Министерство науки и высшего образования РФ
ФГБОУ ВО «Тверской государственный университет»
Математический факультет
Направление 02.03.01 Математика и компьютерные науки
Профиль «Математическое и компьютерное моделирование»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Вычисление операторных экспонент в базисе Паули

Автор:
Алешин Дмитрий Алексеевич

Научный руководитель:
д. ф.-м. н. Цирулёв А.Н.

Допущен к защите:
Руководитель ООП:

_____ В.П. Цветков

Тверь 2023

Оглавление

Введение

В современном математическом моделировании конечномерные малоразмерные квантовые системы являются предметом пристального исследования. Такие системы моделируют особое, недавно обнаруженное, состояние вещества. Материалы, построенные на основе таких состояний называются топологическими [6]. Одно из основных математических задач, которые возникают при исследовании квантовых систем, является вычисление операторных экспонент от гамильтонианов таких систем. Операторная экспонента может моделировать унитарную эволюцию системы, то есть иметь вид $\mathrm{e}^{-it\hat{H}}$, или оператор плотности состояния квантовой подсистемы в термодинамическом равновесии, то есть подсистемы, находящиеся в контакте с окружающей средой - термостатом.

Для малоразмерных квантовых систем гамильтонианы, как правило, имеют достаточно простой вид, так что для них актуальной является задача прямого программного вычисления операторных экспонент. При этом вычислительные алгоритмы должны быть разработаны отдельно для каждого специального класса малоразмерных квантовых систем. Данная выпускная квалификационная работа посвящена разработке и программной реализации одного из таких алгоритмов, а именно алгоритма вычисления операторных экспонент для системы трёх кубитов с гамильтонианом специального вида. Этот гамильтониан имеет трёхпараметрическую форму и представлен суммой трёх операторов Паули с произвольными коэффициентами.

Цель исследования: изучить свойства операторной экспоненты для трехкубитных квантовых систем и разработать алгоритмы её вычисления с помощью разложения генерирующего оператора.

Основные задачи:

1. Изучить основные свойства базиса Паули
2. Рассмотреть возможные типы однородных трехкубитных гамильтонианов
3. Разработать алгоритм вычисления операторной экспоненты для трёх кубитов в базисе Паули

Объектом исследования является формула вычисления операторной экспоненты в базисе Паули. Предметом исследования является разработка алгоритма и программная реализация метода, позволяющего вычислить операторную экспоненту для трёх кубитов в базисе Паули.

В работе присутствуют две главы: первая ознакомительная, которая рассказывает о базисе Паули и его связи со стандартным базисом, а вторая глава посвящена операторной экспоненте, разложению гамильтониана, вычислению операторных экспонент для сумм коммутирующих и антикоммутирующих гамильтонианов и алгоритмам прямого вычисления операторной экспоненты. Кроме того, в работе представлена программная реализация вычисления операторной экспоненты для трех кубитов в базисе Паули на языке программирования $C_{\#}$. В заключении сделаны выводы по проделанной работе. Список литературы состоит из 6 источников. В приложении приведен исходный код реализации вычисления операторной экспоненты для трех кубитов в базисе Паули на языке программирования.

Для написания квалификационной работы были применены следующие методы научного исследования: анализ литературы, изучение и обобщение информации, моделирование и проектирование.

В качестве основы для исследования использовались статьи российских и зарубежных ученых, посвященные темам, связанным с темой данной работы.

Всюду в работе используется система единиц, в которой постоянная Планка $\hbar = 1$ и скорость света $c = 1$, так что энергия измеряется (собственное значение гамильтониана) в обратных сантиметрах.

Глава 1

Базис Паули

1.1 Связь стандартного базиса и базиса Паули

Мы будем рассматривать квантовую систему n различных кубитов, где кубит связан с двумерным Гильбертовым пространством $\mathcal{H}$ и его двумерным (эрмитово сопряжение) пространством $\mathcal{H}^\dagger$. Пусть $\mathcal{H}_n = \mathcal{H}^{\otimes n}$ и $\mathcal{H}_n^\dagger = (\mathcal{H}^\dagger)^{\otimes n}$ будут Гильбертовым пространством системы и его эрмитовым сопряжением, соответственно, и пусть $L(\mathcal{H}_n) = \mathcal{H}_n \otimes \mathcal{H}_n^\dagger$ будут пространством линейных операторов, действующих на $\mathcal{H}$ и $\mathcal{H}^\dagger$ левым и правым сокращениями соответственно. Тогда

$$\dim_{\mathbb{C}} \mathcal{H}_n = \dim_{\mathbb{C}} \mathcal{H}_n^\dagger = 2^n, \quad \dim_{\mathbb{C}} L(\mathcal{H}_n) = 2^{2n}.$$

Будем также считать, что пространство $L(\mathcal{H}_n)$ снабжено внутренним произведением Гильберта-Шмидта,

$$\langle \hat{A}, \hat{B} \rangle = \mathrm{tr}(\hat{A}^\dagger \hat{B}), \quad \hat{A}, \hat{B} \in L(\mathcal{H}_n), \tag{1}$$

которое является естественным продолжением внутреннего продукта в $\mathcal{H}_n$. Вещественное линейное пространство эрмитовых операторов далее обозначается как $H(\mathcal{H}_n)$.

Пусть $\{|0\rangle, |1\rangle\}$ являются ортнормированным базисом в некотором однокубитном пространстве в $\mathcal{H}$. Единичная матрица и матрицы Паули,

$$\sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ \sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

определим четыре оператора Паули

$$\hat{\sigma}_0 = |0\rangle\langle 0| + |1\rangle\langle 1|,\ \hat{\sigma}_1 = |0\rangle\langle 1| + |1\rangle\langle 0|,$$
$$\hat{\sigma}_2 = -i|0\rangle\langle 1| + i|1\rangle\langle 0|,\ \hat{\sigma}_3 = |0\rangle\langle 0| - |1\rangle\langle 1|,$$

которые эрмитовы и унитарны одинаково одновременно, и которые составляют основу в $\mathcal{H}$. Обратное преобразование дается выражением

$$|0\rangle\langle 0| = \frac{\hat{\sigma}_0 + \hat{\sigma}_3}{2},\ |0\rangle\langle 1| = \frac{\hat{\sigma}_1 + i\hat{\sigma}_2}{2},\ |1\rangle\langle 0| = \frac{\hat{\sigma}_1 - i\hat{\sigma}_2}{2},\ |1\rangle\langle 1| = \frac{\hat{\sigma}_0 - \hat{\sigma}_3}{2}.$$

Напомним, что для $k, l, m \in \{1, 2, 3\}$ у нас есть $\mathrm{tr}\hat{\sigma}_k = 0,\ \hat{\sigma}_k^2 = \hat{\sigma}_0$, и

$$\hat{\sigma}_k\hat{\sigma}_l = -\hat{\sigma}_l\hat{\sigma}_k, \quad \hat{\sigma}_k\hat{\sigma}_l = i\mathrm{sign}(\pi)\hat{\sigma}_m, \quad (klm) = \pi(123), \tag{2}$$

где $\pi(123)$ является перестановкой $\{1, 2, 3\}$.

Существует стандартный[1] бинарный базис в $\mathcal{H}_n$ порожденный ортонормированными базисами $\{|0\rangle, |1\rangle\}$ в соответствующих однокубитных пространствах. Математически, положение в тензорном произведении отличает кубиты друг от друга. Поэтому для фиксированного n, элемент этого базиса и соответствующий ему элемент двумерного базиса удобно записать в виде

$$|k\rangle = |k_1 \ldots k_n\rangle = |k_1\rangle \otimes \ldots \otimes |k_n\rangle, \quad \langle k| = \langle k_1 \ldots k_n| = \langle k_1| \otimes \ldots \otimes \langle k_n|,$$

относительно строк $k_1 \ldots k_n$, $(k_1, \ldots, k_n \in \{0, 1\})$ как двоичное число и обозначая его десятичным представлением k. Например, $|101\rangle = |5\rangle$ и $|00110\rangle = |6\rangle$.

[1]Мы не используем общепринятый термин «вычислительный» , потому что это может вызывать путаницу. Базис Паули и стандартный базис являются вычислительными в одном и том же смысле.

В стандартном базисе,

$$|u\rangle = \sum_{k=0}^{2^n-1} u_k |k\rangle, \quad \hat{A} = \sum_{k,l=0}^{2^n-1} a_{kl} |k\rangle\langle l|,$$

где $|u\rangle \in \mathcal{H}_n$ и $\hat{A} \in L(\mathcal{H}_n)$.

Базис Паули $P(\mathcal{H}_n)$ в $L(\mathcal{H}_n)$ определен как

$$\{\hat{\sigma}_{k_1\ldots k_n}\}_{k_1,\ldots,k_n\in\{0,1,2,3\}}, \quad \hat{\sigma}_{k_1\ldots k_n} = \hat{\sigma}_{k_1} \otimes \ldots \otimes \hat{\sigma}_{k_n}, \tag{3}$$

где $\hat{\sigma}_{0\ldots0}$ тождественный оператор. Очевидно, что $P(\mathcal{H}_n)$ состоит из 4^n элементов. Мы будем использовать сокращенное выражение вида

$$\hat{\sigma}_K = \hat{\sigma}_{k_1\ldots k_n},$$

обозначим строку Паули $k_1 \ldots k_n$, где $k_1 \ldots k_n \in \{0, 1, 2, 3\}$, соответствует заглавной букве K. При этом мы часто будем рассматривать K как число, то есть как десятичное представление строки; понятно, что $0 \leq K \leq 4^n - 1$. Обратим внимание, что строка Паули K и элемент $\hat{\sigma}_K$ базиса Паули полностью определяются друг другом и, следовательно, могут быть отождествлены.

Полезно сравнить $P(\mathcal{H}_n)$ со стандартным базисом. Имеем

$$\hat{\sigma}_{k_1\ldots k_n}\hat{\sigma}_{k_1\ldots k_n} = \hat{\sigma}_{0\ldots0}, \quad \mathrm{tr}\hat{\sigma}_{0\ldots0} = 2^n \quad , \mathrm{tr}\hat{\sigma}_{k_1\ldots k_n}\Big|_{k_1\ldots k_n\neq 0\ldots0} = 0. \tag{4}$$

Базис Паули эрмитов, унитарный и ортогональный относительно скалярного произведения (1). Обратим внимание, что оператор $|k\rangle\langle l|$ из стандартного базиса не унитарный, и не эрмитов, если $k \neq l$. Стандартный базис не содержит тождественного оператора, имеющего вид

$$\sum_{k=0}^{2^n-1} |k\rangle\langle k|$$

в этом базисе. В базисе Паули, любой оператор $\hat{U}$ из унитарной группы $U(\mathcal{H}_n)$ (то есть, $\hat{U}^\dagger\hat{U} = \hat{\sigma}_{0\ldots0}$) имеет разложение вида

$$\hat{U} = \sum_{i_1,\ldots,i_n\in\{0,1,2,3\}} U_{i_1\ldots i_n}\hat{\sigma}_{i_1\ldots i_n}, \quad \hat{U}^\dagger = \sum_{i_1,\ldots,i_n\in\{0,1,2,3\}} \overline{U}_{i_1\ldots i_n}\hat{\sigma}_{i_1\ldots i_n},$$

где

$$\sum_{i_1,\ldots,i_n\in\{0,1,2,3\}} \overline{U}_{i_1\ldots i_n} U_{i_1\ldots i_n} = 1, \qquad \sum_{\substack{i_1,\ldots,i_n,j_1,\ldots,j_n\in\{0,1,2,3\} \\ (i_1,\ldots,i_n)\neq(j_1,\ldots,j_n)}} \overline{U}_{i_1\ldots i_n} U_{j_1\ldots j_n} = 0.$$

Обратим внимание, что последнее условие может быть очевидно разложено на $2^{2n-1}(2^n-1)$ независимых условий.

Эрмитовы операторы разлагаются по базису Паули с вещественными коэффициентами.

1.2 Коммутационное и антикоммутационное соотношение

Коммутационное соотношение - это свойство, которое описывает коммутационные отношения между операторами. Для двух операторов A и B коммутационное соотношение определяется следующим образом:

$$[A, B] = AB - BA,$$

если коммутатор $[A, B]$ равен нулю, то операторы A и B коммутируют. Если же коммутатор не равен нулю, то операторы не коммутируют. В базисе Паули коммутационное соотношение может быть выражено следующим образом:

$$[\sigma_i, \sigma_j] = 2i\varepsilon_{ijk}\sigma_k,$$

здесь ε_{ijk} - это символ Леви-Чивиты, который равен 1, если i, j, k образуют четную перестановку, -1, если они образуют нечетную перестановку, и 0 в остальных случаях. Это соотношение означает, что операторы Паули не коммутируют между собой, кроме случая, когда $i = j$. Например, $[\sigma_1, \sigma_2] = 2i\sigma_3$, $[\sigma_2, \sigma_3] = 2i\sigma_1$ и т.д.

Антикоммутационное соотношение - это свойство, которое описывает антикоммутационные отношения между операторами. Для двух операторов A и B антикоммутационное соотношение определяется следующим образом:

$$\{A, B\} = AB + BA,$$

если антикоммутатор $\{A, B\}$ равен нулю, то операторы A и B антикоммутируют. Если же антикоммутатор не равен нулю, то операторы не антикоммутируют. В базисе Паули антикоммутационное соотношение может быть выражено следующим образом:

$$\{\sigma_i, \sigma_j\} = 2\delta_{ij},$$

здесь δ_{ij} - это символ Кронекера, который равен 1, если $i = j$, и 0 в остальных случаях. Это соотношение означает, что операторы Паули антикоммутируют между собой только в случае, когда $i \neq j$. Например, $\{\sigma_1, \sigma_2\} = 0, \{\sigma_2, \sigma_3\} = 0$ и т.д.

Коммутационное и антикоммутационное соотношение имеют множество применений в квантовой механике. Например, они используются для описания свойств квантовых систем, таких как спин электрона или квантовые биты. Коммутационное соотношение может быть использовано для вычисления коммутаторов операторов, что позволяет определить, коммутируют ли они между собой. Это важно для определения свойств квантовых систем, таких как энергия или момент импульса. Антикоммутационное соотношение может быть использовано для вычисления антикоммутаторов операторов, что позволяет определить, антикоммутируют ли они между собой. Это важно для определения свойств квантовых систем, таких как спин электрона или квантовые биты. Кроме того, коммутационное и антикоммутационное соотношение используются в квантовой теории поля для описания взаимодействия между частицами. Они также используются в квантовой информатике для разработки квантовых алгоритмов и квантовых компьютеров.

Рассмотрим несколько примеров коммутационного и антикоммутационного соответствия в базисе Паули.

1. Коммутатор σ_1 и σ_2:
$$[\sigma_1, \sigma_2] = 2i\epsilon_{12k}\sigma_k = 2i\sigma_3;$$
2. Антикоммутатор σ_1 и σ_2:
$$\{\sigma_1, \sigma_2\} = 2\delta_{12}\sigma_0 = 0;$$
3. Коммутатор σ_1 и σ_3:
$$[\sigma_1, \sigma_3] = 2i\epsilon_{13k}\sigma_k = -2i\sigma_2;$$
4. Антикоммутатор σ_1 и σ_3:
$$\{\sigma_1, \sigma_3\} = 2\delta_{13}\sigma_0 = 0;$$
5. Коммутатор σ_2 и σ_3:
$$[\sigma_2, \sigma_3] = 2i\epsilon_{23k}\sigma_k = 2i\sigma_1;$$
6. Антикоммутатор σ_2 и σ_3:
$$\{\sigma_2, \sigma_3\} = 2\delta_{23}\sigma_0 = 0.$$

1.3 Коммутирующие элементы в алгебре Ли группы $SU(2^n)$

Нам понадобится несколько фактов и определений, связанных с базисом Паули и множеством n-длины строк Паули,

$$\mathrm{Str}_n = \{K = k_1 \dots k_n\}_{k_1,\dots,k_n \in \{0,1,2,3\}}.$$

Во-первых, рассмотрим множество $\mathbb{F}_4 = \{0,1,2,3\}$ как квадгруппу Клейна с правилами умножения

$$0 * k = k, \quad k * k = 0, \quad k * l = m,$$

где $k,l,m \in \{1,2,3\}$ и klm есть любая перестановка 123. Во-вторых, пусть функция $s : \mathbb{F}_4 \times \mathbb{F}_4 \to \{1, i, -i\}$ определяется значениями

$$\begin{aligned} &s(0,0) = s(0,k) = s(k,0) = s(k,k) = 1, \quad k = 1,2,3 \\ &s(1,2) = s(2,3) = s(3,1) = i, \quad s(2,1) = s(3,2) = s(1,3) = -i. \end{aligned}$$

Далее, пусть функция $S : \mathrm{Str}_n \times \mathrm{Str}_n \to \{1, -1, i, -i\}$, $(K,L) \mapsto S_{KL}$, будет определена как произведение

$$S_{KL} = s(k_1,l_1)s(k_2,l_2)\dots s(k_n,l_n), \quad K = k_1k_2\dots k_n, \quad L = l_1l_2\dots l_n.$$

Функция S симметрична или антисимметрична в зависимости от количества пар k_r, l_r (r это позиция в строках K и L), таких что $k_r, k_l \in \{1,2,3\}$ и $k_r \neq l_r$, а также в зависимости от относительного порядка в них. Пусть ω^+_{KL} и ω^-_{KL} будут числами пары форм $(1,2),(2,3),(3,1)$ и форм $(2,1),(3,2),(1,3)$ соответственно, и пусть $\omega_{KL} = \omega^+_{KL} + \omega^-_{KL}$. Тогда

$$S_{KL} = (i)^{\omega_{KL}}(-1)^{\omega^-_{KL}}, S_{(KL)} = \frac{S_{KL}}{2}(1 + (-1)^{\omega_{KL}}), S_{[KL]} = \frac{S_{KL}}{2}(1 - (-1)^{\omega_{KL}}), \qquad (5)$$

где круглые и квадратные скобки обозначают симметризацию и антисимметризацию соответственно. Значения S_{KL}, $S_{(KL)}$ и $S_{[KL]}$ приведены в Таблице 2.

Теперь композиция двух базисных элементов Паули и их антикоммутатора и коммутатора можно записать в виде компактных выражений,

$\omega_{KL}\mathrm{mod}4$	0	2	0	2	1	3	1	3
$\omega^{-}_{KL}\mathrm{mod}4$	0	1	1	0	0	1	1	0
S_{KL}		1	-1	-1	i	i	$-i$	$-i$
$S_{(KL)}$	1	1	-1	-1	0	0	0	0
$S_{[KL]}$	0	0	0	0	i	i	$-i$	$-i$

Таблица 1: Множитель до $\hat{\sigma}_M$ в (6) для $\hat{\sigma}_K\hat{\sigma}_L, \{\hat{\sigma}_K, \hat{\sigma}_L\}$, и $[i\hat{\sigma}_K, i\hat{\sigma}_L]$.

удобных для классического компьютерного программирования:

$$\hat{\sigma}_K\hat{\sigma}_L = S_{KL}\hat{\sigma}_M, \quad \{\hat{\sigma}_K, \hat{\sigma}_L\} = S_{(KL)}\hat{\sigma}_M, \quad [i\hat{\sigma}_K, i\hat{\sigma}_L] = -S_{[KL]}\hat{\sigma}_M, \tag{6}$$

где

$$\hat{\sigma}_M = \hat{\sigma}_{m_1\ldots m_n}, \quad m_1 = k_1 * l_1, \ldots, m_n = k_n * l_n. \tag{7}$$

Обратим внимание, что две строки Паули длины n могут коммутировать, даже если они имеют разные ненулевые записи в одних и тех же местах. Например, три оператора $\hat{\sigma}_{11}$, $\hat{\sigma}_{22}$, и $\hat{\sigma}_{33}$ взаимно коммутируют. Так же легко увидеть, что унитарная матрица перехода, преобразующая стандартный базис $\{|i_1\ldots i_n\rangle\langle j_1\ldots j_n|\}$ в базис Паули, состоит только из элементов 0, ± 1, и $\pm i$. В частности

$$|00\ldots0\rangle\langle 00\ldots0| \to \frac{1}{2^n}\sum_{i_1,\ldots,i_n\in\{0,3\}} \hat{\sigma}_{i_1\ldots i_n}.$$

В более общем виде стандартные ортогональные проекторы могут быть выражены как

$$|i_1\ldots i_n\rangle\langle i_1\ldots i_n|_{i_1,\ldots,i_n\in\{0,1\}} = \frac{1}{2^n}\sum_{k_1,\ldots,k_n\in\{0,3\}} \mathcal{X}^{i_1}_{k_1}\cdots\mathcal{X}^{i_n}_{k_n}\,\hat{\sigma}_{k_1\ldots k_n},$$

где

$$\mathcal{X}^0_0 = \mathcal{X}^0_3 = \mathcal{X}^1_0 = 1, \quad \mathcal{X}^1_3 = -1.$$

Выражение (6) показывает, во-перых, что множество $\{i\hat{\sigma}_K\}_{K=0}^{4^n-1}$ составляет ортонормированный базис в $\mathfrak{su}(n)$. И, во-вторых, множество

$$\widetilde{P}(\mathcal{H}_n) = \{\epsilon\hat{\sigma}_K | K \in \mathrm{Str}_n, \epsilon \in \{\pm 1, \pm i\}\},$$

который состоит из 4^{n+1} элементов, это группа; это называется (n-кубитной) группой Паули. Нормализатор группы Паули,

$$\mathcal{C}(\mathcal{H}_n) = \{\hat{U} \in U(\mathcal{H}_n) | \hat{U}\hat{\sigma}_K\hat{U}^\dagger \in \widetilde{P}(\mathcal{H}_n), \;\; \hat{\sigma}_K \in \widetilde{P}(\mathcal{H}_n)\},$$

называется группой Клиффорда. Исходя из (2), (4) и (7) получаем следующее утверждение

Утверждение 1. *Взаимные унитарные преобразования базисных операторов Паули подчиняются соотношениям* $\hat{\sigma}_{i_1\ldots i_n}\hat{\sigma}_{k_1\ldots k_n}\hat{\sigma}_{i_1\ldots i_n} = \pm\hat{\sigma}_{i_1\ldots i_n}$, *где знак плюс стоит тогда и только тогда, когда количество троек* $(i_m k_m i_m)_{m\in\{1,\ldots,n\}}$, *удовлетворяют условиям* $i_m \neq k_m$, $i_m \neq 0$, *и* $k_m \neq 0$ *четности.*

l	0	1	2	3	4	5	6	7
$l_2l_1l_0$	000	001	010	011	100	101	110	111
$k_2k_1k_0$	011	011	011	011	011	011	011	011
$\overline{l} \wedge k$	011	010	001	000	011	010	001	000
$l \wedge k$	000	001	010	011	000	001	010	011
$l \wedge \overline{k}$	000	000	000	000	100	100	100	100
$\hat{\sigma}_I$	$\hat{\sigma}_{011}$	$\hat{\sigma}_{012}$	$\hat{\sigma}_{021}$	$\hat{\sigma}_{022}$	$\hat{\sigma}_{311}$	$\hat{\sigma}_{312}$	$\hat{\sigma}_{121}$	$\hat{\sigma}_{322}$

Таблица 2: Элементы базиса Паули, возникающие для $k = 011$.

Глава 2

Операторная экспонента

2.1 Разложение гамильтониана

Операторная экспонента - это математический объект, который используется в квантовой механике для описания эволюции квантовых систем во времени. Она является аналогом обычной экспоненты в классической механике и определяется через ряд Тейлора оператора. Операторная экспонента может быть использована для вычисления эволюционного оператора, который описывает изменение состояния квантовой системы во времени. Она играет важную роль во многих областях квантовой механики, включая теорию поля, теорию квантовой информации и квантовую оптику. Формула для вычисления операторной экспоненты через ряд Маклорена:

$$\mathrm{e}^{\hat{A}} = \sum_{n=0}^{\infty} \frac{1}{n!} \hat{A}^n,$$

где $\hat{A}$ - оператор, а $\hat{A}^n$ - его n-ая степень.

Разложение гамильтониана в базисе Паули - это представление гамильтониана квантовой системы в виде линейной комбинации произведений матриц Паули, которые образуют базис в пространстве состояний системы. Формула для разложения гамильтониана в базисе Паули:

$$\hat{H} = \sum_{i_1,\ldots,i_n \in \{0,1,2,3\}} h_{i_1 \ldots i_n} \hat{\sigma}_{i_1 \ldots i_n}$$

где n - число кубитов в системе, h_i - коэффициенты, а $\hat{\sigma}_i$ - операторы Паули, определяемые как:

$$\hat{\sigma}_0 = |0\rangle\langle 0| + |1\rangle\langle 1|, \quad \hat{\sigma}_1 = |0\rangle\langle 1| + |1\rangle\langle 0|,$$
$$\hat{\sigma}_2 = -i|0\rangle\langle 1| + i|1\rangle\langle 0|, \quad \hat{\sigma}_3 = |0\rangle\langle 0| - |1\rangle\langle 1|.$$

Представление гамильтониана в базисе Паули позволяет упростить вычисления в квантовой механике и использовать методы линейной алгебры для решения задач. Кроме того, это разложение может быть использовано для построения квантовых алгоритмов и квантовых вычислений.

2.2 Вычисление операторных экспонент для сумм коммутирующих и антикоммутирующих гамильтонианов

Алгоритм реализован на языке программирования C#. Выбор был осуществлен в сторону этого языка из-за его удобства и возможности использовать в будущем этот алгоритм как стороннюю библиотеку. Так же этот язык позволяет разрабатывать алгоритм в стиле объектно-ориентированного программирования.

Для начала представим общий вид гамильтониана для трёх кубитов:

$$\hat{H} = \sum_{i,j,k \in \{0,1,2,3\}} h_{ijk} \hat{\sigma}_{ijk}.$$

Но в нашей работе мы будем использовать гамильтониан взаимодействия, состоящий из трёх слагаемых, в роли которых выступают операторы Паули. Выглядит он следующим образом:

$$\hat{H} = a\hat{\sigma}_A + b\hat{\sigma}_B + c\hat{\sigma}_C, \tag{8}$$

где a, b, c — некоторые коэффициенты, а $\hat{\sigma}_A, \hat{\sigma}_B, \hat{\sigma}_C$ — операторы Паули.

Рассмотрим следующую классификацию гамильтонианов:

1. $[\hat{\sigma}_A, \hat{\sigma}_B] = [\hat{\sigma}_A, \hat{\sigma}_C] = [\hat{\sigma}_B, \hat{\sigma}_C] = 0$;
2. $\{\hat{\sigma}_A, \hat{\sigma}_B\} = \{\hat{\sigma}_A, \hat{\sigma}_C\} = \{\hat{\sigma}_B, \hat{\sigma}_C\} = 0$;
3. $\{\hat{\sigma}_A, \hat{\sigma}_B\} = 0, \quad [\hat{\sigma}_A, \hat{\sigma}_C] = [\hat{\sigma}_B, \hat{\sigma}_C] = [\hat{\sigma}_A + \hat{\sigma}_B, \hat{\sigma}_C] = 0$;
4. $[\hat{\sigma}_A, \hat{\sigma}_B] = 0, \quad \{\hat{\sigma}_A, \hat{\sigma}_C\} = \{\hat{\sigma}_B, \hat{\sigma}_C\} = 0$.

Пункты 1-3 решаются аналитически, поэтому в этой работе мы не будем их рассматривать. Предметом исследования послужит последний пункт классификации гамильтонианов, подробный алгоритм вычисления которого будет описан в параграфе 2.3.

Опишем какие входные данные имеет наш алгоритм. А именно a, b, c — некоторые коэффициенты, а $\hat{\sigma}_A, \hat{\sigma}_B, \hat{\sigma}_C$ — операторы Паули. Перейдём к описанию алгоритму вычисления композиции двух операторов, которая понадобится нам для вычисления операторной экспоненты для трёх кубитов. Пояснение, написанное ниже, объясняет работу на примере двух операторов. Далее $\hat{\sigma}_K$ и $\hat{\sigma}_L$. Для данного алгоритма был реализован класс $Calculation$, состоящий из методов $PauliMatrices, Operations$ и $Factors$. О нём подробнее ниже

Два оператора Паули $\hat{\sigma}_K$ и $\hat{\sigma}_L$ необходимы для расчёта $\omega\hat{\sigma}_M$. Далее, в строках 29-51 происходит расчёт индекса $\hat{\sigma}_M$.

```
29 public static string PauliMatrices(string K, string L)
30 {
31     char[] K1 = K.ToCharArray();
32     char[] L1 = L.ToCharArray();
33     string M = "";
34     int a, b, c;
35     if (K1.Length == L1.Length)
36     {
37      for (int i = 0; i < K1.Length; i++)
38      {
39       a = K1[i] - '0';
40       b = L1[i] - '0';
41       c = Calculation.Operations(a, b);
42       M = M + Convert.ToString(c);
43      }
44      return M;
45     }
46     else
47     {
48      Console.WriteLineОшибка(" размерности кубитов двух операторов");
49      return "";
50     };
51 }
```

В строках 31 и 32 разбиваются строки K и L на массив символов. В строках 39-42 происходит приведение к типу int и вычисление элемента, затем добавление его в строку.

Процедура вычисления элементов у нас начинается в строках 52-60.

```
52 public static int Operations(int a, int b)
53 {
54  int c;
55  if (a == b) c = 0;
56  else if (a == 0) c = b;
57  else if (b == 0) c = a;
```

```
 else c = 6 / (a * b);
 return c;
}
```

Если $a * a = 0, a * 0 = a, a * b = c$, то $a, b, c = 1, 2, 3$

Расчет коэффициента ω описан в строках 61-91 и вычисляется по формуле:

$$\omega = (i)^{p+m}(-1)^m,$$

где p - количество соответствий 1-2, 2-3, 3-1, а m - количество соответствий 2-1, 2-3, 1-3.

```
public static string Factors(string K, string L)
{
 char[] K1 = K.ToCharArray();
 char[] L1 = L.ToCharArray();
 int a, b, skl, p = 0, m = 0;
 string w = "";
 if (K1.Length == L1.Length)
 {
  for (int i = 0; i < K1.Length; i++)
  {
   a = K1[i] - '0';
   b = L1[i] - '0';
   if (a - b == 1 || a - b == -2) m++;
   if (b - a == 1 || b - a == -2) p++;
   p = p + m;
  }
  if (p % 4 == 0 || p % 4 == 1)
  {
   skl = Convert.ToInt32(Math.Pow(-1, m));
   if (skl == 1) w = "";
   else w = "-1";
  }
  if (p % 4 == 2 || p % 4 == 3)
  {
   skl = Convert.ToInt32(Math.Pow(-1, m + 1));
   if (skl == 1) w = "i";
   else w = "-i";
  }
 }
 return w;
}
```

Результатом выполнения данного алгоритма является строка композиции двух операторов Паули.

2.3 Алгоритмы прямого вычисления операторной экспоненты

В этом параграфе будет идти речь о вычислении степеней гамильтониана на основе полученных композиций операторов из параграфа 2.2. Подробнее об алгоритме.

Отправной точкой для этого алгоритма поступят формулы вычисления степеней гамильтониана, полученные аналитически:

$$\begin{aligned} \hat{H}^2 &= \hat{\sigma}_0 + 2ab\hat{\sigma}_A\hat{\sigma}_B, \\ \hat{H}^3 &= \hat{H} + 2ab^2\hat{\sigma}_A + 2a^2b\hat{\sigma}_B + 2abc\hat{\sigma}_A\hat{\sigma}_B\hat{\sigma}_C, \end{aligned} \tag{9}$$

Для данного алгоритма был реализован класс *Hamiltonian*, состоящий из методов *CountingOperators*, *CalculateSecondDegreeOfHamiltonian* и *CalculateThirdDegreeOfHamiltonian*. О них подробнее ниже.

В строках 95-166 происходит расчёт необходимых операторов для вычисления степеней гамильтониана.

```
95 public static string[] CountingOperators(string sigma_1, string sigma_2,
       string sigma_3)
96 {
97  string sigma_12 = Calculation.Factors(sigma_1, sigma_2) + $"*{sigma}_"
      + Calculation.PauliMatrices(sigma_1, sigma_2);
98  string sigma_23 = Calculation.Factors(sigma_2, sigma_3) + $"*{sigma}_"
      + Calculation.PauliMatrices(sigma_2, sigma_3);
99  string sigma_12_indexes = sigma_12.Split("_")[sigma_12.Split("_").
      Length - 1];
100 string sigma_123_draft = Calculation.Factors(sigma_12_indexes, sigma_3)
       + $"*{sigma}_" + Calculation.PauliMatrices(sigma_12_indexes, sigma_3
      );
101 string sigma_123;
102
103 if (sigma_12.StartsWith("-1"))
104 {
105  if (sigma_123_draft.StartsWith("-i"))
106  {
107   sigma_123 = $"i*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
     Split("_").Length - 1]}";
108  }
109  else if (sigma_123_draft.StartsWith("i"))
110  {
111   sigma_123 = $"-i*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
     Split("_").Length - 1]}";
112  }
```

```
 else if (sigma_123_draft.StartsWith("-1"))
 {
  sigma_123 = $"{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
  Split("_").Length - 1]}";
 }
 else
 {
  sigma_123 = $"-1*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
  Split("_").Length - 1]}";
 }
}
else if (sigma_12.StartsWith("-i"))
{
 if (sigma_123_draft.StartsWith("-i"))
 {
  sigma_123 = $"-1*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
  Split("_").Length - 1]}";
 }
 else if (sigma_123_draft.StartsWith("i"))
 {
  sigma_123 = $"{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
  Split("_").Length - 1]}";
 }
 else if (sigma_123_draft.StartsWith("-1"))
 {
  sigma_123 = $"i*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
  Split("_").Length - 1]}";
 }
 else
 {
  sigma_123 = $"-i*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
  Split("_").Length - 1]}";
 }
}
else if (sigma_12.StartsWith("i"))
{
 if (sigma_123_draft.StartsWith("-i"))
 {
  sigma_123 = $"{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
  Split("_").Length - 1]}";
 }
 else if (sigma_123_draft.StartsWith("i"))
 {
  sigma_123 = $"-1*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
  Split("_").Length - 1]}";
 }
 else if (sigma_123_draft.StartsWith("-1"))
 {
  sigma_123 = $"-i*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
  Split("_").Length - 1]}";
 }
```

```
155   else
156   {
157    sigma_123 = $"i*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
      Split("_").Length - 1]}";
158   }
159  }
160  else
161  {
162   sigma_123 = sigma_123_draft;
163  }
164  string[] sigma_array = new string[] { sigma_12, sigma_23, sigma_123 };
165  return sigma_array;
166 }
```

В строках 168-171 происходит расчёт второй степени гамильтониана из формулы (9).

```
168 public static string CalculateSecondDegreeOfHamiltonian(double alpha,
      double beta, string[] arrayOfSigma)
169 {
170  return $"{sigma}_000 + {2 * alpha* beta}*{arrayOfSigma[0]}";
171 }
```

В строках 172-175 происходит расчёт третьей степени гамильтониана из формулы (9).

```
172 public static string CalculateThirdDegreeOfHamiltonian(double alpha,
      double beta, double gamma, string sigma_1, string sigma_2, string[]
      arrayOfSigma)
173 {
174  return $"H + {2 * alpha * Math.Pow(beta, 2)}*{sigma}_{sigma_1}+{2 *
      Math.Pow(beta, 2) * beta}*{sigma}_{sigma_2} + {2 * alpha * beta *
      gamma}*{arrayOfSigma[2]}";
175 }
```

Результатом выполнения данного алгоритма будет две строки H^2 и H^3.

Заключение

В работе получены следующие основные результаты:

1. Изучены свойства базиса Паули
2. Исследована связь стандартного базиса и базиса Паули
3. Разработан алгоритм вычисления операторной экспоненты для трёх кубитов в базисе Паули на языке C#

Литература

[1] V. V. Nikonov, A. N. Tsirulev. *Pauli basis formalism in quantum computations*. Volume 8, No 3, pp. 1 – 14, 2020.
(*doi:10.26456/mmg/2020-831*)

[2] Michael A. Nielsen, Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press The Edinburgh Building, Cambridge CB2 8RU, UK. 2010
(*mmrc:201702/W020170224608149940643*)

[3] David J. Griffiths. *Introduction to Quantum Mechanics*. Prentice Hall, Upper Saddle River, New Jersey 07458, 1995
(*fisica:mecanica-quantica/Griffiths-IntroductionToQuantumMechanics*)

[4] Herbert Goldstein. *Classical Mechanics*. Universitat De Barcelona. Biblioteca de Fizica i Quimica, 2000
(*poincare:zarkom/BookMechanicsGoldsteinClassicalMechanicsOptimized*)

[5] Eleanor Rieffel, Wolfgang Polak. *Quantum Computing: A Gentle Introduction*. The MIT press. Cambridge, Massachusetts. London, England, 2011
(*mmrc:201702/W020170224608150244118*)

[6] Jeonghwan Ahn, Seoung-Hun Kang, Mao-Hua Du. *Procedures for assessing the stability of proposed topological materials*. Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA. 2023
(*arXiv:* 2305.09308)

Приложение C#

```
class Program
{
 static string sigma = "\u03C3";
 static void Main()
 {
  Console.OutputEncoding = Encoding.UTF8;
  Console.Write(" Введите коэффициент первого слагаемого гамильтониана: ");
  string a = Console.ReadLine()!;
  Console.Write(" Введите индексы первого оператора Паули: ");
  string sigma_1 = Console.ReadLine()!;
  Console.Write(" Введите коэффициент второго слагаемого гамильтониана: ");
  string b = Console.ReadLine()!;
  Console.Write(" Введите индексы второго оператора Паули: ");
  string sigma_2 = Console.ReadLine()!;
  Console.Write(" Введите коэффициент третьего слагаемого гамильтониана: ");
  string c = Console.ReadLine()!;
  Console.Write(" Введите индексы третьего оператора Паули: ");
  string sigma_3 = Console.ReadLine()!;
  Console.WriteLine($" Введённый вами гамильтониан: H = {a}*{sigma}_{sigma_1
   } + {b}*{sigma}_{sigma_2} + {c}*{sigma}_{sigma_3}");
  string[] sigma_array = Hamiltonian.countingOperators(sigma_1, sigma_2,
    sigma_3);

  Console.WriteLine($"H^2 = {Hamiltonian.
   calculateSecondDegreeOfHamiltonian(Convert.ToDouble(a), Convert.
   ToDouble(b), sigma_array)}");
  Console.WriteLine($"H^3 = {Hamiltonian.
   calculateThirdDegreeOfHamiltonian(Convert.ToDouble(a), Convert.
   ToDouble(b), Convert.ToDouble(c), sigma_1, sigma_2,sigma_array)}");
 }
 class Calculation
 {
  public static string PauliMatrices(string K, string L)
  {
   char[] K1 = K.ToCharArray();
   char[] L1 = L.ToCharArray();
   string M = "";
   int a, b, c;
   if (K1.Length == L1.Length)
```

```
  {
   for (int i = 0; i < K1.Length; i++)
   {
    a = K1[i] - '0';
    b = L1[i] - '0';
    c = Calculation.Operations(a, b);
    M = M + Convert.ToString(c);
   }
   return M;
  }
  else
  {
   Console.WriteLine(" Ошибка размерности кубитов двух операторов");
   return "";
  };
 }
 public static int Operations(int a, int b)
 {
  int c;
  if (a == b) c = 0;
  else if (a == 0) c = b;
  else if (b == 0) c = a;
  else c = 6 / (a * b);
  return c;
 }
 public static string Factors(string K, string L)
 {
  char[] K1 = K.ToCharArray();
  char[] L1 = L.ToCharArray();
  int a, b, skl, p = 0, m = 0;
  string w = "";
  if (K1.Length == L1.Length)
  {
   for (int i = 0; i < K1.Length; i++)
   {
    a = K1[i] - '0';
    b = L1[i] - '0';
    if (a - b == 1 || a - b == -2) m++;
    if (b - a == 1 || b - a == -2) p++;
    p = p + m;
   }
   if (p % 4 == 0 || p % 4 == 1)
   {
    skl = Convert.ToInt32(Math.Pow(-1, m));
    if (skl == 1) w = "";
    else w = "-1";
   }
   if (p % 4 == 2 || p % 4 == 3)
   {
    skl = Convert.ToInt32(Math.Pow(-1, m + 1));
    if (skl == 1) w = "i";
```

```
85      else w = "-i";
86     }
87    }
88    return w;
89   }
90  }
91  class Hamiltonian
92  {
93   public static string[] countingOperators(string sigma_1, string
     sigma_2, string sigma_3)
94   {
95    string sigma_12 = Calculation.Factors(sigma_1, sigma_2) + $"*{sigma}_
     " + Calculation.PauliMatrices(sigma_1, sigma_2);
96    string sigma_23 = Calculation.Factors(sigma_2, sigma_3) + $"*{sigma}_
     " + Calculation.PauliMatrices(sigma_2, sigma_3);
97    string sigma_12_indexes = sigma_12.Split("_")[sigma_12.Split("_").
     Length - 1];
98    string sigma_123_draft = Calculation.Factors(sigma_12_indexes,
     sigma_3) + $"*{sigma}_" + Calculation.PauliMatrices(sigma_12_indexes,
      sigma_3);
99    string sigma_123;
100
101   if (sigma_12.StartsWith("-1"))
102   {
103    if (sigma_123_draft.StartsWith("-i"))
104    {
105     sigma_123 = $"i*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft
     .Split("_").Length - 1]}";
106    }
107    else if (sigma_123_draft.StartsWith("i"))
108    {
109     sigma_123 = $"-i*{sigma}_{sigma_123_draft.Split("_")[
     sigma_123_draft.Split("_").Length - 1]}";
110    }
111    else if (sigma_123_draft.StartsWith("-1"))
112    {
113     sigma_123 = $"{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
     Split("_").Length - 1]}";
114    }
115    else
116    {
117     sigma_123 = $"-1*{sigma}_{sigma_123_draft.Split("_")[
     sigma_123_draft.Split("_").Length - 1]}";
118    }
119   }
120   else if (sigma_12.StartsWith("-i"))
121   {
122    if (sigma_123_draft.StartsWith("-i"))
123    {
124     sigma_123 = $"-1*{sigma}_{sigma_123_draft.Split("_")[
     sigma_123_draft.Split("_").Length - 1]}";
```

```
125     }
126     else if (sigma_123_draft.StartsWith("i"))
127     {
128       sigma_123 = $"{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
    Split("_").Length - 1]}";
129     }
130     else if (sigma_123_draft.StartsWith("-1"))
131     {
132       sigma_123 = $"i*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft
    .Split("_").Length - 1]}";
133     }
134     else
135     {
136       sigma_123 = $"-i*{sigma}_{sigma_123_draft.Split("_")[
    sigma_123_draft.Split("_").Length - 1]}";
137     }
138   }
139   else if (sigma_12.StartsWith("i"))
140   {
141     if (sigma_123_draft.StartsWith("-i"))
142     {
143       sigma_123 = $"{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft.
    Split("_").Length - 1]}";
144     }
145     else if (sigma_123_draft.StartsWith("i"))
146     {
147       sigma_123 = $"-1*{sigma}_{sigma_123_draft.Split("_")[
    sigma_123_draft.Split("_").Length - 1]}";
148     }
149     else if (sigma_123_draft.StartsWith("-1"))
150     {
151       sigma_123 = $"-i*{sigma}_{sigma_123_draft.Split("_")[
    sigma_123_draft.Split("_").Length - 1]}";
152     }
153     else
154     {
155       sigma_123 = $"i*{sigma}_{sigma_123_draft.Split("_")[sigma_123_draft
    .Split("_").Length - 1]}";
156     }
157   }
158   else
159   {
160     sigma_123 = sigma_123_draft;
161   }
162   string[] sigma_array = new string[] { sigma_12, sigma_23, sigma_123
    };
163   return sigma_array;
164 }
165
166 public static string calculateSecondDegreeOfHamiltonian(double alpha,
    double beta, string[] arrayOfSigma)
```

```
  {
   return $"{sigma}_000 + {2 * alpha* beta}*{arrayOfSigma[0]}";
  }
  public static string calculateThirdDegreeOfHamiltonian(double alpha,
   double beta, double gamma, string sigma_1, string sigma_2, string[]
   arrayOfSigma)
  {
   return $"H + {2 * alpha * Math.Pow(beta, 2)}*{sigma}_{sigma_1}+{2 *
   Math.Pow(beta, 2) * beta}*{sigma}_{sigma_2} + {2 * alpha * beta *
   gamma}*{arrayOfSigma[2]}";
  }
 }
}
```