МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)**

Институт информационных технологий, математики и механики

Кафедра: высокопроизводительных вычислений и системного программирования

Направление подготовки: «Прикладная математика и информатика»
Профиль подготовки: «Прикладная математика и информатика (общий профиль)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему:

«Реализация алгоритма подготовки произвольного начального состояния системы кубитов с учетом ограничений современных квантовых компьютеров»

Выполнил: студент группы 382003_3
_____Ивлев А.Д.
Подпись

Научный руководитель:
к.т.н., доцент
_____Мееров И.Б.
Подпись

Нижний Новгород
2024

0. Оглавление

1. Введение

Подготовка произвольного состояния системы кубитов является важной и актуальной задачей квантовых вычислений. Её важность обуславливается тем, что многие квантовые алгоритмы требуют предварительной подготовки начального состояния, например алгоритм решения СЛАУ [1; 23–28].

На данный момент нет устоявшегося, оптимального решения данной задачи. Исследователи со всего мира предлагают разные подходы, и за последние несколько лет выпустили десятки работ по данной теме. В данной работе будут рассмотрены наиболее актуальные подходы (например, работы [1, 2]) и их практические реализации, которые могут существенно отличаться от теоретических ввиду несовершенства современных квантовых систем.

2. Постановка задачи

Большинство работ опираются на модель идеального квантового компьютера, но постановка задачи может иметь дополнительные ограничения, связанные с отличием современных квантовых компьютеров от их идеальной модели. Под идеальным квантовым компьютером мы понимаем квантовый компьютер с топологией связей кубитов в виде полного графа, без внешних шумов и с абсолютной точностью выполнения всех операций. К сожалению, современные устройства ещё далеки от идеальной модели, поэтому при составлении квантовых схем нам необходимо учитывать множество факторов. В данной работе я сделаю акцент на неполноте связей кубитов и ограниченности базового набора гейтов.

Также отдельно в данной работе будет рассмотрен алгоритм аппроксимации произвольного унитарного оператора, с помощью которого будет решена проблема ограниченности базового набора гейтов для адаптации точного алгоритма. И он будет применён для явного решения задачи подготовки произвольного состояния системы кубитов в виде аппроксимации унитарного оператора.

Реализация алгоритмов будет выполнена на python с использованием библиотеки Qiskit [4], что позволяет проводить запуски экспериментов на реальных сверхпроводниковых квантовых компьютерах, предоставленных облачной платформой IBM Quantum [5]. Также для тестирования схем на языке c++ будет реализован собственный симулятор идеального квантового компьютера.

Все алгоритмы будут построены на основе базовых операций, которые традиционно называют гейтами, по возможности придерживаясь синтаксиса QASM (Quantum Assembly Language [6]).

В ходе выполнения дипломной требуется решить следующие задачи:

1. Разработка алгоритма построения квантовой схемы, выполняющей перевод системы кубитов в произвольное заданное начальное состояние
2. Разработка симулятора идеального квантового компьютера
3. Реализация алгоритма аппроксимации произвольного унитарного оператора
4. Рассмотрение влияния ограничений современных квантовых компьютеров на построение квантовых схем и адаптация вышеописанных алгоритмов под структуру квантовых компьютеров облачной платформы IBM Quantum

5. Выполнение экспериментов, оценивающих размеры и точность работы полученных схем, с использованием симулятора идеального квантового компьютера и квантовых компьютеров IBM

3. Симулятор идеального квантового компьютера

3.1 Общая структура и операция измерения

Состояние квантовой системы из N кубитов описывается комплексным нормированным вектором длины 2^N, с которым мы и будем работать. Следовательно мы ограничены по используемой памяти, и максимальное количество кубитов для симуляции составляет ~30 для системы, на которой проводились эксперименты.

При работе с реальными квантовыми компьютерами мы не знаем вектора состояния, поэтому важной операцией является измерение, которая переводит множество вероятностных состояний к одному конкретному. То есть, пусть квантовая система находится в некотором состоянии $|\psi\rangle \in \mathbb{C}^{2^N}$, тогда $|\psi_i|^2$ – вероятность получить i-е состояние при измерении и при этом состояние $|\psi\rangle$ переходит в состояние $|i\rangle$.

В нашем случае, при реализации симулятора, вектор состояния нам доступен поэтому измерение реализовано с возможностью выбора: провести классическое измерение, описанное выше с изменением вектора состояния или провести серию измерений без необходимости перерасчета всей квантовой схемы.

Измерение производится следующем образом:

1. Выбирается случайное число R в диапазоне [0, 1]
2. Проходимся по вектору состояния с индексом i, суммируя $|\psi_i|^2$, пока полученная сумма меньше R
3. Сохраняем полученное в ходе измерения состояние i
4. Если необходимо коллапсируем состояние системы, иначе переходим к пункту 1, пока не проведём требуемое количество экспериментов

Код операции измерения в приложении [А.1].

3.2 Применение однокубитных гейтов

Рассмотрим проведение операций над системой на примере применения однокубитного гейта. Пусть он будет произвольным и будет описываться унитарной матрицей $U \in \mathbb{C}^{2x2}$. Для примера возьмём N = 2 (будем считать 0 кубит младшим) и пусть система уже находится в некотором состоянии $|\psi\rangle \in \mathbb{C}^{2^N}$. Тогда применение данного гейта U к нулевому кубиту примет вид: $|\psi\rangle \rightarrow (I \otimes U)|\psi\rangle$, где $\otimes$ - кронекерово произведение, а I – единичная матрица размера 2x2. Аналогично для применения к первому кубиту $|\psi\rangle \rightarrow (U \otimes I)|\psi\rangle$. Данный классический подход, крайне медленный из-за

большего количества матричных операций, поэтому сведём его к методу, работающему только с вектором состояния.

Рассмотрим подробнее преобразования:

$$(I \otimes U)|\psi\rangle = \begin{pmatrix} u_1 & u_2 & 0 & 0 \\ u_3 & u_4 & 0 & 0 \\ 0 & 0 & u_1 & u_2 \\ 0 & 0 & u_3 & u_4 \end{pmatrix} \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{pmatrix} = \begin{pmatrix} u_1\psi_1 + u_2\psi_2 \\ u_3\psi_1 + u_4\psi_2 \\ u_1\psi_3 + u_2\psi_4 \\ u_3\psi_3 + u_4\psi_4 \end{pmatrix}$$

Можно заметить, что 1 и 2 строки результирующего вектора состояния будут получены путём преобразования $U\begin{pmatrix}\psi_1 \\ \psi_2\end{pmatrix}$, а 3 и 4 строки из $U\begin{pmatrix}\psi_3 \\ \psi_4\end{pmatrix}$.

$$(U \otimes I)|\psi\rangle = \begin{pmatrix} u_1 & 0 & u_2 & 0 \\ 0 & u_1 & 0 & u_2 \\ u_3 & 0 & u_4 & 0 \\ 0 & u_3 & 0 & u_4 \end{pmatrix} \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{pmatrix} = \begin{pmatrix} u_1\psi_1 + u_2\psi_3 \\ u_1\psi_2 + u_2\psi_4 \\ u_3\psi_1 + u_4\psi_3 \\ u_3\psi_2 + u_4\psi_4 \end{pmatrix}$$

Аналогично 1 и 3 строки результирующего вектора состояния будут получены путём преобразования $U\begin{pmatrix}\psi_1 \\ \psi_3\end{pmatrix}$, а 2 и 4 строки из $U\begin{pmatrix}\psi_2 \\ \psi_4\end{pmatrix}$.

Соответственно, данную закономерность можно обобщить. Пусть q – номер кубита к которому мы применяем однокубитный гейт U, тогда возьмём битовую маску (mask) с 1 на данной позиции. При обходе вектора (с $i = \overline{0, 2^N - 1}$) будем проверять условие (i & mask) == mask, если оно выполнено, тогда применим преобразование $U\begin{pmatrix}\psi_{i-l} \\ \psi_i\end{pmatrix}$, где l = 2^q - расстояние, на котором располагается пара для ψ_i-го элемента.

3.3 Применение гейтов с контролем

Как будет подробнее объяснено далее (в пункте 6.3) для полноты возможностей нам достаточно некоторого универсального базового набора гейтов. В предыдущем пункте было разобрано, как применить произвольный однокубитный гейт, следовательно для полноты нам достаточно добавить реализацию двухкубитного гейта CNOT (гейт контролируемого отрицания), который в общем случае задаётся матрицей:

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

Данный гейт меняет состояние управляемого кубита, если управляющий кубит в состоянии 1.

Соответственно, применение гейта CNOT(0, 1) примет вид:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{pmatrix} = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_4 \\ \psi_3 \end{pmatrix}$$

Можно заметить, что для достижения подобного эффекта достаточно добавить к нашей битовой маске 1 на номере управляющего кубита, то есть обобщим для применения произвольного гейта U с множественным контролем:

Пусть q – номер управляемого кубита, а list_q – номера управляющих кубитов. Тогда возьмём битовую маску (mask) с 1 на позициях из list_q и q. При обходе вектора (с $i = \overline{0, 2^N - 1}$) будем проверять условие (i & mask) == mask, если оно выполнено, тогда применим преобразование $U\begin{pmatrix}\psi_{i-l}\\ \psi_i\end{pmatrix}$, где l = 2^q - расстояние, на котором располагается пара для ψ_i-го элемента.

Можно заметить, что при данном подходе мы избавились от громоздких матричных операций со всем вектором и свели их к операциям с парами значений, причём при работе эти пары не пересекаются. Соответственно, данная задача легко распараллеливается, что и было проделано с помощью технологии OpenMP.

Код применения гейта с контролем в приложении [А.2].

4. Алгоритм подготовки начального состояния системы кубитов

4.1 Описание алгоритма

Формальная постановка задачи выглядит следующим образом. Задается нормированный вектор состояния системы кубитов $|\psi\rangle \in \mathbb{C}^{2^N}$, где N – количество кубитов. Необходимо построить квантовую схему, которая задаёт унитарное преобразование U, которое переводит исходное состояние $|0\rangle^{\otimes N}$ (все кубиты в состоянии 0) в желаемое $|\psi\rangle$, то есть совершает преобразование $U|0\rangle^{\otimes N} = |\psi\rangle$.

Основная идея алгоритма заключается в том, что мы представляем все комплексные числа вектора состояния в показательном виде ($r_k e^{i\psi_k}$), рассматриваем подзадачу построения вектора из действительных чисел (r_k), и далее – подзадачу наложения фазы на полученный вектор.

Первая подзадача сводится к распределению 1 (соответствует начальному состоянию, где все кубиты в состоянии $|0\rangle$) по всему вектору в необходимых пропорциях, для чего будем использовать гейт $Ry(\varphi)$ и его контролируемые версии.

Вторая подзадача в общем случае может быть решена с использованием гейта $P(\varphi)$, но ввиду того, что данный гейт редко входит в базовый набор гейтов реальных квантовых систем, будем использоваться гейт $Rz(\varphi)$, что повлечёт за собой накопление глобальной фазы, но данная проблема будет рассмотрена отдельно.

Гейты $Ry(\varphi)$ и $Rz(\varphi)$ в матричном виде имеют вид

$$Ry(\varphi) = \begin{pmatrix} \cos\left(\frac{\varphi}{2}\right) & -\sin\left(\frac{\varphi}{2}\right) \\ \sin\left(\frac{\varphi}{2}\right) & \cos\left(\frac{\varphi}{2}\right) \end{pmatrix}, \qquad Rz(\varphi) = \begin{pmatrix} e^{-i\varphi/2} & 0 \\ 0 & e^{i\varphi/2} \end{pmatrix}$$

Разделим первую часть алгоритма на N итераций ($i = \overline{1, N}$), на каждой из которых мы будем рассматривать только первые 2^{i} значений вектора. То есть на каждой итерации мы будем задействовать только первые i кубитов, причём старший из них изначально будет находиться в состоянии 0. Это значит, что вторая половина рассматриваемых значений вектора будет заполнена нулями.

Задача итерации распределить значения из первой половины на весь под вектор из 2^{i} значений. Например, для $\mathrm{i} = 2$:

$$\begin{pmatrix} a \\ b \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} a' \\ b' \\ c \\ d \end{pmatrix}$$

Разобьём 2^i значений на пары по их позициям $(j, 2^{i-1}+j)$, $j = \overline{0, 2^{i-1}-1}$. То есть сопоставим каждому значению из первой половины вектора состояния в пару значение из второй. Можно заметить, что данные значения описываю состояния, которые отличаются только значением старшего, для данной итерации, кубита. Следовательно мы можем выделить данные состояния, используя множественный контроль (который далее будет рассмотрен отдельно в пункте 4.2), и применить гейт $Ry(\varphi)$ к каждой паре по отдельности.

То есть мы свели задачу итерации к переходу пар из состояния $(c, 0)$ к (a, b). Но есть существенный недостаток данного подхода это большая теоретическая сложность алгоритма $O(2^N)$.

Пример первых двух итерации построения вектора модулей приведен ниже (Рисунок 1), что соответствует следующему изменению состояния квантового регистра:

$$\binom{1}{0} \rightarrow \binom{a}{b} \left| \begin{pmatrix} a \\ b \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} a' \\ \mathrm{b} \\ c \\ 0 \end{pmatrix} \right| \begin{pmatrix} a \\ b \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} a' \\ b' \\ c \\ d \end{pmatrix}$$

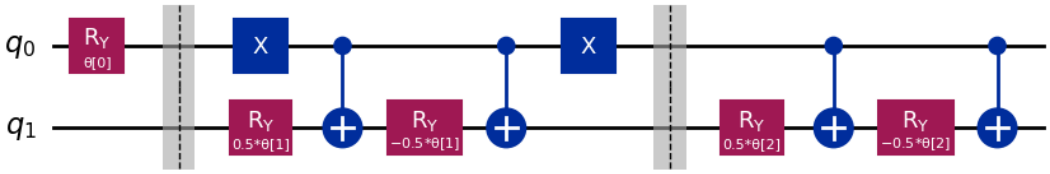

Рисунок 1. Схематичная квантовая схема первых двух шагов построения вектора модулей

Для завершения основной части алгоритма нам осталось определить углы φ для применения гейтов $Ry(\varphi)$. Рассмотрим данный процесс для одной пары:

$$\begin{pmatrix} \cos\left(\frac{\varphi}{2}\right) & -\sin\left(\frac{\varphi}{2}\right) \\ \sin\left(\frac{\varphi}{2}\right) & \cos\left(\frac{\varphi}{2}\right) \end{pmatrix} \binom{c}{0} = \binom{a}{b}$$

Получим что $a = c * \cos\left(\frac{\varphi}{2}\right), b = c * \sin\left(\frac{\varphi}{2}\right)$. Тогда $\varphi = 2\arccos\left(\frac{a}{c}\right)$ при $c \neq 0$ ($\varphi = 0$ при $c = 0$). Вспомним, что нам известно о значениях a, b, c.

При $i = 1$ есть всего одна пара и для неё мы знаем только $c = 1$, при $i = N$ мы знаем a, b для каждой пары, так как это известные нам значения желаемого вектора состояний $|\psi\rangle$. Поэтому для определения углов мы будем идти от последней итерации к первой “сворачивая” вектор к нулевому состоянию, используя соотношение $c = \sqrt{a^2 + b^2}$.

Также вспомним, что мы рассматривали построение вектора модулей r от изначальных комплексных значений. Нам осталось выполнить преобразование для

полученных на последней итерации пар в виде: $\begin{pmatrix} a \\ b \end{pmatrix} \rightarrow \begin{pmatrix} ae^{i\psi_1} \\ be^{i\psi_2} \end{pmatrix}$, причём углы ψ нам известны. Коды определения углов в приложении [А.3].

Следовательно, на последней итерации $\begin{pmatrix} c \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} ae^{i\psi_1} \\ be^{i\psi_2} \end{pmatrix}$, то есть мы можем не задействовать дополнительные гейты CNOT для поворота на фазу, так как второе значение в паре $\begin{pmatrix} c \\ 0 \end{pmatrix}$ равно 0. Пример одного шага последней итерации приведен ниже.

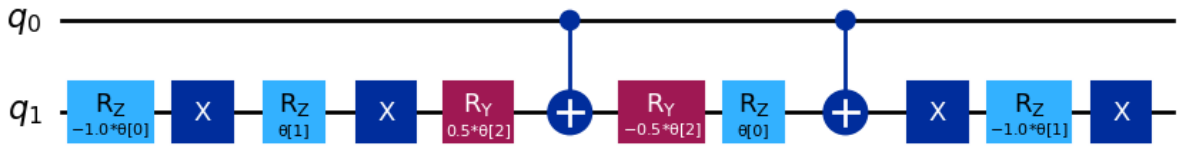

Рисунок 2. Блок квантовой схемы, выполняющий поворот на заданную фазу

На рисунке ниже приведен пример полной схемы данного алгоритма для N = 2 в общем виде.

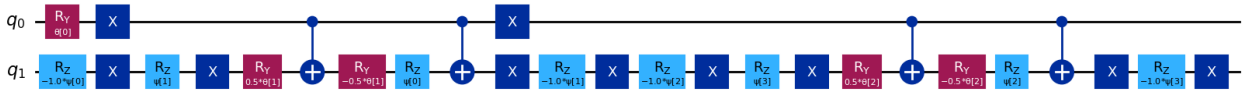

Рисунок 3. Полная квантовая схема для N=2

Данная схема выполнить следующий перевод состояния квантового регистра:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} \cos(\frac{\varphi_0}{2}) \\ \sin(\frac{\varphi_0}{2}) \end{pmatrix} \otimes \begin{pmatrix} \cos(\frac{\varphi_0}{2}) \\ \sin(\frac{\varphi_0}{2}) \\ 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} \cos(\frac{\varphi_0}{2})\cos(\frac{\varphi_1}{2})e^{i\psi_1} \\ \sin(\frac{\varphi_0}{2})\cos(\frac{\varphi_2}{2})e^{i\psi_2} \\ \cos(\frac{\varphi_0}{2})\sin(\frac{\varphi_1}{2})e^{i\psi_3} \\ \sin(\frac{\varphi_0}{2})\sin(\frac{\varphi_2}{2})e^{i\psi_4} \end{pmatrix}$$

Код подготовки состояния системы кубитов в приложении [А.4].

Также стоит отметить, что далеко не всегда требуется подготовить состояние, когда все кубиты запутаны. Следовательно мы можем разделить задачу на подготовку состояний групп, запутанных между собой кубитов, что может значительно уменьшить сложность алгоритма в целом. Например, подготовка состояния для N незапутанных кубитов будет иметь сложность O(N), что является лучшим случаем. Худший случай соответственно полная запутанность и сложность $O(2^N)$, что утверждалось ранее. В общем случае верна оценка $O(m2^{N_{max}})$, где m – количество групп запутанных кубитов, а N_{max} – максимальное количество запутанных кубитов соответственно.

Преимущества данного алгоритма:

- высокая теоретическая точность
- отсутствие дополнительных кубитов
- устойчивость к ошибкам глобальной фазы (будет показано далее)

Недостатки:

- высокая сложность $O(m2^{N_{max}})$
- большое количество 2-х кубитный гейтов CNOT

4.2 Подробности реализации

Для построение квантовых схем описанного выше алгоритма необходимо подробнее рассмотреть построение гейтов с множественным контролем. Его основой выступает 2-х кубитный гейт CNOT.

Сначала рассмотрим гейт дважды контролируемого отрицания (гейт CCNOT или гейт Тоффоли). Данный гейт можно разложить на более простые гейты в виде:

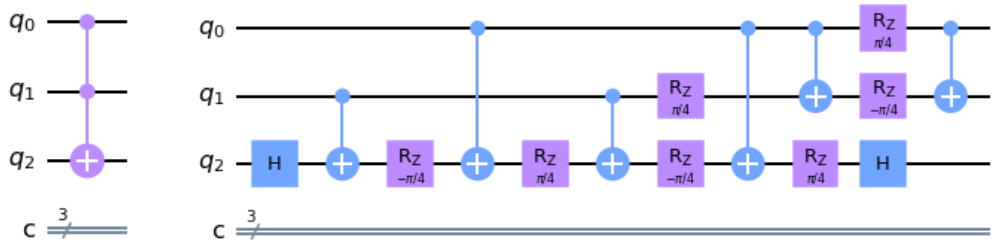

Рисунок 4. Разложение гейта Тоффоли с точностью до глобальной фазы $e^{i\frac{\pi}{8}}$

Такое представление верно с точностью до глобальной фазы равной $e^{i\frac{\pi}{8}}$ Как бороться с накопленной глобальной фазой будет описано ниже.

Соответственно, если количество управляющих кубитов n = 1, то CNOT, если n = 2, то гейт Тоффоли. Далее рассмотрим построение гейта контролируемого отрицания с произвольным количеством управляющих кубитов n.

Заметим, что последовательность гейтов HZH эквивалентна гейту X. Следовательно, достаточно реализовать управляемый гейт Z. Для n = 1 это можно сделать с помощью гейта контролируемой фазы, который ввиду ограничений примет вид:

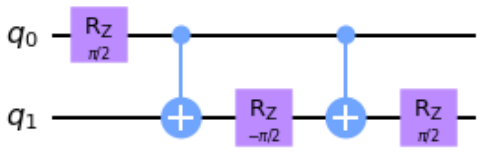

Рисунок 5. Разложение гейта CZ через гейты RZ и CNOT с точностью до глобальной фазы $e^{-i\frac{\pi}{4}}$

Такое представление верно с точностью до глобальной фазы равной $e^{-i\frac{\pi}{4}}$.

Для $n \geq 2$ используем рекурсивный подход ([7; 13–25]), основанный на следующем преобразовании:

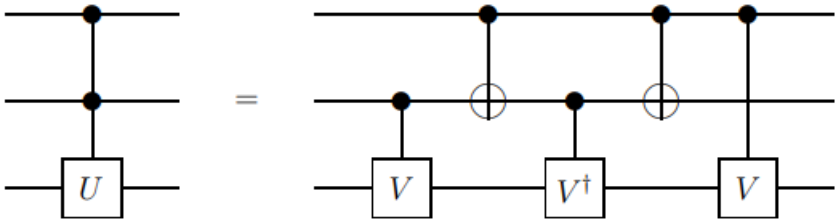

Рисунок 6. Разложение дважды управляемого воздействия U

Где примем $U = RZ(\varphi), U = V^2$. Тогда $V = RZ\left(\frac{\varphi}{2}\right), V^\dagger = RZ\left(\frac{-\varphi}{2}\right)$. Стоит отметить, что внутренняя глобальная фаза компенсируется и остаётся глобальная фаза с углом $\sum_{k=1}^{n} \frac{\varphi}{2^{k+1}}$.

Следовательно, для получения гейта X с множественным контролем (CnNOT) достаточно применить H(c), CnPhase (φ = π, list, c), H(c), где c – управляемый кубит, a list – список из n управляющих кубитов. Код в приложении [В.1].

Вспомним, что в нашем алгоритме подготовки начального состояния на i итерации n = i, причём гейт CnNOT применяется 2^{i+1} раз, следовательно глобальная фаза на итерации равна $2^{i+1}\sum_{k=1}^{i}\frac{\pi}{2^{k+1}} = \pi 2^i \sum_{k=1}^{i}\frac{1}{2^k} = \pi\left(2^i - 1\right)$, то есть угол глобальной фазы кратен π, следовательно, если он равен 0 (при $i = 0$), то всё хорошо, а если π, то требуется его погасить последовательностью гейтов XZYXZY применённых к любому кубиту.

Далее рассмотрим построение произвольного гейта с множественным контролем. Для этого выделим в дополнительный кубит необходимое нам состояние с помощью гейта множественного контроля и гейтов отрицания X. Гейты X применяются до и после гейта множественного контроля на кубитах значения которых в выделяемом состоянии равно 0.

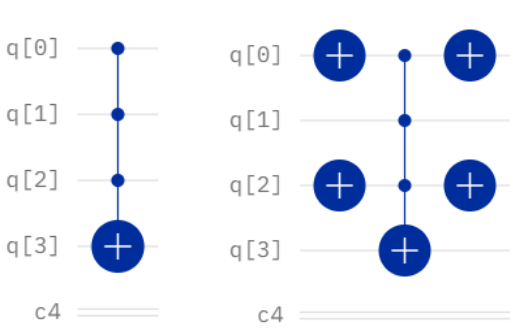

Рисунок 7.Пример выделения состояний |111⟩ и |010⟩ в дополнительный кубит

Далее применим нужный нам гейт с контролем от дополнительного кубита и после необходимо вернуть состояние дополнительного кубита в исходное снова применив контролируемое отрицание. Можно заметить, что гейты X между двумя гейтами контролируемого отрицания можно убрать, так как XX = I. Пример применения гейта H с контролем по состоянию $|010\rangle$:

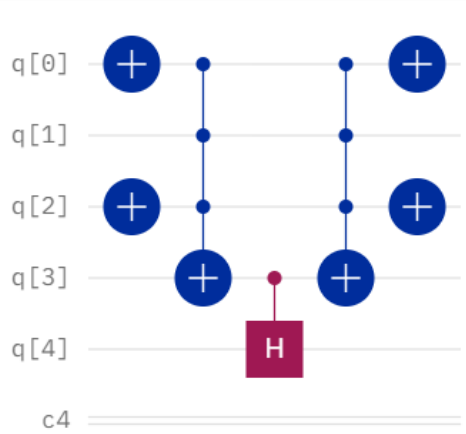

Рисунок 8.Пример применения гейта H к четвёртому кубиту при состоянии $|010\rangle$ первых трёх с использованием 3 кубита в качестве дополнительного

Но, так как мы используем гейты RY, RZ с множественным контролем, то дополнительный кубит не требуется. Например, для n = 1 их можно представить в виде:

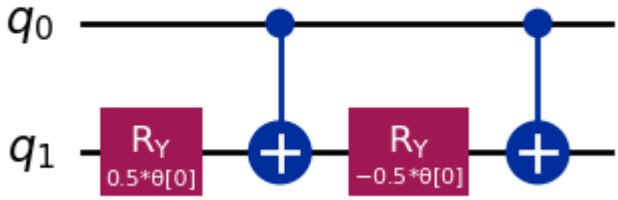

Рисунок 9. Реализация гейта CRY без дополнительного кубита (аналогично для CRZ, CRX)

Для больших n достаточно заменить CNOT на CnNOT.

Стоит отметить, что данный алгоритм был построен с помощью наиболее часто используемого базового набора гейтов: X, Rx, Ry, Rz, CNOT. Адаптация под другие базовые наборы будет рассмотрена на примере в пунктах 6, 7.

4.3 Пример работы алгоритма

Рассмотрим пример нахождения углов для конкретного вектора $(\frac{i}{\sqrt{2}}, 0, 0, -\frac{1}{\sqrt{2}})$. Запишем числа в показательном виде: $(\frac{1}{\sqrt{2}}e^{i\frac{\pi}{2}}, 0, 0, \frac{1}{\sqrt{2}}e^{i\pi})$. Из этого вектора мы сразу можем выделить углы $\psi = (\frac{\pi}{2}, 0, 0, \pi)$, тогда искомый вектор модулей: $(\frac{1}{\sqrt{2}}, 0, 0, \frac{1}{\sqrt{2}})$.

Для нахождения углов φ начнём с последней итерации $i = 2$, где есть две пары $\left(\frac{1}{\sqrt{2}}, 0\right)$ (0 и 2 элементы) и $(0, \frac{1}{\sqrt{2}})$ (1 и 3 элементы). Для первой пары $c_1 = \frac{1}{\sqrt{2}}$ и $\varphi_1 = 2 * \arccos 1 = 0$, а для второй $c_2 = \frac{1}{\sqrt{2}}$ и $\varphi_2 = 2 \arccos 0 = \pi$. Вектор $\binom{c_1}{c_2} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$ является конечным для предыдущей итерации ($i = 1$), поэтому повторим аналогичные действия для него. В полученном векторе всего одна пара, из которой $c_0 = 1$ и $\varphi_0 = \frac{\pi}{2}$. Так, вычислив углы $\varphi = [[\frac{\pi}{2}]. [0, \pi]]$, мы вернулись к исходному вектору нулевого состояния.

Мы можем проверить вычисления подставив все полученные углы в уравнения общего вида для N = 2:

$$\begin{pmatrix} \cos(\frac{\pi}{4})\cos(0)e^{i\frac{\pi}{2}} \\ \sin(\frac{\pi}{4})\cos(\frac{\pi}{2})e^{i0} \\ \cos(\frac{\pi}{4})\sin(0)e^{i0} \\ \sin(\frac{\pi}{4})\sin(\frac{\pi}{2})e^{i\pi} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}}i \\ 0 \\ 0 \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Тогда квантовая схема, задающая данное состояние примет вид:

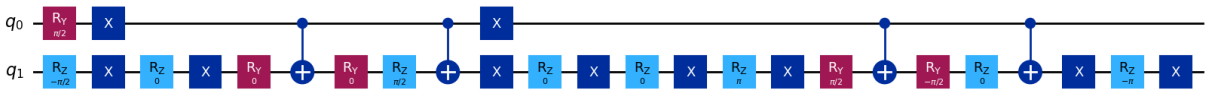

Рисунок 10.Схема подготовки состояния $(\frac{i}{\sqrt{2}}, 0, 0, -\frac{1}{\sqrt{2}})$

4.4 Оптимизация за счет использования дополнительных кубитов

В статье “A divide-and-conquer algorithm for quantum state preparation” [2] приводиться оптимизация за счет использования дополнительных кубитов. Данный алгоритм является модификацией предыдущего алгоритма. Он позволяет за счет $2^N - N - 1$ дополнительных кубитов привести сложность к $O(N^2)$.

Идея заключается в развертке алгоритма в ширину вместо длинs за счет древовидной структуры алгоритма и гейтов CSWAP (гейт управляемой перестановки). Углы вычисляются также, как в предыдущем случаи, тогда общая схема прошлого алгоритма (Рисунок 11) перейдет в схему (Рисунок 12).

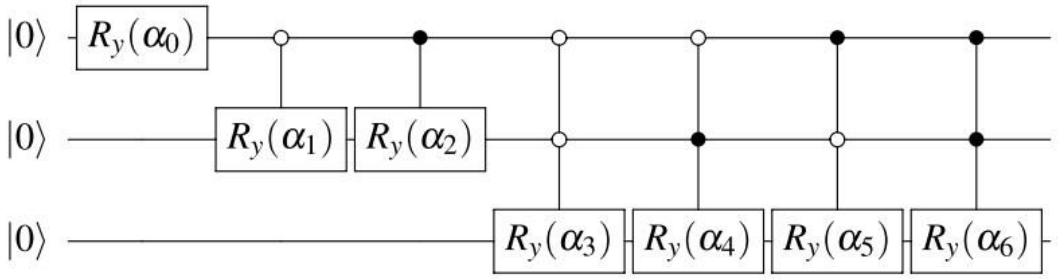

Рисунок 11. Общая квантовая схема точного алгоритма для N=3

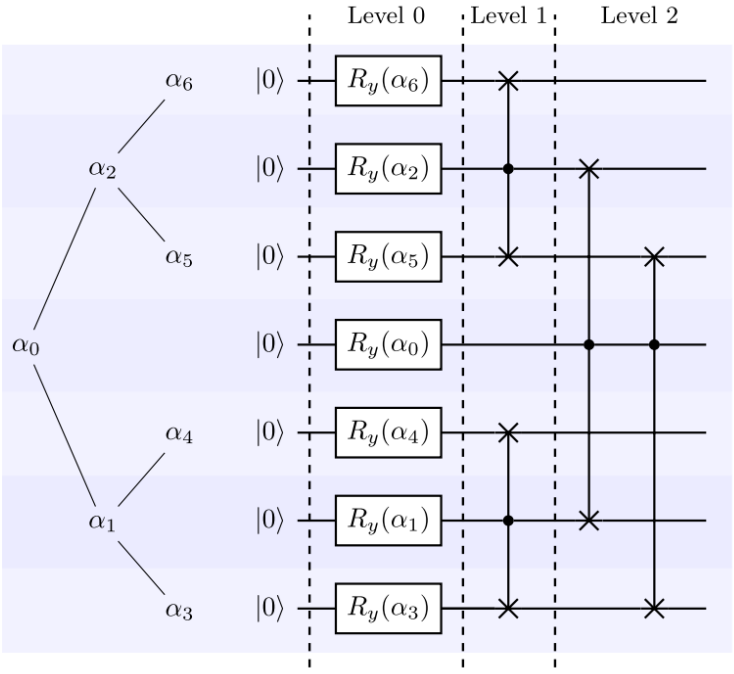

Рисунок 12. Общая квантовая схема алгоритма разделяй и властвуй для N=3

Гейт CSWAP раскладывается на более простые в виде:

Рисунок 13. Разложение гейта CSWAP

Ввиду малой связности кубитов в реальных квантовых компьютерах данная оптимизация даже увеличивает количество гейтов для подготовки состояния для систем из небольшого количество кубитов. Например, для N=3, при обходе ограничений квантовых систем IBM (подробнее главы 5, 6, 7), показанное выше количество кубитов

для явной схемы 1570 и 1780 для оптимизированной соответственно. Поэтому, данный алгоритм был рассмотрен только теоретически.

Преимущества:

- высокая теоретическая точность
- относительно малая сложность $O(N^2)$.

Недостатки:

- большое количество дополнительных кубитов ($2^N - N - 1$) и гейтов CNOT.

5. Аппроксимация произвольного унитарного оператора

5.1 Описание алгоритма

Мы хотим аппроксимировать произвольный унитарный оператор U размера $2^n \times 2^n$ некоторой конечной квантовой схемой на основе некоторого конечного набора базовых гейтов. Это тоже важная задача. Например, одно из применений это представление одного универсального набора гейтов через другой, что будет использоваться в данной работе. Так как базовые наборы гейтов для разных квантовых компьютеров часто отличаются, то таким способом мы обеспечиваем переносимость квантовых схем на разные квантовые процессоры.

Теорема Соловей–Китаева гарантирует, что аппроксимация может быть сделано эффективно, для конечного набора элементов G в SU(2^n) (SU(2^n) - специальная унитарная группа матриц $2^n \times 2^n$ с определителем равным 1) содержащего свои собственные обратные значения (что если $g \in G$, то $g^{-1} \in G$) и такого, что группа которую они порождают, плотна в SU (2^n).

Для n = 1 известна оптимальная оценка требуемого количества гейтов - $\log^c(\frac{1}{\varepsilon})$, где ε – желаемая точность, а c – некоторая константа в диапазоне (1, 4] (доказательство [8; 749–758]).

Идея алгоритма заключается в том, что мы заготавливаем несколько параметризованных шаблонных схем. И пытаемся оптимизировать параметры, порядок применения и количество повторений данных шаблонов, чтобы полученная в результате матрица схемы минимально отличалась от желаемой.

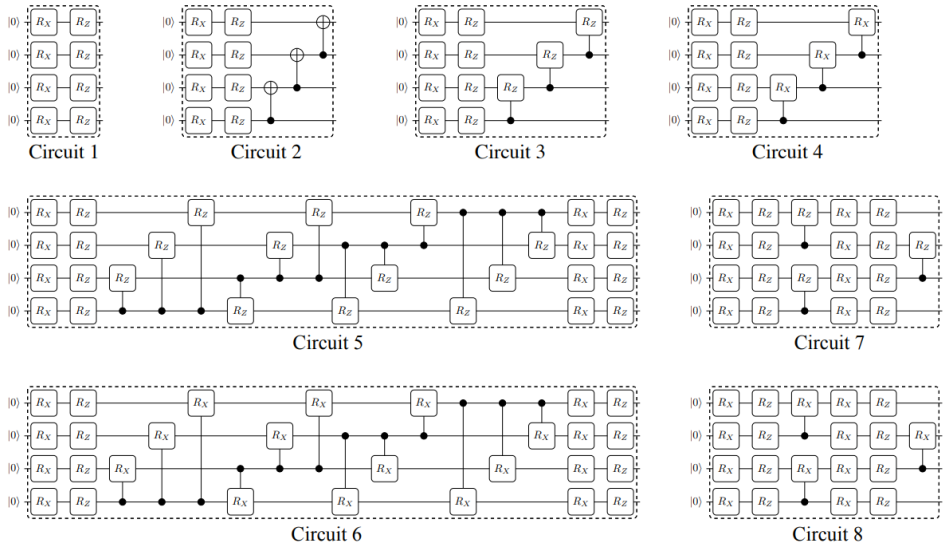

Рисунок 14. Типовые примеры шаблонных схем [9: 8]

Так как мы хотим аппроксимировать произвольный унитарный оператор, то мы изначально ничего не знаем о его виде. Поэтому, для универсальности, за базовый шаблон будем брать некоторые аналоги схемы 2 (Рисунок 14). Наличие в ней гейтов CNOT будет обеспечивать связанность кубитов.

Следовательно, предложенный универсальный метод может быть не оптимальным и количество повторений шаблонных схем для достижения высокой точности может достигать 2^n, но как правило не более.

Подбор оптимальных параметров будет происходить, как решение задачи оптимизации: $\min_p ||U - M(p)||$, где M(p) – матрица параметризованной схемы с набором параметров p – соответственно, и используется норма Фробениуса.

Так же так как все наши параметры углы поворота, то наложим на них ограничения в виде принадлежности диапазону $[-\pi, \pi]$. Решение будет производиться числено с помощью scipy minimize [10] с критерием остановки в виде желаемой точности ε или ограничением по количеству операций N_{max}.

Преимущества данного алгоритма:

- высокая теоретическая точность
- отсутствие дополнительных кубитов
- адаптивность шаблонов под разные ограничения квантовых компьютеров

Недостатки:

- высокая сложность (как квантовая, так и классическая) для достижения высокой точности в общем случае

5.2 Подготовка начального состояния в виде аппроксимации унитарного оператора

Любая квантовая схема представляет собой некоторый унитарный оператор. Соответственно получив вид данного оператора для схемы, описанной в предыдущей главе 4, можно аппроксимировать его. В некоторых случаях это может значительно уменьшить размер квантовой схемы, сравнение в пункте 8.3.

Рассмотрим оператор, получившийся из схемы из пункта 4.3 (Рисунок 10):

$$\begin{pmatrix} \frac{i}{\sqrt{2}} & -\frac{i}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & -\frac{i}{\sqrt{2}} & \frac{i}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix}$$

Можно заметить, что первый столбец данного оператора всегда будет вектором состояния, которое мы хотим построить. Что естественным образом получается из нашей постановки задачи в виде $U|0\rangle^{\otimes N} = |\psi\rangle$, где $|\psi\rangle$ - желаемое состояние. Соответственно для решения данной задачи в общем случае достаточно брать произвольный унитарный оператор с первым столбцом $|\psi\rangle$.

6. Влияние ограничений современных квантовых компьютеров на построение алгоритмов

6.1 Примеры современных квантовых компьютеров и их ограничений

Для примера рассмотрим квантовые компьютеры, находящиеся в открытом доступе через сервис IBM Quantum [6]. В данный момент базовым набором гейтов для всех доступных для общего пользования квантовых машин является набор ECR, I, Rz, SX, X.

Данный набор отличен от используемого нами ранее, но он также представляет собой универсальную систему. Как аппроксимировать ранее используемы гейты, используя представленный набор будет рассказано далее в пункте 6.3.

Также важная часть ограничений — это топология связей кубитов, на данный момент она далека от полного графа (топологии идеального квантового компьютера).

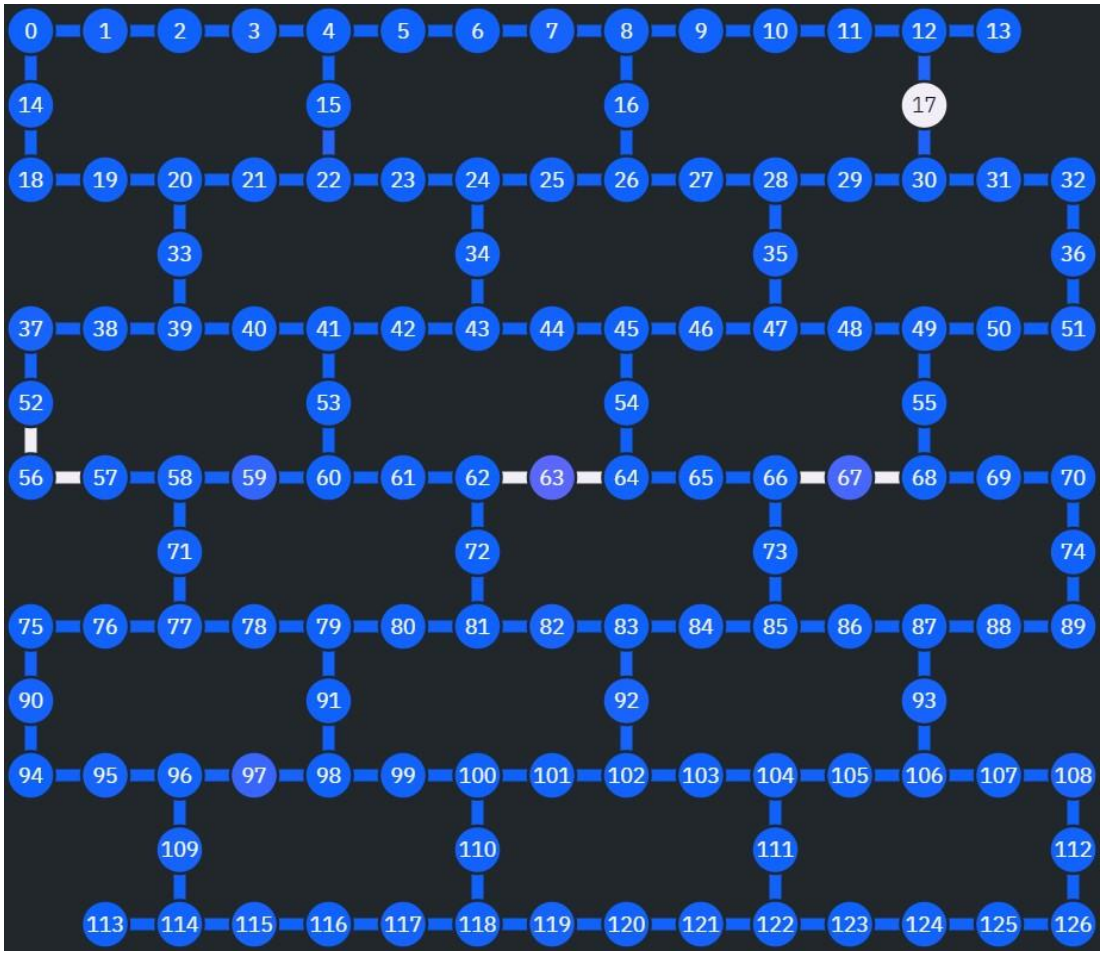

Рисунок 15. Топология ibm_sherbrooke

Цветом кубитов на данном изображении указана средняя ошибка выполнения однокубитной операции SX, а цвет соединения кубитов ошибку 2-х кубитного гейта ECR. Чем ближе цвет к белому, тем больше ошибка. Как положить алгоритм на неполные топологии будет рассказано далее в пункте 6.2.

Средние значения ошибок для ibm_sherbrooke (на 13.05.2024):

ECR: 7.432e-3

SX: 2.279e-4

Ошибка считывания: 1.180e-2

На данный момент существует множество вариантов компенсаций ошибок выполнения гейтов, но мы не будем рассматривать их в данной работе, ввиду сложности и не универсальности решений данной задачи.

6.2 Применение двухкубитных гейтов с учетом топологии

Как видно из приведённого выше примера (Рисунок 15) связность кубитов далека от идеальной, следовательно мы не можем использовать 2-х кубитные гейты для каждой пары. Но что же делать, если например нам нужно реализовать CNOT(0, 5) на ibm_sherbrooke.

Так как любые 2 кубита связаны, либо напрямую, либо по цепочке, то в первом случае обычный CNOT. Иначе, если кубиты не являются соседями, то представим исходный CNOT эквивалентной последовательностью гейтов CNOT между соседними кубитами.

Пусть два кубита a, b соединяет цепочка из n-1 кубитов ($a \neq b$, следовательно $n \neq 0$). Для удобства перенумеруем все кубиты в цепочки от a = 0, до b = n.

Алгоритм CNOT(0, n):

- Если n = 1, то CNOT(0, 1) и алгоритм завершается
- Последовательно применяем CNOT между ближайшими кубитами в цепочке, начиная с 0 и заканчивая n
- Проводим обратные действия предыдущему шагу, исключая CNOT(n-1, n) и CNOT(0, 1).
- Один раз повторяем предыдущие два шага, для возврата состояний промежуточных кубитов в исходные

Например, CNOT (0, 2) и CNOT (0, 3):

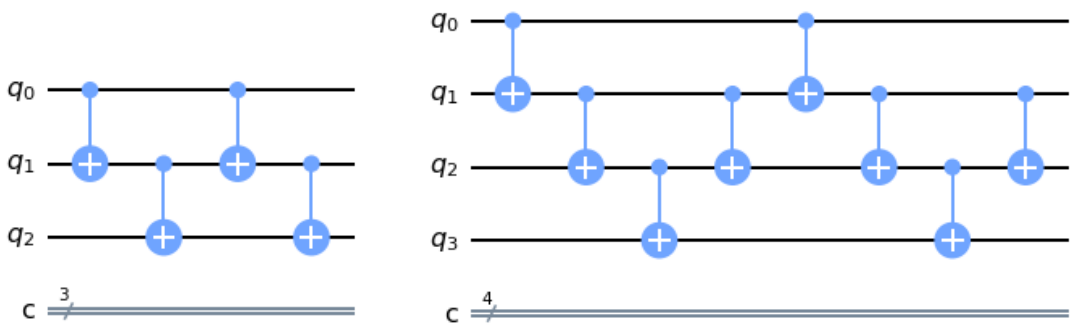

Рисунок 16. Разложение CNOT для несвязанных кубитов

Тогда такой подход будет использовать $4(n-1)$ гейтов CNOT, если $n \geq 2$. При $n = 1$ один обычный CNOT. Любые другие 2-х кубитные гейты можно разложить, используя гейты CNOT, для которых нужно использовать данный подход. Код линейного CNOT [B.2].

Можно заметить, что в топологии квантового компьютера ibm_sherbrooke некоторые кубиты можно связать разными способами. Тогда так как применение гейта ERC имеет некоторую погрешность, то стоит выбрать кратчайший путь с наименьшей общей погрешностью. Данная задача эквивалента нахождению кратчайшего пути на графе с весами, которая для данных небольших графов хорошо решается, например алгоритмом Дейкстры. Так, например, кубиты 39 и 58 в ibm_sherbrooke лучше связать, через 39-40-41-53-60-59-58, а не 39-38-37-52-56-57-58 из-за большой ошибки применения гейта ERC между парами кубитов 52, 56 и 56, 57.

6.3 Базисные наборы гейтов и их универсальность

Назовём универсальным набором квантовых гейтов такой набор, к которому может быть сведена любая операция, возможная на квантовом компьютере, то есть любая другая унитарная операция может быть выражена как конечная последовательность гейтов из данного набора. Технически это невозможно, так как необходим несчетный набор гейтов, поскольку число возможных гейтов неисчислимо, тогда как число конечных последовательностей из конечного набора счетно. Поэтому потребуем только, чтобы любая квантовая операция могла быть аппроксимирована последовательностью элементов из конечного множества. Вышеописанная теорема Соловей–Китаева гарантирует, что это возможно.

Есть множество универсальных наборов квантовых вентилей. Ранее мы использовали набор Rx, Ry, Rz, P, CNOT, универсальность которого доказана. Данные гейты имеют вид:

$$Rx(\varphi) = \begin{pmatrix} \cos\left(\frac{\varphi}{2}\right) & -\mathrm{i}\sin\left(\frac{\varphi}{2}\right) \\ -\mathrm{i}\sin\left(\frac{\varphi}{2}\right) & \cos\left(\frac{\varphi}{2}\right) \end{pmatrix}$$

$$Ry(\varphi) = \begin{pmatrix} \cos\left(\frac{\varphi}{2}\right) & -\sin\left(\frac{\varphi}{2}\right) \\ \sin\left(\frac{\varphi}{2}\right) & \cos\left(\frac{\varphi}{2}\right) \end{pmatrix}$$

$$Rz(\varphi) = \begin{pmatrix} e^{-i\varphi/2} & 0 \\ 0 & e^{i\varphi/2} \end{pmatrix}$$

$$P(\varphi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi/2} \end{pmatrix}$$

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

Квантовые компьютеры IBM использую набор ECR, I, Rz, SX, X, который также является универсальным и имеют вид:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

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

$$SX = \begin{pmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}$$

$$\mathrm{ECR} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1 & 0 & \mathrm{i} \\ 1 & 0 & -\mathrm{i} & 0 \\ 0 & \mathrm{i} & 0 & 1 \\ -\mathrm{i} & 0 & 1 & 0 \end{pmatrix}$$

Покажем, что однокубитные гейты нового набора можно выразить через старый. Для начала выведем гейт U3, который является универсальным однокубитным гейтом.

$$U_3(\theta, \phi, \lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix} = Rz(\phi)Rx(-\frac{\pi}{2})Rz(\theta)Rx(\frac{\pi}{2})Rz(\lambda)$$

Зная разложение U3 несложно представить остальные однокубитные гейты:

$$I = U_3(0,0,0)$$

$$X = U_3(\pi, 0, \pi)$$

$$SX = e^{i\frac{\pi}{4}}U_3\left(\frac{\pi}{2}, -\frac{\pi}{2}, \frac{\pi}{2}\right)$$

Теперь разложим ECR:

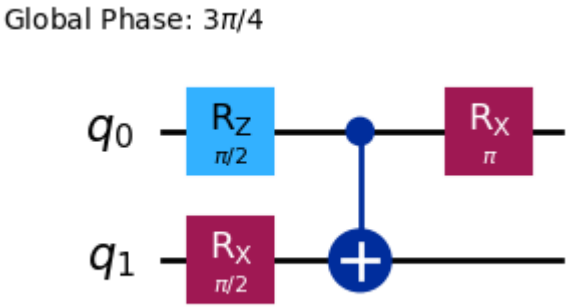

Рисунок 17. Разложение гейта ECR с точностью до глобальной фазы $e^{i\frac{3\pi}{4}}$

Можно заметить, что мы оставили $e^{i\frac{\pi}{4}}$ в разложении SX. Это обозначение глобальной фазы. Гейт $Ph(\varphi) = \begin{pmatrix} e^{i\varphi} & 0 \\ 0 & e^{i\varphi} \end{pmatrix}$ добавляет глобальную фазу к вектору состояния, то есть $Ph(\varphi)|\psi\rangle = e^{i\varphi}|\psi\rangle$. $Ph(\varphi) = P(\varphi)XP(\varphi)X$.

$Ph(\varphi)$ и $P(\varphi)$ не принадлежат SU(2), но их отсутствие в базовом наборе может привести к проблеме накопления глобальной фазы. Далее расскажем, в чем проблема и один из вариантов как с ней бороться.

Как выразить используемые нами гейты с помощью базового набора ECR, I, Rz, SX, X, чтобы непосредственно запустить на квантовом компьютере IBM вышеописанные алгоритмы будет показано в 7 главе.

6.4 Проблема наличия глобальной фазы и её решение

Наличие глобальной фазы как правило не доставляет проблем, так как она не влияет на результат измерения, но всё же есть некоторые алгоритмы, такие что при возникновении глобальной фазы внутри алгоритма мы можем получить результат отличный от желаемого, например алгоритм подготовки начального состояния или оценки фазы [1; 24-27].

Пусть нам доступен набор гейтов Rx, Ry, Rz, CNOT точно выразить из них $Ph(\varphi)$ и $P(\varphi)$ нельзя так как они не принадлежат SU(2), поэтому попробуем найти аппроксимацию для $Ph(\varphi)$. Будем считать, что глобальная фаза на момент запуска алгоритма известна, так как в большинстве случаев мы можем её подсчитать. Например, при замене гейта $P(\varphi)$ на $Rz(\varphi)$ возникнет глобальная фаза $e^{i\frac{\varphi}{2}}$. Также заметим, что схема XZYXZY гасит фазу π, и что последовательное применение двух одинаковых CNOT не изменяет состояние системы.

Как было сказано ранее, для получения гейта X с множественным контролем достаточно CnNOT(c, list) = H(c)CnPhase(**π**, list, c)H(c), где c – номер управляемого кубита,

a list список управляющих и он будет прибавлять глобальную фазу $\frac{\pi}{2^{n+1}}$ (n – количество управляющих кубитов). Соответственно применив два раза данный гейт с одинаковыми параметрами, мы не изменим модуль состояния, но увеличим глобальную фазу на $\frac{\pi}{2^n}$.

Пусть нам нужно убрать глобальную фазу θ. Так как мы можем просто убирать фазу π, то нам достаточно применять двойной CnNOT с различным количеством управляющих кубитов n, прибавляя к θ углы $\frac{\pi}{2^n}$ до тех пор, пока разница общей глобальной фазы и $k * \pi$ не станет по модулю меньше $\frac{\pi}{2^N}$, где k – произвольное целое число, а N – максимально доступное число кубитов в системе. После если k – нечетное, то применим XZYXZY который погасит фазу π.

После применения данного алгоритма останется глобальная фаза по модулю меньше, чем $\frac{\pi}{2^N}$, то есть точность аппроксимации $Ph(\theta)$ ограничена лишь количеством кубитов в системе.

Конечно, это лишь одна из возможных аппроксимаций, возможно не самая точная и оптимальная, но имеющая строгую структуру. Также возможно решение данной задачи в виде аппроксимации унитарного оператора с элементами $e^{i\theta}$ на диагонали.

7. Адаптация алгоритма подготовки начального состояния под квантовые компьютеры IBM

7.1 Представление однокубитных гейтов

В описанном в главе 4 алгоритм подготовки начального состояния использовал гейты X, H, Y, Z, Ry, Rz, CNOT, а как было сказано ранее квантовые компьютеры IBM в данный момент использую базовый набор гейтов ECR, I, Rz, SX, X. Следовательно, нам нужно аппроксимировать первый набор вторым.

Начнём с аппроксимации однокубитных гейтов. X и Rz присутствуют в обоих наборах. Оставшиеся гейты H, Y, Z и Ry попробуем оптимально аппроксимировать, используя шаблонную схему SX-Rz-SX- Rz-SX- Rz-SX:

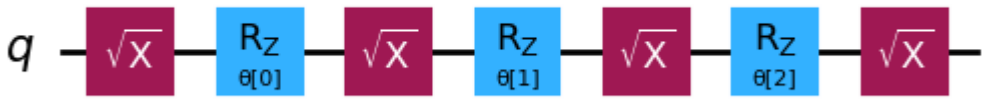

Рисунок 18. Шаблонная схема для аппроксимации однокубитных гейтов

Данную схема была выбрана за шаблонную, так как она с точностью до глобальной фазы аналогична универсальному гейту U3. Также поэтому, как правило, достаточно одного её повторения. Глобальная фаза не станет для нас проблемой, так как данные аппроксимируемые гейты её не накладывают.

Здесь и далее под точностью аппроксимации будем понимать $\varepsilon = ||U - M(p)||$, где M(p) – матрица параметризованной схемы с набором параметров p – соответственно, и используется норма Фробениуса.

Из соображения, что гейт X можно представить в виде HZH для оптимизации, уберём боковые гейты Sx. Тогда для H мы получим значения параметров [-2.5e-09, 1.571, -3.9e-09], а точность аппроксимации 6.04e-09. Легко заметить, что для идеальной аппроксимации набор параметров $[0, \frac{\pi}{2}, 0]$ и схема примет вид:

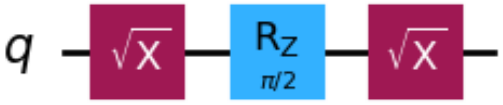

Рисунок 19. Аппроксимация гейта H

Гейты Y, Z в нашем алгоритме применяются в связке ZY, следовательно и аппроксимируем данный составной гейт. Для него мы получим значения параметров [-0.0, 3.142, -3.142], а точность аппроксимации 1.64e-09. Легко заметить, что для идеальной аппроксимации набор параметров $[0, \pi, -\pi]$ и схема примет вид:

Рисунок 20. Аппроксимация гейта ZY

Гейт Ry(φ) зависит от одного параметра попробуем провести аппроксимацию при нескольких его значениях, чтобы увидеть закономерность Результаты представим в виде таблицы:

Угол φ	Найденный набор параметров	Точность
$\frac{\pi}{7}$	[1.4, 0.0, 0.951]	1.836e-09
$\frac{\pi}{7}$	[2.519, -0.0, 2.07]	2.522e-09
$\frac{\pi}{3}$	[2.057, 0.0, 1.01]	1.374e-09
$\frac{\pi}{3}$	[0.989, -0.0, -0.058]	2.256e-09
$\frac{\pi}{3}$	[0.559, 0.0, -0.488]	9.517e-10
$\frac{\pi}{5}$	[0.638, 0.0, 0.009]	1.305e-09

Таблица 1. Подбор параметров для аппроксимации гейта Ry

Заметим, что даже для одного угла результат подбора параметров может быть не однозначен. Но явно видны закономерности: второй параметр всегда 0, модуль разности первого и третьего параметра равен углу φ. Из данных соображений построим оптимальную аппроксимирующую схему:

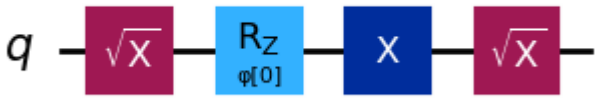

Рисунок 21. Аппроксимация гейта Ry(φ)

Теперь нам осталось аппроксимировать гейт CNOT, что будет показано в следующем пункте.

7.2 Представление двухкубитного гейта CNOT

Чтобы сохранить все выводы, касающиеся глобальной фазы для алгоритма подготовки начального состояния, описанного в главе 4, аппроксимируем не чистый CNOT, а CNOT со сдвигом глобальной фазы на $\frac{\pi}{4}$. За основу возьмём шаблоны:

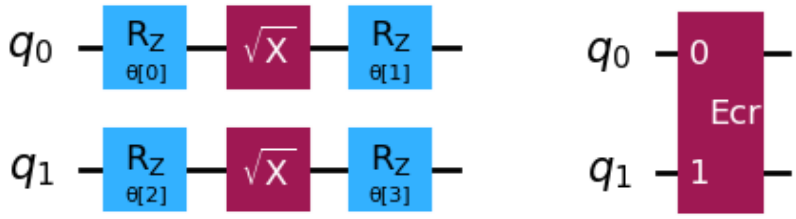

Рисунок 22. Шаблонные схемы для аппроксимации CNOT с точностью до глобальной фазы $e^{i\frac{\pi}{4}}$

Первая схема будет обеспечивать повороты кубитов, а вторая их связность. Чтобы получить необходимую точность, чередуя, применим данные схемы. Тогда параметризованная схема примет вид:

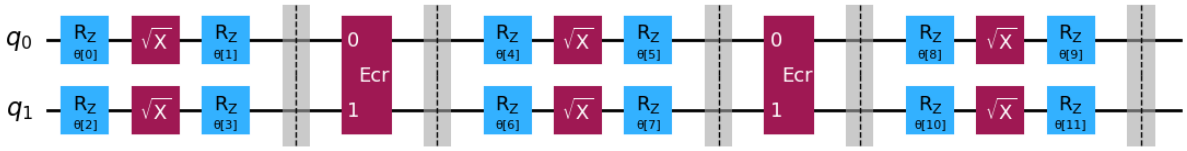

Рисунок 23. Параметризованная схема для аппроксимации CNOT с точностью до глобальной фазы $e^{i\frac{\pi}{4}}$

По аналогии с предыдущем пунктом для оптимизации будем искать закономерности в поведении параметров. Например, один из результатов [2.915, -0.262, -0.0, 3.142, 2.879, -0.007, -0.0, 3.142, 3.135, -1.344, 0.0, 0.0] даёт точность 1.527e-08.

В ходе исследования удалось привести схему к оптимальному относительно данных шаблонов виду:

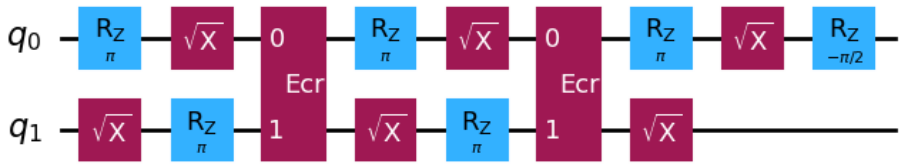

Рисунок 24. Представление CNOT с точностью до глобальной фазы $e^{i\frac{\pi}{4}}$

Возможно, при выборе других шаблонов мы могли бы получить более оптимальную схему. Но теперь мы можем заменить старые базовые гейты на их представление и провести эксперименты. Но уже можно заметить, что данный подход в несколько раз увеличивает количество используемых гейтов.

8. Оценка размера квантовых схем и точности работы алгоритмов на симуляторе идеального квантового компьютера

8.1 Аппроксимация произвольного унитарного оператора

В данном разделе мы проведём ряд экспериментов на идеальной модели квантового компьютера. Как было сказано ранее подбор оптимальных шаблонных схем для аппроксимации конкретного унитарного оператора U отдельная, достаточно сложная задача. Поэтому ради универсальности возьмём следующие шаблонные схемы:

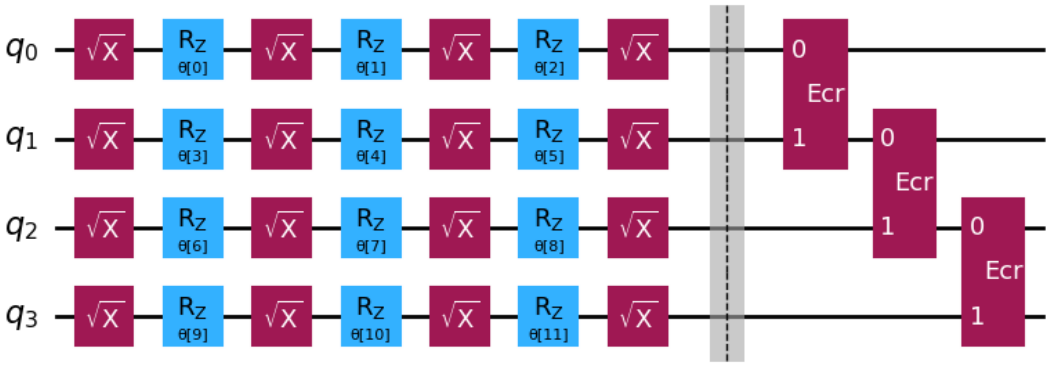

Рисунок 25. Универсальный шаблонный блок

Можно заметить, что они естественным образом расширяются на произвольное количество кубитов. Для простоты будем применять их последовательно, повторяя несколько раз.

Проведём эксперименты, чтобы установить среднюю точность аппроксимации в зависимости от количества кубитов и повторений шаблонной схемы. Для чистоты эксперимента для отслеживания глобальной фазы добавим в конец схемы параметризованный гейт глобальной фазы.

Суть эксперимента посчитать среднюю точность аппроксимации для произвольного унитарного оператора U действующего на n кубитов при k повторениях шаблонной схемы. На данный момент было произведены эксперименты для n = 1,2,3, и $k = \overline{1, 2^n}$ соответственно. Усреднение происходило при 5 различных U для каждой пары (n, k). Стоит ответить, что данный универсальный подход приводит к решению задачи оптимизации с $3nk$ параметрами, что можно значительно улучшить, оптимально подбирая шаблоны и их последовательность для каждого унитарного преобразования, но это отдельная задача, которая здесь не рассматривается.

Результаты представим в виде таблицы, где столбцы количество повторений шаблонной схемы, строки количество кубитов, а значение ячейки – полученная средняя

ошибка. Прочерки в таблице означают, что при данных параметрах эксперимент не проводился.

Кол-во кубитов\повторений	1	2	3	4	5	6	7	8
1	0.426	1.203e-8	7.857e-9	-	-	-	-	-
2	1.427	0.895	0.145	4.062e-8	2.499e-8	-	-	-
3	3.06	2.641	2.325	1.62	1.389	0.954	0.481	3.28e-6

Таблица 2. Результаты по аппроксимации произвольного унитарного преобразования в зависимости от количества повторений шаблонной схемы

Можно заметить, что вышеописанная оценка, что требуется как правило не больше 2^n повторений верна. Но также для некоторых унитарных операторов требуется меньшее количество повторений.

8.2 Алгоритм подготовки начального состояния

В отличие от предыдущему случаю будем оценивать точность работы алгоритма как норму разности результирующего вектора состояния и исходного вектора. Проведём испытания на 10 произвольных векторах для систем из $n = \overline{1,6}$ кубитов.

Стоит отметить, что в связи с переводом в новый набор базовых гейтов размер схемы увеличился в несколько раз. Например, общий вид схемы при n = 2 (можно сравнить с Рисунок 10):

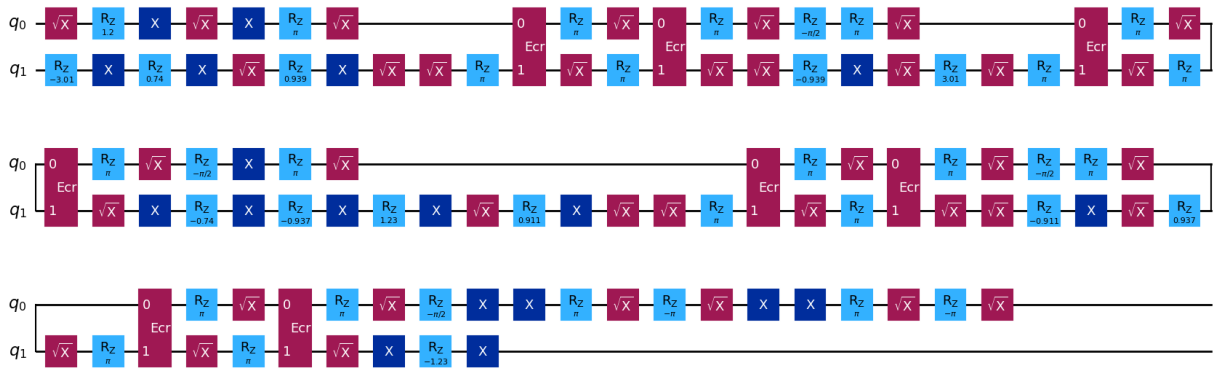

Рисунок 26. Общий вид схемы при n = 2

Ниже приведены результаты для произвольного комплексного вектора с присутствием фазы и без.

Количество кубитов	1	2	3	4	5	6
Точность с фазой	0.747	1.055e-15	2.092e-14	2.237e-13	1.721e-12	1.145e-11
Точность без фазы	1.586e-16	8.941e -16	2.119e-14	2.255e-13	1.734e-12	1.123e-11

Таблица 3. Точность подготовки начального состояния для симулятора

Можно заметить аномально низкую точность для 1 кубита при подготовке произвольного комплексного вектора состояния, она объясняется тем, что для данного случая мы не всегда можем погасить глобальную фазу (если не использовать дополнительные кубиты). Для прочих n заметно, что с его увеличение точность падает в среднем на порядок, хотя и остаётся достаточно высокой. Наиболее вероятно, что падение точности связано с большим объёмом вычислений с использованием вещественных значений.

8.3 Подготовка начального состояния в виде аппроксимации унитарного оператора и сравнение двух подходов

Пусть за аппроксимируемый унитарный оператор возьмём матрицу, получаемую в результате построения явного алгоритма подготовки начального состояния. Точность подготовки начального состояния будем оценивать аналогично предыдущему пункту. Проведём испытания на 3 произвольных векторах для систем из $n = \overline{1,3}$ кубитов. Количество повторений шаблонной схемы возьмём 2^n, следовательно его сложность O($n2^n$).

Количество кубитов	1	2	3
Явный алгоритм	0.818	1.15e-15	2.153e-14
Аппроксимация	0.818	1.41e-08	4.54e-07

Таблица 4. Сравнение точности подготовка начального состояния в виде аппроксимации унитарного оператора и явного алгоритма

Можно заметить, что точность существенно меньше при сравнении с точным алгоритмом. Прежде, чем делать выводы об эффективности данного подхода сравним общее количество гейтов и гейтов ECR (указаны в скобках) в полученных схемах.

Количество кубитов	1	2	3	4	5	6	7
Явный алгоритм	17 (0)	106 (8)	1596 (200)	16358 (2184)	122520 (16648)	809250 (110600)	5057556 (692744)
Аппроксимация	14 (0)	60 (4)	184 (16)	496 (48)	1248 (128)	3008 (320)	7040 (768)

Таблица 5. Сравнение количество гейтов в полученных схемах

За счёт того, что шаблонная схема была заготовлена сразу с учётом ограничений квантовых компьютеров IBM, она достигает меньшего количества используемых кубитов. При этом существенно понижается точность вычислений, но главную сложность данного подхода это решение задачи оптимизации для подбора $3n2^n$ параметров. А явный алгоритм при действиях для обхода ограничений приобрёл сложность O(10^n)

9. Результаты экспериментов на симуляторе и квантовых компьютерах IBM

9.1 Аппроксимация произвольного унитарного оператора с адаптированными шаблонными схемами.

Проверим как изменится точность, если мы запустим алгоритм на реальном квантовом компьютере. Ввиду невозможности отследить эволюцию квантового оператора на реальной системе, будем оценивать погрешность самым простым способом, то есть нормой разницы результатов при применении гейта к нулевому начальному состоянию поделённую на количество измерений. Более точные оценки возможны, только при детальном анализе квантового компьютера с помощью квантовой томографии [16].

Все эксперименты были произведены на квантовом компьютере ibm_sherbrooke, с количеством измерений 4096.

Так как ресурсы использования облачных квантовых компьютеров ограничены, то проведём 3 эксперимента для 1 кубита с 2 повторениями шаблонной схемы и 1 эксперимент для 2 кубитов с 4 повторениями шаблонной схемы. В таблице результатов будем отображать результат измерений и точность.

Симулятор	Квантовый компьютер	Погрешность
[3310, 786]	[3269, 827]	0.014
[1554, 2542]	[1548, 2548]	0.002
[2755, 1341]	[2692, 1404]	0.021
[1603, 214, 1676, 603]	[1662, 235, 1520, 679]	0.044

Таблица 6. Результаты экспериментов по аппроксимации произвольного унитарного оператора

Результаты для одного кубита находятся примерно в пределах заявленной ошибки измерения 0.0118, что показывает высокую точность аппроксимации малых операторов. При увеличении количества кубитов увеличилось количество задействованных гейтов и погрешность увеличилась из-за накопившейся ошибки их исполнения.

9.2 Адаптированные алгоритмы подготовки начального состояния

Проверим как изменится точность, если мы запустим алгоритмы на реальном квантовом компьютере. Так как при измерении на реальном квантовом компьютере мы можем получить, только приближённый вектор модулей состояний, то для экспериментов будем подготавливать, только вектора состояния с нулевой фазой.

Проведём 1 эксперимента для 1 кубита и 1 эксперимент для 2 кубитов. Для первого эксперимента возьмём вектор $[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]$, а для второго эксперимента возьмём вектор $[\frac{1}{\sqrt{2}}, 0, 0, \frac{1}{\sqrt{2}}]$ (вектор, разобранный в пункте 4.3, но с нулевой фазой всех элементов). Будем проводить 4096 измерений, тогда идеальными результатами можно считать [2048, 2048] и [2048, 0, 0, 2048] соответственно.

Тогда результаты при использовании точного алгоритма:

Симулятор	Квантовый компьютер	Погрешность
[2017, 2079]	[1982, 2114]	0.012
[2067, 0, 0, 2029]	[2035, 111, 124, 1826]	0.065

Таблица 7. Результаты экспериментов по подготовке начального состояния точного алгоритмом

В первом случае погрешность объясняется неточностью считывания, а во втором случае ошибка уже непосредственно накапливается из-за большого количества исполняемых гейтов.

Проведём аналогичный эксперимент, но используя алгоритм аппроксимирующий унитарный оператор.

Симулятор	Квантовый компьютер	Погрешность
[2057, 2039]	[2068, 2028]	0,004
[2059, 0, 0, 2037]	[2162, 90, 54, 1790]	0,071

Таблица 8. Результаты экспериментов по подготовке начального состояния аппроксимирующим алгоритмом

Можно заметить, что за счёт меньшего количества гейтов для первого случая лучше результат для аппроксимирующего алгоритма, но для второго лучше для точного. Это можно объяснить, исходя из того, что сравнительно низкая теоретическая точность аппроксимирующего алгоритма, не может в полной мере компенсироваться малым количеством используемых гейтов по сравнению с явным алгоритмом для систем с малым количеством кубитов.

10. Заключение

В ходе выполнения дипломной работы были решены следующие задачи:

- Реализован собственный симулятор идеального квантового компьютера
- Реализован алгоритм подготовки произвольного состояния системы кубитов
- Реализован алгоритм аппроксимации произвольного унитарного оператора размера $2^n \times 2^n$
- Разобраны ограничения современных квантовых компьютеров: не полнота топологии и универсальность базового набора гейтов квантового компьютера. Для не полной топологии разобран алгоритм связи двух произвольных кубитов по цепочке гейтом CNOT
- Представлена проблема глобальной фазы и её возможное решение
- Проведены эксперименты для оценки точности работы алгоритмов и размеров их схем, как на симуляторе, так и на реальных квантовых компьютерах

Также, аналогичная описанной, адаптация алгоритма подготовки произвольного начального состояния системы кубитов под квантовый компьютер, предоставленный сервисом OriginQ [17], позволила мне получить гранд приз в студенческой группе международных соревнований The second CCF “Pilot Cup” Quantum Computing Programming Challenge. Данные соревнования были организованы китайской компьютерной федерацией.

11. Приложение

A. Участки кода из реализации симулятора идеального квантового компьютера

1. Реализация операции измерения

```
vector<size_t> condition_exp(size_t start, size_t end, size_t count = 10, bool col_flag =
false) {
	vector<size_t> temp(1i64 << (end - start));
	size_t j = 0;
	if (col_flag) {
		count = 1;
	}

	for(size_t i = 0; i < count; i++) {
		T r = T(rand()) / RAND_MAX;
		T sum = 0;
		j = 0;
		while (sum < r && j < data.size()) {
			sum += norm(data[j]);
			j++;
		}
		temp[((j-1) % (1i64 << end))>>start]++;
	}

	if (col_flag) {
		clear();
		data[((j-1) % (1i64 << end)) >> start] = 1;
	}

	return temp;
}
```

2. Применение однокубитного гейта с произвольным контролем

```
void gate(vector<size_t> control_q, complex<T> matrix[4]) {
	int mask = 0;
	int step = 1 << control_q.back();
	for (size_t i = 0; i < control_q.size(); i++) {
		if (control_q[i] >= size) {
			data.resize(data.size() * 2, 0);
		}
		mask += (1 << control_q[i]);
	}

#pragma omp parallel for
	for (int i = 0; i < data.size(); i++) {
		if ((i & mask) == mask) {
			complex<T> temp = data[i - step];
			data[i - step] = temp*matrix[0] + data[i]*matrix[1];
			data[i] = temp * matrix[2] + data[i] * matrix[3];
		}
	}
}
```

3. Просчет углов для алгоритма подготовки состояния системы кубитов

```
template<typename T>
vector<vector<T>> abs_ang(vector<complex<T>> vec_) {
	size_t step = vec_.size();
	vector<vector<T>> level_ang{};
	vector<T> vec{};

	for (size_t i = 0; i < vec_.size(); i++) {
		vec.push_back(abs(vec_[i]));
	}

	while (step != 1) {
		step = step >> 1;
		level_ang.push_back(vector<T>{});
		vector<T> temp_vec{};
```

```
                for (size_t j = 0; j < step; j++) {
                        T a = vec[j];
                        T b = vec[j + step];
                        T c = sqrt(a * a + b * b);
                        temp_vec.push_back(c);
                        if (c != 0) {
                                level_ang.back().push_back(2 * acos(a / c));
                        }
                        else {
                                level_ang.back().push_back(0);
                        }
                vec.clear();
                vec = temp_vec;
                }
        }

        reverse(level_ang.begin(), level_ang.end());

        return level_ang;
}

template<typename T>
vector<T> phase_ang(vector<complex<T>> vec_) {
        vector<T> level_ang{};

        for (size_t i = 0; i < vec_.size(); i++) {
                level_ang.push_back(arg(vec_[i]));
                if (level_ang.back() < 0) {
                        level_ang.back() += 2 * PI;
                }
        }

        return level_ang;
}
```

4. Общий вид алгоритма подготовки состояния системы кубитов

```
vector<complex<double>> State_preparation(vector<complex<double>> vec) {
        size_t vec_size = vec.size();
        size_t N = log2(vec_size);
        Qbit<double> cir(N);
        cir[0] = 1;

        vector<vector<double>> level_angles = abs_ang(vec);
        vector<double> phase = phase_ang(vec);

        vector<double> angles = level_angles[0];

        if (angles[0] != 0) {
                cir.RY(0, angles[0]);
        }
        if (N == 1) {
                if (phase[1] != 0) {
                        cir.P(0 ,phase[1]);
                }
                if (phase[0] != 0) {
                        cir.X(0);
                        cir.P(0, phase[0]);
                        cir.X(0);
                }
        }
        vector<size_t> temp{ 0 };

        for (size_t i = 1; i < N; i++) {
                size_t p_2 = (1 << i);
                angles.clear();
                angles = level_angles[i];
                size_t last_0 = p_2 - 1;

                for (size_t j = 0; j < p_2; j++) {
                        if (angles[j] != 0 || i == N - 1 || j == 0) {
                                X_bit_mask(cir, ((p_2 - j - 1) ^ (p_2 - last_0 - 1)));
                                last_0 = j;
                        }

                        if (i == N - 1 && phase[j] != 0) {
```

```
                        cir.RZ(i, -phase[j]);
                }
                if (i == N - 1 && phase[j + p_2] != 0) {
                        cir.X(i);
                        cir.RZ(i, phase[j + p_2]);
                        cir.X(i);
                }

                cir.RY(i, angles[j] / 2);
                cir.CnNOT(temp, i);
                cir.RY(i, -angles[j] / 2);

                if (i == N - 1 && phase[j] != 0) {
                        cir.RZ(i, phase[j]);
                }

                cir.CnNOT(temp, i);

                if (i == N - 1 && phase[j + p_2] != 0) {
                        cir.X(i);
                        cir.RZ(i, -phase[j + p_2]);
                        cir.X(i);
                }
        }

        X_bit_mask(cir, p_2 - last_0 - 1);
        temp.push_back(i);
    }

    return cir.get_data();
}
```

B. Код с использованием библиотеки Qiskit

1. Реализация CnNot

```
def CnPhase(ang, list, c, q_list, cir): #Multi controled P
    if(len(list) == 1):
        cir = CP_(ang, list[0], c, q_list, cir)
        return cir


    for i in range(1, len(list)):
        cir = CP_(ang/(2**i), list[-i], c, q_list, cir)
        cir = CnNOT(list[:-i], list[-i], q_list, cir)
        cir = CP_(-ang/(2**i), list[-i], c, q_list, cir)
        cir = CnNOT(list[:-i], list[-i], q_list, cir)


    cir = CP_(ang/(2**(len(list) - 1)), list[0], c, q_list, cir)
    return cir
def CnNOT(list, c, q_list, cir):
    if(len(list) == 1):
        cir = CNOT_lin(list[0], c, q_list, cir)
    elif(len(list) == 2):
        cir = C2NOT(list[0], list[1], c, q_list, cir)
    else:
        cir = H(q_list[c], cir)
        cir = CnPhase(np.pi, list, c, q_list, cir)
        cir = H(q_list[c], cir)
    return cir
```

2. Реализация линейного CNot

```
def CNOT_lin(q0, c, q_list, cir, flag_non_lin = False):
    if(flag_non_lin or abs(q0 - c) == 1):
```

```
            cir = CX(q_list[q0], q_list[c], cir)
    else:
            i = 0
            while(q0 + i < c):
                    cir = CX(q_list[q0 + i], q_list[q0 + i + 1], cir)
                    i+=1
            i -= 2
            while(i > 0):
                    cir = CX(q_list[q0 + i], q_list[q0 + i + 1], cir)
                    i-=1
            while(q0 + i < c):
                    cir = CX(q_list[q0 + i], q_list[q0 + i + 1], cir)
                    i+=1
            i -= 2
            while(i > 0):
                    cir = CX(q_list[q0 + i], q_list[q0 + i + 1], cir)
                    i-=1
    return cir
```

Полная версия кода: https://github.com/Faert/IvlevAD_State_preparation_sim

12. Литература

1. Quantum Algorithm Implementations for Beginners Abhijith J., Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Alexander Malyzhenkov, David Mascarenas, Susan Mniszewski, Balu Nadiga, Daniel O'Malley, Diane Oyen, Scott Pakin, Lakshman Prasad, Randy Roberts, Phillip Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter J. Swart, James G. Wendelberger, Boram Yoon, Richard Zamora, Wei Zhu, Stephan Eidenbenz, Andreas Bärtschi, Patrick J. Coles, Marc Vuffray, Andrey Y. Lokhov: arXiv:1804.03719v3 (2022)
2. A divide-and-conquer algorithm for quantum state preparation Israel F. Araujo, Daniel K. Park, Francesco Petruccione, Adenilton J. da Silva: arXiv:2008.01511v2
3. Quantum-state preparation with universal gate decompositions Martin Plesch, Časlav Brukner: arXiv:1003.5760v2 (2021)
4. QISKIT Documentation: https://docs.quantum.ibm.com
5. IBM Quantum Platform: https://quantum.ibm.com
6. Open Quantum Assembly Language Andrew W. Cross, Lev S. Bishop, John A. Smolin, Jay M. Gambetta: arXiv:1707.03429v2 (2017)
7. Elementary gates for quantum computation A. Barenco (Oxford), C.H. Bennett (IBM), R. Cleve (Calgary), D.P. DiVincenzo (IBM), N. Margolus (MIT), P. Shor (AT&T), T. Sleator (NYU), J. Smolin (UCLA), H. Weinfurter (Innsbruck): arXiv:quant-ph/9503016v1 (1995)
8. М. Нильсен, И. Чанг КВАНТОВЫЕ ВЫЧИСЛЕНИЯ И КВАНТОВАЯ ИНФОРМАЦИЯ Перевод с английского под редакцией М. Н. Вялого и П. М. Островского с предисловием К. А. Валиева, Москва «Мир» (2006)
9. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms Sukin Sim, Peter D. Johnson, Alan Aspuru-Guzik: arXiv:1905.10876v1 (2019)
10. NumPy and SciPy Documentation: https://docs.scipy.org/doc/
11. Quantum state preparation using tensor networks Ar.A. Melnikov, Alena Termanova, S.V. Dolgov, Florian Neukart, M.R. Perelshtein: https://www.researchgate.net/publication/371160444_Quantum_state_preparation_using_tensor_networks (2023)

12. Decompositions of general quantum gates M. Mottonen, J.J. Vartiainen: arXiv:quant-ph/0504100v1 (2005)
13. Synthesis of Quantum Logic Circuits Vivek V. Shende, Stephen S. Bullock, Igor L. Markov: arXiv:quant-ph/0406176v5 (2006)
14. Constructive Quantum Shannon Decomposition from Cartan Involutions Byron Drury, Peter J. Love: arXiv:0806.4015v1 (2008)
15. Wavefunction preparation and resampling using a quantum computer Alexei Kitaev, William A. Webb: arXiv:0801.0342v2 (2009)
16. Quantum Process Tomography on Cloud-accessible Quantum Computing Platforms Vedrukov, P.E., Ivlev, A.D., Liniov, A.V. et al. Lobachevskii J Math 45, 119–129: https://doi.org/10.1134/S1995080224010529 (2024)
17. Quantum Cloud-Original Quantum: https://qcloud.originqc.com.cn