Минобрнауки России

Федеральное государственное автономное образовательное учреждение высшего образования
«Омский государственный университет им. Ф.М. Достоевского»

Физический факультет

Квазипособие студентам от студентов

Суперпозиция успеха и отчаяния: Как выжить в мире квантовых вычислений

Составители:

Студент Журавлева Анастасия

Студент Коцур Павел

Студент Бахтин Артур

Омск – 2025

Содержание

Введение

Настоящие квазипособие представляет собой вводный курс по основам квантовых вычислений, разработанный студенческим коллективом с целью облегчения процесса освоения данного перспективного направления компьютерных наук для учащихся младших курсов. Она также является вводным пособием, разработанным для обеспечения базового понимания квантовых вычислений. Её основная цель – заложить прочный фундамент знаний, необходимый для успешного изучения более сложных материалов, в том числе научных трудов и учебной литературы, предназначенных для продвинутого уровня. Материал данного пособия является результатом анализа и систематизации знаний, полученных авторами в ходе изучения квантовой информатики.

Квантовые вычисления, представляющие собой принципиально новый подход к обработке информации, обладают потенциалом для революционного изменения существующих алгоритмов и технологий. Область применения квантовых вычислений охватывает широкий спектр задач, включая, но не ограничиваясь: разработкой новых лекарственных препаратов и материалов, усовершенствование методов искусственного интеллекта, а также развитие криптографических систем.

Предлагаемое квазипособие ориентировано на студентов, начинающих знакомство с квантовыми вычислениями. Материал изложен в доступной форме, позволяющей усвоить основные принципы, избегая излишней математической детализации, при этом сохраняя научную корректность. В квазипособии рассматриваются ключевые концепции, такие как квантовые биты (кубиты), принципы суперпозиции и квантовой запутанности, а также базовые квантовые алгоритмы и архитектуры. Основной акцент сделан на предоставлении студентам понимания фундаментальных основ, необходимых для дальнейшего обучения и работы с более сложными источниками информации.

Мы надеемся, что это квазипособие станет для вас эффективным средством подготовки к изучению специализированной литературы, облегчит понимание сложных концепций и терминологии, а также позволит вам чувствовать себя увереннее при работе с оригинальными научными публикациями и учебными пособиями по квантовым вычислениям.

Авторы выражают надежду на вашу успешную работу и желают вам плодотворного обучения.

С уважением,
Авторы методички

1 Основные положения

В данной главе будут рассмотрены основные понятия для квантовых вычислений.

1.1 Кубит

В классическом понимании информация измеряется битами, которые можно представить как элементарные триггерные системы, принимающие одно из двух двоичных значений: 0 или 1. Биты могут подвергаться изменениям с помощью классических логических операций. В каждый момент времени состояние бита подлежит строгому детерминизму. Однако такая бинарная единица хранения информации не подходит для описания состояния квантовых систем. Поэтому в теории квантовых вычислений вместо классических битов используются кубиты (квантовые биты), для описания состояния которых применяются волновые функции. Таким образом, главное различие между битами и кубитами заключается в том, что кубиты не являются детерминированными и могут находиться в суперпозиции своих логических состояний [1]. Как следствие можно записать:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,. \tag{1.1}$$

Можно переписать формулу (1.1) как:

$$|\psi\rangle = e^{i\gamma}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right). \tag{1.2}$$

Это уравнение задаёт положение на сфере Блоха.

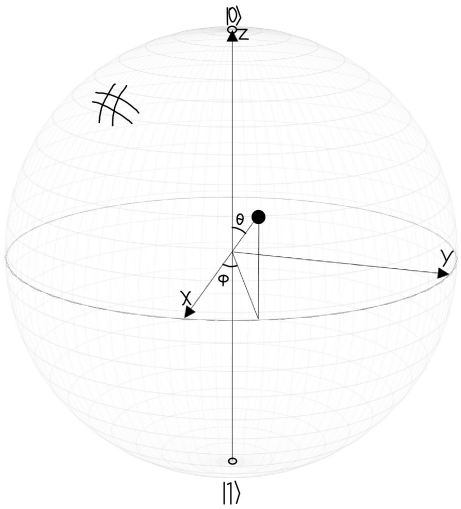

Рис. 1: Представление кубита в виде сферы Блоха

В этой сфере континуальное множество точек, как следствие может показаться, что можно закодировать информацию любого объёма. В реальности кубит коллапсирует из

суперпозиции в определённое состояние, соответствующее результату измерения.

1.2 Квантовое преобразование Фурье

В ортонормированном базисе $|0\rangle, ..., |N--1\rangle$ квантовое преобразование Фурье определяется как линейный оператор, действующий на базисные состояния по формуле [1]:

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N--1} e^{2\pi ijk/N} |k\rangle . \tag{1.3}$$

Аналогичным образом действие этого оператора на произвольное состояние можно записать в виде:

$$\sum_{j=0}^{N--1} x_j |j\rangle \longrightarrow \sum_{k=0}^{N--1} y_k |k\rangle , \tag{1.4}$$

где амплитуды y_k – дискретные преобразования Фурье амплитуд x_j. Этот оператор является унитарным, а значит может быть реализован на квантовом компьютере.

Будем считать, что $N = 2^n$, где n – целое число, и что базис $|0\rangle, ..., |2^n - -1\rangle$ есть вычислительный базис для n–кубитового квантового компьютера.
Можно представить квантовое преобразование Фурье в виде произведения:

$$|j_1, ..., j_n\rangle \to \frac{(|0\rangle + e^{2\pi i 0.j_n} |1\rangle)(|0\rangle + e^{2\pi i 0.j_{n--1}j_n} |1\rangle)...(|0\rangle + e^{2\pi i 0.j_1...j_n} |1\rangle)}{2^{n/2}}. \tag{1.5}$$

Это представление можно считать определением квантового преобразования Фурье. Данное представление позволяет построить квантовую схему, эффективно вычисляющую преобразование Фурье, и это доказывает унитарность такого преобразования. Указанное представление позволяет понять алгоритмы, основанные на квантовом преобразовании Фурье.

Покажем эквивалентность формулы (1.5) и определения (1.3):

$$\begin{aligned}
|j\rangle \to \frac{1}{2^{n/2}} \sum_{k=0}^{2^n--1} e^{2\pi ijk/2^n} |k\rangle = \\
= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} ... \sum_{k_n=0}^{1} e^{2\pi ij \sum_{l=1}^{n} k_l 2^{--l}} |k_1...k_n\rangle = \\
= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} ... \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi ijk_l 2^{--l}} |k_l\rangle = \\
= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi ijk_l 2^{--l}} |k_l\rangle \right] = \\
= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi ij 2^{--l}} |1\rangle \right] = \\
= |l_1, ..., j_n\rangle \to \frac{(|0\rangle + e^{2\pi i 0.j_n} |1\rangle)(|0\rangle + e^{2\pi i 0.j_{n--1}j_n} |1\rangle)...(|0\rangle + e^{2\pi i 0.j_1...j_n} |1\rangle)}{2^{n/2}}.
\end{aligned} \tag{1.6}$$

Рассмотрим вычисление квантового преобразования Фурье для входного состояния $|j_1...j_n\rangle$. Для этого введём унитарное преобразование:

$$R_k \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}. \tag{1.7}$$

Применяя элемент Адамара к первому биту, получим состояние:

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1} |1\rangle \right) |j_2...j_n\rangle . \tag{1.8}$$

Поскольку $e^{2\pi i 0.j_1}$ равно $--1$ при $j_1 = 1$ и $+1$ иначе. Применяя управляемое R_2, имеем:

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2} |1\rangle \right) |j_2...j_n\rangle . \tag{1.9}$$

Далее будем применять управляемое R_3, управляемое R_4 и т.д. до R_n (см. п. **??**), добавляя на каждом шаге по биту к фазе первого из векторов $|1\rangle$. По завершении этой процедуры придем к состоянию:

$$\frac{1}{2^{1/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 ... j_n} |1\rangle \right) |j_2...j_n\rangle . \tag{1.10}$$

Теперь используем ту же процедуру для второго кубита. Элемент Адамара переводит в состояние:

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 ... j_n} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0.j_2} |1\rangle \right) |j_3...j_n\rangle , \tag{1.11}$$

Применение управляемых $R_2, ..., R_{n--1}$ даст состояние вида:

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 ... j_n} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0.j_2 ... j_n} |1\rangle \right) |j_3...j_n\rangle , \tag{1.12}$$

Продолжая выполнять те же действия для остальных кубитов, придем в конце концов к состоянию:

$$\frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_1 j_2 ... j_n} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0.j_2 ... j_n} |1\rangle \right) ... \left(|0\rangle + e^{2\pi i 0.j_n} |1\rangle \right) . \tag{1.13}$$

Теперь с помощью операции обмена обратим порядок кубитов. После этого получим состояние:

$$\left(|0\rangle + e^{2\pi i 0.j_n} |1\rangle \right) \frac{1}{2^{2/2}} \left(|0\rangle + e^{2\pi i 0.j_{n--1} j_n} |1\rangle \right) ... \left(|0\rangle + e^{2\pi i 0.j_2 ... j_n} |1\rangle \right) . \tag{1.14}$$

2 Ошибки

В квантовых вычислениях, подобно классическим, информация подвержена ошибкам. В этой главе мы рассмотрим основные проблемы, связанные с возникновением ошибок в квантовых вычислениях, и почему методы, применяемые в классических вычислениях, здесь не всегда работают. Ключевым препятствием является декогерентизация – взаимодействие квантового компьютера с окружающей средой. Это взаимодействие приводит к потере квантовой информации, подобно тому, как контакт с воздухом разрушает хрупкий предмет.

2.1 Основные проблемы

2.1.1 Декогерентизация

Контакт квантового компьютера с окружающей средой (декогерентизация) является основной причиной ошибок, разрушающих квантовую информацию [4]. Представьте себе кубит (квантовый бит) как монету, которая может находиться в состоянии "орла"($|0\rangle$) или "решки"($|1\rangle$), а также в суперпозиции этих состояний (одновременно и орел, и решка). Взаимодействие с внешним миром, например, из-за тепла или электромагнитных помех, может "сбить" монету, преждевременно определив ее состояние, что эквивалентно потере квантовой информации.

2.1.2 Неидеальность операций

Декогерентизация – не единственная проблема. Даже идеальное изолирование от окружающей среды не гарантирует отсутствия ошибок. Представьте себе, что мы хотим применить определенную операцию U_0 к двум кубитам. В реальности эта операция никогда не будет выполнена идеально. Фактическое преобразование будет немного отличаться от запланированного, что можно выразить как небольшую погрешность, порядка ϵ. Применение операций (квантовых вентилей) многократно, например, около $\frac{1}{\epsilon}$ раз, приведет к накоплению этих ошибок, что сделает вычисления неточными или даже невозможными.

2.1.3 Почему классические методы не работают

Стандартные методы борьбы с ошибками, используемые в классических компьютерах, например, охлаждение, неэффективны в квантовых вычислениях. Охлаждение может повысить точность классической информации, но способно разрушить квантовую информацию, вызвав декогерентизацию. Чтобы сохранить работоспособность квантового компьютера, необходимы методы коррекции ошибок, адаптированные к квантовым особенностям.

2.2 Специфические трудности коррекции ошибок в квантовых вычислениях

Квантовая информация более хрупка, чем классическая, и требует специальных методов коррекции ошибок. Рассмотрим основные проблемы:

2.2.1 Фазовые ошибки

В дополнение к обычным ошибкам инвертирования битов ($|0\rangle \to |1\rangle$ и $|1\rangle \to |0\rangle$), в квантовых вычислениях возникают фазовые ошибки. Эти ошибки изменяют не только значение бита, но и фазу его состояния. Примером фазовой ошибки может быть:

$$|0\rangle \to |0\rangle \tag{2.15}$$
$$|1\rangle \to - - |1\rangle \tag{2.16}$$

Фазовая ошибка особенно опасна, поскольку может перевести состояние в ортогональное ему (математически "противоположное"). Классические методы кодирования, которые хорошо справляются с инверсией битов, не обеспечивают защиты от фазовых ошибок.

2.2.2 Малые ошибки

Если кубит находится в суперпозиции:

$$\alpha |0\rangle + \beta |1\rangle , \tag{2.17}$$

где α и β – комплексные числа, описывающие вероятность нахождения кубита в состояниях $|0\rangle$ и $|1\rangle$ соответственно, ошибка может изменить значения α и β, пусть и незначительно (на величину порядка ϵ). Однако при многократном применении операций эти небольшие ошибки могут накапливаться и в итоге привести к значительным искажениям. Классические же методы коррекции ошибок, как правило, предназначены для исправления больших, а не малых, ошибок.

2.2.3 Измерение – причина возмущения

В классических системах для обнаружения и исправления ошибок часто используется принцип голосования (например, если три копии бита, и две из них равны, то считаем значение верным). Для обнаружения и исправления ошибок в квантовых вычислениях требуется измерять кубиты в коде. Однако, измерение кубита в квантовой системе неизбежно возмущает его состояние, то есть разрушает или изменяет информацию, которую он хранит. Это фундаментальное ограничение квантовой механики.

2.3 Обобщение

Преодоление проблем, связанных с ошибками в квантовых вычислениях, является одним из самых сложных и важных направлений исследований в этой области. Разработка методов квантовой коррекции ошибок – ключевой шаг на пути к созданию надежных и работоспособных квантовых компьютеров.

3 Квантовые вычисления и квантовые компьютеры

В данной главе будут рассматриваться квантовые вычисления, квантовые алгоритмы, устройство и применение квантовых компьютеров, а также знакомство с синтаксисом языка для квантового компьютера *OPENQASM* 2.0 на примере генератора случайных чисел.

3.1 Квантовые вычисления. Однокубитовые и многокубитовые гейты. Состояние Белла

Изменения, происходящие с квантовым состоянием, можно описать на языке квантовых вычислений. Аналогично классическому компьютеру, квантовый компьютер строится из квантовых схем, состоящих из проводов и элементарных квантовых элементов, которые позволяют передавать квантовую информацию и управлять ею. Вычисления реализуют алгоритмически некоторую функцию. Алгоритм имеет множество входных данных и соответствующие им выходные данные.

Любое изменение состояния квантовой системы выражается действием унитарного оператора (гейта). Алгоритм для квантового компьютера представляет собой последователь-

ное применение различных унитарных операторов к некоторому вектору состояния. Оператор U унитарен, если его сопряженный оператор совпадает с обратным:

$$UU^* = U^*U = I, \tag{3.18}$$

где U^* – эрмитово сопряженный оператор, I – единичный оператор.

- **Оператор Адамара**
  Оператор Адамара H является одним из самых полезных квантовых элементов и определяется следующим образом:

$$H = H^* = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & --1 \end{pmatrix} \tag{3.19}$$

  Когда оператор Адамара применяется к состоянию кубита, он переводит его в суперпозицию состояний:

$$\begin{aligned} |0\rangle &\to \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle\right), \\ |1\rangle &\to \frac{1}{\sqrt{2}} \left(|0\rangle - -|1\rangle\right). \end{aligned} \tag{3.20}$$

- **Гейт** X
  Гейт X – квантовый аналог классического вентиля NOT.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{3.21}$$

  Применим гейт X к состояниям $|0\rangle$ и $|1\rangle$:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle, \tag{3.22}$$

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle. \tag{3.23}$$

- **Гейт** $CNOT$
  Гейт $CNOT$ применяется для системы из двух кубитов, и предназначен для четырехмерного пространства:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{3.24}$$

  Гейт $CNOT$ действует как условный NOT, а именно если первый кубит (управляющий) равняется единице, то на второй (управляемый) действует гейт NOT. Если первый равняется нулю, то на второй ничего не действует. Базис после применения данного гейта выглядит следующим образом:

$$\begin{aligned} CNOT\ |00\rangle &= |00\rangle, \\ CNOT\ |01\rangle &= |01\rangle, \\ CNOT\ |10\rangle &= |11\rangle, \\ CNOT\ |11\rangle &= |10\rangle. \end{aligned} \tag{3.25}$$

Рассмотрим ситуацию, в которой содержится гейт Адамара с последующим применением гейта $CNOT$. Сначала преобразование Адамара переводит верхний кубит в состояние суперпозиции, затем эта суперпозиция поступает на управляющий вход $CNOT$, и управляемый кубит меняет значение, если значение управляющего кубита равно 1. Полученные выходные состояния называются состояниями Белла, или ЭПР – состояниями, в честь тех людей, которые первыми указали на странные свойства подобных состояний – Белла, Эйнштейна, Подольского, Розена [1]:

$$\begin{aligned} |\beta_{00}\rangle &= \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \\ |\beta_{01}\rangle &= \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \\ |\beta_{10}\rangle &= \frac{|00\rangle - -|11\rangle}{\sqrt{2}}, \\ |\beta_{11}\rangle &= \frac{|01\rangle - -|10\rangle}{\sqrt{2}}. \end{aligned} \tag{3.26}$$

Состояния Белла демонстрируют квантовую запутанность. Если две частицы находятся в состоянии Белла, их состояния нельзя описать независимо. Измерение одной частицы мгновенно определяет состояние другой, даже если они находятся на огромном расстоянии (ЭПР–парадокс).

3.2 Квантовый язык программирования *OPENQASM* 2.0. Генератор случайных чисел на квантовом компьютере

Квантовые языки программирования предназначены для описания квантовых алгоритмов, моделирования квантовых схем и взаимодействия с реальными квантовыми компьютерами. Язык *OPENQASM (Quantum Assembly Language)* 2.0 [5] служит для описания квантовых алгоритмов в императивном стиле. Это один из ключевых низкоуровневых языков, используемых в квантовых вычислениях. Он используется в квантовых фреймворках (например, *Qiskit* преобразует *Python*–код в *OPENQASM* 2.0 перед отправкой на квантовый компьютер), квантовых симуляторах *IBM Quantum Lab*, эмуляторе квантового компьютера «Телеквант» [6].

Программа начинается со строки *OPENQASM* 2.0, далее следует объявление квантовых и классических переменных которые будут использоваться как ячейки хранения информации. После объявления переменных следует тело программы, в котором к квантовым переменным применяются квантовые операции и квантовые измерения с последующей записью результата измерения в классическую переменную. Квантовый регистр используется для генерации случайного бита, а классический – для хранения результата измерения. По завершению программы результатом ее работы являются данные, записанные в классические переменные. Состояния квантовых переменных не доступны для пользователя в силу фундаментальных принципов квантовой механики. Извлечь информацию из квантовых переменных можно только средствами квантовых измерений. Все объявленные классические переменные инициализируются 0, все объявленные квантовые переменные инициализируются состоянием $|0\rangle$. Время жизни всех квантовых и классических переменных совпадает со временем жизни программы [6]. Как пример рассмотрим

генератор случайных чисел, написанный на языке *OPENQASM* 2.0 для эмулятора квантового компьютера «Телеквант» [6]. В рамках реализации генератора случайных чисел для 10 кубитов используется следующий алгоритм:

1. Инициализация
   Создается квантовый регистр q[10] из 10 кубитов и классический регистр c[10] для хранения результатов измерений.

2. Применение гейта Адамара
   После преобразования Адамара состояния кубитов распределятся с равной вероятностью.

3. Измерение состояния кубитов
   На этом этапе производится измерение состояния кубитов с сохранением результатов. Полученные значения интерпретируются как случайные биты.

Программный код для измерения 10 кубитов:

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[10];
creg c[10];
h q[0];
⋮
h q[9];
measure q[0] → c[0];
⋮
measure q[9] → c[9];
```

Программа начинается со строки *OPENQASM* 2.0, где 2.0 – используемая версия языка. Далее подключается стандартная библиотека квантовых операций, включающая в себя необходимый для генерации случайных чисел гейт Адамара h. Далее объявляется квантовый qreg и классический creg регистры. Квантовый регистр используется для генерации случайного бита, а классический – для хранения результата измерения. После следует применения гейта Адамара h, который переводит кубит из состояния $|0\rangle$ в суперпозицию. Это означает, что при измерении кубит с равной вероятностью окажется в состоянии $|0\rangle$ или $|1\rangle$. Все объявленные классические переменные инициализируются нулем, все объявленные квантовые переменные инициализируются состоянием $|0\rangle$. Последняя строка измеряет кубит $q[0]$ и сохраняет результат в классический бит $c[0]$. Результат измерения будет случайным: 0 или 1.

3.3 Квантовый компьютер

Основной интерес к теории квантовых вычислений и квантовой информации связан с тем, что необходимые для обработки алгоритмов устройства действительно могут быть реализованы, а не просто являются математическими моделями. Экспериментальная реализация квантовых схем является очень сложным процессом, в котором должны обязательно одновременно выполняться несколько принципов: чтобы сохранять квантовые состояния,

квантовый компьютер должен быть хорошо изолирован от окружающей среды, и его кубиты должны быть достаточно доступны, чтобы можно было выполнять вычисления и считывать результаты [1]. Трудности в реализации связаны с тем, что чаще всего можно удовлетворить только часть требований. Например, монета является хорошим представлением бита, но плохим представлением кубита, поскольку находится в суперпозиции очень короткое время. При оценке качества и перспективности системы ключевую роль играет понятие квантового шума.

4 Квантовая информация

В этой главе мы погрузимся в основы квантовой информации, изучая ключевые концепции и их удивительные приложения. Во многом затронутые темы будут носить фундаментальный характер. Настоятельно рекомендуется перевести дух и только после этого окунуться в изучение квантовой информации с новыми силами и зарядом бодрости.

4.1 Квантовая информация в квантовых каналах

В области квантовых вычислений распространены два разных значения для термина квантовая информация. Во–первых, он применяется в качестве общего названия для всех видов деятельности, связанных с обработкой информации на основе квантовой механики. Во втором значении этот термин более специализирован: он относится к изучению элементарных задач по обработке квантовой информации [1].

В классической теории информации доказываются теоремы Шеннона о кодировании применительно к сжатию и развертыванию квантовых состояний. Рассмотрим различные квантовые аналоги этих теорем. Для начала выберем за квантовый источник информации квантовые состояния $|\psi_j\rangle$, описываемые набором вероятности p_j. Таким образом каждое обращение независимо и даёт с вероятностью p_j состояние $|\psi_j\rangle$. Здесь возникает двоякая ситуация. К примеру мы имеем состояния $|0\rangle$ и $|1\rangle$, описываемые вероятностями p и $1--p$, соответственно, тогда можно применить теорему Шеннона точно так же как и для классического случая. Мы получим что необходимое число кубитов будет $H(p, 1--p)$, где H – энтропия Шеннона. ($H(p_j) = --\sum_j p_j \log(p_j)$). Но если мы будем иметь в каком–то из случаев, допустим, состояние Белла, использовать классический подход в лоб нельзя, так как мы не различаем состояния Белла и, условно, $|0\rangle$.

Даже в этом случае можно произвести сжатие, только оно перестаёт быть безошибочным в том смысле, что квантовые состояния на выходе источника могут слегка искажаться процедурой сжатия–развертывания. Чтобы охарактеризовать искажения вводят понятие совпадения (fidelity), которое определяет среднее искажение, вносимое этим алгоритмом. Идея состоит в том, что сжатые данные должны восстанавливаться с очень большой точностью.

Теорема Шумахера о кодировании для канала без шума устанавливает, какое количество ресурсов требуется для сжатия квантовых данных при условии, что источник можно восстановить с точностью близкой к 1, что означает отсутствие ошибок. Теорема в нашем определении источника информации сводится к утверждению о том, что возможно сжатие не более, чем до классического предела $H(p_j)$. Здесь кроется одна деталь – это верно для ортогональных $|\psi_j\rangle$. В общем случае надо вводить энтропию фон Неймана, которая совпадает с энтропией Шеннона только в случае ортогональности. В противном случае

энтропия фон Неймана строго меньше энтропии Шеннона. Так для примера с состоянием Белла и $|0\rangle$ понадобится меньше число кубитов, чем $H(p, 1 - -p)$.

4.2 Квантовая различимость

В квантовомеханическом случае не всегда можно различить произвольные состояния, например возникает неразличимость неортогональных квантовых состояний. Тем самым квантовое состояние содержит скрытую информацию, недоступную для измерения. Неразличимость на «пальцах» доказывается от обратного, если бы мы могли точно различить состояния неортогональной системы, то передача информации была бы больше скорости света, что невозможно по физическим причинам [1].

4.3 Квантовый параллелизм

Квантовый параллелизм – это фундаментальное свойство квантовых вычислений, позволяющее выполнять вычисления над всеми возможными входными данными одновременно. В отличие от классических компьютеров, которые обрабатывают данные последовательно, квантовые компьютеры используют суперпозицию квантовых битов (кубитов), чтобы находиться во всех возможных состояниях одновременно. Это позволяет квантовому алгоритму одновременно исследовать все возможные решения задачи, что значительно ускоряет вычисления для определенного класса задач. Однако, важно отметить, что квантовый параллелизм не означает мгновенного получения всех результатов; информация о всех возможных исходах содержится в суперпозиции, и извлечение нужного результата требует процесса измерения, который разрушает суперпозицию. Тем не менее, эта одновременная обработка информации является ключевым фактором, обеспечивающим превосходство квантовых компьютеров над классическими в решении определенных типов задач.

4.4 Плотное кодирование

Плотное кодирование использует зацепленные состояния Белла, например, $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Алиса, имеющая один кубит из зацепленной пары, применяет к нему один из четырех унитарных операторов $\{I, \sigma_x, \sigma_z, \sigma_x\sigma_z\}$, где I – единичный оператор, σ_x – оператор Паули X, σ_z – оператор Паули Z. Каждый оператор соответствует одному из четырех возможных двухбитных состояний $(00, 01, 10, 11)$. Затем Алиса отправляет свой кубит Бобу. Боб, выполнив совместное измерение на своей и полученной от Алисы паре кубитов в базисе Белла, определяет, какой оператор был применен Алисой, восстанавливая таким образом два классических бита информации.

4.5 Квантовая телепортация

Квантовая телепортация – это процесс передачи квантового состояния от одного кубита (отправителя) к другому (получателя) с использованием зацепленных частиц и классической связи. Важно отметить, что сама *материя* не перемещается, а лишь квантовое состояние.

4.5.1 Схема телепортации

Для телепортации требуется:

- **Отправитель (Алиса)**: Кубит в неизвестном состоянии $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, где α и β – комплексные амплитуды.
- **Получатель (Боб)**: Кубит в известном состоянии, например, $|0\rangle$.
- **Зацепленная пара кубитов**: Пара зацепленных кубитов, по одному у Алисы и Боба, в состоянии Белла $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

4.5.2 Процесс

1. **Совместное измерение**: Алиса выполняет совместное измерение своего кубита $|\psi\rangle$ и своего кубита из зацепленной пары в базисе Белла. Результат измерения может быть одним из четырех состояний Белла: $|\Phi^+\rangle$, $|\Phi^-\!-\rangle$, $|\Psi^+\rangle$, $|\Psi^-\!-\rangle$.
2. **Классическая коммуникация**: Алиса передает результаты своего измерения (два классических бита) Бобу по классическому каналу связи.
3. **Восстановление состояния**: В зависимости от полученных от Алисы битов, Боб выполняет соответствующую унитарную операцию над своим кубитом из зацепленной пары. Это восстанавливает исходное состояние $|\psi\rangle$ на кубите Боба.

4.5.3 Математическое описание

Состояние всей системы до измерения:

$$|\psi\rangle \otimes \left|\phi^+\right\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes \frac{1}{\sqrt{|00\rangle + |11\rangle}} \tag{4.27}$$

В зависимости от результатов измерения Алисы, Боб применяет следующие операции:

- $|\Phi^+\rangle$: I (единичный оператор)
- $|\Phi^{--}\rangle$: σ_z (оператор Паули Z)
- $|\Psi^+\rangle$: σ_x (оператор Паули X)
- $|\Psi^{--}\rangle$: $\sigma_x\sigma_z$

4.6 Квантовая криптография

4.6.1 Распределение квантового ЭПР–ключа

У каждого есть секреты. Боб и Алиса хотят передать информацию в совершенном секрете, проблема в том, что их подруга Ева хочет всё подслушать. Предположим Боб и Алиса не способны придумать классический шифр, который не взломала бы Ева. Есть единственный способ передать биты так, чтобы Ева их не узнала, но предположим Алиса и Боб хотят установить общий случайный ключ. Можно использовать протоколы открытых ключей, но Ева и здесь перехитрила наших героев, она обладает невероятно сильной вычислительной машиной. Но Боб и Алиса будут использовать квантовые ключи, тем самым смогут защититься от посягательств от Евы.

Предположим, что Алиса и Боб делят запас запутанных пар, приготовленных в состоянии $|\phi^+\rangle$. Чтобы приготовить известный только им ключ необходимо выполнить следующий протокол.

Для каждого кубита они измеряют σ_1 или σ_3. Это решение псевдослучайное. Затем они открыто объявляют о том, какие наблюдаемые были измерены, но не открывают полученные ими результаты. Кубиты, измеренные вдоль разных осей, получаются нескоррелированные, их отбрасывают. В тех случаях, где измерения выполнены вдоль одних и тех же осей, их результаты хотя и случайны, но идеально скоррелированны. Ева не может, измеряя свои кубиты, вычислить код. Чтобы убедиться в том, что их состояния скоррелированы, Алиса и Боб открывают часть ключа и сверяются. Если Ева попытается вмешаться в передачу, то Боб и Алиса выбросят ключи и создадут новый. Конечно же можно предположить ещё варианты. В реальности будут возникать помехи, которые можно устранить разными способами.

4.7 Невозможность клонирования

В классической физике копирование информации – тривиальная задача. Однако в квантовой механике это невозможно из-за принципа суперпозиции и теоремы о запрете клонирования. Эта теорема утверждает, что невозможно создать точную копию произвольного неизвестного квантового состояния. Попытка клонирования неизбежно приведет к нарушению принципов квантовой механики.

4.7.1 Формулировка теоремы

Рассмотрим два кубита: исходный кубит в неизвестном состоянии $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ и вспомогательный кубит, который изначально находится в известном состоянии $|0\rangle$. Цель – создать устройство (квантовый клоннер), которое преобразует состояние $|\psi\rangle \otimes |0\rangle$ в состояние $|\psi\rangle \otimes |\psi\rangle$.

Предположим, существует унитарный оператор U, который выполняет клонирование:

$$U(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle \tag{4.28}$$

Для доказательства невозможности клонирования воспользуемся методом от противного. Предположим, что такой оператор U существует. Рассмотрим два разных состояния: $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ и $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$. Если клонирование возможно, то:

$$U(|\psi_1\rangle \otimes |0\rangle) = |\psi_1\rangle \otimes |\psi_1\rangle \tag{4.29}$$

$$U(|\psi_2\rangle \otimes |0\rangle) = |\psi_2\rangle \otimes |\psi_2\rangle \tag{4.30}$$

Теперь вычислим скалярное произведение этих двух состояний:

$$\langle\psi_1 \otimes \psi_1|\psi_2 \otimes \psi_2\rangle = (\langle\psi_1|\psi_2\rangle)^2 \tag{4.31}$$

С другой стороны, поскольку U – унитарный оператор, он сохраняет скалярное произведение:

$$\langle\psi_1 \otimes \psi_1|\psi_2 \otimes \psi_2\rangle = \langle\psi_1 \otimes 0|\, U^\dagger U\, |\psi_2 \otimes 0\rangle = \langle\psi_1|\psi_2\rangle \tag{4.32}$$

Из этих двух уравнений следует:

$$(\langle\psi_1|\psi_2\rangle) = (\langle\psi_1|\psi_2\rangle)^2 \tag{4.33}$$

Это уравнение выполняется только в двух случаях: $\langle\psi_1|\psi_2\rangle = 0$ или $\langle\psi_1|\psi_2\rangle = 1$. Это означает, что состояния $|\psi_1\rangle$ и $|\psi_2\rangle$ либо ортогональны, либо идентичны. Однако это противоречит тому, что $|\psi_1\rangle$ и $|\psi_2\rangle$ – произвольные состояния. Следовательно, наше предположение о существовании унитарного оператора U неверно.

5 Практическая реализация

Это заключительная глава, направленная на практическую часть. Ниже представлен программный код на языке Python, использующий библиотеку Qiskit, который демонстрирует построение и применение (в данном случае, обратного) квантового преобразования Фурье. После ознакомления с примером кода будет дано подробное пояснение используемых компонентов Qiskit, а также базовых концепций языка Python, необходимых для понимания представленной реализации.

5.1 Квантовое преобразование Фурье

Программа:

```
 from qiskit import QuantumCircuit,transpile
 from qiskit.quantum_info import Kraus, SuperOp
 from qiskit.visualization import plot_histogram
 from qiskit.transpiler.preset_passmanagers import
generate_preset_pass_manager
 from qiskit_aer import Aer
 from qiskit.visualization import plot_histogram
 import matplotlib.pyplot as plt
 import numpy as np

 def qft(circuit, n):
 for q in range(n):
 circuit.h(q)
 for k in range(q + 1, n):
 circuit.cp(np.pi / (2 ** (k - q)), k, q)
 for i in range(n // 2):
 circuit.swap(i, n - i - 1)

 def iqft (circuit,n ) :
 for i in range ( n //2) :
 circuit . swap ( i , n - i -1)
 for q in range (n -1 , -1 , -1) :
 circuit.h(q)
 for k in range (q -1 , -1 , -1) :
 circuit . cp ( - np . pi / float (2**( q - k ) ) , k , q )
 n = int(input(" (n): "))
 n_shots = int(input(" (n_shots): "))
 shots=n_shots*1024
 qc = QuantumCircuit(n, n)
 iqft(qc,n)
 qft(qc, n)
 qc.measure(range(n),range(n))
 backend = Aer.get_backend('qasm_simulator')
 transpiled_qc = transpile(qc, backend)
 job = backend.run(transpiled_qc , shots=shots)
```

```
    result = job.result()
    counts = result.get_counts(transpiled_qc)
    print("Result:")
    print(counts)
    print('\n')
    title = f'implementation for {n} qubits for shots {shots} '
    figsize = (5,5)
    color = ['crimson']
    fig0 = plot_histogram(counts, title=title, figsize=figsize, color=color)
    fig0.show()

```

5.2 Ознакомление с языком

Модуль Qiskit предназначен для работы с квантовыми компьютерами на уровне расширенных квантовых схем, операторов и примитивов. Некоторые компоненты модуля и их функции:

- Terra – Предоставляет инструменты для создания квантовых схем на уровне квантового машинного кода или близко к нему, позволяет оптимизировать квантовые схемы для устройства, управлять пакетами заданий и запускать их на квантовых устройствах и симуляторах с удалённым доступом.
- Aer – Предоставляет высокопроизводительные симуляторы квантовых вычислений с реалистичными моделями шума. Симуляторы могут имитировать эффекты шума для простых и сложных моделей шума.
- Модуль в Python – это отдельный файл с кодом на Python, который содержит функции, классы и переменные. Модуль имеет расширение .py, а его имя соответствует имени файла (позволяет структурировать код и делить его на мелкие, легко управляемые части).
- Библиотека – это набор файлов, модулей, классов, функций, в которых реализован весь функционал решения конкретной проблемы. Библиотеки обычно распространяются и устанавливаются отдельно, в то время как модули могут быть созданы и использованы в собственной кодовой базе. Для установки библиотеки используется менеджер пакетов `pip`. (`pip install` имя_библиотеки)
- `from math import` – выборочный импорт атрибутов из модуля math.
- `pow` – это встроенная функция Python, которая позволяет возводить числа в степень.
- Язык Python чувствителен к регистру, поэтому переменные и функции регистрозависимы: их имена должны точно соблюдать верхний и нижний регистр. Регистр в Python – это свойство строк, которое означает различие между прописными (большими) и строчными (маленькими) буквами (которое позволяет изменять порядок символов в строке).
- QuantumRegister – это коллекция кубитов, используемая для хранения квантовой информации.

- ClassicalRegister – используется для создания классического регистра с определённым количеством битов.
- В пакете Qiskit для квантовых вычислений на языке Python классический регистр – это структура, которая служит для записи и отображения результата в эмуляторе, так как просчёт идёт на квантовом регистре.
- Обычно в схемах используется комбинация классических и квантовых регистров: квантовые регистры применяют для выполнения квантово–механических операций с кубитами, а классические – для выполнения классических операций с полученными измерениями (часть классического управления в рамках алгоритма)
- QuantumCircuit в Python – это модуль библиотеки Qiskit, который позволяет создавать и манипулировать квантовыми схемами.
- Квантовая схема представляет собой набор квантовых операций, называемых квантовыми вентилями, которые применяются к кубитам.
- BasicAer – модуль в программе Qiskit, предоставляющий доступ к некоторым базовым симуляторам. Эти симуляторы подходят в основном для учебных целей, а не для высокопроизводительных задач.
- Симуляторы в Qiskit – это высокопроизводительные инструменты для моделирования квантовых схем. Они позволяют понять поведение запрограммированных квантовых алгоритмов перед их выполнением на реальном оборудовании.
- Среди доступных симуляторов: QASM Simulator, Statevector Simulator и Unitary Simulator. Есть различные виды симуляторов в Qiskit, применяемые для разных задач.

Некоторые виды симуляторов в Qiskit:

- StatevectorSimulator. Отслеживает полное квантовое состояние для бесшумной отладки схем.
- QASMSimulator. Имитирует сигналы шума реального квантового компьютера.
- StabilizerSimulator. Быстро имитирует схему Клиффорда, широко используется для коррекции ошибок.
- ExtendedStabilizerSimulator. Подходит для шумной схемы Клиффорда.
- UnitarySimulator. Моделирует идеализированные квантовые схемы через простую унитарную эволюцию матрицы.
- MatrixProductStateSimulator. Для квантовых приложений в физике конденсированного вещества.
- Выбор симулятора зависит от необходимости сбалансировать точность моделирования и скорость для анализа целевых алгоритмов на заданных аппаратных архитектурах.

- `Execute` в Python – это метод для выполнения одной команды SQL (язык запросов, предназначенный для работы с базами данных, структурированных особым образом.).
  - Позволяет выполнять различные команды SQL, такие как CREATE, INSERT, UPDATE, DELETE и SELECT.
  - Поддерживает параметризированные запросы для безопасности.
  - Возвращает объект курсора для связывания.
  - Может выполнять любые допустимые команды SQL.
  - Перед использованием `execute()` необходимо установить соединение с базой данных и создать объект курсора (интерфейс для перемещения по результатам запроса, извлечения информации и внесения изменений в записи).
- Некоторые рекомендации по работе с `execute()`:
  - Всегда закрывать соединения после завершения операций с базой данных, чтобы освободить системные ресурсы;
  - Реализовывать правильную обработку ошибок для эффективного управления операциями с базой данных;
  - Поддерживать чистое управление соединениями для оптимальных операций с базой данных.
- `def` – ключевое слово в языке программирования Python, которое используется для определения функций.
- Однострочные комментарии. Начинаются с символа #
- Многострочные комментарии. В Python официально не существует специального синтаксиса для многострочных комментариев, но их можно создать с помощью нескольких однострочных комментариев или использовать тройные кавычки.
- Документирующие строки (docstrings). Это строки, которые обычно располагаются в начале модулей, классов, методов и функций для описания их назначения. Они заключаются в тройные двойные кавычки и могут быть использованы для автоматической генерации документации. К примеру:

```
1       def add(a, b):
2       """ A function for adding two numbers.
3       :param a: the first term.
4       :param b: the second term.
5       :return: sum of a and b"""
6       return a + b
7
```

- `Range` – это встроенная функция, которая возвращает итерируемый объект, содержащий целые числа. С её помощью можно сгенерировать последовательность чисел с определённым шагом и перебирать её с помощью цикла for. Синтаксис функции: `range(stop)` или `range(start, stop, step=1)`.
  - `start` – начальное значение последовательности (по умолчанию 0).

- `stop` – конечное значение последовательности (не включается).
- `step` – шаг последовательности (по умолчанию 1).

Функция `range()` может принимать от одного до трёх целочисленных аргументов: `range(n)`; `range(k, n)`; `range(k, n, s)`.

- `Swap()` в Python – это метод, который используется для обмена значениями двух переменных. Как правило, он принимает две переменные и меняет их значения местами.
- Qin – пакет для языка.
- `circuit.h(0)` означает, что к кубиту 0 применяется адамаровский вентиль (Hadamard gate), который переводит его в состояние суперпозиции.
- Точка указывают путь к функции, где `circuit` – имя модуля, h – класс модуля, а qin – функция в этом классе. Также точки могут использоваться для обозначения атрибутов функций. Ещё точки могут означать доступ к файлам: одинарная точка означает текущий файл каталога,а двойная точка – один уровень выше в каталоге.
- Некоторые особенности вентиля h:
  - Создание состояний суперпозиции. Если применить вентиль h к кубиту, изначально находящемуся в состоянии $|0\rangle$, то кубит преобразуется в суперпозицию состояний $|0\rangle$ и $|1\rangle$.
  - Преобразование вычислительных базисных состояний. Вентиль h преобразует состояния $|0\rangle$ и $|1\rangle$ в базисные состояния Адамара $|+\rangle$ и $|--\rangle$ соответственно.
  - Самообратность. Вентиль h является самообратным, то есть его двойное применение возвращает кубит в исходное состояние.
  - Математически вентиль h можно представить в виде матрицы 2 на 2:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{5.34}$$

- `circuit.z(qin[2])` – это применение квантового вентиля z к второму элементу регистра qin. С помощью таких вентилей в квантовых вычислениях можно получить из одного базисного вектора другой.
- circuit.z(qin) – это применение квантового вентиля z, который позволяет получить из одного из базисных векторов базиса Адамара (обозначается как $|--\rangle$) другой базисный вектор ($|+\rangle$). После circuit пишется сам вентиль.
- z – фундаментальный однокубитный вентиль. Его также называют вентилем фазовой инверсии или инверсии знака. Действие вентиля заключается в повороте состояния кубита на величину π вокруг оси z на сфере Блоха. В результате вентиль z меняет местами состояния $|+\rangle$ и $|--\rangle$, а также $|i\rangle$ и $|--i\rangle$, при этом оставляя без изменений состояния $|0\rangle$ и $|1\rangle$. Некоторые особенности:
  - Вероятности измерения состояний $|0\rangle$ и $|1\rangle$ не изменяются после применения вентиля z.

– Относительная фаза произвольного квантового состояния изменяется на π плюс скорректированная относительная фаза, которая должна находиться между 0 и 2π.

– Вентиль z способен генерировать состояния суперпозиции и выполнять базисные преобразования, из-за чего играет важную роль в квантовых алгоритмах и протоколах.

- x (также известен как вентиль Паули x) действует аналогично логическому оператору «НЕ»: кубит, на который он воздействует, меняет своё значение на противоположное. Преобразует состояние $|0\rangle$ в $|1\rangle$, а $|1\rangle$– в $|0\rangle$. С точки зрения сферы Блоха вентиль x соответствует повороту вектора вокруг оси x на π радиан (или 180°).

- s (операция сдвига фазы) – одна из базовых операций в квантовых вычислениях. Умножает состояние кубитов $|1\rangle$ на комплексное число i, что соответствует сдвигу фазы на 90 градусов. Это означает, что фаза состояния $|1\rangle$ изменяется от 0 до $\pi/2$. Математическое определение: $S|0\rangle = |0\rangle, S|1\rangle = i|1\rangle$.

- Вентиль T (или t-операция) в квантовых вычислениях — это однокубитная операция, которая изменяет фазу состояния $|1\rangle$ кубита на угол $\pi/4$. Математически операция T определяется следующим образом: $T|0\rangle = |0\rangle$. $T|1\rangle = e^{i\pi/4}|1\rangle = \left(\frac{1+i}{\sqrt{2}}\right)|1\rangle$. Этот вентиль, наряду с вентилем Адамара и CNOT, образует универсальный набор гейтов, позволяющий выполнять любые квантовые вычисления. Он является ключевым компонентом для реализации квантового преобразования Фурье, управляемой фазовой оценки и многих других квантовых алгоритмов.

В Qiskit информацию о существующих вентилях можно найти, например, на сайте Deep Learning University в руководстве «Qiskit : Quantum Gates Tutorial». В нём представлен краткий обзор различных однокубитных квантовых вентилей.

Также в руководстве есть таблица с названиями и описанием некоторых из них. Узнать больше о вентилях (гейтах) в Cirq можно, например, на странице документации Gates.

Также на сайте quantumai.google есть раздел, посвящённый основам работы с Cirq, где можно найти, в том числе, информацию о доступных вентилях и способах построения схем. Для изучения Cirq рекомендуется ознакомиться с документацией и примерами использования, которые доступны на официальном сайте и в репозитории на GitHub.

Compose – это функция, которая объединяет две или более функции для создания новой функции. Кроме того, на сайте Quantum Computing Stack Exchange есть пост, в котором рассказывается о создании произвольного вентиля для произвольного количества кубитов в Cirq. Позволяет применять серию функций к вводу, при этом результат одной функции становится входом для другой.

В Python есть несколько способов компилировать функции, например:

- С помощью вложенных вызовов функций.

- С использованием лямбда–функций.

- С помощью функции functools.compose.

Композиция функций полезна для создания более выразительного и модульного кода, разбивая сложные операции на небольшие, повторно используемые функции.

Measure – специальная инструкция в языке квантового программирования QASM, которая описывает квантовое измерение.

из-за фундаментальных ограничений квантовой механики не существует способа напрямую «измерять» состояние квантовой системы. Квантовые измерения – единственный способ извлечь информацию из квантовой системы. Результат измерения – один бит классической информации.

Пример применения инструкции Measure: measure qbar – > rbar;. Эта строка означает проведение квантового измерения над третьим кубитом квантового регистра qbar и занесение результата измерения во второй бит классического регистра rbar.

Inverse – функция, возвращающая обратную функцию, предоставленной. Метод должен возвращать функцию, а не выражение. Если обратная функция представляет собой более сложное выражение, чем отдельная функция, то inverse() может вернуть лямбда–функцию. Метод inverse() следует определять только для функций, которые являются взаимно однозначными.

Благодарности

P.S.

Отдельно хотелось бы выразить глубокую признательность Поповой Анне Павловне за её неоценимую помощь и экспертное редактирование данного методического пособия. Её профессионализм, внимательное отношение к деталям и ценные замечания значительно повысили качество, ясность и методологическую точность представленного материала. Благодарим за существенный вклад в создание этого руководства.

Список литературы

[1] M.A. Nielsen, I. Chuang. Quantum Computation and Quantum Information. – Cambridge University Press, 2001.

[2] Hales L., Hallgren S. An improved quantum Fourier transform algorithm and applications Proceedings 41st Annual Symposium on Foundations of Computer Science. 2000.

[3] Abhijiyh J., Adetokunbo A. et al. Quantum Algorithm Implementations for Beginners ACM Transactions on Quantum Computing. 2022. – V. 3. – P. 17–32.

[4] J. Preskill. Lecture Notes for Physic 229: Quantum Information and Computation. – Cambridge University Press, 1998.

[5] A. W. Cross, L. S. Bishop, J. A. Smolin, J. M. Gambetta. Open Quantum Assembly Language arXiv. – 2017. – P. 24. [doi]

[6] Телеквант (образовательная платформа квантового программирования). – URL: telequant.ru