Efficient Classical-Quantum Algorithms for Matrix Encoding

Liron Mor Yosef, Haim Avron

Abstract

We introduce an efficient classical-quantum algorithm for encoding arbitrary dense Hermitian matrices as Block Encoding circuits ($\mathbf{U}_{\mathbf{A}} \in \mathbf{BE}_{\alpha,\theta}(\mathbf{A})$). Our work is motivated by Block Encoding's fundamental role as the leading paradigm for quantum linear algebra, providing a unified framework for leveraging quantum computing to accelerate numerical linear algebra operations. Our algorithms, accepts four distinct input representations: (1) classical matrix description $\mathbf{A} \in \mathbb{C}^{n\times n}$, (2) an $4 \times 4 \times \cdots \times 4$ ($\log n$ times) Pauli coefficients tensor $\mathbf{A}_P$, (3) matrix state preparation circuit $\mathcal{U}_{\mathbf{A}}$, or (4) matrix state preparation circuit for the Pauli tensor $\mathcal{U}_{\mathbf{A}_P}$. This flexibility optimizes performance across different data availability scenarios, with the classical matrix input achieving $O(n^2 \log n)$ run-time complexity in the worst case. Moreover, the third input model demonstrates a significant breakthrough: the first known method to construct a Block Encoding circuit directly from a matrix state preparation circuit without requiring additional classical information (such as row norms) or additional quantum hardware (such as QRAM). This establishes a new bidirectional equivalence between block encoding and matrix state preparation input models, providing a unified framework for matrix encoding in quantum algorithms.

1 Introduction

1.1 Motivation and problem statement

Quantum computers hold hope for significant speedups in scientific computing and machine learning due to their ability to handle matrix operations efficiently [3]. However, unlocking this potential hinges the algorithm's ability to efficiently access classical data within the quantum system. The mechanism in which classical input is fed into a quantum algorithm is known as the algorithm's *input model*.

Leveraging breakthroughs in quantum linear algebra, researchers have proposed many quantum algorithms for scientific computing and machine learning. However, the feasibility of their input model assumptions remains critical to their effectiveness. As shown by Chakraborty et al. [4], these assumptions often significantly impact the performance and efficiency of such algorithms. Prime examples of quantum linear algebra algorithms include the HHL algorithm [8], and others [5, 7, 14, 2, 11].

Given the essential role of the input model in defining how classical data interacts with the quantum system, researchers have explored various approaches. Two noteworthy examples include the sparse-data access model [1, 5] and various quantum data structure based models [9, 10].

In this work, we study the use of Pauli decomposition in developing efficient algorithms to encode arbitrary dense or sparse Hermitian matrices into Block Encoding circuits, either provided as classical data or as quantum circuits, into Block Encoding circuits.

1.2 Brief overview of Block Encoding and State preparation

Chakraborty et al. [4] showed that a variety of the aforementioned widely used input models can be reduced to an input model in which matrices are inputed using *block encodings* and vectors are inputed as *state preparation circuits*:

Definition 1 (State preparation Circuit)**.** We say that a $\log_2 n$-qubit circuit $\mathcal{U}$ is a *state preparation circuit* for a vector $\mathbf{x} \in \mathbb{C}^n$ if applying $\mathcal{U}$ to the state $|0\rangle_{\log_2 n}$ results in the state $|\mathbf{x}\rangle_{\log_2 n}$.

Definition 2 (Block encoding of a matrix)**.** For $\alpha \geq 0$ and $\theta \in [0, 2\pi)$, a circuit $\mathcal{U}$ is a (α, θ)-*Block Encoding* of $\mathbf{A} \in \mathbb{C}^{m \times n}$, denoted as $\mathcal{U} \in \mathbf{BE}_{\alpha,\theta}(\mathbf{A})$, if

$$\alpha e^{i\theta} \mathbf{M}(\mathcal{U}) = \begin{bmatrix} \mathbf{A} & * \\ * & * \end{bmatrix}$$

where $*$ denotes arbitrary entries, and $\mathbf{M}(\mathcal{U})$ denote the unique unitary matrix of the circuit $\mathcal{U}$. We refer to α as the *scale* and θ as the *phase*.

We refer to the input model in which matrices are accessed using block encodings and vectors are accessed as state preparation circuits as the *block encoding input model*. There are powerful algorithms that operate under the block encoding model. In particular, in the block encoding model we can perform Quantum Singular Value Transformation [7], a powerful technique that leads to efficient algorithms for solving linear equations, amplitude amplification, quantum simulation, and more [12].

Another relevant input model is the *state preparation input model*. In this model, matrices accessed via *matrix state preparation circuit* and vectors are accessed via state preparation circuits. Mor-Yosef et al. [15] recently introduced an algorithm for multivariate trace estimation and spectral sums estimation under this model.

Definition 3 (Matrix state preparation circuit)**.** We say that a $(\log_2 n + \log_2 m)$-qubit circuit $\mathcal{U}$ is a *matrix state preparation circuit* for a matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ if applying $\mathcal{U}$ to the state $|0\rangle_{\log_2 mn}$ results in the state $||\mathbf{A}\rangle\rangle := |\mathbf{vec}(\mathbf{A})\rangle$. Equivalently, the first column of $\mathbf{M}(\mathcal{U})$ is $\mathbf{vec}(\mathbf{A})$.

For convenience, where appropriate, we add the matrix as sub-index when denoting state preparation circuits, e.g. $\mathcal{U}_{\mathbf{A}}$. In such cases, with an abuse of notation, the number of gates in $\mathcal{U}_{\mathbf{A}}$ is denoted by $g_{\mathbf{A}}$, and the depth by $d_{\mathbf{A}}$.

1.3 Statement of main results

While existing block encoding methods typically exploit specific matrix properties like structure, sparsity, or rank, we introduces an efficient general-purpose technique for arbitrary matrices (dense and sparse) through Pauli decomposition. These techniques will utilize the principles of Pauli decomposition, and the use of a quantum multiplexer.

The Pauli decomposition represents matrices as a sum of tensor products of Pauli matrices. Formally, let $\Sigma = \{\mathbf{I} = 0, \mathbf{X} = 1, \mathbf{Y} = 2, \mathbf{Z} = 3\}$ represent a set of indices corresponding to the four 2×2 Pauli matrices: $\sigma_I, \sigma_X, \sigma_Y, \sigma_Z$. Assume that $n = 2^q$. Given a *word* (i.e., sequence) $\mathcal{W} = (w_1, w_2, \ldots, w_q) \in \Sigma^q$, we define the corresponding *q-wise Pauli matrix* as $\sigma_{\mathcal{W}} := \sigma_{w_1} \otimes \sigma_{w_2} \otimes \cdots \otimes \sigma_{w_q}$. Then, the Pauli decomposition of matrix $\mathbf{A}$ can be expressed mathematically as:

$$\mathbf{A} = \sum_{\mathcal{W} \in \Sigma^q} \alpha_{\mathcal{W}} \sigma_{\mathcal{W}}$$

where $\alpha_{\mathcal{W}} \in \mathbb{R}$ are real coefficients.

The quantum multiplexer [13] acts like a switch within a quantum circuit. It uses control qubits to selectively apply different unitary operations to a target qubit. Given a set of quantum circuits $\mathcal{U}_0, \ldots, \mathcal{U}_{k-1}$ the $\log k$-qubit multiplexer is defined as:

$$\mathcal{MX}_{\log k} \coloneqq \sum_{i=0}^{k-1} |i\rangle \langle i| \otimes \mathcal{U}_i.$$

Importantly, $\mathcal{MX}_{\log k}$ acts as a multiplexer. In other words:

$$\mathcal{MX}_{\log k}(\underbrace{|i\rangle}_{control} \underbrace{|\psi\rangle}_{input}) = \underbrace{|i\rangle}_{control} \underbrace{\mathcal{U}_i |\psi\rangle}_{output}.$$

The matrix that represents the multiplexer is a block diagonal matrix of the corresponding operators:

$$\mathbf{M}(\mathcal{MX}_{\log k}) = \mathbf{diag}\left(\mathbf{M}(\mathcal{U}_0), \ldots, \mathbf{M}(\mathcal{U}_{k-1})\right)$$

Once we have obtained the Pauli coefficients of the matrix, we can utilize a multiplexer to construct a block-diagonal matrix composed of the corresponding q-wise Pauli matrices. By employing a state preparation circuit for the coefficients, we can efficiently implement linear combinations of coefficients multiplied by matrices, effectively creating a block encoding of the matrix $\mathbf{A}$.

To efficiently determine the Pauli coefficients classically, we require some theoretical groundwork.

1.4 Contribution

This work makes three key contributions: (a) the first bidirectional equivalence between block encoding and matrix state preparation input models, (b) a novel classical Pauli decomposition algorithm with $O(n^2 \log n)$ run-time complexity, and (c) an efficient quantum circuit implementation for multiplexed Pauli tensor products with $O(n^2)$ gate complexity. This general-purpose approach improves upon existing techniques for arbitrary matrix encoding, achieving particular efficiency when the Pauli decomposition is sparse.

2 Block encoding equivalence

Given a matrix state preparation circuit for $\mathbf{A}$ and a state preparation circuit for a vector $\mathbf{w}$ whose entries are the row norms of $\mathbf{A}$, it is possible to construct a block encoding of $\mathbf{A}$ [6, Section I.D]. We are unaware of any efficient algorithm that given *only* a matrix state preparation circuit for $\mathbf{A}$ constructs a block encoding of $\mathbf{A}$. In this section we will show a way to create block encoding from state preparation circuits and vise versa.

2.1 Block encoding → matrix state preparation circuit

Mor-Yosef et al. [15] show that given a circuit $\mathcal{U}$ we can construct a matrix state preparation of $\mathbf{M}(\mathcal{U})$. Thus, given a block encoding of $\mathbf{A}$ we can immediately construct a matrix state preparation circuit for a matrix that contains $\mathbf{A}$.

The following provides a proof for creating a state preparation circuit, including auxiliary (garbage) quantum states, from Block Encoding.

Proposition 4. Suppose that $\mathcal{U} \in \mathbf{BE}_{\alpha,\theta}(\mathbf{A})$. Applying the 'qml.matrix' results $\mathcal{U}_{\mathbf{M}(\mathcal{U})}$ s.t $\mathcal{U}_{\mathbf{M}(\mathcal{U})} \in \mathbf{MS}_{\alpha,\theta}(\mathbf{A})$

Proof. We have that,

$$\begin{aligned}
\mathcal{U}_{\mathbf{M}(\mathcal{U})} &= \begin{bmatrix} \mathbf{vec}(\mathbf{M}(\mathcal{U})) & * \end{bmatrix} \\
&= \alpha^{-1} e^{-i\theta} \begin{bmatrix} \mathbf{vec}\left(\begin{bmatrix} \mathbf{A} & * \\ * & * \end{bmatrix}\right) & * \end{bmatrix} \\
&= \alpha^{-1} e^{-i\theta} \begin{bmatrix} \mathbf{vec}(\mathbf{A}) & * \\ \psi & * \end{bmatrix}
\end{aligned}$$

□

2.2 Matrix state preparation circuit → block encoding

We introduce a method to create block encoding solely from a matrix state preparation circuit. Preliminary examples are provided to illustrate this approach, with the full methodology detailed in the paper. At a high level, we construct $\mathcal{U}_{\mathbf{A}_p}$ from $\mathcal{U}_{\mathbf{A}}$, and then apply the technique from the previous section to block encode $\mathbf{A}$.

In high level, we construct $\mathcal{U}_{\mathbf{A}_p}$ from $\mathcal{U}_{\mathbf{A}}$, then use the technique from the previous section to block encode $\mathbf{A}$. We will now demonstrate how to compute $\mathcal{U}_{\mathbf{A}_p}$ in the following section.

2.2.1 Construct $\mathcal{U}_{\mathbf{A}_p}$ from $\mathcal{U}_{\mathbf{A}}$(Warm-up: $q = 1$ and Real A)

As a warm-up, let us first consider the case of $q = 1$, i.e. the $\mathbf{A} \in \mathbb{R}^{n\times n}$ a 2 by 2 real and Hermitian matrices. To keep notation simple, we use the 1 base index, i.e.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad a_{12} = a_{21}$$

Note that we can write $\mathbf{A}$ as a linear combination of the following matrices,

$$\mathbf{E}_{11} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{E}_{12} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \mathbf{E}_{21} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \mathbf{E}_{22} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Indeed we have that

$$\mathbf{A} = \sum_{i,j} a_{ij}\mathbf{E}_{i,j} = a_{11}\mathbf{E}_{11} + a_{12}\mathbf{E}_{12} + a_{21}\mathbf{E}_{21} + a_{22}\mathbf{E}_{22}$$

From linearity of $(\cdot)_p$we have that

$$\mathbf{A}_p = \left(\sum_{i,j} a_{ij}\mathbf{E}_{ij}\right)_p = \mathbf{A} = \sum_{i,j} a_{ij}\left(\mathbf{E}_{ij}\right)_p$$

Note that we can create state preparation circuits $\{\mathcal{U}_{(\mathbf{E}_{ij})_p}\}_{i,j}$, using the standard state prepartion operation ([13]). Now we can create the $\mathcal{U}_{\mathbf{A}_p}$ as follow:

This observation can easily be used to implement, via qMSLA operations, an algorithm that takes $\mathcal{U}_{\mathbf{A}}$ and outputs $\mathcal{U}_{\mathbf{A}_p}$.

References

[1] Andris Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *STACS'12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 636–647. LIPIcs, 2012.

[2] Dong An and Lin Lin. Quantum linear system solver based on time-optimal adiabatic quantum computing and quantum approximate optimization algorithm. *ACM Transactions on Quantum Computing*, 3(2):1–28, 2022.

[3] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[4] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The Power of Block-Encoded Matrix Powers: Improved Regression Techniques via Faster Hamiltonian Simulation. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 33:1–33:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[5] 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.

[6] 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.

[7] 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*, pages 193–204, 2019.

[8] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009.

[9] Iordanis Kerenidis and Anupam Prakash. Quantum Recommendation Systems. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:21, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[10] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101(2):022316, 2020.

[11] Lin Lin and Yu Tong. Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum*, 4:361, 2020.

[12] John M Martyn, Zane M Rossi, Andrew K Tan, and Isaac L Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2(4):040203, 2021.

[13] Vivek V Shende, Stephen S Bullock, and Igor L Markov. Synthesis of quantum logic circuits. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, pages 272–275, 2005.

[14] Yigit Subasi, Rolando D Somma, and Davide Orsucci. Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing. *Physical Review Letters*, 122(6):060504, 2019.

[15] Liron Mor Yosef, Shashanka Ubaru, Lior Horesh, and Haim Avron. Multivariate trace estimation using quantum state space linear algebra. *arXiv preprint arXiv:2405.01098*, 2024.