New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
10 October 2024

REVISED
29 January 2025

ACCEPTED FOR PUBLICATION
10 February 2025

PUBLISHED
28 February 2025

PAPER

Pauli decomposition via the fast Walsh-Hadamard transform

Timothy N Georges[1,2,3], **Bjorn K Berntson**[1,3], **Christoph Sünderhauf**[1,3] **and Aleksei V Ivanov**[1,3,*]

[1] Riverlane Ltd, St Andrews House, 59 St Andrews Street, Cambridge CB2 3BZ, United Kingdom
[2] Department of Chemistry, Physical and Theoretical Chemistry Laboratory, University of Oxford, Oxford OX1 3QZ, United Kingdom
[3] All authors contributed equally.
[*] Author to whom any correspondence should be addressed.

E-mail: aleksei.ivanov@riverlane.com

Keywords: Pauli decomposition, linear combination of unitaries, fast Walsh–Hadamard transform

Abstract

The decomposition of a square matrix into a sum of Pauli strings is a classical pre-processing step required to realize many quantum algorithms. Such a decomposition requires significant computational resources for large matrices. We present an exact and explicit formula for the Pauli string coefficients which inspires an efficient algorithm to compute them. More specifically, we show that up to a permutation of the matrix elements, the decomposition coefficients are related to the original matrix by a multiplication of a generalised Hadamard matrix. This allows one to use the Fast Walsh-Hadamard transform and calculate all Pauli decomposition coefficients in $\mathcal{O}(N^2 \log N)$ time and using $\mathcal{O}(1)$ additional memory, for an $N \times N$ matrix. A numerical implementation of our equation outperforms currently available solutions.

1. Introduction

The Pauli matrices $\mathcal{P} := \{I, X, Y, Z\}$, where

$$I := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y := \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{1}$$

form a set of elementary gates, ubiquitous in the field of quantum computation and information [1]. In order to apply a quantum algorithm to a specific application area such as simulation of quantum systems [2–5] or solving systems of linear equations [6], it is often desirable to express the problem input, such as a Hamiltonian or more generally a complex matrix, $A \in \mathbb{C}^{2^n \times 2^n}$ with a positive integer n, as a weighted sum of tensor products of Pauli matrices, also known as Pauli strings:

$$A = \sum_{P \in \mathcal{P}_n} \alpha(P) P, \tag{2}$$

where

$$\mathcal{P}_n := \left\{ \bigotimes_{j=0}^{n-1} P_j : P_j \in \mathcal{P} \right\} \tag{3}$$

and $\alpha : \mathcal{P}_n \to \mathbb{C}$. The problem is then to compute the Pauli coefficients $(\alpha(P))_{P \in \mathcal{P}_n}$ efficiently.

An exact expression for the Pauli coefficients is given by

$$\alpha(P) = \frac{1}{2^n} \mathrm{tr}\left(A^\dagger P\right) \quad (P \in \mathcal{P}_n), \tag{4}$$

which follows from equation (2) and the fact that $\mathcal{P}_n$ forms an orthonormal basis under the Hilbert–Schmidt inner product $\frac{1}{2^n}\mathrm{tr}(A^\dagger B)$ on the vector space $\mathbb{C}^{2^n \times 2^n}$. Unfortunately, naive implementation of equation (4) suffers from poor scaling, $\mathcal{O}(2^{5n})$, and this equation is not easy to use for a derivation of an explicit linear-combination-of-unitaries decomposition of a Hamiltonian or a matrix used in specific applications.

In this work, we derive an exact and explicit representation of α based on a Walsh–Hadamard transform, defined as multiplication by the nth tensor power $H^{\otimes n}$ of the Hadamard matrix

$$H := \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{5}$$

Namely, for a matrix $A = (a_{p,q})_{p,q=0}^{2^n-1} \in \mathbb{C}^{2^n \times 2^n}$, the following theorem holds.

Theorem 1 (Pauli decomposition of $A \in \mathbb{C}^{2^n \times 2^n}$ via the Walsh–Hadamard transform). *The following decomposition*

$$A = \sum_{r,s=0}^{2^n-1} \alpha_{r,s} P_{r,s}, \tag{6}$$

where

$$P_{r,s} := \bigotimes_{j=0}^{n-1} \left(i^{r_j \wedge s_j} X^{r_j} Z^{s_j}\right) \in \mathcal{P}_n \quad \left(r,s \in [2^n-1]_0\right) \tag{7}$$

and

$$\alpha_{r,s} := \frac{i^{-|r\wedge s|}}{2^n} \sum_{q=0}^{2^n-1} a_{q\oplus r,q} \left(H^{\otimes n}\right)_{q,s} \quad \left(r,s \in [2^n-1]_0\right), \tag{8}$$

holds. Conversely, A can be recovered from its Pauli string decomposition according to

$$a_{r,s} = i^{|\bar{r}\wedge s|} \sum_{q=0}^{2^n-1} \alpha_{r\oplus s,q} \left(H^{\otimes n}\right)_{q,s}. \tag{9}$$

In theorem 1, $\wedge$, $\oplus$, and $\bar{\cdot}$ are the bitwise AND, XOR, and NOT operators, respectively, $|\cdot|$ is the Hamming weight, $\{r_j\}_{j=0}^{n-1}$ and $\{s_j\}_{j=0}^{n-1}$ are the binary expansion coefficients of r and s, respectively, and $[2^n-1]_0 := \{0,1,\ldots 2^n-1\}$. Precise definitions of these standard notations can be found at the beginning of section 2. As we show in the proof of theorem 1, $P_{r,s}$ is indeed a Pauli string because $i^{r_j\wedge s_j} X^{r_j} Z^{s_j} \in \mathcal{P} := \{I,X,Y,Z\}$ for $r_j, s_j \in \{0,1\}$.

A naive implementation of matrix multiplication performed in equation (8) scales as $\mathcal{O}(2^{3n})$. However, one can use the Fast Walsh–Hadamard transform [7, chapter 7], which scales as $\mathcal{O}(n2^{2n})$ and requires only $n2^{2n}$ addition and $n2^{2n}$ subtraction operations on complex numbers, with $\mathcal{O}(1)$ additional memory. Furthermore, because XOR transformation can be implemented in place, full Pauli decomposition can be implemented with $\mathcal{O}(1)$ memory overhead.

Relation to previous work. To the best of our knowledge, the explicit equations (8) and (9) and their proofs have not been published before. There are several algorithms developed for calculating the Pauli coefficients α [8–16]. To date, the best scaling methods achieve the time complexity $\mathcal{O}(n2^{2n})$ [13–16] of the present algorithm. The previous fastest approach, as measured by CPU time, is the optimized Rust implementation of the tensorised Pauli decomposition (TPD) algorithm of Hantzko, Binkowski, and Gupta [14] in Qiskit [17, 18]. As shown in section 3, our approach outperforms this implementation. The most similar algorithm is that of Gidney [15] which we learnt about after posting this work online. It was proposed in Stack-Overflow comment [15] without though an explanation of how this algorithm was derived. Gidney's code then was also incorporated in Pennylane [19]. The main difference of [15] to ours is that our implementation carries out all operations, including XOR transformation, in-place. Hamaguchi, Hamada and Yoshioka proposed an algorithm [16] which is based on the Fast Walsh–Hadamard transformation. While the central primitive/subroutine in their algorithm is also the Fast Walsh–Hadamard transformation, we observe that our implementation is around 3x faster as shown in the results section. We also note that the (Fast) Walsh–Hadamard transformation is well understood and its fast implementations are known [7, 20]. This transformation is used in other areas of quantum computation. For example, in the context of Pauli channels, it was shown that Pauli error rates and Pauli eigenvalues are related to each other through such a transformation [21] and its fast implementation is used in several papers [22, 23]. However, as evident by equation (8), the Walsh–Hadamard transformation by itself does not provide the Pauli decomposition of a matrix as the application of XOR transformation and the correct phase factors must still be applied.

The paper is organised as follows. Our theoretical results, including a rigorous proof of theorem 1, are contained in section 2. In this section, we also consider special cases where A possesses certain symmetries, establishing three corollaries of theorem 1. The numerical algorithm inspired by equations (6)–(8) is presented in section 3 together with comparison against the Qiskit implementation of TPD and the algorithm of Hamaguchi *et al* [16]. In section 4, we draw conclusions from the results presented herein and discuss the outlook for Pauli decomposition in the context of our results.

2. Theoretical results

In this section, we prove our main result, theorem 1. Additionally, we state and prove three corollaries concerning the special cases where the matrix to be decomposed is (i) Hermitian, (ii) real symmetric or (iii) complex symmetric.

To make our statements and proofs precise, we introduce some technology. Given a positive integer N, we define $[N]_0 := \{0, 1, \ldots, N\}$. For any $p \in [2^n - 1]_0$, there exists a unique binary decomposition

$$p = \sum_{j=0}^{n-1} p_j 2^j \quad \left(p_j \in \{0,1\}\right), \tag{10}$$

which defines a map $p \mapsto (p_j)_{j=0}^{n-1} \in \{0,1\}^n$. Moreover, we may use equation (10) to extend operations $\{0,1\} \times \{0,1\} \to \{0,1\}$ on bits to operations $[2^n-1]_0 \times [2^n-1]_0 \to [2^n-1]_0$ on n-bit integers, by bitwise application. Let $x, y \in \{0,1\}$ and recall the following operations

$$\text{NOT: } \bar{x} := 1 - x, \quad \text{XOR: } x \oplus y := \bar{x}y + x\bar{y}, \quad \text{AND: } x \wedge y := xy. \tag{11}$$

Bitwise operations on integers are defined by composing the $\oplus$ (XOR) and $\wedge$ (AND) operators with equation (10) yielding the definitions

$$\text{XOR: } p \oplus q := \sum_{j=0}^{n-1} \left(p_j \oplus q_j\right) 2^j, \quad \text{AND: } p \wedge q := \sum_{j=0}^{n-1} \left(p_j \wedge q_j\right) 2^j \quad \left(p, q \in [2^n - 1]_0\right). \tag{12}$$

We also make use of the Hamming weight,

$$|p| := \sum_{j=0}^{n-1} p_j \quad \left(p \in [2^n - 1]_0\right). \tag{13}$$

2.1. Proof of theorem 1

Let $E_{p,q}$ be a matrix unit for $\mathbb{C}^{2^n \times 2^n}$, i.e.

$$E_{p,q} := \left(\delta_{j,p}\delta_{k,q}\right)_{j,k=0}^{2^n-1} \quad \left(p, q \in [2^n - 1]_0\right). \tag{14}$$

We write

$$E_{p,q} = \bigotimes_{j=0}^{n-1} e_j(p,q), \tag{15}$$

where

$$e_j(p,q) := \left(\delta_{k,p_j}\delta_{l,q_j}\right)_{k,l=0}^{1} \quad \left(j \in [n-1]_0; p, q \in [2^n - 1]_0\right). \tag{16}$$

The Pauli matrices form a basis for $\mathbb{C}^{2\times 2}$; for the matrix units equation (16), the explicit decomposition is

$$e_j(p,q) = \frac{1}{2}\left(X^{p_j \oplus q_j} + (-1)^{p_j} (iY)^{p_j \oplus q_j} Z^{\overline{p_j \oplus q_j}}\right). \tag{17}$$

Next, using the identity $XZ = -iY$, we obtain

$$e_j(p,q) = \frac{1}{2}\left(X^{p_j \oplus q_j} + (-1)^{q_j} X^{p_j \oplus q_j} Z\right). \tag{18}$$

Putting equation (18) into equation (15) yields

$$E_{p,q} = \frac{1}{2^n} \bigotimes_{j=0}^{n-1} \left(X^{p_j \oplus q_j} + (-1)^{q_j} X^{p_j \oplus q_j} Z \right) = \frac{1}{2^n} \bigotimes_{j=0}^{n-1} \sum_{s_j=0,1} (-1)^{q_j \wedge s_j} X^{p_j \oplus q_j} Z^{s_j} = \frac{1}{2^n} \prod_{j=0}^{n-1} \sum_{s_j=0,1} (-1)^{q_j \wedge s_j} X_j^{p_j \oplus q_j} Z_j^{s_j}, \tag{19}$$

where

$$X_j := I^{\otimes j} \otimes X \otimes I^{\otimes (n-j-1)}, \quad Z_j := I^{\otimes j} \otimes Z \otimes I^{\otimes (n-j-1)} \quad \left(j \in [n-1]_0 \right). \tag{20}$$

It follows from equation (19) that

$$E_{p,q} = \frac{1}{2^n} \left(\sum_{s_0=0,1} \cdots \sum_{s_{n-1}=0,1} \right) \prod_{j=0}^{n-1} (-1)^{q_j \wedge s_j} X_j^{p_j \oplus q_j} Z_j^{s_j}, \tag{21}$$

which is equivalent to

$$E_{p,q} = \frac{1}{2^n} \sum_{s=0}^{2^n-1} \prod_{j=0}^{n-1} (-1)^{q_j \wedge s_j} X_j^{p_j \oplus q_j} Z_j^{s_j} \tag{22}$$

with s defined via equation (10).

To proceed, we will express the constant factors in equation (22) using tensor products of Hadamard matrices, equation (5). Note that

$$H_{q_j, s_j} = (-1)^{q_j \wedge s_j} \quad \left(q_j, s_j \in \{0, 1\} \right) \tag{23}$$

and hence,

$$\left(H^{\otimes n} \right)_{q,s} = \prod_{j=0}^{n-1} (-1)^{q_j \wedge s_j} \quad \left(q, s \in [2^n - 1]_0 \right). \tag{24}$$

Using this observation in equation (22) gives

$$E_{p,q} = \frac{1}{2^n} \sum_{s=0}^{2^n-1} \left(\prod_{j=0}^{n-1} (-1)^{q_j \wedge s_j} \right) \prod_{j=0}^{n-1} X_j^{p_j \oplus q_j} Z_j^{s_j} = \frac{1}{2^n} \sum_{s=0}^{2^n-1} \left(H^{\otimes n} \right)_{q,s} \prod_{j=0}^{n-1} X_j^{p_j \oplus q_j} Z_j^{s_j}. \tag{25}$$

Decomposing A into a sum of elementary matrices and using equation (25), we write

$$A = \frac{1}{2^n} \sum_{p,q,s=0}^{2^n-1} a_{p,q} \left(H^{\otimes n} \right)_{q,s} \prod_{j=0}^{n-1} X_j^{p_j \oplus q_j} Z_j^{s_j}. \tag{26}$$

Let us introduce a new variable of summation, $r := p \oplus q$. Note that for each $q \in [2^n - 1]_0$, the map $p \mapsto p \oplus q$ is a bijection $[2^n - 1]_0 \to [2^n - 1]_0$ and that $p = q \oplus r$ and $r_j = p_j \oplus q_j$. Hence,

$$A = \frac{1}{2^n} \sum_{r,s=0}^{2^n-1} \sum_{q=0}^{2^n-1} a_{q \oplus r, q} \left(H^{\otimes n} \right)_{q,s} \prod_{j=0}^{n-1} X_j^{r_j} Z_j^{s_j} = \sum_{r,s=0}^{2^n-1} \left(\frac{i^{-|r \wedge s|}}{2^n} \sum_{q=0}^{2^n-1} a_{q \oplus r, q} \left(H^{\otimes n} \right)_{q,s} \right) \left(i^{|r \wedge s|} \prod_{j=0}^{n-1} X_j^{r_j} Z_j^{s_j} \right), \tag{27}$$

which concludes the proof of equations (6)–(8) if we can show that $P_{r,s}$, defined in (7), is an element of $\mathcal{P}_n$. Indeed, using the identity $XZ = -iY$, we compute

$$X^{r_j} Z^{s_j} = \begin{cases} I & (r_j, s_j) = (0, 0) \\ X & (r_j, s_j) = (1, 0) \\ Z & (r_j, s_j) = (0, 1) \\ -iY & (r_j, s_j) = (1, 1), \end{cases} \tag{28}$$

which shows that

$$i^{|r \wedge s|} \prod_{j=0}^{n-1} X_j^{r_j} Z_j^{s_j} = \bigotimes_{j=0}^{n-1} i^{r_j \wedge s_j} X^{r_j} Z^{s_j} \in \mathcal{P}_n. \tag{29}$$

To establish the converse, equation (9), we first write

$$a_{s\oplus r,s} = \frac{1}{2^n}\sum_{u=0}^{2^n-1}\sum_{q=0}^{2^n-1} a_{u\oplus r,u}\left(H^{\otimes n}\right)_{u,q}\left(H^{\otimes n}\right)_{q,s} = i^{|r\wedge s|}\sum_{q=0}^{2^n-1}\alpha_{r,q}\left(H^{\otimes n}\right)_{q,s}, \tag{30}$$

using equation (8) and the fact that $H^{\otimes n}$ is symmetric and satisfies $(H^{\otimes n})^2 = 2^n I^{\otimes n}$. By making the replacement $r \to r\oplus s$ in (30) and using the identity $(r\oplus s)\wedge s = \bar{r}\wedge s$, we obtain the result equation (9).

2.2. Special cases

We consider the special cases where A is (i) Hermitian, (ii) real symmetric, or (iii) (complex) symmetric.

Corollary 1.1 (Decomposition of Hermitian matrices). *Suppose that $A \in \mathbb{C}^{2^n\times 2^n}$ satisfies $A^\dagger = A$. Then, $\alpha_{r,s}\in\mathbb{R}$ for $r,s\in[2^n-1]_0$.*

Proof. As Pauli strings are Hermitian, it follows from equation (6) and our Hermiticity assumption that

$$A - A^\dagger = \sum_{r,s=0}^{2^n-1}\left(\alpha_{r,s}-\alpha^*_{r,s}\right)\bigotimes_{j=0}^{n-1}\left(i^{r_j\wedge s_j}X^{r_j}Z^{s_j}\right) = 0. \tag{31}$$

By the linear independence of $\mathcal{P}_n$, the prefactors of the Pauli strings in (31) must be identically zero. Thus, $\alpha_{r,s}-\alpha^*_{r,s}=0$ for $r,s\in[2^n-1]_0$ and the result follows. □

In the case of symmetric matrices, we can use the following proposition to make statements about the sparsity of $(\alpha_{r,s})_{r,s=0}^{2^n-1}$.

Proposition 1.2 (Distribution of ± 1 in $H^{\otimes n}$). *The matrix $H^{\otimes n}\in\mathbb{C}^{2^n\times 2^n}$ has $2^{n-1}(2^n\pm 1)$ entries with values ± 1, respectively.*

Proof. The claim is easily verified in the case $n=1$. Assume, inductively, that the claim holds for an arbitrary positive integer n. We write $H^{\otimes(n+1)} = H^{\otimes n}\otimes H$; a straightforward calculation shows that $H^{\otimes(n+1)}$ has $2^n(2^{n+1}\pm 1)$ entries with values ± 1, respectively. The result follows. □

Corollary 1.3 (Decomposition of real symmetric matrices). *Suppose that $A\in\mathbb{R}^{2^n\times 2^n}$ satisfies $A^\top = A$. Then, $\alpha_{r,s}\in\mathbb{R}$ when $|r\wedge s|$ is even and $\alpha_{r,s}=0$ when $|r\wedge s|$ is odd. Correspondingly, $(\alpha_{r,s})_{r,s=0}^{2^n-1}$ has a sparsity of $(2^n-1)/2^{n+1}$.*

Proof. The first claim follows immediately from corollary 1.1. To prove the second claim, we use the identity

$$\left(i^{r_j\wedge s_j}X^{r_j}Z^{s_j}\right)^\top = (-1)^{r_j\wedge s_j}\left(i^{r_j\wedge s_j}X^{r_j}Z^{s_j}\right) \tag{32}$$

to group Pauli strings in equation (6):

$$A - A^\top = \sum_{r,s=0}^{2^n-1}\left(\alpha_{r,s}-(-1)^{|r\wedge s|}\alpha_{r,s}\right)\bigotimes_{j=0}^{n-1}\left(i^{r_j\wedge s_j}X^{r_j}Z^{s_j}\right) = 0. \tag{33}$$

By the linear independence of $\mathcal{P}_n$, we must have $\alpha_{r,s}-(-1)^{r\wedge s}\alpha_{r,s}=0$ for $r,s\in[2^n-1]_0$. In the case $|r\wedge s|$ odd, this implies $\alpha_{r,s}=0$.

To compute the sparsity of $(\alpha_{r,s})_{r,s=0}^{2^n-1}$, we note that

$$\left(H^{\otimes n}\right)_{r,s} = (-1)^{|r\wedge s|}\qquad\left(r,s\in[2^n-1]_0\right), \tag{34}$$

i.e. $(H^{\otimes})_{r,s}$ is equal to the parity of $|r\wedge s|$. Hence, using proposition 1.2, we obtain the stated sparsity: The locations of zeros correspond to the locations of -1 in $H^{\otimes n}$. □

Corollary 1.4 (Decomposition of complex symmetric matrices). *Suppose that $A\in\mathbb{C}^{2^n\times 2^n}$ satisfies $A^\top = A$. Then, $\alpha_{r,s}=0$ when $|r\wedge s|$ is odd. Correspondingly, $(\alpha_{r,s})_{r,s=0}^{2^n-1}$ has a sparsity of $(2^n-1)/2^{n+1}$.*

Proof. We write

$$A = \mathrm{Re}(A) + i\,\mathrm{Im}(A) \tag{35}$$

and note that $\mathrm{Re}(A)$ and $\mathrm{Im}(A)$ are real symmetric matrices. By corollary 1.3 and linearity, we obtain the first claim.

The proof of the sparsity claim is similar to that in the proof of corollary 1.3. □

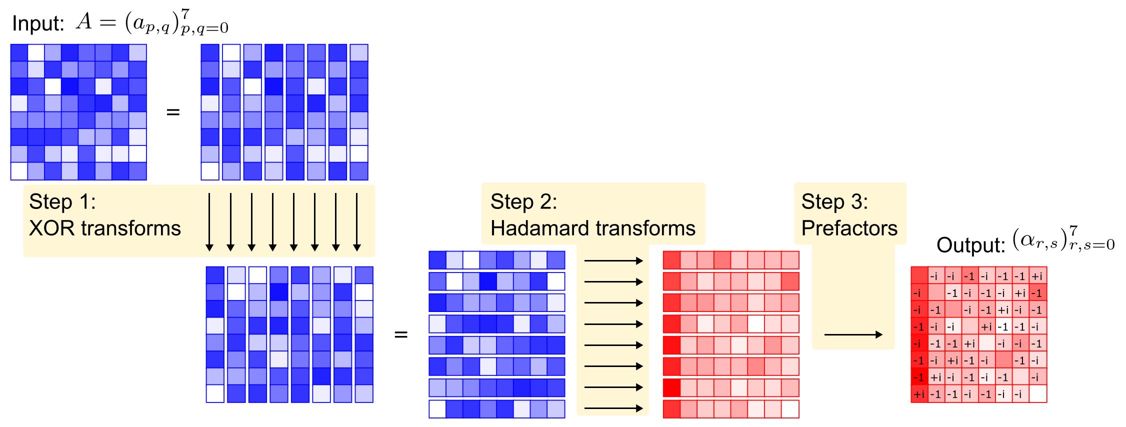
Input: $A=(a_{p,q})_{p,q=0}^{7}$
=
Step 1: XOR transforms
=
Step 2: Hadamard transforms
Step 3: Prefactors
Output: $(\alpha_{r,s})_{r,s=0}^{7}$

Figure 1. Illustration of algorithm 1 and theorem 1 for an 8×8 matrix. In Step 1, the XOR transform is applied, which permutes the elements in each matrix column according to (36). Then, in Step 2, the Walsh–Hadamard transform is applied to each row according to (38). This can be done efficiently with the Fast Walsh–Hadamard algorithm [7]. Finally, in Step 3, the elements are multiplied by prefactors according to (39).

3. Algorithm and numerical results

In this section, we present our implementation of the Pauli decomposition and compare its efficiency to the latest (still unreleased) Qiskit implementation [17] of the TPD algorithm [14]. The following algorithm is also illustrated in figure 1.

Algorithm 1 (Pauli decomposition of $A\in\mathbb{C}^{2^n\times 2^n}$ via the Walsh–Hadamard transform)
Input:

- The matrix elements $(a_{p,q})_{p,q=0}^{N-1}$ of a matrix A of dimension $N\times N$ $(N=2^n)$.

Output:

- The coefficients of the Pauli decomposition $A=\sum_{r,s=0}^{2^n-1}\alpha_{r,s}P_{r,s}$ in equation (6). They are computed in-place, i.e. they are stored in $(a_{p,q})_{p,q=0}^{N-1}$.

Complexity:

- Runtime: $\mathcal{O}(N^2\log N)$
- Additional space: $\mathcal{O}(1)$

Algorithm:

1. XOR-Transform: Permute the elements of the matrix $A=(a_{p,q})_{p,q=0}^{N-1}$ according to

$$a_{r,q}\leftarrow a_{r\oplus q,q}\qquad \left(r,q\in[N-1]_0\right). \tag{36}$$

   This transformation can be done in place requiring only $2^{n-1}(2^n-1)$ SWAP operations.
2. Fast Walsh–Hadamard Transform: Apply the Walsh–Hadamard transform

$$a_{r,s}\leftarrow\sum_{q=0}^{N-1}a_{r,q}H_{q,s}^{\otimes n}\qquad \left(r,s\in[N-1]_0\right). \tag{37}$$

   This can be done in place using only $n2^{2n}$ addition and $n2^{2n}$ subtraction operations on complex numbers [7]. This is because the generalised Hadamard matrix

$$H^{\otimes n}=\prod_{j=0}^{n-1}H_j\text{ with }H_j:=I^{\otimes j}\otimes H\otimes I^{\otimes(n-j-1)}\qquad\left(j\in[n-1]_0\right), \tag{38}$$

   and H_j is highly sparse and structured: Each column of H_j has only 2 non-zero elements, and they take values in $\{\pm 1\}$. Other decompositions of generalised Hadamard matrices are possible [7, 20].

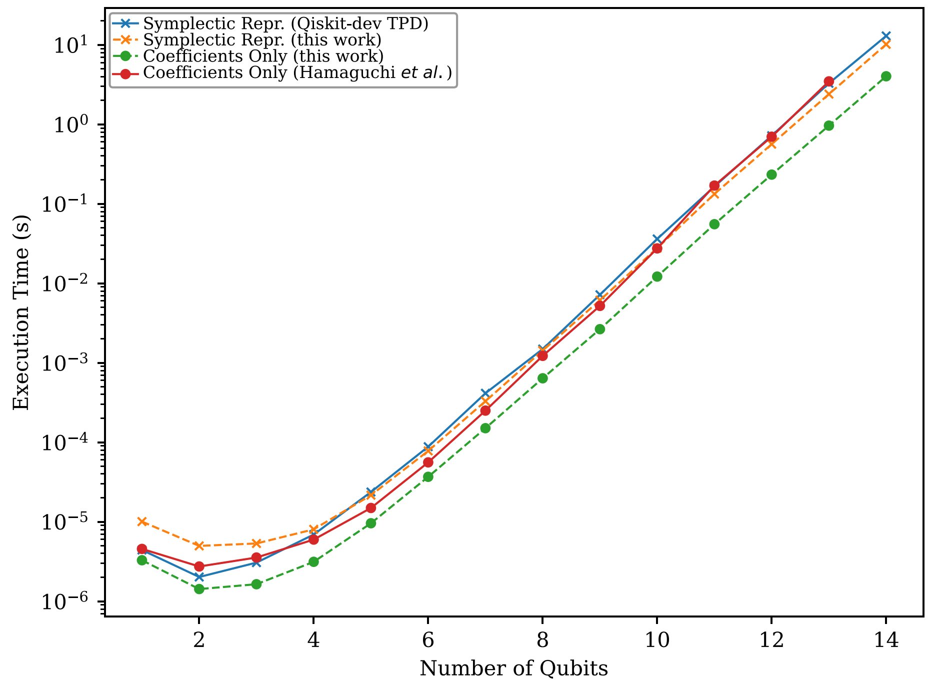

Figure 2. The execution time to perform a Pauli decomposition on random Hermitian matrices, as a function of the number of qubits. This calculation was performed on a single core AMD EPYC 7763 processor, 2.4GHz. Each data point was obtained by averaging over 10 runs for each of 10 different random matrices (100 runs contribute to each data point). The orange and green lines show this work, with and without calculation of a symplectic representation of Pauli strings, respectively. Qiskit-dev TPD refers to to the latest (still unreleased) Qiskit implementation [17] of the TPD algorithm [14]. Hamaguchi *et al* refers to the work [16, 26].

3. Prefactors: Multiply each element of A according to

$$a_{r,s} \leftarrow a_{r,s}\frac{(-i)^{|r\wedge s|}}{2^n} \qquad \left(r,s\in[N-1]_0\right). \tag{39}$$

   This can be done in place.

Reference implementation:

- A reference implementation in C with Python & Numpy bindings is freely available on Github [24] under the MIT license. It can also be installed with `pip install pauli_lcu`.

The execution time of this algorithm for random Hermitian matrices as a function of n is presented in figure 2(green line). We compare our implementation to Qiskit's function `decompose_dense` from `qiskit._accelerate.sparse_pauli_op` in the yet unpublished version [17] (blue line). Qiskit includes calculation of the symplectic representation [25] of the Pauli strings along with the coefficients; to make a fair comparison, we also show our algorithm including this calculation (orange line), and time our Python bindings rather than C code throughout. We observe that for more than 4 qubits, our implementation outperforms Qiskit and the maximum speed up we achieved is around a factor of ≈ 1.4. We also compare our results with the approach of Hamaguchi, Hamada and Yoshioka, which is based on the Fast Walsh–Hadamard transformation as well. We used their C++ implementation as available at the GitHub repository [26]. We observe that our algorithm provides a maximum speedup of a factor 3.6. For further details of our implementation we refer to our open-source implementation at [24]. Finally, we note that the reverse transform equation (9) can be achieved by effectively running algorithm 1 backwards.

To evaluate the performance of our algorithms further, we consider two more examples from chemistry and physics. One is the transformation of the electron repulsion integrals:

$$h_{IJ} = \int_{\mathbb{R}^3\times\mathbb{R}^3} \mathrm{d}\mathbf{r}'\mathrm{d}\mathbf{r}\,\frac{\rho_I^*(\mathbf{r},\mathbf{r}')\,\rho_J(\mathbf{r},\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}, \quad I,J\in\left[D^2-1\right]_0, \tag{40}$$

where D is the number of basis functions, and $\rho_I(\mathbf{r},\mathbf{r}')$ is a pair-density matrix. We use precomputed integrals for the Fe_2S_2 complex from [27]. Table 1 shows the execution time of our algorithms, the algorithm

Table 1. The execution time (in seconds) to perform a Pauli decomposition on electronic integrals in a molecular orbital basis set. This calculation was performed on a single core AMD EPYC 7763 processor, 2.4GHz. Qiskit-dev TPD refers to the Qiskit implementation [17] of the TPD algorithm [14]. Hamaguchi *et al* refers to the work [16, 26].

Number of basis functions	Coefficients only (this work)	Symplectic repr. (this work)	Symplectic repr. (Qiskit-dev TPD)	Coefficients only (Hamaguchi *et al*)
16	0.000 877	0.00 132	0.00 188	0.00 204
32	0.0103	0.0269	0.0271	0.0327
64	0.226	0.526	0.607	0.788
128	3.95	9.52	11.1	memory error

Table 2. The execution time (in seconds) to perform a Pauli decomposition on a kinetic energy matrix in real space. This calculation was performed on a single core AMD EPYC 7763 processor, 2.4GHz. Qiskit-dev TPD refers to the Qiskit implementation [17] of the TPD algorithm [14]. Hamaguchi *et al* refers to the work [16, 26].

Number of grid points	Coefficients only (this work)	Symplectic repr. (this work)	Symplectic repr. (Qiskit-dev TPD)	Coefficients only (Hamaguchi *et al*)
8	$2.95 \cdot 10^{-5}$	$2.13 \cdot 10^{-5}$	$10.2 \cdot 10^{-5}$	$5.62 \cdot 10^{-5}$
64	$6.23 \cdot 10^{-5}$	$8.35 \cdot 10^{-5}$	0.00 018	$8.5 \cdot 10^{-5}$
512	0.00227	0.00546	0.0113	0.007068
4096	0.221	0.528	0.94	0.79
32 768	17.2	41.3	81.6	memory error

of Hantzko *et al* [14] as implemented on Qiskit, and the algorithm of Hamaguchi *et al* [16, 26]. For the largest basis set size, we observe a speedup of 1.16 for the symplectic representation , and further factor of 2.4 (2.78 in total as compared to Qiskit-dev TPD) when one is interested in Pauli coefficients only.

Our second example is the transformation of the kinetic energy operator represented in real space using the dual plane-wave basis [28–30]. For a cubic cell with lattice constant of one and for uniformly distributed grid points, the kinetic energy matrix is

$$T'_{\mathbf{n},\mathbf{n}} = 2\pi^2 \sum_{m_1,m_2,m_3=-N^{1/3}/2}^{N^{1/3}/2-1} \left(m_1^2+m_2^2+m_3^2\right) \exp\left(\frac{2\pi i}{N^{1/3}} \sum_{j=1}^{3} m_j \left(n_j - n'_j\right)\right) \tag{41}$$

where N, the total number of grid points, is a power of 2^3; $n_1,n_2,n_3,n'_1,n'_2,n'_3 \in \left[N^{1/3}-1\right]_0$, and $\mathbf{n}=(n_1,n_2,n_3)$, $\mathbf{n}'=(n'_1,n'_2,n'_3)$. In addition to being symmetric, the kinetic energy matrix also contains translational symmetry. We provide the python script for generating such a matrix in [31]. Table 2 shows the execution time of different algorithms. For the largest calculations, where the total number of grid points is $N=2^{15}$, we observe a factor of 1.65 speedup for the symplectic representation as compared to Qiskit-dev TPD, and a further factor of 2.5 when one is interested in Pauli coefficients only (a factor of 4.12 speed up in total as compared to Qiskit-dev TPD).

As can be seen from figure 1, our algorithm allows for straightforward parallelization. First, the XOR transformation (Step 1 in figure 1) is applied to each column, then Hadamard transformation and phase factor multiplication are applied to each row independently (Steps 2–3 in figure 1). We have implemented this parallelization with OpenMP and tested it on the largest example from table 2 ($2^{15}\times 2^{15}$ matrix). As is shown on figure 3, we observe nearly ideal speedup for 2 and 4 cores. For 8 cores, the wall time is reduced by a factor of 7 and the Pauli coefficients can be calculated in around 2.5 s as compared to 17.2 s on a single AMD EPYC 7763 core.

4. Conclusion

In this work, we have presented an exact and explicit approach to the Pauli decomposition of matrices $A\in\mathbb{C}^{2^n\times 2^n}$. Practically, our theoretical result inspires an efficient algorithm for the computation of the Pauli string coefficients. We develop this algorithm and demonstrate that it outperforms the previous leading solutions in the literature. We emphasize that, as we employ the fast Walsh–Hadamard transform, our algorithm achieves favorable performance versus existing solutions without sophisticated optimisation techniques.

Our results have immediate applications in quantum chemistry. Pauli decompositions of Hamiltonians in second quantization are available through fermion-to-qubit mappings such as that of Jordan–Wigner [4, 32]. However, they are not applicable to Hamiltonians in first quantization. Our theorem applied to Hermitian matrices and their higher-dimensional tensorial extensions allows one to derive the

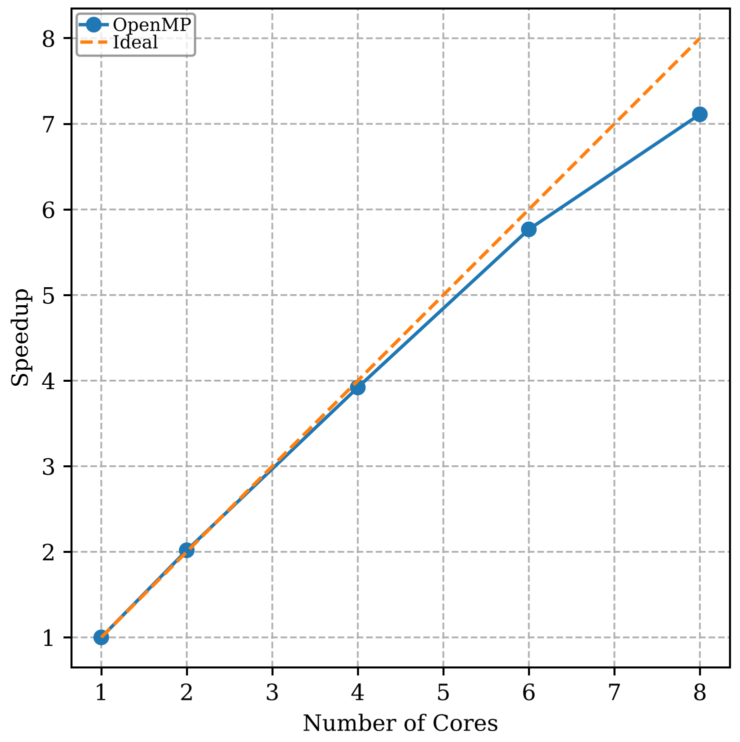

Figure 3. Speedup of Pauli decomposition due to parallelization using OpenMP. The same $2^{15} \times 2^{15}$ kinetic energy matrix as in table 2 was used as an example. This calculation was performed on AMD EPYC 7763 processors, 2.4GHz, and 386 GB RAM.

explicit Pauli decomposition of first quantization Hamiltonians for the study of chemical systems on quantum computers [33]. More generally, we expect that our results will be useful in the block-encoding of matrices. Finally, our solution is available as an open source package at [24].

5. Supplementary information

Scripts used to produce results for this paper are available at Zenodo [31]. Our code is further available on Github [24] and with `pip install pauli_lcu`.

Data availability statement

The data that support the findings of this study are openly available at reference [31].

Acknowledgments

We thank Marius Bothe, Róbert Izsák, and Matt Ord for collaboration on closely related topics. AVI also thanks Mark Turner for helpful discussion about parallelisation. The work presented in this paper was partially funded by a grant from Innovate UK under the 'Commercialising Quantum Technologies: CRD & Tech round 2' competition (Project Number 10004793).

ORCID iDs

Timothy N Georges https://orcid.org/0000-0003-4458-6819
Bjorn K Berntson https://orcid.org/0000-0001-8039-7949
Christoph Sünderhauf https://orcid.org/0009-0003-8455-8465
Aleksei V Ivanov https://orcid.org/0000-0001-7403-3508

References

[1] Nielsen M A and Chuang I L 2001 *Quantum Computation and Quantum Information* vol 2 (Cambridge University Press)
[2] Abrams D S and Lloyd S 1997 Simulation of many-body fermi systems on a universal quantum computer *Phys. Rev. Lett.* **79** 2586
[3] Abrams D S and Lloyd S 1999 Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors *Phys. Rev. Lett.* **83** 5162

[4] Ortiz G, Gubernatis J E, Knill E and Laflamme R 2001 Quantum algorithms for fermionic simulations *Phys. Rev.* A **64** 022319
[5] Aspuru-Guzik A Dutoi A D, Love P J and Head-Gordon M 2005 Simulated quantum computation of molecular energies *Science* **309** 1704
[6] Harrow A W, Hassidim A and Lloyd S 2009 Quantum algorithm for solving linear systems of equations *Phys. Rev. Lett.* **103** 150502 (arXiv:0811.3171)
[7] Yarlagadda R K and Hershey J E 1997 *Hadamard Matrix Analysis and Synthesis: With Applications to Communications and Signal/Image Processing* (Springer)
[8] Romero S V and Santos-Suárez J 2023 Paulicomposer: compute tensor products of pauli matrices efficiently *Quantum Inf. Process.* **22** 449 (arXiv:2301.00560)
[9] Jones T 2024 Decomposing dense matrices into dense Pauli tensors (arXiv:2401.16378)
[10] Koska O, Baboulin M and Gazda A 2024 A tree-approach pauli decomposition algorithm with application to quantum computing (arXiv:2403.11644)
[11] Ying J Shen J, Zhou L, Zhong W, Du M and Sheng Y 2023 Preparing a fast Pauli decomposition for variational quantum solving linear equations *Ann. Phys., Lpz.* **535** 2300212
[12] Lapworth L 2022 A hybrid quantum-classical cfd methodology with benchmark hhl solutions (arXiv:2206.00419)
[13] Gunlycke D, Palenik M C, Fischer S A and Emmert A R 2020 Efficient algorithm for generating Pauli coordinates for an arbitrary linear operator (arXiv:2011.08942)
[14] Hantzko L Binkowski L and Gupta S 2024 Tensorized Pauli decomposition algorithm *Phys. Scr.* **99** 085128
[15] Gidney C 2023 How to write the iSWAP unitary as a linear combination of tensor products between 1-qubit gates?, StackOverflow (available at: https://quantumcomputing.stackexchange.com/a/31790)
[16] Hamaguchi H, Hamada K and Yoshioka N 2024 Handbook for efficiently quantifying robustness of magic (arXiv:2311.01362)
[17] Lishman J 2024 Qiskit implementation of tensorized Pauli decomposition (available at: https://github.com/Qiskit/qiskit/pull/11557)
[18] Javadi-Abhari A *et al* 2024 Quantum computing with qiskit (arXiv:2405.08810)
[19] Bergholm V *et al* 2022 PennyLane: automatic differentiation of hybrid quantum-classical computations (arXiv:1811.04968)
[20] Fino B J and Algazi V R 1976 Unified matrix treatment of the fast walsh-hadamard transform *IEEE Trans. Comput.* C **25** 1142
[21] Flammia S T and Wallman J J 2020 Efficient estimation of Pauli channels *ACM Trans. Quantum Comput.* **1** 1 (arXiv:1907.12976 [quant-ph])
[22] Harper R, Flammia S T and Wallman J J 2020 Efficient learning of quantum noise *Nat. Phys.* **16** 1184 (arXiv:1907.13022 [quant-ph])
[23] Harper R, Yu W and Flammia S T 2021 Fast estimation of sparse quantum noise *PRX Quantum* **2** 010322 (arXiv:2007.07901 [quant-ph])
[24] Ivanov A V and Sünderhauf C 2024 Pauli decomposition via the fast Walsh-Hadamard transform (available at: https://github.com/riverlane/pauli_lcu)
[25] Aaronson S and Gottesman D 2004 Improved simulation of stabilizer circuits *Phys. Rev.* A **70** 052328
[26] Hamaguchi H, Hamada K and Yoshioka N 2024 paulidecomp (available at: https://github.com/quantum-programming/paulidecomp)
[27] Georges T, Bothe M, Sunderhauf C, Berntson B, Izsak R and Ivanov A V 2025 Integrals for "Quantum simulations of chemistry in first quantization with any basis set" (https://doi.org/10.5281/zenodo.14906508)
[28] Skylaris C-K, Mostofi A A, Haynes P D, Diéguez O and Payne M C 2002 Nonorthogonal generalized wannier function pseudopotential plane-wave method *Phys. Rev.* B **66** 035119
[29] Mostofi A A, Skylaris C-K, Haynes P D and Payne M C 2002 Total-energy calculations on a real space grid with localized functions and a plane-wave basis *Comput. Phys. Commun.* **147** 788
[30] Babbush R, Wiebe N, McClean J McClain J, Neven H and Chan G K-L 2018 Low-depth quantum simulation of materials *Phys. Rev.* X **8** 011044
[31] Georges T N, Berntson B K, Sünderhauf C and Ivanov A V 2025 Pauli decomposition via the fast walsh-hadamard transform v3 (https://doi.org/10.5281/zenodo.14905815)
[32] Jordan P and Wigner E 1928 Über das paulische äquivalenzverbot *Z. Phys.* **47** 631
[33] Georges T N, Bothe M, Sünderhauf C, Berntson B K, Izsák R and Ivanov A V 2024 Quantum simulations of chemistry in first quantization with any basis set (arXiv:2408.03145)