ [29],

$$p(r,z) = \frac{\imath}{\rho(z_s)\sqrt{8\pi r}} e^{-i\pi/4} \sum_m \varphi_m(z_s)\,\varphi_m(z)\,\frac{e^{ik_m r}}{\sqrt{k_m}},$$

where r is the horizontal distance, z is the depth, z_s is the depth of the sound source, $\rho(z_s)$ is the density of the medium at z_s, k_m is the horizontal wave number and $i^2 = -1$. Meanwhile, the sound pressure modes $\varphi_m(z)$ satisfy the Helmholtz equation

$$\frac{d^2\varphi_m(z)}{dz^2} + \left(\frac{\omega^2}{c^2(z)} - k_m^2\right)\varphi_m(z) = \mathbf{0}, \tag{C.1}$$

where ω is the angle frequency of sound wave and $c(z)$ is the sound velocity.

On the other hand, the displacement field includes the horizontal displacement field $u(r,z)$ and the vertical displacement field $w(r,z)$, which can be computed from the horizontal displacement modes $d_m^{(1)}(z)$ and the vertical displacement modes $d_m^{(2)}(z)$ [28],

$$u(r,z) = \sum_m \frac{d_m^{(1)}(z)}{8c_m U_m I_m}\sqrt{\frac{2}{\pi k_m r}}\, e^{\imath\left(k_m r - \frac{\pi}{4}\right)}\left\{k_m d_m^{(1)}(z_s) + \frac{\mathrm{d}d_m^{(2)}(z)}{\mathrm{d}z}\bigg|_{z_s}\right\},$$

$$w(r,z) = \sum_m \frac{d_m^{(2)}(z)}{8c_m U_m I_m}\sqrt{\frac{2}{\pi k_m r}}\, e^{\imath\left(k_m r + \frac{\pi}{4}\right)}\left\{k_m d_m^{(1)}(z_s) + \frac{\mathrm{d}d_m^{(2)}(z)}{\mathrm{d}z}\bigg|_{z_s}\right\},$$

where $U_m = \frac{\mathrm{d}\omega}{\mathrm{d}k_m}$ is the group velocity of the displacement mode, $c_m = \frac{\omega}{k_m}$ is the phase velocity of the displacement mode and $I_m = \frac{1}{2}\int \rho(z)\left(d_m^{(1)^2}(z) + d_m^{(2)^2}(z)\right)\mathrm{d}z$. Denote $\xi_m(z) = \left(\xi_{m,1}(z), \xi_{m,2}(z), \xi_{m,3}(z), \xi_{m,4}(z)\right)^{\mathrm{T}} = \left(\frac{d_m^{(1)}(z)}{ik_m}, \imath d_m^{(2)}(z), \frac{\tau_m^{(zx)}(z)}{\imath k_m}, \tau_m^{(zz)}(z)\right)^{\mathrm{T}}$ be the stress-displacement mode, where $\tau_m^{(zx)}(z)$ and $\tau_m^{(zz)}(z)$ are tangential stress mode and normal stress mode, respectively. The stress-displacement mode satisfies the following system of differential equations,

$$\begin{cases} \frac{d\xi_{m,1}(z)}{dz} = -\xi_{m,2}(z) + \frac{1}{\mu(z)}\xi_{m,4}(z), \\ \frac{d\xi_{m,2}(z)}{dz} = \frac{\lambda(z)k_m^2}{\lambda(z)+2\mu(z)}\xi_{m,1}(z) + \frac{1}{\lambda(z)+2\mu(z)}\xi_{m,4}(z), \\ \frac{d\xi_{m,3}(z)}{dz} = \left(\frac{4\mu(z)(\lambda(z)+\mu(z))k_m^2}{\lambda(z)+2\mu(z)} - \rho(z)\omega^2\right)\xi_{m,1}(z) - \frac{\lambda(z)}{\lambda(z)+2\mu(z)}\xi_{m,4}(z), \\ \frac{d\xi_{m,4}(z)}{dz} = -\rho(z)\omega^2\xi_{m,2}(z) + k_m^2\xi_{m,3}(z), \end{cases} \tag{C.2}$$

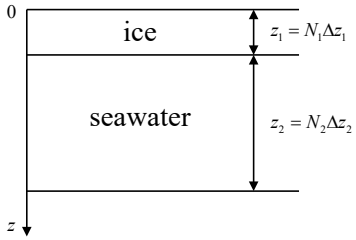


Figure 7: A simple ocean environment model.

where $\lambda(z)$ and $\mu(z)$ are the Lamé coefficients of ice.

A simplified ocean environment model is depicted in Figure 7, the depths of ice and seawater are discretized into $z_1 = N_1\Delta z_1$ and $z_2 = N_2\Delta z_2$. Coupled with the boundary conditions and by finite difference methods, Equation (C.1) and (C.2) can be transformed into the GEPs of the form

$$AV = BV\Sigma, \tag{C.3}$$

where A and B are sparse structured matrices, V is a matrix whose columns are generalized eigenvectors which store modes $\varphi_m(z)$, $d_m^{(1)}(z)$, $d_m^{(2)}(z)$, and Σ is a diagonal matrix with its diagonal elements being generalized eigenvalues which are the square of the horizontal wave numbers k_m. The dimensions of A, B are $4N_1 + N_2 + 5$, which are typically quite large. All non-zero-value elements are located near the diagonal with strong repeatability. We present the structure of matrices A and B in Figure 8.

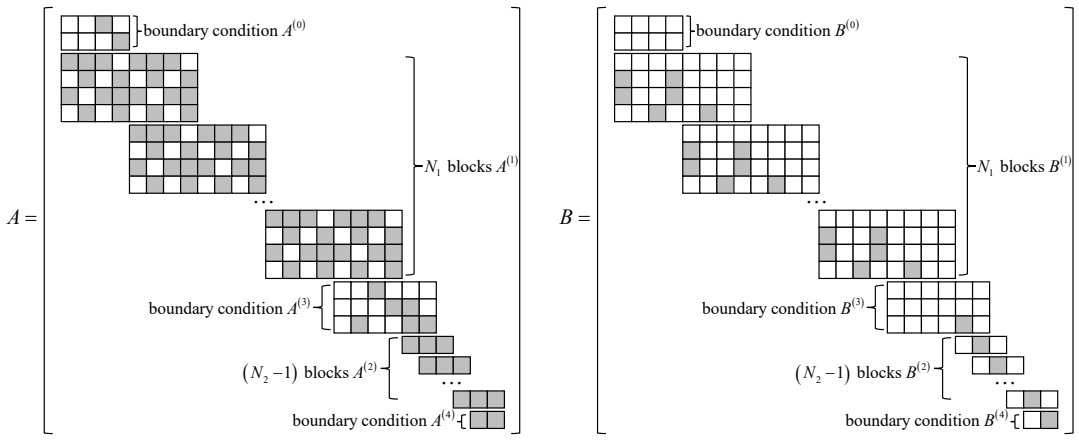


Figure 8: The matrices A and B of GEPs in Equation (C.3), where the gray and white squares represent a non-zero element and a zero element, respectively.

The sub-matrix $\{A^{(i)}\}_{i=0}^4$ and $\{B^{(i)}\}_{i=0}^4$ in the Figure 8 are defined as

$$A^{(0)} = \begin{pmatrix} 0 & 0 & a_3 & 0 \\ 0 & 0 & 0 & a_3 \end{pmatrix}, \quad A^{(1)} = \begin{pmatrix} a_0 & a_2 & a_4 & 0 & a_7 & a_2 & a_4 & 0 \\ 0 & a_0 & 0 & a_5 & 0 & a_7 & 0 & a_5 \\ a_1 & 0 & a_0 & a_6 & a_1 & 0 & a_7 & a_6 \\ 0 & a_1 & 0 & a_0 & 0 & a_1 & 0 & a_7 \end{pmatrix}, \quad A^{(2)} = \begin{pmatrix} a_3 & a_{10} & a_3 \end{pmatrix},$$

$$A^{(3)} = \begin{pmatrix} 0 & 0 & a_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_3 & a_3 & 0 \\ 0 & a_8 & 0 & 0 & a_9 & a_3 \end{pmatrix}, \quad A^{(4)} = \begin{pmatrix} a_{11} & a_{12} \end{pmatrix},$$

$$B^{(0)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad B^{(1)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_0 & 0 & 0 & b_0 & 0 & 0 & 0 & 0 \\ b_2 & 0 & 0 & b_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & b_1 & 0 & 0 & b_1 & 0 & 0 \end{pmatrix}, \quad B^{(2)} = \begin{pmatrix} 0 & b_4 & 0 \end{pmatrix},$$

$$B^{(3)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & b_3 & 0 \end{pmatrix}, \quad B^{(4)} = \begin{pmatrix} 0 & b_5 \end{pmatrix}.$$

Keys	**Values**
0	$\{(a_{ij}, i, j) : a_{ij} = a_0, (i,j) \in (j+2, [0, 4N_1 - 1])\}$
1	$\{(a_{ij}, i, j) : a_{ij} = a_1, (i,j) \in (j+4, \{[0, 4N_1 - 3] : \mathrm{mod}(j,4) = 0 \text{ or } 1\})\}$
2	$\{(a_{ij}, i, j) : a_{ij} = a_1, (i,j) \in (j, \{[4, 4N_1 + 1] : \mathrm{mod}(j,4) = 0 \text{ or } 1\})\}$
3	$\{(a_{ij}, i, j) : a_{ij} = a_2, (i,j) \in (j+1, \{[1, 4N_1 - 3] : \mathrm{mod}(j,4) = 1\})\}$
4	$\{(a_{ij}, i, j) : a_{ij} = a_2, (i,j) \in (j-3, \{[5, 4N_1 + 1] : \mathrm{mod}(j,4) = 1\})\}$
5	$\{(a_{ij}, i, j) : a_{ij} = a_3, (i,j) \in (j+1, \{[4N_1 + 4, 4N_1 + N_2 + 2]\})\}$ $\bigcup \{(a_{ij}, i, j) : a_{ij} = a_3, (i,j) \in (j, \{4N_1 + 2, 4N_1 + 3\})\}$ $\bigcup \{(a_{ij}, i, j) : a_{ij} = a_3, (i,j) \in (j-2, \{2, 3\})\}$
6	$\{(a_{ij}, i, j) : a_{ij} = a_3, (i,j) \in (j-1, \{[4N_1 + 4, 4N_1 + N_2 + 4]\})\}$
7	$\{(a_{ij}, i, j) : a_{ij} = a_4 (i,j) \in (j, \{[2, 4N_1 - 2] : \mathrm{mod}(j,4) = 2\})\}$
8	$\{(a_{ij}, i, j) : a_{ij} = a_4, (i,j) \in (j-4, \{[6, 4N_1 + 2] : \mathrm{mod}(j,4) = 2\})\}$
9	$\{(a_{ij}, i, j) : a_{ij} = a_5, (i,j) \in (j-4, \{[7, 4N_1 + 3] : \mathrm{mod}(j,4) = 3\})\}$
10	$\{(a_{ij}, i, j) : a_{ij} = a_5, (i,j) \in (j, \{[3, 4N_1 - 1] : \mathrm{mod}(j,4) = 3\})\}$
11	$\{(a_{ij}, i, j) : a_{ij} = a_6, (i,j) \in (j-3, \{[7, 4N_1 + 3] : \mathrm{mod}(j,4) = 3\})\}$
12	$\{(a_{ij}, i, j) : a_{ij} = a_6, (i,j) \in (j+1, \{[3, 4N_1 - 1] : \mathrm{mod}(j,4) = 3\})\}$
13	$\{(a_{ij}, i, j) : a_{ij} = a_7, (i,j) \in (j-2, [4, 4N_1 + 3])\}$
14	$\{(a_{ij}, i, j) : a_{ij} = a_8, (i,j) \in (j+3, \{4N_1 + 1\})\}$
15	$\{(a_{ij}, i, j) : a_{ij} = a_9, (i,j) \in (j, \{4N_1 + 4\})\}$
16	$\{(a_{ij}, i, j) : a_{ij} = a_{10}, (i,j) \in (j, [4N_1 + 5, 4N_1 + N_2 + 3])\}$
17	$\{(a_{ij}, i, j) : a_{ij} = a_{11}, (i,j) \in (j+1, \{4N_1 + N_2 + 3\})\}$
18	$\{(a_{ij}, i, j) : a_{ij} = a_{12}, (i,j) \in (j, \{4N_1 + N_2 + 4\})\}$

Table 9: The dictionary data structure of the matrix A of GEPs in Equation (5.2).

Keys	**Values**
0	$\{(b_{ij}, i, j) : b_{ij} = b_0, (i,j) \in (j+3, \{[0, 4N_1 - 4] : \mathrm{mod}(j,4) = 0\})\}$
1	$\{(b_{ij}, i, j) : b_{ij} = b_0, (i,j) \in (j-1, \{[4, 4N_1] : \mathrm{mod}(j,4) = 0\})\}$
2	$\{(b_{ij}, i, j) : b_{ij} = b_1, (i,j) \in (j+3, \{[2, 4N_1 - 2] : \mathrm{mod}(j,4) = 2\})\}$
3	$\{(b_{ij}, i, j) : b_{ij} = b_1, (i,j) \in (j-1, \{[6, 4N_1 + 2] : \mathrm{mod}(j,4) = 2\})\}$
4	$\{(b_{ij}, i, j) : b_{ij} = b_2, (i,j) \in (j+4, \{[0, 4N_1 - 4] : \mathrm{mod}(j,4) = 0\})\}$
5	$\{(b_{ij}, i, j) : b_{ij} = b_2, (i,j) \in (j, \{[4, 4N_1] : \mathrm{mod}(j,4) = 0\})\}$
6	$\{(b_{ij}, i, j) : b_{ij} = b_3, (i,j) \in (j, \{4N_1 + 4\})\}$
7	$\{(b_{ij}, i, j) : b_{ij} = b_4, (i,j) \in (j, [4N_1 + 5, 4N_1 + N_2 + 3])\}$
8	$\{(b_{ij}, i, j) : b_{ij} = b_5, (i,j) \in (j, \{4N_1 + N_2 + 4\})\}$

Table 10: The dictionary data structure of the matrix B of GEPs in Equation (5.2).