arXiv:2601.02233v1 [quant-ph] 5 Jan 2026

PauliEngine：面向量子操作的高性能符号运算

Leon M¨uller,[1] Adelina B¨arligea,[1] Alexander Knapp,[1] 和 Jakob S. Kottmann[1, 2, *]

[1] *Institute for Computer Science, University of Augsburg, Germany*

[2] *Center for Advanced Analytics and Predictive Sciences, University of Augsburg, Germany*（日期：2026年1月6日）

量子计算本质上是混合式的，而要确保量子软件的可扩展性，就必须快速地对量子比特算符进行经典操作。我们推出了 PauliEngine，这是一款高性能 C++ 框架，提供高效的原语，用于泡利串乘法、对易子、符号相位跟踪和结构变换。PauliEngine 基于二元辛表示构建，并针对位运算进行了优化，既支持数值系数，也支持符号系数，还可通过 Python 接口使用。运行时间基准测试表明，与当前最先进的实现相比，PauliEngine 的速度显著提升。PauliEngine 为基于算符的量子软件工具和模拟提供了可扩展的后端。

一、引言

经典计算是量子算法学的核心组成部分，而且今后仍将如此。一个显而易见的方面是量子计算机的模拟 [1], ，这既有助于理解量子算法，也有助于验证近期硬件；但另一个常被忽视的方面，是负责构建量子计算协议经典描述的经典框架——例如，以量子门和测量的形式呈现的描述。在模拟领域，近年来人们重新关注可直接在泡利基上运行的模拟方法，例如泡利传播方法 [2——4] 或其他模拟器 [5]。然而，这些方法的有效性在很大程度上取决于能否获得高性能的基本操作，以便大规模处理泡利串：以尽可能低的运行时间和内存开销，对数百万个泡利串进行乘法、求对易子和变换。

本文介绍 PauliEngine，这是一款精简的 C++后端，旨在为量子软件工具提供快速且节省内存的泡利串算术。PauliEngine 实现了二进制辛表示，并针对乘法、对易子计算与判定以及符号相位跟踪，优化了位级运算。该框架支持数值系数和符号系数（通过 SymEngine [6]），并提供轻量级 Python 接口，便于集成到量子软件工作流中。

II. 预备知识与符号

泡利矩阵构成了描述量子计算的基本算符基。除单位算子 I外，单量子比特矩阵为

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

一个 N 量子比特泡利串是单量子比特泡利算符的张量积，

$$P_{\mathbf{k}} = \bigotimes_{i=1}^{N} \sigma_{k_i}, \tag{2}$$

其中，多重索引$\mathbf{k} = (k_1, k_2, k_3, \ldots, k_N)$ 的每个取值均属于 $k_i \in \{0, 1, 2, 3\}$，分别对应四个矩阵 I, X, Y, Z。下文中，若含义明确，我们通常省略粗体$\mathbf{k}$ ，并在方便时直接写作 $k_i \in \{I, X, Y, Z\}$。

加权泡利串是一个由复系数 $c \in \mathbb{C}$ 和 c, P 组成的有序对。加权泡利串的线性组合足以表示量子计算中涉及的所有算子，尤其是厄米算符和酉算符。

厄米算符 加权泡利串的线性组合，

$$H = \sum_{k=1}^{M} c_k P_k \tag{3}$$

是厄米的，当且仅当所有系数 c_k 都是实数。同理，如果系数均为纯虚数，该算子就是反厄米的。由于 N 个量子比特的泡利算子构成了 $\mathfrak{su}(2^N)$ 的标准正交基，因此每个 N 量子比特可观测量以及每个量子演化的生成元都可以写成这样的线性组合。

酉算符 An 个量子比特量子操作由一个酉 $2^N \times 2^N$ 矩阵表示。任意酉$U \in U(2^N)$都可以通过其生成元 G写成指数形式

$$U = e^{iG}, \tag{4}$$

* 电子邮箱：jakob.kottmann@uni-a.de

其中 G 是厄米算子，并可在泡利基下展开。然而在实践中，使用由单个泡利串生成的基本量子门更为方便。

对于加权字符串 $(\theta,\ P)$，其对应的泡利旋转为

$$U_P(\theta) = e^{-i\frac{\theta}{2}P} = \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\right) P \tag{5}$$

其中因子 $_{1\ 2}$ 遵循量子信息论中的一种常见约定。

一般的酉算符可以表示为这类量子门的乘积

$$U = \prod_k U_{P_k}(\theta_k), \tag{6}$$

即，量子电路。泡利旋转构成一组通用量子门集，这一点可通过对式（4）进行 Trotter 分解，或通过更专门的分解得到验证。常见量子门对应的泡利生成元列于式（1）中。此外，给定运算 U 的生成元 G，可通过将 G 替换为 $_{1\,2}(1 - Z_m)\,G$，在量子比特 m 上实现受控版本。

基本算术规则 上述数据类型的算术运算直接遵循泡利矩阵的代数性质。

在单量子比特层面，

$$\sigma_k\sigma_l = i\epsilon_{klm}\sigma_m, \tag{7}$$

与 ϵ 列维-奇维塔张量。在特别情况下，$XY = iZ$和 $XZ = iY$ 以及泡利矩阵只要彼此不同且均非恒等矩阵，就会反对易。

基本算术运算对各种数据类型的影响总结见表 1

我们在此通过给定的运算展示了各种数据类型之间的转换。其中，闭合表示数据类型保持不变，而不变表示数据类型及其内容均保持不变。

III. 快速算术

A. 泡利字符串的高效数据结构 ngs

泡利算符在符号计算中有几种便捷的数据结构。若泡利串 P以式（2）的形式给出，那么任何由这类泡利串线性组合而成的量子比特算符（如式（3）所示），都可以自然地表示为一个字典，将多重索引$\mathbf{k}$ 映射到相应的复系数。例如，`QubitOperator` 类在`OpenFermion`中采用的就是这种原生表示。

另一种在各种实现中广泛采用的格式 [7–12] 是二进制辛表示法。在这种表示中，N量子比特泡利串 $P_{\mathbf{k}}$ 由两个 N比特向量 $\mathbf{x}$ 和 $\mathbf{y}$编码，

$$P_{\mathbf{k}} \to (\mathbf{x},\mathbf{y}) = (x_1x_2x_3\ldots x_N, y_1, y_2, \ldots, y_N), \tag{8}$$

其中，x_i 表示量子比特 i上存在 X或 Z算子，而 y_i 则类似地跟踪 Y或 Z算子。具体而言，

$$x_i = \begin{cases} 1 & \text{if } k_i \in \{X, Z\} \\ 0 & \text{else} \end{cases} \qquad y_i = \begin{cases} 1 & \text{if } k_i \in \{Y, Z\} \\ 0 & \text{else} \end{cases} \tag{9}$$

该映射是可逆的：

$$k_i = \begin{cases} I & \text{if } (x_i, y_i) = (0,0) \\ X & \text{if } (x_i, y_i) = (1,0) \\ Y & \text{if } (x_i, y_i) = (0,1) \\ Z & \text{if } (x_i, y_i) = (1,1). \end{cases} \tag{10}$$

例如：

$$X(2)\,Y(3)\,X(5)\,Z(7)\,Z(8) \to 01001011|00100011 \tag{11}$$
$$Z(3)\,X(4)\,Z(5) \to 00111|00101 \tag{12}$$
$$Z(2)\,Y(5)\,X(6) \to 0100001|010010, \tag{13}$$

其中，我们使用“|”在视觉上将$\mathbf{x}$与$\mathbf{y}$分隔开。

B. 按位乘法与相位重构

辛编码的一个关键优势是，泡利串的乘法可转化为一种开销很低的按位异或运算，以下记作 $\oplus$。对于两个字符串

$$P = (\mathbf{x},\mathbf{y}), \quad P' = (\mathbf{x}',\mathbf{y}'), \tag{14}$$

它们的乘积是另一个泡利串 $P'' = (\mathbf{x}'',\ \mathbf{y}'')$，其中

$$\mathbf{x}'' = \mathbf{x} \oplus \mathbf{x}', \quad \mathbf{y}'' = \mathbf{y} \oplus \mathbf{y}'. \tag{15}$$

这与单量子比特泡利矩阵的乘法表完全一致，除了复相位 $\pm i$ 必须单独处理：

$*$	I	X	Y	Z
1	1	X	Y	Z
X	X	1	iZ	$-iY$
Y	Y	$-iZ$	I	iX
Z	Z	iY	$-iX$	1

(16)

运算	P	$(c,\ P)$	厄米	酉
乘法	$(c,\ P)$	闭合	算子	闭合
数乘	$(c,\ P)$	闭合	算子*	算子**
加法	算子	闭合	闭合	算子
共轭	不变	闭合	不变	闭合
酉变换			闭合	闭合
厄米				
厄米				

* 对实标量闭合 , ** 对单位根闭合

图 1. 数据类型和运算概览。

$\oplus$	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
$(0,0)$	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
$(1,0)$	$(1,0)$	$(0,0)$	$(1,1)$	$(0,1)$
$(0,1)$	$(0,1)$	$(1,1)$	$(0,0)$	$(1,0)$
$(1,1)$	$(1,1)$	$(0,1)$	$(1,0)$	$(0,0)$

(17)

对于多量子比特泡利串，同样的逻辑适用于每个量子比特。例如，

$$\begin{aligned} XYZXYZ \quad &\cdot \quad YZXZXY \quad = \quad ZXYYZX \\ 101101\ 011011 \quad &\cdot \quad 011110\ 110101 \quad = \quad 110011\ 101110. \end{aligned}$$

然而，单靠基于异或的乘法无法追踪全局相位，而全局相位源于单量子比特泡利算符的非对易性。为了通过快速位运算恢复这一相位，我们预先计算真值表，以标示局部乘积何时会产生因子$+i$或 $-i$ （见图2）。

根据这些表，我们得到比特串 F_+ 和 F_-的布尔表达式：

$$\begin{aligned} F_+ =&(\neg\mathbf{x}\wedge\mathbf{x}'\wedge\mathbf{y}\wedge\mathbf{y}') \\ &\oplus\ (\mathbf{x}\wedge\neg\mathbf{x}'\wedge\neg\mathbf{y}\wedge\mathbf{y}')\ \oplus\ (\mathbf{x}\wedge\mathbf{x}'\wedge\mathbf{y}\wedge\neg\mathbf{y}'), \\ F_- =&(\neg\mathbf{x}\wedge\mathbf{x}'\wedge\mathbf{y}\wedge\neg\mathbf{y}') \\ &\oplus\ (\mathbf{x}\wedge\neg\mathbf{x}'\wedge\mathbf{y}\wedge\mathbf{y}')\ \oplus\ (\mathbf{x}\wedge\mathbf{x}'\wedge\neg\mathbf{y}\wedge\mathbf{y}'), \end{aligned} \tag{18}$$

其中， $|F_\pm|$ 表示汉明重量（置位比特的数量）。乘法的总相位因子为

$$c = i^{(|F_+|-|F_-|) \mod 4}. \tag{19}$$

这个新公式只涉及一次减法和两次置位计数运算。结合前面给出的异或规则，它能够仅使用位运算快速计算任意泡利串的乘积。

C. 符号扩展与可参数化结构

该框架的一项扩展是支持泡利串的符号系数。系数不再仅限于复数，也可以使用变量。

这使得符号微分、参数化算符的解析运算以及参数值的延迟替换成为可能。为此，我们集成了 SymEngine [6]，这是一个用于高性能符号运算的库。

IV. 初步应用

a. 快速对易子：量子力学中，两个算子 A 和 B 之间的对易子 $[A,B]$ $=AB-BA$ 是一个基本概念，它决定了两个可观测量能否同时测量。原则上，可以通过直接计算两个乘积 AB 和 BA来求得对易子。然而，在泡利基中，无需显式进行这些计算；相反，我们可以根据两个泡利串的二进制辛表示，直接推断它们之间的对易关系。

对于两个泡利串 $P=(\mathbf{x},\ \mathbf{y})$ 和 $P'=(\mathbf{x}',\ \mathbf{y}')$，当它们在某个量子比特上都非平凡作用或彼此反对易时，其乘积会产生相位因子 $\pm i$。利用由式18导出的位掩码，乘法例程会统计每个量子比特上乘积产生 $+i$ 和 $-i$因子的次数。令 $F_+(P,\ P')$ 和 $F_-(P,\ P')$ 分别表示这些计数。它们的差值$\tau = F_+ - F_-$ 决定了乘积产生的全局相位。
我们的快速对易子实现只需进行一次乘法，即可判断对易子是否为零：

- If τ is even, then $PP' = P'P$, and thus $[P,P'] = 0$.
- 如果 τ是奇数，则 $PP' = -P'P$，且 $[P,P'] = 2iPP'$，相差的仅是逐位乘法累积产生的相位。因此无需进行第二次乘法。

从算法角度看，该例程仅需执行一次异或来确定所得的泡利串 PP'，执行两次位求和（总体计数）以计算 F_+ 和F_-，最后对 τ 执行模 2 奇偶性检查。这样，对于 N量子比特泡利串，即可完成 $\mathcal{O}(N)$ 对易子计算，而无需显式构造乘积 PP'和 $P'P$。

	x	y	x'	y'	i
II	0	0	0	0	
IY	0	0	1	0	
IX	0	1	0	1	
IZ	0	1	1	0	
YI	0	1	0	0	0
YY	0	1	0	1	0
YX	0	1	1	0	0
YZ	0	1	1	1	1
XI	1	0	0	0	0
XY	1	0	0	1	1
XX	1	0	1	0	0
XZ	1	0	1	1	0
ZI	1	1	0	0	0
ZY	1	1	0	1	0
ZX	1	1	1	0	1
ZZ	1	1	1	1	0

	x	y	x'	y'	$-i$
II	0	0	0	0	0
IY	0	0	1	0	0
IX	0	1	0	1	0
IZ	0	1	1	0	0
YI	0	1	0	0	0
YY	0	1	0	1	0
YX	0	1	1	0	1
YZ	0	1	1	1	0
XI	1	0	0	0	0
XY	1	0	0	1	0
XX	1	0	1	0	0
XZ	1	0	1	1	1
ZI	1	1	0	0	0
ZY	1	1	0	1	1
ZX	1	1	1	0	0
ZZ	1	1	1	1	0

图 2. 用于确定二进制辛形式下两个单量子比特泡利算符相乘时产生的相位因子的系数确定表。每一行对应一对局部泡利算符，其比特表示分别为 $(x,\ y)$ 和 $(x',\ y')$。左表标示产生因子 $+i$ 的情况，右表则标示产生因子 $-i$ 的情况。

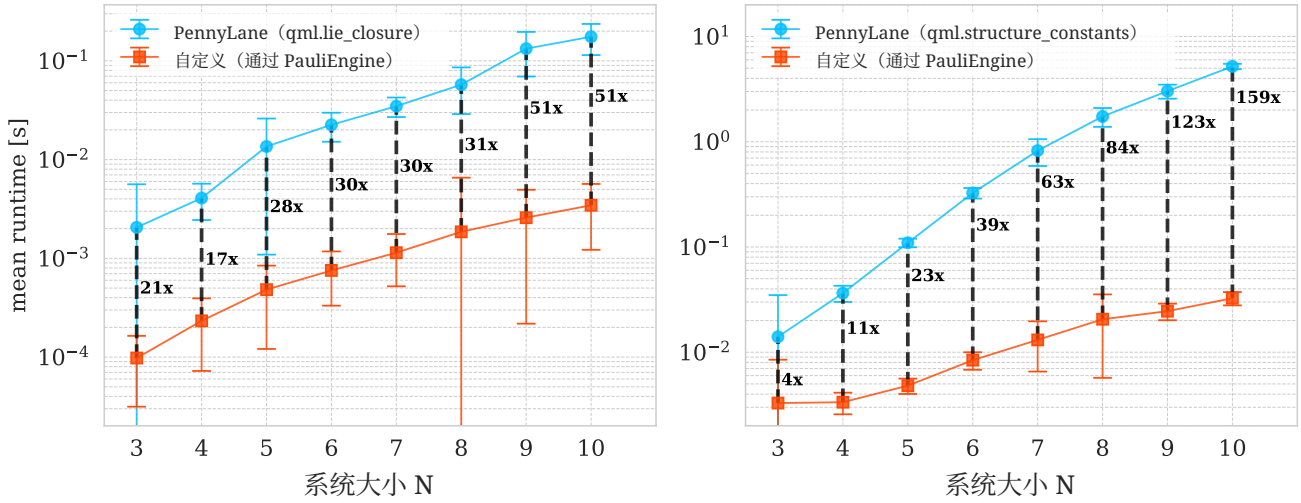

图 3. 使用 PauliEngine 算术与 PennyLane 进行动力李代数计算的运行时间基准测试。左：计算与 $\mathfrak{so}(2N)$ 同构的动力李代数的李闭包，运行 1000 次的平均时间。右：计算相应结构常数，运行 100 次的平均时间。两个坐标轴均为对数刻度。对于每个 N，虚线垂直标记表示加速倍数（PauliEngine 相对于 PennyLane）。所有基准测试均在配备 24 GB 内存的 Apple M4 处理器上进行。

下面，我们将讨论这种快速的对易子计算与检查能够显著缩短运行时间的场景。动力学李代数：近期研究揭示了变分量子算法（VQA）中的贫瘠高原（损失和梯度方差呈指数级消失）与 [13] 拟设所生成的动力学李代数（DLA）的维数 [14–16]之间存在紧密联系。此后，动力学李代数已成为刻画参数化量子电路的表达能力和可训练性的核心工具。此外，

一般结果表明，可证明不存在贫瘠高原的 ans¨atze 往往可以进行高效的经典模拟 [17]。据认为，这是因为相关的动力李代数（DLA）仅随量子比特数量呈多项式地增长，因此仍易于操作和模拟 [5]。

VQA 由参数化量子电路定义，

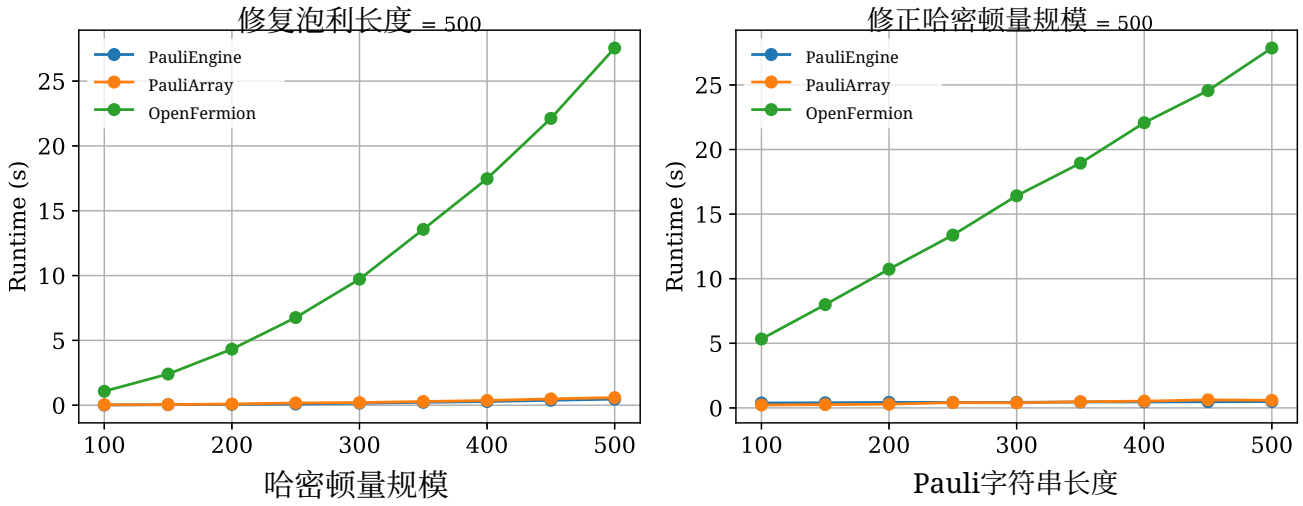

图4. 使用 PauliEngine、PauliArray 和 OpenFermion 进行哈密顿量乘法的运行时间对比。左：固定泡利字符串长度（500）时，运行时间随哈密顿量规模的变化。右：固定哈密顿量规模（500）时，运行时间随泡利字符串长度的变化。在所有测试范围内，PauliEngine 和 PauliArray 的性能均明显优于 OpenFermion；对于中等规模，OpenFermion 就已成为瓶颈。测试平台为 Intel i9-11900KF，配备 32GB 内存。

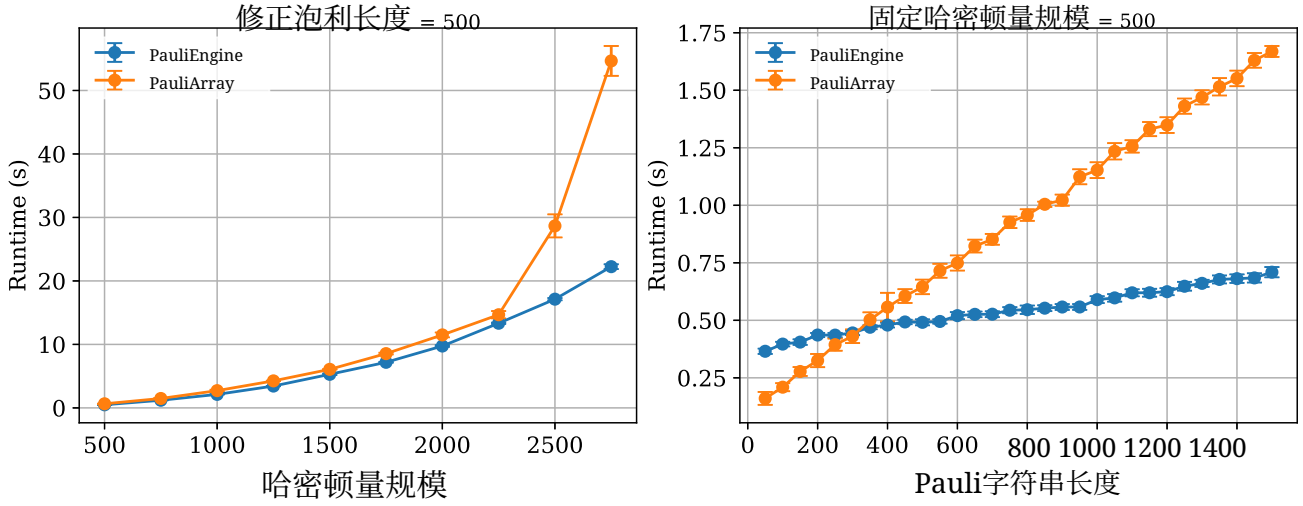

图5. PauliEngine 与 PauliArray 的直接对比。左：固定泡利字符串长度（500）时，10 次运行的平均运行时间随哈密顿量规模的变化。右：固定哈密顿量规模（500）时，100 次运行的平均运行时间随泡利字符串长度的变化。对于较小的实例，PauliArray 速度更快；但随着内存占用显著增加，其运行时间会急剧增长，而 PauliEngine 则保持稳定，其运行时间随哈密顿量规模呈二次增长，随泡利字符串长度近似线性增长。测试平台为 Intel i9-11900KF，配备 32GB 内存。

它通常可以写成这种逐层形式，

$$U(\theta) = \prod_{l=1}^{L} \exp\left(-\mathrm{i}\theta_l H_l\right), \tag{20}$$

作用于初始态 ρ_{in} 并使用可观测量 O进行测量。拟设的许多性质完全取决于其生成元集 $\mathcal{G} = \{H_l\}_{l=1}^{L}$。相应的动力李代数是通过李闭包得到的，

$$\mathfrak{g} = \langle \mathrm{i}\mathcal{G} \rangle_{\mathrm{Lie}} = \{h_k\}_{l=1}^{\dim(\mathfrak{g})} \subseteq \mathfrak{su}(2^N), \tag{21}$$

即包含所有生成元以及它们之间所有（非零）嵌套对易子的最小李代数。$\dim(\mathfrak{g})$ 随系统大小的变化规律与变分量子算法损失函数方差 [15],的变化规律直接相关，这促使人们开发高效方法，显式计算具体 ans¨atz 的动力李代数。

只要 $\dim(\mathfrak{g})$ 呈多项式增长，就可以通过动力李代数的伴随表示高效模拟期望值 [5]。此类模拟器完全在 $\mathfrak{g}$的线性张成空间内演化期望值，运行时间关于动力李代数维数呈多项式关系。主要的计算瓶颈在于构造结构常数，

$$[h_\alpha, h_\beta] = \sum_\gamma f^\gamma_{\alpha\beta} h_\gamma, \quad f^\gamma_{\alpha\beta} = \frac{\langle h_\gamma, [h_\alpha, h_\beta]\rangle}{\langle h_\gamma, h_\gamma\rangle}, \tag{22}$$

在希尔伯特–施密特内积 $\langle A, B\rangle = \mathrm{tr}\left[A^\dagger B\right]$ 下。由此得到的张量 f 定义了完整的伴随表示。

对于许多物理相关的 ans¨atze，动力李代数基中的每个 h_k 本身都是单个泡利串。在这种情况下，稠密矩阵方法会造成不必要的高昂开销：对每一对基元素进行矩阵级对易子计算，都需要 $\mathcal{O}((2^N)^3)$ 次运算；而确定基闭合则需要对 $2^N \times 2^N$ 个矩阵反复进行线性无关性检验。使用泡利表示可以避免这一指数级瓶颈。借助上一节介绍的技术，整个过程变为纯符号运算，速度也能显著提升。李闭包和结构常数的计算都可归结为：（i）通过位运算快速计算泡利串之间的对易子；（ii）通过常数时间的字典查找，判断得到的泡利串是否已在基中。整个过程无需大型矩阵或稠密线性代数。

为实证展示性能优势，我们使用 `PauliEngine`，将 DLA 和伴随表示例程与最先进的`PennyLane` 实现进行基准测试 [18](另请参阅博客文章 [19]) 。图3 展示了与 $\mathfrak{so}(2N)$ 同构的动力李代数的平均运行时间。在所有测试的系统大小下，我们基于`PauliEngine`编写的自定义函数都将运行时间缩短了一个到两个数量级。对于结构常数的计算，这一提升尤为显著：加速比从 $N = 3$ 处的大约 4 倍增长到 $N = 10$处（右图）的近 160 倍。这些提升将使基于动力李代数的分析与模拟能够应用于以往无法处理的更大系统。

对易团：变分量子算法中的一个主要计算瓶颈，是确定所关注期望值所需的单独测量次数。在电子结构应用中，这需要 $O(N^4)$ 次独立运行才能确定一个能量值。 [20, 21]。尽管近期出现了启发式方法 [22] 和随机化方法， [23, 24],占主导地位的误差缓解方法仍是将哈密顿量分解为对易团 [25——28] 快速算术运算可加速相关算法，而快速对易子则可快速验证检测到的团。

生成元梯度：许多量子算法应用都涉及参数化期望值

$$\langle\psi(\theta)|H|\psi(\theta)\rangle = \langle 0|U^\dagger(\theta) H U(\theta)|0\rangle \equiv \langle H\rangle_{U(\theta)}, \tag{23}$$

其中，参数化量子态 $\psi(\theta)$ 通过量子电路 $U(\theta)$ 生成。在自适应量子线路构造中，会根据其梯度筛选新的量子门 $V(\varphi)$

$$\frac{\partial}{\partial\varphi}\langle H\rangle_{U(\theta)+V(\varphi)} \tag{24}$$

梯度既可以通过参数移位规则 [29] 计算；当 $\varphi = 0$ 时，也可以通过算子 H 与生成元 G 或量子门 V 的对易子计算

$$\frac{\partial}{\partial\varphi}\langle H\rangle_{U(\theta)+V(\varphi)} = \frac{1}{4}\langle i\,[H, G]\rangle_{U(\theta)}. \tag{25}$$

如果要计算其他点处的梯度，且关注的量子门不是量子电路末尾的量子门，则可以采用类似方法来加快经典计算。 [10, 30]此时，还需要考虑算符折叠技术（见下一节）。

b. 算符折叠：当能够快速进行泡利算术时，一种在实践中效果显著的技术是算符折叠，通常用于计算期望值

$$\begin{aligned}\langle H\rangle_U &= \langle 0|U^\dagger H U|0\rangle = \langle 0|U_1^\dagger U_2^\dagger U_3^\dagger U_4^\dagger H U_4 U_3 U_2 U_1|0\rangle \\ &= \langle 0|U_1^\dagger U_2^\dagger \left(U_3^\dagger U_4^\dagger H U_4 U_3\right) U_2 U_1|0\rangle = \langle\tilde{H}\rangle_{\tilde{U}}\end{aligned} \tag{26}$$

在此示例中 $\tilde{H} = U_3^\dagger U_4^\dagger H U_4 U_3$和 $\tilde{U} = U_2 U_1$。通常，尤其是在变分设置 [31–33], 中，各个量子门都带有参数，因此我们得到一个˜参数化变换 $H \to H(\theta)$。能够快速执行这些运算，往往有助于分析和测试。例如，各种海森堡绘景技术 [34–36], ，以及带有克利福德尾部的量子电路；对于期望值计算，这类量子电路可以折叠进算子中，而不改变泡利串的数量 [37–39]。此外，还有高度专门化的方法，例如电子结构领域中的转关联哈密顿量 [40–42]。在非对称折叠场景中也有此类应用，例如 在跃迁振幅计算中 $\langle\psi|H|\phi\rangle$ ，不同状态由不同的量子电路构造。实践中的典型例子包括非正交变分量子本征求解器 [43–45] 以及降阶技术 [46], ；在这些场景中，部分折叠可作为分析、数值基准测试或算法组件的有用工具。

V. 基准测试

使用 PauliEngine 框架进行符号泡利算术的一项自然压力测试，是两个哈密顿量的乘法，

$$H_1 = \sum_i (c_i, P_i), \tag{27}$$

$$H_2 = \sum_i (d_j, Q_j), \tag{28}$$

$$H_1 H_2 = \sum_i \sum_j (c_i d_j)(P_i Q_j), \tag{29}$$

其中涉及计算所有泡利串的两两乘积并累加其系数。这项任务直接检验底层数据结构、泡利乘法例程以及内存使用的效率。

我们将 PauliEngine 的实现与事实上的社区标准 OpenFermion [47]，以及近期开发的PauliArray [9]，进行基准比较，后者采用高度优化的位并行方法。对于每个基准实例，我们通过对固定长度的泡利串进行采样来生成随机哈密顿量，其中局部泡利算符均匀随机（包括恒等算符）。我们考察了两项互补的基准测试：

- 随哈密顿量规模（即泡利项数量）变化的扩展性，同时将泡利串长度固定为 500。
- 随泡利串长度变化的扩展性，同时将哈密顿量规模固定为 500 项。

所有计算均在配备 32 GB 内存的 Intel i9-11900KF CPU 上执行。这些基准测试的结果如图 4 和 5所示。

图 4显示，对于所有测试的系统大小，PauliEngine和PauliArray的性能都优于OpenFermion 一个至两个数量级，且扩展性显著更好。由于OpenFermion 在哈密顿量的项数超过几百项后便会变得慢到难以实用，因此后续测试着重直接比较PauliEngine与PauliArray 。

图 5 中的趋势显示出明显的性能区间划分。对于较短的泡利串（大致不超过 250 个量子比特）或较小的哈密顿量，PauliArray凭借极其紧凑的基于 SIMD 的表示，实现了更短的运行时间。然而，随着哈密顿量规模或泡利串长度增加，会出现明显的交叉点：PauliArray 的内存使用量显著增长，最终耗尽可用内存并导致运行时间急剧增加。相比之下，PauliEngine 在这两方面都保持了平滑的扩展性参数。其节省内存的符号字典和位运算避免了大规模情况下影响 PauliArray 的快速膨胀。

从用户的角度来看，依赖这三个软件包之一的应用程序可以轻松地在它们之间切换，这一点令人安心。应用程序接口经过特意设计，以模仿 OpenFermion 的 QubitOperator，确保这种便利性。目前，我们建议在以下两种场景中使用 PauliEngine：1. 需要大型算符时。2. 需要参数化（且可微分）算符时。在其他场景中，PauliArray 和 OpenFermion 使用起来可能更方便，因为它们完全由 Python 编写。

VI. 结论

本文介绍了 PauliEngine，这是一款紧凑、高性能的 C++ 后端，用于符号泡利串算术。PauliEngine 将二进制辛表示与针对乘法、对易子计算和相位跟踪优化的位运算相结合，为大规模算符操作提供快速且节省内存的基本操作。该框架支持数值系数和符号系数，并可通过轻量级 Python 接口调用，因此适合作为量子软件工具的构建模块。

在一系列广泛的基准测试中，PauliEngine始终优于现有库，例如PennyLane、OpenFermion 我们认为它们代表了当前最先进的水平；此外，它也优于PauliArray等专用库。与所有这些库相比，在处理大型哈密顿量或长泡利串时，它的运行速度提升了数个数量级，并具有明显的可扩展性优势。由于PauliArray软件包已经与qiskit [11]，进行了基准测试，因此本文未将其纳入比较。

我们进一步展示了 PauliEnding 在实际应用中的效用，包括加速计算动力学李代数，这对于分析和模拟变分电路至关重要。我们认为，它在支持参数化和可微结构的软件开发工具包中最具应用潜力，例如 PennyLane 和 Tequila。在后者中，PauliEngine 几乎可以无缝集成；我们预计它也能以同样的方式集成到前者中。

总的来说，PauliEngine 为基于 Pauli 的量子模拟方法提供了高效且可扩展的基础，使得在此前难以处理的系统规模下开展经典研究和分析成为可能。

致谢

本研究获得了巴伐利亚高科技议程（JSK）、通过慕尼黑量子谷博士奖学金提供资助的慕尼黑量子谷（AB），以及德国……

德国联邦研究、技术与太空部（BMFTR）通过量子技术项目：HoliQC2（LM、AB）。作者感谢 Oliver H¨uttenhofer 就各种富有成果的讨论所给予的帮助。本项目使用`NanoBind` [48] 来绑定C++和 Python。

---

[1]X. Xu、S. Benjamin、J. Chen、J. Sun、X. Yuan 和 P. Zhang，《艰巨的任务：量子计算机的经典模拟》，Science Bulletin 10.1016/j.scib.2025.10.016（2025）。[2] P. Rall、D. Liang、J. Cook 和 W. Kretschmer，《通过泡利传播模拟量子比特量子电路》，Physical Review A 99，10.1103/physreva.99.062337（2019）。[3] M. S. Rudolph、T. Jones、Y. Teng、A. Angrisani 和 Z. Holmes，《泡利传播：模拟量子系统的计算框架》，arxiv:2505 21606 10 48550 2505 21606 2025 . . /ARXIV. .（）。[4] Z.-L. Li 和 S.-X. Zhang，《低权重泡利传播的双重作用：有缺陷的模拟器，却是变分量子算法的强大初始化器》，arxiv:2508.06358（2025）。[5]M. L. Goh、M. Larocca、L. Cincio、M. Cerezo 和 F. Sauvage，《用于量子计算的李代数经典模拟》，《物理评论研究》7，10 1103 3 65 5 6 2025 . / y -f w（）。[6] S. Developers，《Symengine：快速符号运算库》（2025）。[7] C. Gidney，《Stim：快速稳定子电路模拟器》，Quantum 5，497（2021）。[8] O. Higgott 和 C. Gidney，《Sparse Blossom：利用最小权匹配每核每秒纠正一百万个错误》，Quantum 9，1600（2025）。[9] M. Dion、T. Belabbas 和 N. Bastien，《使用 PauliArray 高效处理泡利串》（2024），arXiv:arXiv:2405.19287 [quant-ph]。[10] T. Jones、A. Brown、I. Bush、*et al.*，《QuEST 及量子计算机的高性能模拟》，《科学报告》9，10736（2019）。[11] A. Javadi-Abhari、M. Treinish、K. Krsulich、C. J. Wood、J. Lishman、J. Gacon、S. Martiel、P. D. Nation、L. S. Bishop、A. W. Cross、B. R. Johnson 和 J. M. Gambetta，《使用 Qiskit 进行量子计算》（2024），arXiv:arXiv:2405.08810 [quant-ph]。[12] Cirq Developers，《Cirq》（2025）。[13] M. Larocca、S. Thanasilp、S. Wang、K. Sharma、J. Biamonte、P. J. Coles、L. Cincio、J. R. Mc-Clean、Z. Holmes 和 M. Cerezo，《变分量子计算中贫瘠高原综述》，arXiv 预印本 arXiv:2405.00781 10.48550/arXiv.2405.00781（2024）。[14] M. Larocca、P. Czarnik、K. Sharma、G. Muraleedharan、P. J. Coles 和 M. Cerezo，《利用量子最优控制工具诊断贫瘠高原》，Quantum 6，824（2022）。[15] M. Ragone、B. N. Bakalov、F. Sauvage、A. F. Kemper、C. Ortiz Marrero、M. Larocca 和 M. Cerezo，《一个李贫瘠高原的代数理论，适用于深度参数化量子电路，《自然·通讯》15，7172（2024）。[16] E. Fontana、D. Herman、S. Chakrabarti、N. Kumar、R. Yalovetzky、J. Heredge、S. H. Sureshbabu 和 M. Pistoia，利用伴随表示刻画量子 ans¨atz 中的贫瘠高原，《自然·通讯》15，7171（2024）。[17] M. Cerezo、M. Larocca、D. Garc´ıa-Mart´ın、N. L. Diaz、P. Braccia、E. Fontana、M. S. Rudolph、P. Bermejo、A. Ijaz、S. Thanasilp、E. R. Anschuetz 和 Z. Holmes，可证明地不存在贫瘠高原是否意味着经典可模拟性？《自然·通讯》16，10 1038 41467 025 63099 6 2025。/s - - -（）。[18] V. Bergholm、J. Izaac、M. Schuld、C. Gogolin、S. Ahmed、V. Ajith、M. S. Alam、G. Alonso-Linaje、B. AkashNarayanan、A. Asadi、 *et al.*，Pennylane：量子-经典混合计算的自动微分，arXiv 预印本 arXiv:1811 04968 10 48550 1811 04968 2018。./arXiv。。（）。[19] K. Kottmann，为量子领域从业者介绍（动态）李代数，PennyLane 演示（2025）。[20] J. F. Gonthier、M. D. Radin、C. Buda、E. J. Doskocil、C. M. Abuan 和 J. Romero，测量是化学领域近期实用量子优势的障碍：资源分析，Physical Review Research 4，033154（2022）。[21] S. Patel、P. Jayakumar、T.-C. Yen 和 A. F. Izmaylov，量子计算机上的量子化学量子测量，《化学评论》125，7490（2025）。[22] D. Bincoletto 和 J. Kottmann，针对费米子可观测量期望值的物理信息测量协议，《数字发现》，（2025），DOI:10.1039/D5DD00251F。[23] P. Naldesi、A. Elben、A. Minguzzi、D. Cl´ement、P. Zoller 和 B. Vermersch，可编程原子量子器件中通过随机化测量获得费米子关联函数，Physical Review Letters 131，060601 2023（）。[24] A. Elben、S. T. Flammia、H.-Y. Huang、R. Kueng、J. Preskill、B. Vermersch 和 P. Zoller，随机化测量工具箱，Nature Reviews Physics 5，9 2023（）。[25] T.-C. Yen、V. Verteletskyi 和 A. F. Izmaylov，使用酉变换在一系列单量子比特测量中测量所有相容算符，Journal of Chemical Theory and Computation 16，2400（2020）。[26] V. Verteletskyi、T.-C. Yen 和 A. F. Izmaylov，利用最小团覆盖优化变分量子本征求解器中的测量，《化学物理学杂志》

152，10.1063/5.0004875（2020）。[27] T.-C. Yen、A. Ganeshram 和 A. F. Izmaylov，通过相容算符分组、非局域变换和协方差估计确定性地改进量子测量，npj Quantum Information 9，14（2023）。[28] A. Gresch 和 M. Kliesch，使用 shadowgrouping 高效估计量子多体哈密顿量的能量并提供保证，《自然·通讯》16，689（2025）。[29] M. Schuld、V. Bergholm、C. Gogolin、J. Izaac 和 N. Killoran，在量子硬件上评估解析梯度，Physical Review A 99，032331（2019）。[30] T. Jones 和 J. Gacon，在变分量子算法的经典模拟中高效计算梯度，arXiv 预印本 arXiv:2009 02823 10 48550 2009 02823 2020。./arXiv。。（）。[31] A. Anand、P. Schleich、S. Alperin-Lea、P. W. Jensen、S. Sim、M. D´ıaz-Tinoco、J. S. Kottmann、M. Degroote、A. F. Izmaylov 和 A. Aspuru-Guzik，幺正耦合簇理论的量子计算视角，Chemical Society Reviews 51，1659（2022）。[32] K. Bharti 和 T. Haug，迭代量子辅助本征求解器，Physical Review A 104，L050401（2021）。[33] M. Cerezo、A. Arrasmith、R. Babbush、S. C. Benjamin、S. Endo、K. Fujii、J. R. McClean、K. Mitarai、X. Yuan、L. Cincio、 *et al.*，变分量子算法，Nature Reviews Physics 3，625（2021）。[34] I. G. Ryabinkin、R. A. Lang、S. N. Genin 和 A. F. Izmaylov，带有生成元高效筛选的迭代量子比特耦合簇方法，Journal of Chemical Theory and Computation 16，1055（2020）。[35] Z.-X. Shang、M.-C. Chen、X. Yuan、C.-Y. Lu 和 J.-W. Pan，薛定谔-海森堡变分量子算法，Physical Review Letters 131，060406（2023）。¨Y. Zhang、L. Cincio、C. F. Negre、P. Czarnik、P. J. Coles、P. M. Anisimov、S. M. Mniszewski、S. Tretiak 和 P. A. Dub，具有降低电路复杂度的变分量子本征求解器，npj Quantum Information 8，96（2022）。[36] J. Sun、L. Cheng 和 W. Li，借助浅层量子电路实现化学精度：一种基于克利福德门的哈密顿量工程方法，Journal of Chemical Theory and Computation 20，695（2024）。[37] J. Sun、L. Cheng 和 S.-X. Zhang，用于模拟量子多体物理的稳定子基态：ory、算法与应用，《Quantum》9，1782（2025）。[39]A. Anand 和 K. R. Brown，《稳定子组态相互作用：利用错误检测特性寻找分子子空间》，Physical Review A 112，032421（2025）。[40]W. Dobrautz、I. O. Sokolov、K. Liao、P. L. R´ıos、M. Rahm、A. Alavi 和 I. Tavernelli，《通过变相关方法在当前量子硬件上实现真正的化学精度》，Journal of Chemical Theory and Computation 20，4146（2024）。[41] I. O. Sokolov、W. Dobrautz、H. Luo、A. Alavi 和 I. Tavernelli，《通过精确的变相关方法，在量子计算机上将量子多体问题的精度提高数个数量级》，《物理评论研究》5，023174（2023）。[42] A. Kumar、A. Asthana、C. Masteran、E. F. Valeev、Y. Zhang、L. Cincio、S. Tretiak 和 P. A. Dub，《利用转关联哈密顿量模拟分子电子态：以更少的量子比特实现更高精度》，Journal of Chemical Theory and Computation 18，5312（2022）。[43] W. J. Huggins、J. Lee、U. Baek、B. O'Gorman 和 K. B. Whaley，《一种非正交变分量子本征求解器》，《新物理学杂志》22，073009（2020）。[44] J. S. Kottmann 和 F. Scala，《多组态价键理论的量子算法方法：从可解释的电路设计中获得的见解》，Journal of Chemical Theory and Computation 20，3514（2024）。[45] N. H. Stair、R. Huang 和 F. A. Evangelista，《一种用于强关联电子的多参考量子克雷洛夫算法》，Journal of Chemical Theory and Computation 16，2236（2020）。[46] N. P. Bauman、B. Peng 和 K. Kowalski，《耦合簇降阶技术：化学系统中经典计算与量子计算现有应用综述》，《量子化学进展》87，141（2023）。[47] J. R. McClean、N. C. Rubin、K. J. Sung、I. D. Kivlichan、X. Bonet-Monroig、Y. Cao、C. Dai、E. S. Fried、C. Gidney、B. Gimby、 *et al.*，《OpenFermion：面向量子计算机的电子结构软件包》，《量子科学与技术》5，034014（2020）。[48] W. Jakob，《nanobind：轻巧高效的 C++/Python 绑定》（2022），https://github.com/wjakob/nanobind。