UNIVERSIDADE FEDERAL DE SÃO CARLOS-UFSCar

CENTRO DE CIÊNCIAS EXATAS E DA TERRA-CCET
PROGRAMA DE PÓS GRADUAÇÃO EM FÍSICA

Daniel Yoshio Akamatsu

ALGORITMOS QUÂNTICOS PARA RESOLUÇÃO DE EQUAÇÕES DIFERENCIAIS: ANÁLISE DE COMPLEXIDADE E APLICABILIDADE

São Carlos-SP

2022

Daniel Yoshio Akamatsu

ALGORITMOS QUÂNTICOS PARA RESOLUÇÃO DE EQUAÇÕES DIFERENCIAIS: ANÁLISE DE COMPLEXIDADE E APLICABILIDADE

Dissertação apresentada ao Programa de Pós-Graduação em Física para obtenção do título de Mestre em Física.

Orientação Prof. Dr. Celso Jorge Villas-Boas

São Carlos-SP

2022

Dedico este trabalho aos meu pais e amigos

Agradecimentos

Agradeço ao Prof. Dr. Celso Jorge Villas-Boas pela compreensão, ajuda e paciência nesses tempos deveras obscuros que estamos atravessando. Agradeço também a meus pais, Vitar Helena dos Reis Akamatsu e Tadaomi Akamatsu, por todo esforço dedicado a me possibilitar adquirir a formação necessária par estar hoje realizando este trabalho.

Agradeço a meus amigos de Atibaia: Flávio Moura, Augusto Costa, Otávio Netto Zani e Armando Franco, e minha amiga Alessandra Valéria de Franceschi Souza, pessoas cujos momentos compartilhados comigo fazem da minha vida algo agradável.

Agradeço também ao departamento de Física da UFSCar e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pela infraestrutura e recursos disponibilizados.

Resumo

Este trabalho consiste na análise de complexidade de um algoritmo para resolução de equações diferenciais, que compartilha características como outros algoritmos designados para a mesma tarefa. Operações como preparação de estados, implementação de portas de múltiplos *q-bits* e portas controladas estão presentes em vários algoritmos, e a complexidade destas, cujos detalhes são por vezes negligenciados em análises de algoritmos novos, são tratadas detalhadamente, visando fazer uma análise de complexidade o mais fiel à realidade possível. É possível concluir que a negligência destas operações pode levar a uma análise errônea da complexidade, através de exemplos práticos.

Palavras-chave: Computação quântica, algoritmo quântico, equações diferenciais, complexidade.

Abstract

This work consists in a complexity analysis of an algorithm for solving differential equations, which shares common characteristics with other algorithms designed for the same task. Operations such as state preparation, implementation of multiple q-bit gates and controlled gates are found in several algorithms and their complexity, whose details are sometimes neglected when new algorithms are analyzed, are approached with details, intending to do an analysis as consistent with the reality as possible. It is possible to conclude that neglecting these operations can lead to a wrong complexity analysis with aid of practical examples.

Keywords: Quantum computing, quantum algorithm, differential equations, complexity.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A computação quântica é uma das ferramentas tecnológicas que surgiram com o advento da mecânica quântica, um aparato teórico moderno que possibilita a modelagem e descrição matemática de fenômenos que se mostraram indescritíveis através das teorias clássicas antes existentes .Dentre estes, é possível citar espectros discretos de emissão e absorção de alguns átomos e respostas de alguns corpos microscópicos a campos magnéticos tais quais não são observadas na predição da teoria clássica [1].

Desde que foi descoberta a possibilidade de codificação e processamento de informação utilizando sistemas quânticos [2], advento este que possibilitou a criação de computadores quânticos, tem existido procura por melhores computadores de tal categoria, bem como também por problemas cuja solução seja comprovadamente mais eficiente nestes que em um computador clássico. Nesse contexto, o termo *supremacia quântica* se refere a uma situção onde tal superioridade de sistemas quânticos seja de fato atingida através da exibição de problemas insolúveis para sistemas clássicos [3].

Um problema com aplicabilidade científica e tecnológica é a resolução de equações diferenciais, uma vez que uma ampla gama de modelos matemáticos de interesse tecnológico utilizam-nas para fazer previsões. Dentre estes, é possível citar a área de economia [4], da medicina epidemiológica [5] e da mecânica de flúidos [6]. Uma vez que soluções analíticas de equações diferenciais nem sempre são factíveis, algoritmos que possibilitam utilização de recursos computacionais tornam-se ferramentas indispensáveis [7].

A análise de complexidade é uma subárea da ciência da computação que visa estabelecer correlação entre variáveis características das entradas fornecidas (como exemplo é possível citar a dimensão de uma matriz da qual se deseja extrair o determinante) a um algoritmo e os recursos consumidos pelo mesmo para a resolução do problema para o qual foi designado (tempo e quantidade de memória, por exemplo) [8], fornecendo parâmetros objetivos para a comparação de algoritmos.

Devido à importância enfatizada, este trabalho visa discutir a análise de complexidade de algoritmos quânticos para resolução de equações diferenciais de forma detalhada, com o objetivo de verificar se os mesmos de fato apresentam ganho em termos de recursos com relação a suas contrapartes clássicas.

2 Fundamentos Teóricos

2.1 Mecânica quântica

A mecânica quântica é uma teoria que foi gradualmente desenvolvida e aprimorada ao longo do começo do sec XX, através de contribuições notáveis de cientistas como Einstein, Schrödinger, de Broglie e Bohr [9, 10, 11, 12], com o intuito de explicar o comportamento da matéria em escalas atômicas, nas quais as teorias clássicas (como o eletromagnetismo e mecânica newtoniana) falham em fazer predições adequadas.

O trabalho de Planck para explicar a falha catastrófica da estatística clássica ao descrever a radiação de corpo negro [13] já forneceu indícios de que as teorias clássicas vigentes poderiam ser inadequadas para lidar com alguns aspectos da matéria. Uma nova constante natural (h), com valor $6,626.10^{-34}$ J.s, foi descoberta e batizada constante de Planck. De Broglie foi o primeiro a atribuir comportamento ondulatório à matéria [11], com comprimento de onda como função da quantidade de movimento p:

$$\lambda = \frac{h}{p}. \tag{2.1}$$

Outros fenômenos contribuíram para a estruturação da mecânica quântica. O efeito fotoelétrico foi eficazmente explicado por Einstein, inspirado pelo trabalho de Planck, através da quantização da energia absorvida por elétrons ao interagirem com radiação eletromagnética [9]. A existência de energias discretas em espectros de emissão e absorção foram explicados de forma bem sucedida através do modelo atômico de Bohr [12] com órbitas discretas (e consequentemente níveis de energia discretos). Estas foram apenas algumas das manifestações naturais que culminaram em uma teoria ondulatória da matéria apresentada por Schrödinger, na qual a toda partícula há uma onda associada, cuja evolução temporal é regida pela equação de Schrödinger [10]:

$$-\frac{\hbar^2}{2m}\nabla^2\Psi(\mathbf{r},t) + V\Psi(\mathbf{r},t) = i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t). \tag{2.2}$$

Em tal teoria, a posição das partículas não é mais uma medida determinística, já que está associada uma distribuição de probabilidades dependente da função de onda dada pela equação (2.3).

$$Pr(\mathbf{r},t) = |\Psi(\mathbf{r},t)|^2. \tag{2.3}$$

Embora a mecânica ondulatória seja uma parte importante da teoria quântica, o formalismo atual da teoria engloba, além desta, outros aspectos não correlacionados a ondas [1]. Certas generalidades, como espectros discretos de certas variáveis características

do sistema e probabilidades de obter certos valores para um dada medição foram observadas na velha teoria quântica e na formulação ondulatória de Schrödinger, e também estão presentes na teoria quântica atual.

Um dos fenômenos de escopo puramente quântico, e cuja descrição não está relacionada a ondas é fato de entidades corpusculares regidas pela mecânica quântica exibirem *spin*. O *spin* é uma variável que determina como partículas reagem a campos magnéticos, influenciando, dentre outros fenômenos, a maneira pela qual a partícula sofre deflexão ao interagir com o campo.

A teoria quântica pode ser, de forma concisa, sintetizada em um conjunto de seis postulados, e a ferramenta matemática utilizada para formalizá-la é a álgebra linear em espaços complexos. Uma descrição detalhada da teoria pode ser encontrada na literatura especializada [1, 14].

2.1.1 Postulados da mecânica quântica

2.1.1.1 Primeiro Postulado

O primeiro Postulado da Mecânica Quântica afirma que o estado de um sistema em um dado instante "t" é descrito por um vetor *ket* $|\psi\rangle$ normalizado em um espaço de Hilbert *H*. Espaços de Hilbert são espaços vetoriais complexos dotados de produto interno, porém não necessitam ser finitamente gerados. Isto é, um dado vetor pertencente ao espaço pode ser escrito como combinação linear de infinitos elementos de base. Sistemas como o oscilador harmônico quântico, por exemplo, são descritos por um espaço de dimensão infinita [1]. Denomina-se vetor *bra* o vetor dual correspondente ao *ket*, o qual é denotado por $\langle\psi|$.

Caso o conjunto (não necessariamente finito) dos elementos $|i\rangle$ seja uma base do espaço que descreve o sistema, então é possível representar o estado deste através de um conjunto $\{\alpha_i\}$ de coeficientes complexos:

$$|\psi\rangle = \sum \alpha_i |i\rangle . \tag{2.4}$$

O vetor *bra* correspondente será consequentemente descrito por:

$$\langle\psi| = \sum \alpha_i^* \langle i| . \tag{2.5}$$

O produto interno entre dois vetores $|\psi\rangle$ e $|\phi\rangle$ é denotado por $\langle\psi|\phi\rangle$, e obedece à condição $\langle\psi|\phi\rangle = \langle\phi|\psi\rangle^*$. Uma dada base formada pelo conjunto dos $|i\rangle$ é ortogonal caso seja obedecida a condição estabelecida pela equação (2.6). Neste caso, a condição de normalização implica na equação (2.7)

$$\langle i|j\rangle = \delta_{ij}. \tag{2.6}$$

$$\langle\psi|\psi\rangle = \sum \alpha_i^* \alpha_i = 1. \tag{2.7}$$

Tanto a condição de normalização quanto os coeficientes α_i possuem significados físicos importantes que serão posteriormente discutidos.

2.1.1.2 Segundo Postulado

O segundo Postulado estabelece que observáveis como energia, momentum e posição (dentre outros), devem ser representados por operadores hermitianos, que são transformações lineares denotadas por $\hat{\mathbf{O}} : H \to H$, cuja atuação leva um vetor no espaço de Hilbert a outro neste mesmo espaço, preservando todas as propriedades de transformações lineares cujo escopo detalhado foge à presente discussão, mas que podem ser consultadas em [15], bastando enfatizar a ação deste operador em cada um dos elementos da base:

$$\hat{\mathbf{O}} \left|j\right\rangle = \sum O_{ij} \left|i\right\rangle . \tag{2.8}$$

Para uma dada base ortogonal, é possível então obter uma representação matricial de $\hat{\mathbf{O}}$:

$$\left\langle i\right| \hat{\mathbf{O}} \left|j\right\rangle = O_{ij}. \tag{2.9}$$

O operador adjunto correspondente a $\hat{\mathbf{O}}$ é definido por:

$$\left\langle \hat{O}^\dagger \phi \middle| \psi \right\rangle = \left\langle \phi \middle| \hat{O}\psi \right\rangle \to O_{ij}^\dagger = O_{ji}^*. \tag{2.10}$$

Um operador é dito hermitiano se ele e seu adjunto são iguais. Tal classe apresenta a importante propriedade de possuir autovalores reais e autovetores ortogonais.

2.1.1.3 Terceiro Postulado

O terceiro Postulado afirma que a medição de uma grandeza física associada a um observável $\hat{\mathbf{O}}$ poderá fornecer como resultado somente um dos autovalores de $\hat{\mathbf{O}}$. A teoria dos espaços de Hilbert é suficientemente ampla e elegante para englobar espectros contínuos de autovalores (e evidentemente isto é uma exigência, visto que não se observa grandezas como posição apresentarem valores discretos mesmo para sistemas quânticos), mas os espectros discretos que foram observados antes da formalização completa da Mecânica Quântica estão, de certa forma, relacionados ao terceiro Postulado.

O operador associado à energia como observável física, por exemplo, denomina-se Hamiltoniano e, para um oscilador harmônico de frequência ν, possui espectro de autovalores descrito por funções de números inteiros:

$$E = (n + \frac{1}{2})h\nu. \tag{2.11}$$

Este espectro explica as energias quantizadas observadas no efeito fotoelétrico.

2.1.1.4 Quarto Postulado

O quarto Postulado estabelece que a probabilidade de se obter um determinado autovalor λ na medição de um observável $\hat{\mathbf{O}}$ é dada pelo quadrado do módulo do produto interno entre $|\lambda\rangle$ (autovetor de $\hat{\mathbf{O}}$ cujo autovalor é λ) e o vetor de estado do sistema. Matematicamente, o quarto Postulado pode ser expresso pelas equações (2.12) e (2.13).

$$P_\lambda = |\langle\lambda|\psi\rangle|^2; \tag{2.12}$$

$$dP_\lambda = |\langle\lambda|\psi\rangle|^2 d\lambda. \tag{2.13}$$

A equação (2.13) é a contraparte da (2.12) para espectros contínuos.

Se $|\lambda_i\rangle$ for o conjunto dos possíveis autovetores de um observável $\hat{\mathbf{O}}$, a condição de hermiticidade implica que este forma uma base ortonormal, e portanto as probabilidades associadas às medições devem obedecer a seguinte relação:

$$|\psi\rangle = \sum_i \alpha_i |\lambda_i\rangle \rightarrow P(\lambda_i) = |\alpha_i|^2. \tag{2.14}$$

Para um estado expandido em termos dos autovetores de um dado observável, o fato de os coeficientes da expansão estarem associados a probabilidades implica no valor médio do observável em questão descrito pela seguinte relação:

$$\langle O\rangle = \langle\psi| \hat{\mathbf{O}} |\psi\rangle . \tag{2.15}$$

Esta relação é consequência direta da ortogonalidade entre autovetores, e tal média deve ser interpretada como média de *ensamble*, ou seja, se for efetuada uma quantidade muito grande de medições do observável $\hat{\mathbf{O}}$ em vários sistemas distintos, porém idênticos, o valor médio obtido será dado pela equação (2.15).

2.1.1.5 Quinto Postulado

O quinto Postulado estabelece o que ocorre com o sistema imediatamente após a realização de uma medição. O efeito destas pode ser representado por um conjunto de operadores $\hat{\mathbf{M}}_\mathbf{i}$, cada um associado a um dos possíveis resultados, e imediatamente após a medição, o estado do sistema é alterado de acordo com a seguinte relação:

$$|\psi\rangle \rightarrow \frac{\hat{\mathbf{M}}_\mathbf{i} |\psi\rangle}{\sqrt{\langle\psi| \hat{\mathbf{M}}_\mathbf{i} |\psi\rangle}}. \tag{2.16}$$

A equação (2.16) revela que a medição efetuada em um sistema quântico pode alterar consideravelmente seu estado. De fato, excetuando-se os casos nos quais o sistema já se encontra em um estado descrito por um dos autoestados da observável $\hat{\mathbf{O}}$, este será

drasticamente alterado após a medição, uma vez que esta implica em uma projeção do vetor de estados no autovetor correspondente ao autovalor encontrado como resultado. Desta maneira, estudos descritivos a respeito de sistemas quânticos podem requerer, em um âmbito mais geral, o uso de outros processos ferramentas estatísticas (como tomografia de estados [16]).

Há, associado a cada um dos possíveis valores mensuráveis de um observável (valores estes caracterizados pelo conjunto de autovalores e autovetores $(\lambda_i,|i\rangle)$), um operador que representa o efeito da medição do observável em questão no sistema, cuja notação representativa, bem como também a condição geral que deve ser satisfeita pelo conjunto de operadores, é expressa nas equações (2.17) e (2.18) para variáveis discretas e contínuas, respectivamente.

$$\sum_i |i\rangle \langle i| = \hat{\mathbf{I}}. \tag{2.17}$$

$$\int_{-\infty}^{\infty} |\tau\rangle \langle \tau| \, d\tau = \hat{\mathbf{I}}. \tag{2.18}$$

As equações (2.17) e (2.18) denominam-se relações de completeza, e podem ser deduzidas (no caso destas medições específicas) a partir da multiplicação de $|\psi\rangle$ pela soma expressa nestas, que resulta no próprio $|\psi\rangle$, ou seja, é equivalente à identidade.

A notação utilizada para representar operadores como $|i\rangle \langle i|$, por exemplo, é um símbolo para produto direto, o qual obedece as propriedades listadas nas equações (2.19)-(2.22). Este tipo de notação é prática do ponto de vista operacional.

$$|\alpha\rangle \langle \beta|\gamma\rangle = (\langle \beta|\gamma\rangle) |\alpha\rangle , \tag{2.19}$$

$$\langle \alpha|\beta\rangle \langle \gamma| = (\langle \alpha|\beta\rangle) |\gamma\rangle , \tag{2.20}$$

$$(|\alpha\rangle + |\beta\rangle) \langle \gamma| = |\alpha\rangle\langle\gamma| + |\beta\rangle\langle\gamma| , \tag{2.21}$$

$$(c |\alpha\rangle) \langle \beta| = |\alpha\rangle (c \langle \beta|) = c |\alpha\rangle\langle\beta| . \tag{2.22}$$

Os operadores $|i\rangle\langle i|$ relacionados na equação (2.17) são chamados de projetores, uma vez que sua atuação em um vetor de estados obedece a equação (2.23) como consequência das propriedades anteriormente listadas e da formalização do quinto Postulado (equação (2.16)).

$$|\psi\rangle = \sum_i \alpha_i |i\rangle \rightarrow |j\rangle\langle j| \, |\psi\rangle = |j\rangle . \tag{2.23}$$

As medições descritas até então, que podem ser representadas por projetores, denominam-se medições projetivas e não são a classe mais geral possível de medições.

A generalização do conceito de medição envolve também a generalização do conceito de estado quântico, a qual pertence ao escopo da mecânica estatística quântica, que estuda não só estados puros (associados a um único vetor unitário), mas também sistemas aos quais é possível associar probabilidades de existência em diferentes estados. Neste contexto mais amplo, ainda assim é possível afirmar que uma medição consiste em associar um conjunto de possíveis eventos (e suas respectivas probabilidades) a um estado generalizado. A cada evento, há também um operador $\hat{\mathbf{M}}_i$ associado, de tal forma que o conjunto total destes obedeça a uma relação de completeza:

$$\sum_i \hat{\mathbf{M}}_i^\dagger \hat{\mathbf{M}}_i = \hat{\mathbf{I}}. \tag{2.24}$$

Uma das características da Mecânica Quântica que a diferencia da Mecânica Clássica é a existência de observáveis que não podem ser medidas simultaneamente, para as quais a medição precisa de uma prejudica o grau de precisão associado à outra. Um exemplo conhecido é o caso da posição e do *momentum*, que obedecem o princípio da incerteza de Heisenberg:

$$\sigma_x \sigma_p \geq \frac{\hbar}{2}. \tag{2.25}$$

O que determina se duas observáveis $\hat{\mathbf{O}}_1$ e $\hat{\mathbf{O}}_2$ podem ser medidas simultaneamente é a relação de comutação existente entre elas:

$$[\hat{\mathbf{O}}_1, \hat{\mathbf{O}}_2] = \hat{\mathbf{O}}_1\hat{\mathbf{O}}_2 - \hat{\mathbf{O}}_2\hat{\mathbf{O}}_1. \tag{2.26}$$

Se os observáveis comutam (isto é, $[\hat{\mathbf{O}}_1, \hat{\mathbf{O}}_2] = 0$), então a medição pode ser realizada de forma simultânea com precisão arbitrariamente alta para ambos. Caso contrário, é inevitável que a medição precisa de um afete a precisão do outro. Para a posição e o *momentum*, vale relação descrita pela equação (2.27), ou seja, estes não comutam, e por isso observa-se a ocorrência intrínseca de incerteza descrita pela equação (2.25).

$$[\hat{\mathbf{x}}, \hat{\mathbf{p}}_\mathbf{x}] = i\hbar. \tag{2.27}$$

2.1.1.6 Sexto Postulado

O sexto Postulado concerne a dinâmica dos sistemas regidos pela Mecânica Quântica, e estabelece que a evolução temporal destes é regida pela equação de Schrödinger, na qual $\hat{\mathbf{H}}$ é o Hamiltoniano, o mesmo operador associado à energia E como observável física:

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = \hat{\mathbf{H}} |\psi\rangle . \tag{2.28}$$

É comum referenciar a equação (2.28) como equação de Schrödinger dependente do tempo, e a equação de autovetores e autovalores relacionada à energia ((2.29)) como equação independente do tempo:

$$\hat{\mathbf{H}} |\psi\rangle = E |\psi\rangle . \tag{2.29}$$

É importante ressaltar que a equação (2.2) representa um caso particular da equação de Schrödinger (2.28): Neste caso, a variável que representa a posição x está associada a um espectro contínuo de valores. Se $|i\rangle$ for um dos possíveis autovetores de um observável, então a amplitude α_i dada por $\langle i|\psi\rangle$ pode também ser expressa como uma função de onda. No caso das variáveis contínuas, torna-se conveniente fazer manipulações algébricas com o intuito de reescrever a equação (2.28) em termos da função de onda, por esta ser um objeto mais apropriado para cálculos.

A integração da equação (2.28) resulta na equação de evolução temporal de sistemas quânticos:

$$|\psi(t)\rangle = U(t)\,|\psi_0\rangle\,, \tag{2.30}$$

$$U(t) = exp(-i/\hbar\hat{\mathbf{H}}t). \tag{2.31}$$

Ou seja, a evolução temporal do sistema é dada pela ação de um operador unitário que é a função do Hamiltoniano descrita pela equação (2.31). Operadores unitários preservam a norma dos vetores nos quais agem e obedecem a seguinte condição:

$$\hat{\mathbf{U}}^{\dagger}\hat{\mathbf{U}} = \hat{\mathbf{U}}\hat{\mathbf{U}}^{\dagger} = \hat{\mathbf{I}}. \tag{2.32}$$

Caso a ação não obedecesse esta condição, o estado obtido após a evolução poderia não ter significado físico, estando inclusive em desacordo com o primeiro e o quarto Postulados.

2.1.2 Partículas de *spin* 1/2

O *spin* é um excelente exemplo de uma variável quântica que não possui análogo clássico. Assim como o elétron foi uma das primeiras partículas subatômicas a serem descobertas, seu respectivo *spin* constitui-se como a primeira grandeza desta nova classe de objetos a ser observada através de experimentos. Atualmente, são conhecidas outras partículas elementares como quarks, múons, pósitrons, fótons, dentre outras. O fato de exibirem *spin* múltiplo de semi-inteiros (1/2,3/2,...) ou de inteiros (0,1,2,...) permite classificar as partícula como férmions no primeiro caso e como bósons, no segundo.

A classe dos férmions engloba uma série de partículas que formam a matéria com a qual a humanidade tem contato corriqueiro (sendo o próprio elétron um exemplo), enquanto a dos bósons engloba os entes mediadores das forças naturais (como exemplo, há o fóton, que é o *quantum* da força eletromagnética) [17].

A importância do *spin* não se deve só a sua relevância teórica e ao fato de ser uma excelente ferramenta didática para compreensão dos Postulados da Mecânica Quântica, mas também ao fato de ser um sistema quântico de dois níveis , que é representado por um

ket bidimensional (outro exemplo são fótons com seu estado de polarização). Este tipo de sistema (bem como as técnicas empregadas para manipulá-los) são de suma importância para a computação quântica.

2.1.2.1 Experimento de *Stern-Gerlach*

O procedimento mais conhecido associado à detecção da grandeza *spin* é o experimento de *Stern-Gerlach*, que consiste em submeter feixes de átomos de prata aquecidos em um forno a sequências de campos magnéticos orientados em diferentes direções, e observar as implicações.

Inicialmente, o feixe de átomos é submetido a um campo magnético em uma direção específica, que é definida (por convenção) como sendo o eixo z. Observa-se então a ocorrência de uma bifurcação, conforme ilustrado na Fig. 1.

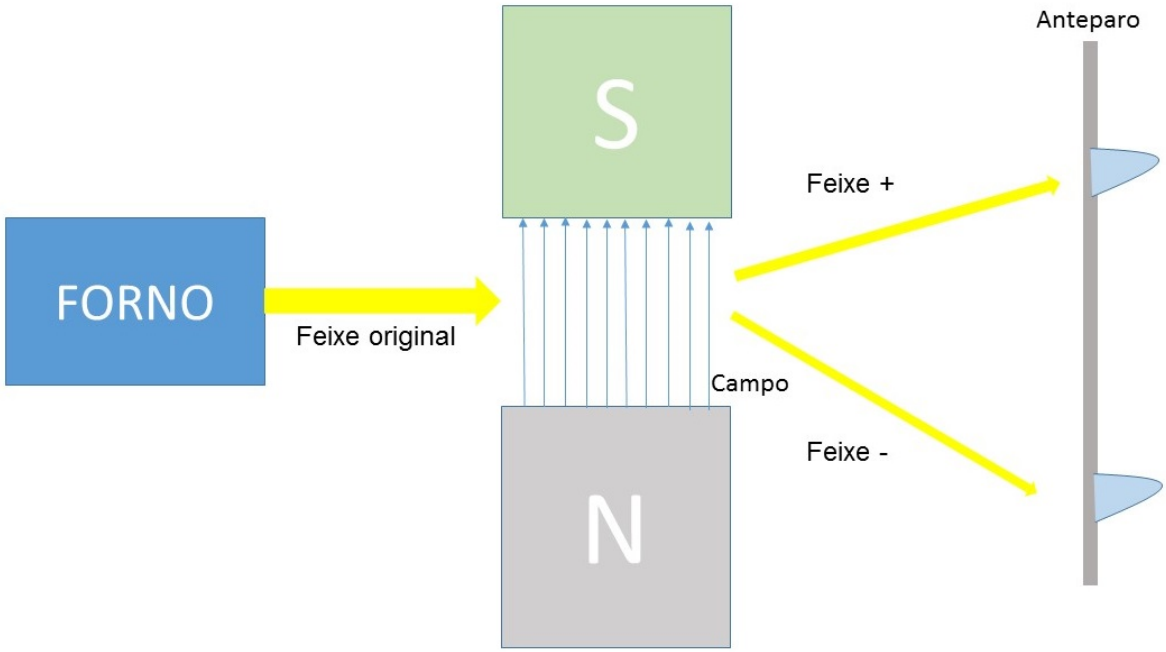

Figura 1 – Experimento de Stern-Gerlach. FONTE: Elaborada pelo próprio autor.

Ao atravessar a região que contém o campo magnético expresso por $\vec{\mathbf{B}} = B_z\hat{\mathbf{z}}$, o átomo sofre uma força determinada pela direção do momento magnético μ. Observa-se que a direção para a qual o feixe é desviado depende, portanto, do sinal do momento magnético μ_z, conforme estabelece a relação a seguir:

$$F_z = \mu_z \frac{\partial B_z}{\partial z}. \tag{2.33}$$

O que se esperaria como resultado clássico típico seria a ocorrência de um pico de detecção de larga abertura no anteparo, uma vez que o valor de μ_z ficaria livre para oscilar entre um valor máximo e um valor mínimo (denotados, respectivamente, por μ e $-\mu$). O

observado, porém, é a existência de dois picos estreitos e bem definidos, conforme ilustrado na Fig. 1. Logo, o momento magnético é mais um exemplo de grandeza quantizada, ou seja, uma variável que assume um conjunto finito de valores bem definidos (neste caso, apenas dois).

Este fato pode ser explicado de forma sucinta e elegante pelo formalismo expresso nos seis postulados discutidos anteriormente.

O momento magnético na direção z pode ser definido como uma grandeza proporcional a um observável $\hat{\mathbf{S}}_z$ ($u_z \propto \hat{\mathbf{S}}_z$), cujos autovalores possuem o mesmo valor absoluto, mas diferem quanto ao sinal. Há, desta maneira, dois estados que podem ser assumidos pelo átomo: $|+\rangle$ e $|-\rangle$, correspondentes aos dois respectivos autovalores de $\hat{\mathbf{S}}_\mathbf{z}$ ($\hbar/2$ e $-\hbar/2$). Portanto, o estado do átomo pode ser descrito pela equação (2.34), enquanto os elementos da matriz que representa o operador $\hat{\mathbf{S}}_z$ em função de seus autoestados são dados pelas equações (2.35) e (2.36)

$$|\psi\rangle = \alpha_+ |+\rangle + \alpha_- |-\rangle \,, \tag{2.34}$$

$$\langle +| \hat{\mathbf{S}}_\mathbf{z} |+\rangle = \hbar/2, \langle -| \hat{\mathbf{S}}_\mathbf{z} |-\rangle = -\hbar/2, \langle -| \hat{\mathbf{S}}_\mathbf{z} |+\rangle = \langle -| \hat{\mathbf{S}}_\mathbf{z} |+\rangle = 0, \tag{2.35}$$

$$\hat{\mathbf{S}}_z = \frac{\hbar}{2}\hat{\sigma}_z, \ \hat{\sigma}_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.36}$$

O experimento revela resultados surpreendentes que contradizem o padrão esperado pela intuição baseada na Física Clássica, como ilustrado na Fig. 2.

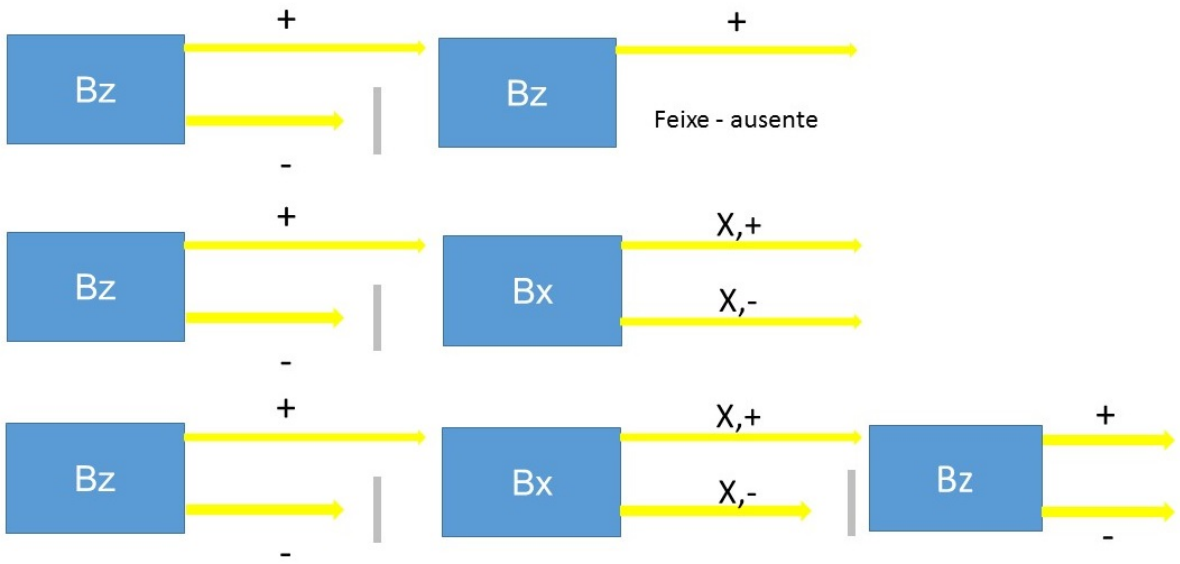

Figura 2 – Arranjos sequenciais de Stern-Gerlach
. FONTE: Elaborada pelo próprio autor.

No primeiro caso, o padrão observado concorda com ponto de vista intuitivo, ou seja, o feixe que teve sua componente negativa bloqueada continua apresentando apenas a componente positiva ao passar novamente pelo campo direcionado ao longo do eixo x. Caso o feixe + resultante da ação do campo na direção z atravesse o campo direcionado em x, resultará uma nova bifurcação, que poderia facilmente levar o observador a deduzir que o feixe - é composto de frações idênticas de átomos com componente positiva e negativa em x. O que ocorre, porém, caso o feixe X,+ seja submetido novamente ao campo na direção z mostra que esse não é o caso, pois este, que supostamente deveria ser composto unicamente de átomos com componente positiva de momento magnético na direção z exibe novamente as duas componentes (+ e -). Ou seja, submeter os feixes a campos em diferentes direções modifica drasticamente o estado destes, resultado este que está de acordo com o quinto postulado da mecânica quântica. Assim, quando um átomo atravessa um campo que está alinhado em uma determinada direção, sua configuração de spin será projetada para um dos autoestados associados ao operador que representa esse campo. Apesar de certas características antiintuitivas, o resultado observado permite, ainda assim, afirmar que um átomo no estado $|X,+\rangle$ possui igual probabilidade de apresentar tanto + quanto - como resultado para uma medição de S_z. De fato, os estados citados se correlacionam de acordo com a seguinte relação:

$$|X,+\rangle = \frac{|+\rangle + |-\rangle}{\sqrt{2}}. \tag{2.37}$$

A realização de tais experimentos sequencias com as devidas sucessões nas direções x, y e z permite correlacionar as bases de autovetores de $\hat{\mathbf{S}}_\mathbf{z}, \hat{\mathbf{S}}_\mathbf{y}$ e $\hat{\mathbf{S}}_\mathbf{x}$, para obter relações semelhantes às da Eq. (2.37) e construir as respectivas matrizes associadas a estes operadores:

$$\hat{\mathbf{S}}_z = \frac{\hbar}{2}\hat{\sigma}_z, \hat{\sigma}_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.38}$$

$$\hat{\mathbf{S}}_y = \frac{\hbar}{2}\hat{\sigma}_y, \hat{\sigma}_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \tag{2.39}$$

$$\hat{\mathbf{S}}_x = \frac{\hbar}{2}\hat{\sigma}_x, \hat{\sigma}_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2.40}$$

As matrizes $\hat{\sigma}_x$, $\hat{\sigma}_y$ e $\hat{\sigma}_z$ denominam-se matrizes de Pauli, e em conjunto com a matriz identidade, formam uma base do espaço vetorial das matrizes bidimensionais. Tais matrizes são importantes para a computação quântica, visto que estão relacionadas a operações fundamentais presentes nesta [8].

Ademais, as direções x, y e z não são as únicas nas quais é possível efetuar a medição do *spin* ou momento magnético. Para uma dada direção arbitrária representada por um vetor unitário $\tilde{\mathbf{u}} = \frac{1}{r}(x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}})$, haverá um operador de *spin* dado pela equação

(2.41).

$$\tilde{\sigma} = \hat{\sigma}_x\hat{\mathbf{x}} + \hat{\sigma}_y\hat{\mathbf{y}} + \hat{\sigma}_z\hat{\mathbf{z}};\ \hat{\mathbf{S}}_\mathbf{u} = \tilde{\mathbf{u}}.\tilde{\sigma} = \frac{1}{r}(x\hat{\sigma}_x + y\hat{\sigma}_y + z\hat{\sigma}_z). \tag{2.41}$$

2.1.2.2 Evolução temporal do *spin*

O experimento de Stern-Gerlach e suas respectivas conclusões são exemplos de cenários cuja compreensão está fundamentada nos cinco primeiros Postulados, mas não exibem ocorrência de evolução temporal, ou seja, não é evidenciada uma manifestação natural do sexto Postulado.

Um exemplo de situação descritível pelo sexto Postulado é a evolução de uma partícula com *spin*.

O operador relacionado à energia é o Hamiltoniano, que no caso de sistemas com *spin* é dado pela equação (2.42), com o "vetor" $\overrightarrow{\mathbf{S}}$ dado por $\overrightarrow{\mathbf{S}} = (\hat{\mathbf{S}}_\mathbf{x}, \hat{\mathbf{S}}_\mathbf{y}, \hat{\mathbf{S}}_\mathbf{z})$.

$$\hat{\mathbf{H}} = \left(-\frac{e}{m_e c}\right)\vec{\mathbf{S}}.\vec{\mathbf{B}}. \tag{2.42}$$

Para um campo magnético orientado puramente na direção z, resultará as seguintes relações para $\hat{\mathbf{H}}$ e seus respectivos autovalores:

$$\hat{\mathbf{H}} = \left(-\frac{e}{m_e c}\right)B_z\hat{\mathbf{S}}_z. \tag{2.43}$$

$$E\pm = \mp\frac{e\hbar B_z}{2m_e c}. \tag{2.44}$$

O operador de evolução para esse sistema será consequentemente dado por:

$$\hat{\mathbf{U}}(\mathbf{t}) = exp\left(\frac{-i\omega\hat{\mathbf{S}}_\mathbf{z}\mathbf{t}}{\hbar}\right), \omega = \frac{|e|B_z}{m_e c}. \tag{2.45}$$

Uma função de um dado operador $\hat{\mathbf{O}}$, $f(\hat{\mathbf{O}})$, compartilha com este os autovetores, e se λ_i for um autovalor de $\hat{\mathbf{O}}$ correspondente ao autovetor $|i\rangle$, então $f(\lambda_i)$ será autovalor de $f(\hat{\mathbf{O}})$ para o mesmo autovetor. Isso se deve à expansão em série de Taylor da função f [18]. Consequentemente, se o estado inicial for expresso em termos dos autovetores de $\hat{\mathbf{S}}_\mathbf{z}$ segundo a equação (2.46), então após a evolução este será dado pela equação (2.47).

$$|\alpha_0\rangle = c+|+\rangle + c-|-\rangle, \tag{2.46}$$

$$|\alpha(t)\rangle = \hat{\mathbf{U}}(\mathbf{t})|\alpha_0\rangle = c+exp\left(\frac{-i\omega t}{2}\right)|+\rangle + c-exp\left(\frac{i\omega t}{2}\right)|-\rangle. \tag{2.47}$$

Os valores das constantes $c\mp$ dependem da configuração do estado inicial, de tal forma que se este for o estado $|X,+\rangle$ dado pela equação (2.37), por exemplo, então $c+ = c- = 1/\sqrt{2}$.

Ter uma equação como (2.47) descrevendo o estado do sistema significa que os valores esperados irão mudar com o tempo também. O observável $\hat{\mathbf{S}}_\mathbf{x}$, por exemplo, que tem como autovetores $|X,+\rangle$ e $|X,-\rangle$ e é descrito pela equação (2.48), possui valor médio descrito pela equação (2.49).

$$|\alpha_0\rangle = |X,-\rangle = \frac{|+\rangle - |-\rangle}{\sqrt{2}}, \tag{2.48}$$

$$\langle S_x\rangle = \langle\alpha(t)|\,\hat{\mathbf{S}}_\mathbf{z}\,|\alpha(t)\rangle = \frac{\hbar}{2}cos(\omega t). \tag{2.49}$$

Novamente, é válido ressaltar que tal valor médio se refere a vários sistemas supostamente idênticos analisados no mesmo instante. Este exemplo apresenta o procedimento comumente adotado para analisar a evolução de sistemas quânticos com variáveis discretas. Uma vez que o Hamiltoniano é o operador relacionado à energia como observável, e também determina a evolução temporal dos sistemas, convém resolver a equação de Schrödinger independente do tempo (equação (2.29)) e expandir o estado do sistema em termos dos autovetores de $\hat{\mathbf{H}}$. Em seguida, utiliza-se a relação entre os autovetores de uma função de um operador e os do operador (no caso $\hat{\mathbf{H}}$) para determinar o resultado da evolução temporal.

O caso de variáveis contínuas, apresenta algumas similaridades. Neste caso, resolve-se a equação para a função de onda ((2.2)) através do método de separação de variáveis para gerar duas equações subsequentes, onde apenas uma é dependente do tempo. A independente do tempo é resolvida primeiro, e as autofunções espaciais (ou de *momentum*) encontradas permitem determinar e evolução temporal.

Caráter contínuo ou discreto não são características de sistemas específicos, mas sim do conjunto de observáveis de interesse. O oscilador harmônico quântico, por exemplo, exibe um espectro discreto de energias, mas é um sistema cuja essência física faz desejável conhecimento a respeito da posição. Este sistema pode ser analisado tanto através da mecânica ondulatória quanto através de métodos algébricos como o exemplificado nessa seção. A escolha do método depende de quais observáveis deseja-se extrair informação a respeito.

2.2 Complexidade computacional

A análise de complexidade computacional está inserida em um campo de conhecimento extremamente relevante para a ciência da computação e áreas correlatas, pois é a principal ferramenta disponível para classificar algoritmos de forma comparativa, utilizando como critério o quanto estes consomem de algum recurso pré-estabelecido para a resolução da tarefa para a qual foram designados.

O tipo de recurso a ser analisado depende consideravelmente do modelo computacional e do estado da arte da tecnologia [8]. As estruturas miniaturizadas atualmente disponíveis na área eletrônica (memórias, circuitos, dentre outros) tornam a preocupação com espaço menos crucial que em tempos nos quais os elementos possuíam maior escala e poucos *bits* disponíveis.

Para quantificar a complexidade computacional, define-se um recurso (o mais comumente analisado é o tempo de execução, mas por vezes também há interesse em recursos como energia e espaço), e estuda-se como este varia como função de algum parâmetro característico das variáveis iniciais que são fornecidas ao algoritmo em análise para o processamento que levará à resolução do problema. Um exemplo simples é o procedimento para calcular o traço de uma matriz M de dimensão n (aqui apresentado em pseudocódigo).

Algoritmo 1 Traço de uma matriz

```
traço ← 0
for i ← 0 to n − 1 do
    traço ← traço + M[i][i]
end for
```

Nota-se, neste algoritmo, que o comando de somar o i-ésimo elemento da diagonal da matriz à variável que armazena o valor do traço será repetido n vezes. Cada comando referente a soma demanda um intervalo de tempo constante K, então o tempo total consumido pelo algoritmo é $t = Kn$.

A função que expressa a relação entre o consumo de recurso e as variáveis da entrada denomina-se complexidade. A análise de complexidade é sempre realizada considerando o melhor algoritmo conhecido para um determinado problema, e costuma-se expressar ordens assintóticas das funções, não valores exatos. O principal objetivo da análise de complexidade é estabelecer, matematicamente, limites inferiores para o consumo de recursos [8], mas existem situações nas quais limites inferiores ou ordens exatas são almejados.

Há três notações amplamente utilizadas para representar ordem assintótica das funções de complexidade: O, Ω e Θ. A ordem O diz respeito ao limite superior das funções, ou seja: Dadas duas funções $f(n)$ e $g(n)$, diz-se que $f(n)$ é $O(g(n))$ se existem duas constantes c e n_0 tais que se $n \geq n_0$, então $f(n) \leq cg(n)$. Esta notação é utilizada para expressar como o algoritmo se comporta no pior caso possível.

A ordem Ω representa a situação oposta à O, e representa o comportamento do algoritmo para o melhor caso possível: O limite inferior. Diz-se que uma função $f(n)$ é $\Omega(g(n))$ se existem constante n_0 e c tais que se $n \geq n_0$, então $f(n) \geq cg(n)$.

Por fim, há a ordem Θ, que representa o comportamento assintótico exato. Diz-se que uma função $f(n)$ é $\Theta(g(n))$ se existe n_0 tal que se $n \geq n_0$, então $f(n)$ é simultaneamente

$\Omega(n)$ e O(n). O algoritmo utilizado para calcular o traço de uma matriz possui ordens O e Ω que seguem a mesma variação em relação a n. Neste caso, O(n) e $\Omega(n)$ são idênticas, e consequentemente a ordem deste algoritmo é $\Theta(n)$. Qualquer constante numérica é desprezada no cálculo destas ordens.

Alguns exemplos de funções e suas ordens assintóticas são apresentados na tabela 1 a seguir.

Tabela 1 – Ordens assintóticas.

Função	**O**	**Ω**	**Θ**
$f(n) = 3n$	O(n), $f(n) \leq (3n)$,	$\Omega(n)$,f(n)$\geq n$	$\Theta(n)$
$f(n) = 7n^2 + \sqrt{(n)log(n)}$	O(n^2) $f(n) \leq (7n^2)$	Ω(log(n)),f(n)$\geq log(n)$	N.A

É importante que não haja confusão entre os limites inferiores e superiores das funções, expressos na tabela 1, e o limite inferior de eficiência para os algoritmos. Quando afirma-se que o limite inferior de eficiência para a resolução de um dado problema é polinomial, por exemplo, não significa que utilizou-se a ordem Ω na análise em questão, mas tão somente que analisou-se a ordem assintótica O ou θ para o melhor algoritmo conhecido para tal tarefa.

Uma vez escolhidos os recursos de interesse (tempo, energia, espaço, dentre outros) e estabelecidas as ordens assintóticas de suas respectivas variações com relação às variáveis de entrada (a dimensão de uma matriz ou vetor, por exemplo), é possível agrupar os algoritmos em classes de complexidade.

Algumas classes notáveis são as classes P e NP, que dizem respeito a problemas de decisão, que podem ser solucionados através de respostas como "sim" ou "não", ou seja, consistem em rejeitar ou aceitar uma hipótese. Um exemplo é a decisão sobre a primalidade de um número. A hipótese é "o número N é primo". Após a execução do algoritmo, aceita-se a hipótese com uma resposta "sim" ou rejeita-se, com a resposta "não".

A classe P diz respeito a um conjunto de problemas de decisão cuja solução consome um tempo polinomial com relação a uma determinada variável "n"da entrada, ou seja, t=O(n). A classe NP diz respeito a problemas cuja solução pode não ser factível em tempo polinomial, mas se for apresentada uma aceitação da hipótese, em conjunto com uma variável w denominada testemunha, então é fácil checar se esta reposta está correta (isto pode ser executado em tempo polinomial). Na fatoração de um número, por exemplo, que pode ser definida pela hipótese "Um dado número N possui um fator menor que l, sendo $l < N$", caso haja a aceitação da hipótese e seja exibido um número w que cumpra os devidos requisitos, então é fácil verificar se esta resposta em prol da aceitação está correta fazendo a divisão de N por w, que é executável em tempo polinomial.

Um dos problemas importantes ainda sem solução na ciência da computação é

a relação existente entre P e NP. Conjectura-se que P seja um subconjunto de NP, ou seja, $P \subset NP$, restando saber se os conjuntos são ou não iguais. Uma classe diretamente correlacionada a este problema é a NP-completa, que consiste em um subconjunto de NP que possui a particularidade de que qualquer outro problema em NP pode ser modelado de forma equivalente a algum elemento NP-completo, e isto implica que a resolução de um destes problemas em tempo polinomial automaticamente prova que NP=P. Tal demonstração garantiria um limite inferior de eficiência para uma quantidade muito grande de algoritmos, de tal forma que esse feito acarretaria em um grande avanço nas ciências da computação.

Um exemplo de problema NP completo é encontrar soluções inteiras e positivas de um polinômio descrito por:

$$Ax + By - C = 0. \tag{2.50}$$

Os fatores A,B,C são inteiros positivos. É fácil verificar através de uma substituição direta se valores testemunha fornecidos de x, y são soluções, logo esse problema é claramente NP. Detalhes de como outros problemas NP podem ser reduzidos a este estão além do escopo do presente trabalho, e podem ser consultados em [19].

Algumas classes de complexidade dizem respeito a problemas cuja solução pode ser dada por um algoritmo probabilístico, ou seja, um algoritmo que ao final de sua execução exibe uma probabilidade p de apresentar um solução correta.

Um exemplo de algoritmo probabilístico é o *random sort*, que consiste em um processo que pode levar ao ordenamento de um dado arranjo (um conjunto de elementos com posições definidas, que é por vezes referido na computação como vetor, de tal forma que esta palavra adquire, neste contexto, um significado diferente da definição matemática). O processo consiste em sortear novas posições para cada um dos elementos do arranjo, e realocá-los nestas posições. Se o número de elementos do arranjo for N, nota-se que a probabilidade de sucesso de tal algoritmo é 1/N!, de tal forma que necessita-se, em média, repetir o processo N! vezes até obter o sucesso. Muito embora esse processo seja extremamente demorado e custoso, ilustra as características deste tipo de algoritmo.

A classe BPP diz respeito a algoritmos probabilísticos. Sua nomenclatura é derivada de *bounded-error probabilistic polinomial time*, e esta inclui os problemas de decisão que podem ser resolvidos em tempo polinomial com uma probabilidade superior a 2/3 de fornecer a resposta correta (seja ela "sim" ou "não"). A escolha do limite 2/3 é arbitrária, e assim como no caso do *random sort*, é possível repetir o processo de tal forma a aproximar tanto quanto se deseje a probabilidade de sucesso de um.

Devido à natureza estatística de sistemas quânticos (evidenciada pelo quarto Postulado) é esperado que algoritmos derivados destes exibam natureza probabilística, uma vez que estão comumente condicionados à medição de estados específicos. Embora

classe BQP seja equivalente à BPP para algoritmos quânticos, ela engloba problemas de decisão que podem ser solucionados em tempo polinomial em um computador quântico, com probabilidade de acerto superior a 2/3.

2.3 Computação quântica

A computação quântica é um ramo de pesquisa e desenvolvimento de tecnologias que surgiu com a possibilidade de utilizar/controlar sistemas quânticos para processamento de informação [2]. Costuma-se usar computação quântica como termo oposto a computação clássica, visto que estas diferem tanto nos fundamentos quanto nas características dos sistemas físicos utilizados.

Em primeiro lugar, enquanto a computação clássica usa como unidade básica de informação o *bit*, a computação quântica usa o *q-bit*. O *bit* é uma variável que pode assumir dois valores discretos, denotados 0 e 1, que podem representar cenários como ausência ou presença de corrente elétrica ou a magnetização de um determinado elemento de um circuito. O *q-bit* é uma variável que corresponde a uma superposição de estados aos quais podem ser associados os rótulos 0 e 1, e pode ser representado por um estado quântico bidimensional descrito por:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,. \tag{2.51}$$

Outra diferença fundamental baseia-se na reversibilidade das operações lógicas: Enquanto a computação quântica opera em regime reversível (pois a evolução dos sistemas corresponde e operadores unitários), a computação clássica é usualmente irreversível devido à irreversibilidade de alguns elementos como a porta XOR [8]. Embora busque-se modelos de computação clássica reversível [20], estes não são considerados usuais.

Na computação clássica é possível clonar *bits* irrestritamente, enquanto *q-bits* não podem ser clonados segundo o teorema da não clonagem [8], além de haver restrições quanto ao tempo necessário para efetuar as operações devido à decoerência, que distorce os estados quânticos e leva à perda de informação. A presença de ruido é outro problema tecnológico relacionado à computação quântica. A implementação de computadores quânticos funcionais enfrenta uma série de desafios tecnológicos. Projetar sistemas que forneçam possibilidade de geração de estados emaranhados, controle da evolução, viabilidade de medição e ainda assim sejam economicamente viáveis não é uma tarefa trivial. De uma forma geral, é possível dizer que a necessidade de isolamento e resfriamento para evitar ruído e decoerência são necessários e contribuem para o elevado custo dos computadores quânticos [21].

Em relação à implementação, há diversos sistemas para computação quântica, cada qual com suas vantagens, desvantagens e métodos específicos para manipulação do Hamiltoniano, aplicação de portas lógicas, preparação de estados iniciais, medições

e criação de emaranhamento. Dentre os possíveis sistemas, há computadores à base de armadilhas iônicas [22], de ressonância magnética [23], de cavidades ópticas [24] e de osciladores harmônicos [25]. Neste contexto, o termo "estado da máquina" se refere ao estado quântico composto pelos *q-bits* do sistema físico usado pelo computador.

Um modelo computacional é a forma utilizada para representar as operações que um computador realiza para processamento informacional. É possível citar como exemplo os modelos de máquina de Turing, que utiliza um conjunto de estados, um cabeçote de leitura e escrita, memórias e símbolos graváveis para processar informação, e o modelo de circuito que traduz uma série de operações lógicas na forma de arranjos de elementos eletrônicos, processando o estado inicial até que nele esteja codificado um dado resultado desejado. O modelo mais comum para representar a computação quântica é o de circuitos de operações unitárias, embora haja outros, como computação quântica adiabática [26] e computação quântica topológica [27].

A busca por problemas que podem ser resolvidos mais eficientemente por computadores quânticos do que por computadores clássicos vem sendo alvo de ampla investigação. O termo “supremacia quântica” se refere, neste contexto, a casos onde tal feito é possível [3]. Não existe critério universal que estabeleça em quais casos os computadores quânticos serão mais eficientes, uma vez que o projeto de algoritmos, tanto quânticos quanto clássicos, não é tarefa trivial que possua diretrizes universais [8]. Há, porém, alguns casos históricos de projeção de algoritmos quânticos mais eficientes que os equivalentes clássicos já conhecidos, como o algoritmo de Deutsch-Jozsa [8].

Acredita-se que a computação quântica tenha potencial de contribuição para áreas tecnológicas como comunicação, aprendizagem de máquina, otimização [28] e simulação de sistemas quânticos [8]. O paralelismo quântico, que pode implicar na possibilidade de manipulação de informação em paralelo devido ao fato de estados quânticos serem superposições de diversos elementos, é um dos fatores que contribuem para aumentar o interesse nestes algoritmos.

2.3.1 Circuitos quânticos e notações correlatas

O modelo circuital de computação quântica consiste na aplicação de portas lógicas em um conjunto finito de *q-bits* que comumente exibe interações entre seus constituintes através do fenômeno de emaranhamento, que pode resultar de diversas formas de interação entre sistemas quânticos, mas uma vez que ocorre, o estado do sistema composto não pode ser descrito através do tratamento de suas partes independentemente.

Matematicamente, n *q-bits* podem ser, em certas situações, descritos por um produto tensorial segundo a notação expressa em:

$$|\psi\rangle = |\psi_1\rangle\,|\psi_2\rangle\,...\,|\psi_n\rangle = (\alpha_1\,|0\rangle + \beta_1\,|1\rangle)(\alpha_2\,|0\rangle + \beta_1\,|1\rangle)....(\alpha_n\,|0\rangle + \beta_n\,|1\rangle), \qquad (2.52)$$

$$|\psi\rangle = \sum_{i=0}^{2^n-1} \lambda_i |i\rangle . \tag{2.53}$$

Este produto é distributivo, de tal forma que os coeficientes λ_i, expressos na equação (2.53) serão produtos cujos fatores são as 2^n possíveis permutações dos coeficientes dos n *qbits*, mas não é comutativo, de tal forma que a ordem dos *q-bits* altera o estado superposto gerado. Caso tenha sido gerado emaranhamento, o estado do sistema não poderá ser descrito como um produto de estados individuais como apresentado na (2.52), mas ainda assim será uma superposição de 2^n estados.

As portas lógicas correspondem a operadores unitários, que obedecem à condição descrita pela equação (2.32).

Operadores unitários que atuam em espaços complexos representam a generalização das rotações em espaços reais, e portanto preservam a norma do vetor no qual atuam (caso não fosse dessa maneira, haveria uma contradição com os Postulados da Mecânica Quântica, uma vez que os estados devem ser normalizados).

Um *q-bit* é um estado quântico descrito pela equação (2.51). Logo, um operador que atua em tal estado pode ser descrito como uma matriz 2x2. A forma matemática para a representação matricial de um operador unitário é definida por quatro parâmetros $(\delta, \alpha, \theta, \beta)$ dada pela equação (2.54):

$$\hat{\mathbf{U}}(\delta,\alpha,\theta,\beta) = \begin{bmatrix} e^{i\left(\delta+\frac{\alpha}{2}+\frac{\beta}{2}\right)} cos\left(\frac{\theta}{2}\right) & e^{i\left(\delta+\frac{\alpha}{2}-\frac{\beta}{2}\right)} sen\left(\frac{\theta}{2}\right) \\ -e^{i\left(\delta-\frac{\alpha}{2}+\frac{\beta}{2}\right)} sen\left(\frac{\theta}{2}\right) & e^{i\left(\delta-\frac{\alpha}{2}-\frac{\beta}{2}\right)} cos\left(\frac{\theta}{2}\right) \end{bmatrix} . \tag{2.54}$$

Nota-se que se as fases α, β e δ forem reduzidas a 0, o resultado é idêntico à parametrização usual da matriz de rotação real em termos do ângulo θ, e que a fase δ (denominada fase global) pode ser expressa em termos de um fator multiplicativo escalar $e^{i\delta}$.

A tabela 2 apresenta as principais portas lógicas de um *q-bit* e suas respectivas representações matriciais.

A porta X é por vezes chamada de porta NOT, pois tem efeito de converter o estado $|0\rangle$ no $|1\rangle$, e vice-versa.

Além das portas de um *q-bit* há também as de múltiplos *q-bits*. Estas podem ser representadas por matrizes 2^nx2^n, que são, por vezes, descritas como produtos tensoriais entre matrizes de portas atuando em $m \leq n$ *q-bits*.

A Fig. 3 exemplifica um circuito genérico, com representação matricial equivalente dada pelo produto expresso na equação (2.55), na qual a notação $\otimes$ se refere a produto tensorial

$$\mathbf{M} = (\mathbf{U_1} \otimes \mathbf{X} \otimes \mathbf{I}^{\otimes \mathbf{2}})\mathbf{U_2}(\mathbf{I} \otimes \mathbf{H} \otimes \mathbf{I}^{\otimes \mathbf{4}}). \tag{2.55}$$

Tabela 2 – Portas de um *q-bit*.

Nome da porta	**Representação matricial**
Pauli-X (X)	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y(Y)	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z(Z)	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Haddamard(H)	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase(S,P)	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$(T)	$\begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}$

Cada linha (denominada fio) representa um *q-bit*. As portas lógicas são representadas por retângulos que envolvem os *q-bits* em que atuam, e a letra dentro dos retângulos se refere o nome desta (um quadrado com H dentro, por exemplo, se refere à porta de Haddamard apresentada na tabela 2). Portas agrupadas na mesma coluna devem ser processadas através de produtos tensoriais dos elementos da coluna, seguindo a ordem de cima para baixo (lembrando que este produto não é comutativo), e com espaços vazios sendo substituídos pela matriz identidade. Após essa etapa, os resultados das colunas adjacentes devem ser multiplicados através do produto matricial padrão, da esquerda para a direita.

Além de portas de um e múltiplos *q-bits*, há também o conceitos de portas controladas. Uma porta controlada age em um conjunto de *q-bits* (denominados alvo) somente se os outros (denominados controle) apresentarem algum valor pré-definido, e é graficamente representada pelo usual retângulo indicando a porta controlada, com um conjunto de pontos unidos a esta por uma linha vertical passando pelos *q-bits* de controle. Um ponto escuro indica que a ação é condicionada ao *q-bit* de controle no estado "1", e um ponto claro indica que é condicionada ao controle no estado "0". A Fig. 4 exibe um exemplo de porta U controlada. Esta porta somente agirá nos estados $|000\rangle$ e $|001\rangle$, aplicando a porta U no último *q-bit* que compõe o sistema, e deixando os demais estados intactos.

A representação matricial de uma porta desse tipo (controlada) segue exemplificada

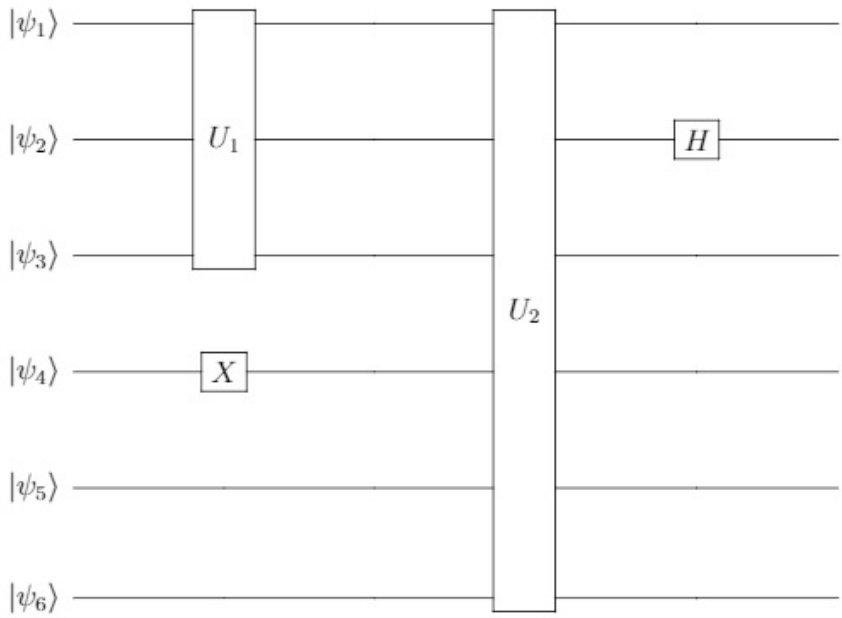

Figura 3 – Circuito quântico genérico.Cada linha (denominada fio) representa um *q-bit*.As portas lógicas são representadas por retângulos que envolvem os *q-bits* nos quais elas atuam, e a letra dentro dos retângulos se refere o nome da porta (um quadrado com H dentro, por exemplo, se refere à porta de Haddamard apresentada na tabela 2).

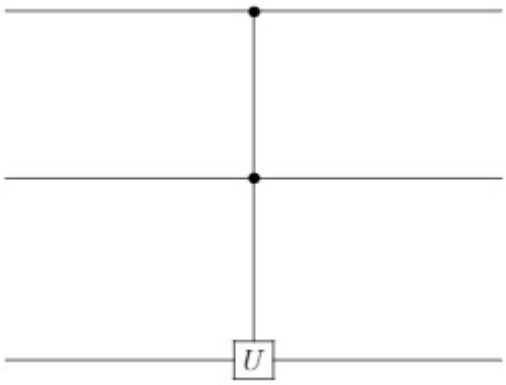

Figura 4 – Porta U controlada. Os pontos de controle escuros, unidos pela linha vertical a sua respectiva porta lógica, indicam o condicionamento da aplicação de U ao *q-bit* alvo (terceiro) somente à situação na qual os q-bits de controle valem "1".

na equação (2.56), para a porta $\hat{\mathbf{U}}$ da figura 4:

$$\mathbf{U_c} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & 0 & 0 & 0 & 0 & u_{10} & u_{11} \end{bmatrix}. \tag{2.56}$$

Tratam-se de matrizes bloco-diagonais, nas quais os blocos dos *q-bits* de controle correspondem à matriz identidade, e o dos *q-bits* alvo corresponde à representação matricial da operação controlada.

A porta controlada anteriormente ilustrada na Fig. 4 corresponde a um caso no qual a ação é condicionada a todos os *q-bits* de controle apresentarem estado com valor "1". Este não é o único tipo de controle possível. Outro exemplo seria o de uma porta que age somente se os primeiros *q-bits* estiverem em estado "1"e "0", respectivamente, modificando apenas os estados $|100\rangle$ e $|101\rangle$ e deixando os demais intactos. A Fig. 5 ilustra a notação para esta porta, bem como também uma forma equivalente de estruturá-la através do posicionamento de uma porta com controles fechados (ativados por "1") intercalada entre várias X. Uma vez que a mesma troca os estados dos *q-bits*, basta utilizá-la para converter 0 em 1 e vice-versa, para que possam controlar a ação da porta central, e depois revertê-los ao valor original através da aplicação de outra porta X.

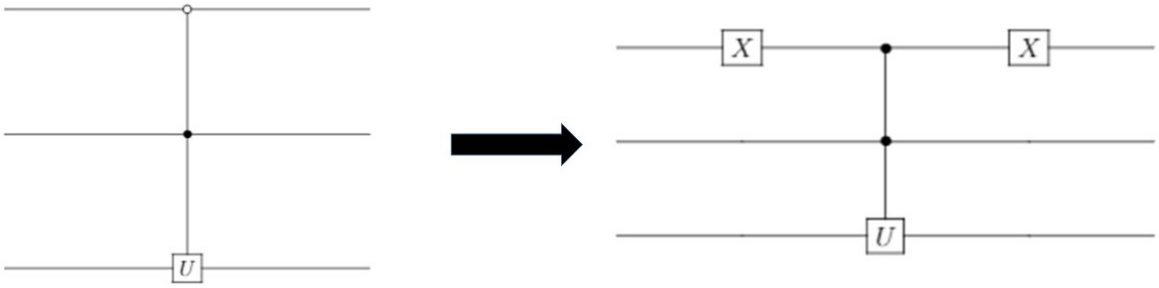

Figura 5 – Equivalência de controles. Um controle condicionado ao valor zero (ponto branco) pode ser substituído por um condicionado ao valor um (ponto escuro) entremeado por duas portas NOT (denotadas por X), uma vez que estas intercambiam o valor do *q-bit* no qual agem.

Outros símbolos importantes são os de medida e controle clássico, ilustrados na Fig. 6.

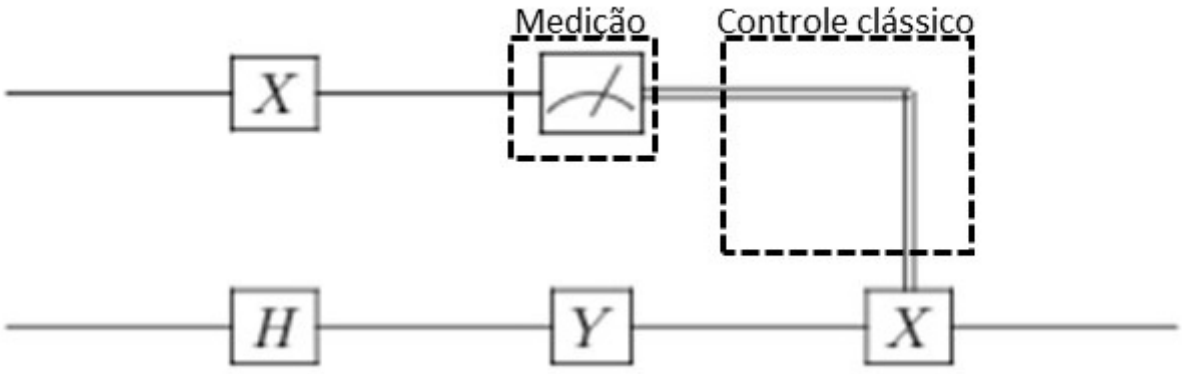

Figura 6 – Medições e controles clássicos.

O símbolo de medição indica que será realizada uma medida nos *q-bits* sobre os

quais este é posicionado. Após esta, resulta um estado (n-m)-dimensional (sendo m a quantidade de *q-bits* medidos), para o qual as probabilidades associadas aos *q-bits* restantes também devem estar de acordo com a estatística estabelecida pelo estado imediatamente anterior à medição. O resultado obtido pode ser usado como parâmetro de controle para outras portas quânticas aplicadas ao longo do circuito. O símbolo com linhas duplas, similar a uma alavanca, une o símbolo de medição à porta que esta controla.

3 Análise da complexidade de algoritmos quânticos.

Algoritmos quânticos, como discutido no capítulo anterior, possuem uma estrutura descrita por uma sequência de portas lógicas (operações unitárias) que atuam em um conjunto de *q-bits*. Quantificar o consumo de elementos físicos (proporcionais ao espaço) e tempo consumidos por um algoritmo quântico significa, portanto, estabelecer como a quantidade de portas lógicas e o total de *q-bits* que precisam ser utilizados dependem de alguma característica das variáveis que serão processadas no decorrer da execução do algoritmo (a dimensão de uma matriz a se inverter, por exemplo).

Utiliza-se o termo "largura" para indicar a quantidade total de *q-bits* necessários para a execução do algoritmo, e o termo "profundidade" para referência à quantidade total de grupos de portas que podem ser executadas simultaneamente (ou seja, portas atuando em grupos de *q-bits* distintos, cujo tempo de aplicação é igual ao da mais demorada). O circuito exibido na figura 3, por exemplo, possui largura 6 e profundidade 3. Há, porém, certas sutilezas que devem ser levadas em consideração.

Primeiramente, portas que atuam em n *q-bits* demandam um tempo de aplicação que depende de n, e tal dependência deve ser considerada para a análise das ordens assintóticas abordadas no capítulo 2. Além disso, estas mesmas podem ser decompostas em termos de operações em uma quantidade menor de *q-bits*. Esta decomposição não é única, de tal forma que há diversas maneiras de representar o mesmo algoritmo através de diferentes conjuntos de portas lógicas. Desta maneira, analisar efetivamente o tempo consumido por um algoritmo quântico significa analisar o tempo consumido por uma representação do mesmo em termos de elementos cujo tempo individual de execução não dependa das variáveis de entrada.

Um dado conjunto de portas é dito universal caso qualquer outra porta possa ser representado em termos de uma quantidade finita de elementos desse conjunto. Tal representação deve ser feita, algebricamente, através de produtos como o expresso na equação (2.55). Embora matrizes possam ser expressas como combinações lineares de um grupo finito de elementos, visto que essas formam um espaço vetorial, tal representação é de pouca utilidade para designar circuitos quânticos, pois para tal finalidade o ideal é expressa-las como produtos, não como somas.

O conjunto de portas lógicas executáveis depende, a rigor, do sistema, mas ainda assim é possível especificar um conjunto de portas que, devido a possuírem caráter amplo e geral, podem ser efetuadas na ampla gama de sistemas comumente utilizados . Estas

são as portas de um *q-bit* e a porta C-NOT (um tipo de porta controlada cuja notação está expressa na Fig. 7, e a representação matricial expressa na equação (3.1). O trabalho desenvolvido por Barenco *et. al* aborda a complexidade computacional relativa à construção de uma série de portas lógicas a partir destas duas[29].

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.1}$$

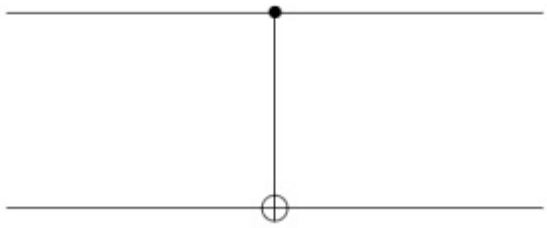

Figura 7 – Porta C-NOT. Esta porta efetua a inversão do segundo *q-bit* somente se o primeiro for igual a $|1\rangle$.

3.1 Complexidade de operações controladas

Operações controladas são recorrentes em uma série de algoritmos quânticos por motivos similares aos de serem também nos clássicos: Elas indicam para o sistema instruções do tipo "se A for verdadeiro, então faça B", algo que é estritamente necessário, tanto no cenário quântico quanto no clássico. Estas possuem representação matricial descrita pela equação (2.56), e são ilustradas em circuitos conforme a figura 4. Será utilizada, para a presente discussão, a notação $C^n\hat{\mathbf{U}}$ para indicar uma operação controlada por n *q-bits*. A ação dessas operações em um estado genérico do sistema, com os estados $|\phi_1\rangle ... |\phi_n\rangle$ atuando como *q-bits* de controle, é descrita por:

$$\mathbf{C^n\hat{U}} |\phi_1\rangle ... |\phi_n\rangle |\psi\rangle = \begin{cases} |\phi_1\rangle ... |\phi_n\rangle \hat{\mathbf{U}} |\psi\rangle , & \text{se } \phi_1 = \phi_2 = ...\phi_n = 1, \\ |\phi_1\rangle ... |\phi_n\rangle |\psi\rangle , & \text{em outros casos.} \end{cases} \tag{3.2}$$

Evidentemente, o caso mais simples corresponde a uma porta controlada por um *q-bit*. A construção de tal porta tem como base a fatoração da forma matricial descrita pela equação (2.54), tal que:

$$\hat{\mathbf{U}} = e^{i\delta}\mathbf{IR_z}(\alpha)\mathbf{R_y}(\theta)\mathbf{R_z}(\beta) = \begin{bmatrix} e^{i\delta} & 0 \\ 0 & e^{i\delta} \end{bmatrix} \begin{bmatrix} e^{i\frac{\alpha}{2}} & 0 \\ 0 & e^{-i\frac{\alpha}{2}} \end{bmatrix} \begin{bmatrix} cos\left(\frac{\theta}{2}\right) & sen\left(\frac{\theta}{2}\right) \\ -sen\left(\frac{\theta}{2}\right) & cos\left(\frac{\theta}{2}\right) \end{bmatrix} \begin{bmatrix} e^{i\frac{\beta}{2}} & 0 \\ 0 & e^{-i\frac{\beta}{2}} \end{bmatrix}. \tag{3.3}$$

Para $\hat{\mathbf{A}} = \mathbf{R_z}(\alpha)\mathbf{R_y}\left(\frac{\theta}{2}\right)$, $\hat{\mathbf{B}} = \mathbf{R_y}\left(\frac{\theta}{2}\right)\mathbf{R_z}\left(\frac{-\alpha-\beta}{2}\right)$ e $\hat{\mathbf{C}} = \mathbf{R_z}\left(\frac{\beta-\alpha}{2}\right)$, é possível então representar a operação descrita pela equação (3.3) por $\hat{\mathbf{U}} = e^{i\delta}\hat{\mathbf{A}}\hat{\mathbf{X}}\hat{\mathbf{B}}\hat{\mathbf{X}}\hat{\mathbf{C}}$ pelo circuito exibido na Fig. 8, na qual $\hat{\mathbf{E}} = e^{i\delta}\hat{\mathbf{I}}$.

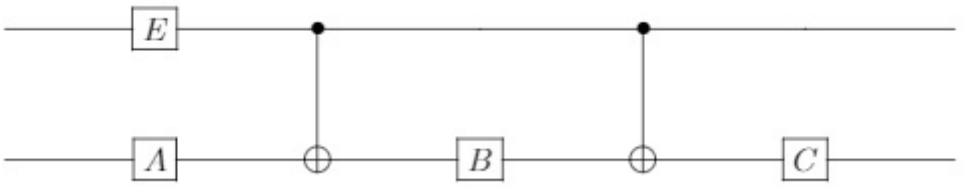

Figura 8 – Implementação de $\mathbf{C^1\hat{U}}$. Para $\hat{\mathbf{A}} = \mathbf{R_z}(\alpha)\mathbf{R_y}\left(\frac{\theta}{2}\right)$, $\hat{\mathbf{B}} = \mathbf{R_y}\left(\frac{\theta}{2}\right)\mathbf{R_z}\left(\frac{-\alpha-\beta}{2}\right)$ e $\hat{\mathbf{C}} = \mathbf{R_z}\left(\frac{\beta-\alpha}{2}\right)$, a aplicação das C-NOTs garante que o produto final seja o descrito pela equação (3.3).

Para n>1, há uma divisão em dois casos. O primeiro caso diz respeito a $2 \leq n \leq <8$, e envolve arranjos consideravelmente complexos de operadores $\hat{\mathbf{V}}$, definidos como a n-ésima raiz do operador $\hat{\mathbf{U}}$ para o qual se deseja implementar a porta $\mathbf{C^n\hat{U}}$. Tal implementação exige uma quantidade de portas que varia exponencialmente com o número de *q-bits*, sendo eficiente somente para muitos poucos *q-bits*.

A implementação mais versátil e geral apresentada por Barenco [29] envolve dois passos: Mostrar que uma porta $\mathbf{C^n\hat{X}}$ pode ser implementada utilizando apenas portas C-NOT, e então utilizar estas portas (em conjunto com portas de um *q-bit*) para construir as portas controladas por múltiplos *q-bits*.

3.1.1 Operações controladas por n q-bits.

Para operações controladas por n *q-bits*, é comumente utilizada, por questão de conveniência e clareza, uma porta denominada porta de Toffoli (ou porta CC-NOT), cuja notação circuital é ilustrada na Fig. 9 e a representação matricial é exibida na equação (3.4):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.4}$$

Esta porta pode ser implementada pelo arranjo ilustrado na Fig. 10, no qual $\hat{\mathbf{A}} = \mathbf{R_y}(\pi/4)$. Ou seja, também pode ser construída através da utilização de portas de

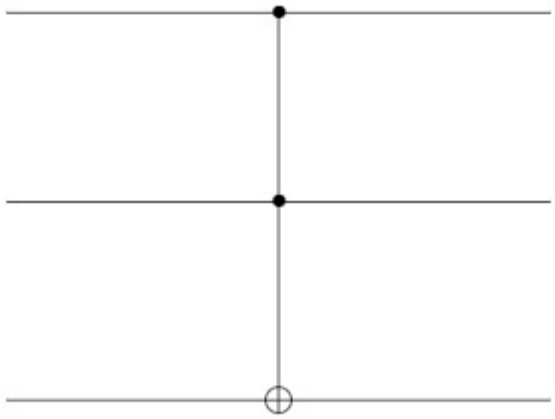

Figura 9 – Porta de Toffoli.A ação somente troca um por zero (e vice-versa) no valor do último *q-bit* se os dois primeiros forem um.

um *q-bit* e de combinações de C-NOT. Segundo a notação adotada, esta pode ser também chamada de $\mathbf{C^2\hat{X}}$.

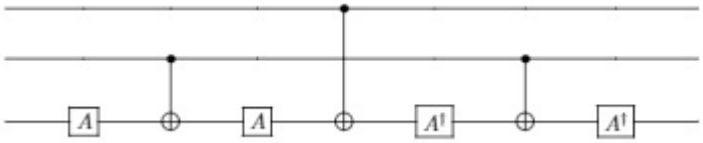

Figura 10 – Implementação da porta de Toffoli utilizando a porta de um *q-bit* $\mathbf{\hat{A} = R_y(\pi/4)}$ e C-NOTs para garantir que os produtos de rotações no *q-bit* alvo forneçam o resultado correto.

Estas portas $\mathbf{C^2\hat{X}}$ podem ser utilizadas para construir portas $\mathbf{C^m\hat{X}}$ que atuam em um total de n *q-bits* (implicando que uma quantidade de *q-bits* correspondente a $n - m$ não é alterada após a execução da porta, nem serve como controle). Tal construção está ilustrada na Fig. 11. As primeiras sete portas de Toffoli têm efeito de negar o sexto *q-bit* apenas se os dois primeiros forem 1, o sétimo será negado apenas se os três primeiros forem 1, e assim sucessivamente, resultando no nono *q-bit* ser negado apenas se os cinco primeiros forem 1, obtendo, desta maneira, o efeito esperado para uma porta $\mathbf{C^m\hat{X}}$. Para obter o efeito desejado, os três penúltimos *q-bits* tem seu valor alterado, e a aplicação das próximas quatro portas de Toffoli tem como objetivo restaurá-los a seus valores originais, para completar o efeito esperado da porta [29].

Este tipo de construção só é possível para as condições nas quais $n \geq 5$ e $m \in \{3, ..., [n/2]\}$ [29], e apesar de necessitar $n - m$ *q-bits* livres, estes mantêm-se inalterados após a execução da porta, e seu o valor inicial não interfere no funcionamento correto da mesma, permitindo então a combinação de arranjos como este para construir qualquer porta $\mathbf{C^{n-2}\hat{X}}$, conforme ilustrado na Fig. 12. A construção é realizada alternando duas portas $\mathbf{C^{n-m-1}\hat{X}}$ entre duas portas $\mathbf{C^m\hat{X}}$, contanto que $n \geq 5$ e $m \in \{2, .., n-3\}$. Estes arranjos são possíveis porque uma vez que os *q-bits* livres não possuem seu valor alterado, e

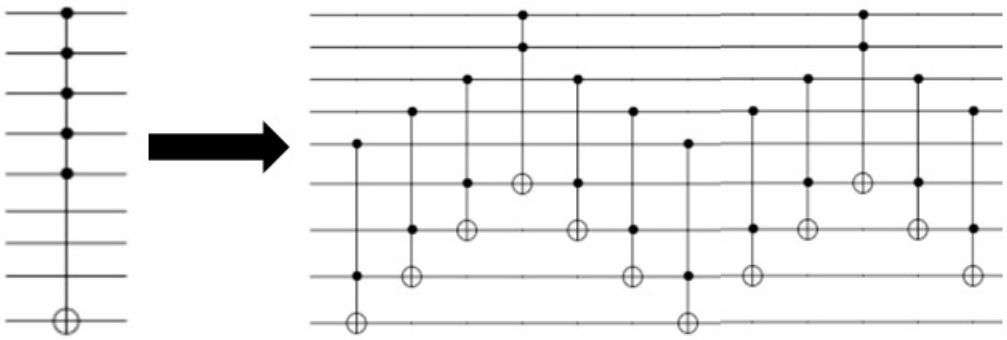

Figura 11 – Porta $\mathbf{C^m\hat{X}}$ com n *q-bits* totais ($m = 5, n = 9$), decomposta em termos de Portas de Toffoli. Esse tipo de construção é utilizada para estruturar outras portas controladas

a porta funciona para qualquer valor inicial destes, torna-se então possível utilizar os *q-bits* deixados livres pela $\mathbf{C^m\hat{X}}$ como controle da $\mathbf{C^{n-m-1}\hat{X}}$, e os que serviram como controle na $\mathbf{C^m\hat{X}}$ ficam vagos para que a $\mathbf{C^{n-m-1}\hat{X}}$ possa ser efetuada de forma adequada.

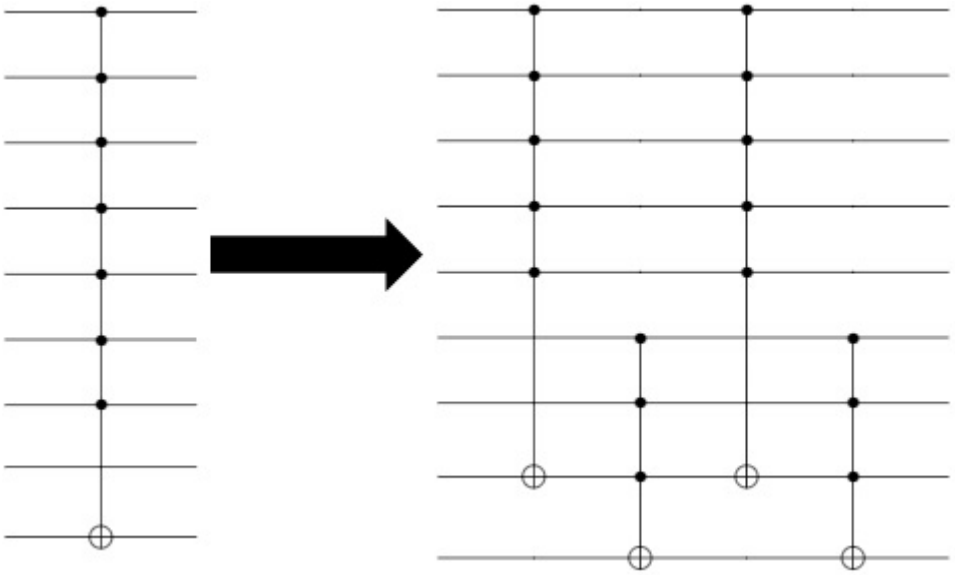

Figura 12 – Porta $\mathbf{C^{n-2}\hat{X}}$ ($m = 5, n = 9$).

A Fig. 11 ilustra como cada uma das portas $\mathbf{C^k\hat{X}}$ pode ser simulada utilizando um total de 4(k-2) portas de Toffoli. Logo, a construção exibida na Fig. 12 exigirá um total de $8(m-2) + 8(n-m-3) = 8(n-5)$ destas, e uma vez que a porta de Toffoli pode ser executada com a utilização de um número fixo de portas C-NOT e de um *q-bit*, resulta que uma $\mathbf{C^{n-2}\hat{X}}$ poderá ser executada através do uso de uma quantidade de portas que escala com O(n) (assim como também o tempo de execução) e mostra-se mais uma vez que um amplo conjunto de operações pode ser decomposto em termos de portas de um *q-bit* e C-NOT.

A importância dessas $\mathbf{C^k\hat{X}}$ consiste no fato de serem componentes necessárias para a construção das portas gerais de um *q-bit* controladas por $n-1$ *q-bits*, denotadas

por $\mathbf{C^{n-1}\hat{U}}$ (sendo n o total de *q-bits* nos quais a porta age). A Fig. 13 mostra como tal elemento pode ser construído usando duas portas $\mathbf{C^{n-2}\hat{X}}$, uma $\mathbf{C^1\hat{V}}$, sua inversa e uma $\mathbf{C^{n-2}\hat{V}}$. A porta de um *q-bit* $\hat{\mathbf{V}}$ é tal que $\hat{\mathbf{V}}^2 = \hat{\mathbf{U}}$.

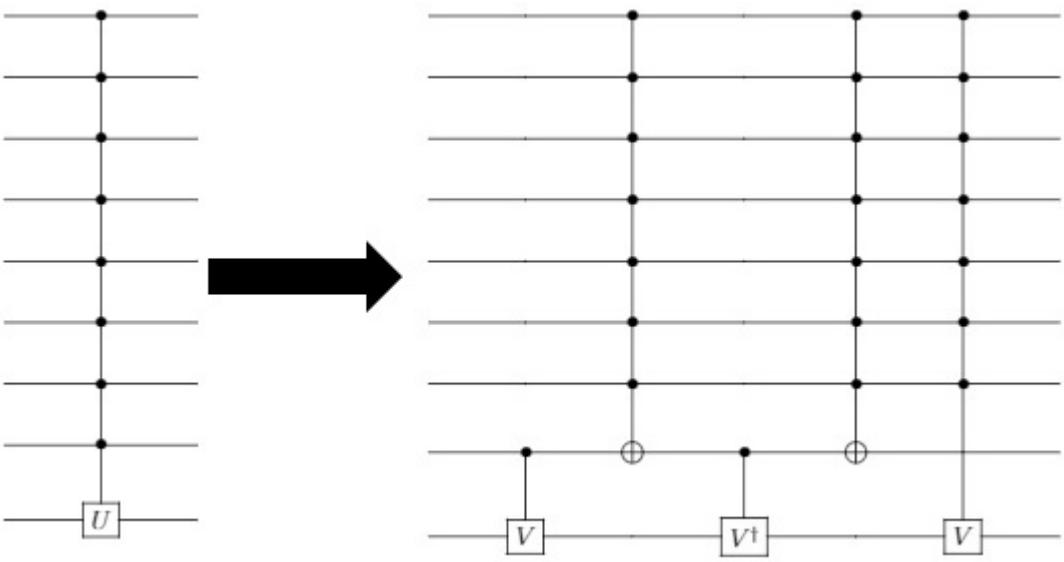

Figura 13 – Porta $\mathbf{C^{n-1}\hat{U}}$. As portas $\hat{\mathbf{V}}$ são tais que $\hat{\mathbf{V}}^2 = \hat{\mathbf{U}}$, e as $\mathbf{C^{n-2}\hat{X}}$ garantem a aplicação do produto correto no *q-bit* alvo para obtenção do resultado desejado.

A porta $\mathbf{C^{n-2}\hat{V}}$ pode ser decomposta recursivamente através do mesmo método. Observa-se que a construção inicial de $\mathbf{C^{n-1}\hat{U}}$ exigirá $16(n-5)$ portas de Toffoli (pois as $\mathbf{C^{n-2}\hat{X}}$ utilizadas demandam $8(n-5)$ cada uma), e as portas $\hat{\mathbf{V}}$ controladas demandam O(1) operações. Além disso, a decomposição recursiva deverá ser repetida O(n) vezes, resultando portanto em uma complexidade temporal de $O(n^2)$ para a implementação de $\mathbf{C^{n-1}\hat{U}}$.

É possível obter um ganho polinomial na implementação de uma porta controlada geral $\mathbf{C^k\hat{U}}$ através da utilização de apenas um *q-bit ancilla* (posteriormente, o termo *ancilla* será utilizado para referenciar *q-bits* que são necessários para a execução dos algoritmos, mas cujo valor não é diretamente relevante para o resultado destes). A implementação de uma porta $\mathbf{C^{n-2}\hat{U}}$ com um *q-bit ancilla* é ilustrada na Fig. 14. Neste caso, nota-se que a complexidade temporal da implementação é O(n), pois esta é realizada através da utilização de duas $\mathbf{C^{n-2}\hat{X}}$.

Em suma, para $n > 8$, qualquer porta de um *q-bit* controlada pode ser implementada através de portas de um *q-bit* e portas C-NOT, com complexidade $O(n^2)$, que pode ser reduzida para $O(n)$ com a introdução de um *q-bit ancilla* .

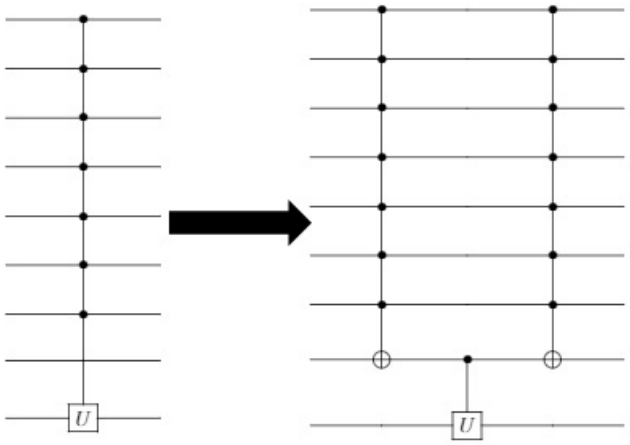

Figura 14 – Porta $\mathbf{C^{n-2}\hat{U}}$ com um *qbit ancilla* (penúltimo). A utilização deste poupa a necessidade de portas $\hat{\mathbf{V}}$ controladas seguindo níveis recursivos, reduzindo assim a complexidade de $O(n^2)$ para $O(n)$.

3.2 Operações gerais de n *q-bits*.

Operações gerais de múltiplos *q-bits*, representadas por matrizes 2^nx2^n (ou seja, 4^n graus de liberdade) representam uma fração significativa da totalidade dos algoritmos quânticos, de tal forma que algoritmos que não efetuam medições podem ser descritos por uma matriz desse tipo atuando em um determinado estado inicial.

Barenco *et. al.* [29] e Mötönen *et. al.* [30, 31] abordaram a construção do caso mais geral possível de tais operações. O limite inferior teórico para a complexidade expressa em termos da quantidade de elementos de um conjunto fixo de portas é $\frac{1}{4}(4^n - 3n - 1)$ [32] (trata-se do caso para matrizes com total grau de liberdade com respeito a seus parâmetros, desde que preservando a unitariedade. Para casos específicos, como transformadas de Fourier, essa complexidade pode ser menor, pois os graus de liberdade são reduzidos).

Barenco *et. al.* [29] utilizou a decomposição de uma porta de n *q-bits* expressa em termos de matrizes de dois níveis e implementa a ação destas últimas através de portas $\mathbf{C^{n-1}\hat{X}}$ e $\mathbf{C^{n-1}\hat{U}}$, resultando em uma complexidade computacional de $O(n^3 4^n)$. Reck *et. al.* [33] havia anteriormente desenvolvido um trabalho semelhante, mas com enfoque em operações em sistemas ópticos.

Vertiainen e Möttönen [30] modificaram o método de Barenco *et. al.* e utilizaram uma permutação entre os elementos da base para para eliminar a necessidade das portas $\mathbf{C^{n-1}\hat{X}}$, reduzindo assim a complexidade de $O(n^3 4^n)$ para $O(n4^n)$, e posteriormente utilizaram a remoção de *q-bits* de controle supérfluos para reduzir a complexidade para $O(4^n)$ em alguns casos.

Um trabalho posterior de Möttönen [31] estabelece a utilização de outro tipo de decomposição de matrizes, a decomposição seno-cosseno, que consiste em expressar uma matriz em três fatores, sendo esta uma generalização da fatoração expressa na equação

(3.3). A utilização recursiva dessa fatoração permite escrever a operação em n *q-bits* em termos de um produto de matrizes que pode ser implementado através de rotações (operações de um *q-bit*) uniformemente controladas, para as quais o autor desenvolve, no mesmo trabalho, um método eficiente para realização. A complexidade alcançada neste trabalho para a implementação de uma operação de n *q-bits* é de $O(4^n)$.

3.2.1 Método das matrizes de dois níveis

O método desenvolvido por Barenco *et. al.* [29] consiste em fatorar a matriz que representa a operação de n *q-bits* em matrizes de dois níveis, que representam um operador agindo não trivialmente (ou seja, de forma diferente da identidade) apenas em um subespaço bidimensional do espaço de Hilbert 2^n dimensional do sistema. Estas possuem apenas duas linhas que diferem da matriz identidade, e cada uma delas contem apenas dois elementos. A representação matricial da porta de Toffoli, exibida na equação (3.4), é um exemplo.

As matrizes de dois níveis utilizadas na decomposição, denotadas $\hat{\mathbf{L}}(i,j)$, são tais que a multiplicação por estas resulta em uma nova matriz com os coeficientes na posição i e j garantidamente nulos, conforme exemplificado na equação (3.5).

$$\hat{\mathbf{L}}(i,j)\begin{bmatrix} m_{11} & \cdots & m_{1j} & \cdots & \cdots & m_{1N} \\ \vdots & & \vdots & & & \vdots \\ m_{i1} & \cdots & m_{ij} & \cdots & \cdots & m_{iN} \\ \vdots & & \vdots & & & \vdots \\ m_{N1} & \cdots & m_{Nj} & \cdots & \cdots & m_{NN} \end{bmatrix} = \begin{bmatrix} m'_{11} & \cdots & m'_{1j} & \cdots & \cdots & m'_{1N} \\ \vdots & & \vdots & & & \vdots \\ m'_{i1} & \cdots & 0 & \cdots & \cdots & m'_{iN} \\ \vdots & & \vdots & & & \vdots \\ m'_{N1} & \cdots & m'_{Nj} & \cdots & \cdots & m'_{NN} \end{bmatrix}. \quad (3.5)$$

Devido à propriedade exibida na equação (3.5), o produto por matrizes $\hat{\mathbf{L}}(i,j)$ com coeficientes específicos produzirá o efeito descrito pela equação (3.6), resultando na fatoração descrita pela equação (3.7). O fato do produtório percorrer somente os índices correspondentes à parte inferior à diagonal é consequência da unitariedade de $\hat{\mathbf{U}}$, a qual garante que após a anulação dos coeficientes inferiores também ocorrerá a dos superiores. A matriz $\hat{\mathbf{D}}$ é uma matriz diagonal, cujos coeficientes correspondem a fases que podem ser implementadas através da aplicação de sucessivas portas de fase em paralelo, portanto a complexidade total de sua implementação é $O(1)$.

$$\hat{\mathbf{D}}\prod_{i=1}^{N}\prod_{j=0}^{i-1}\hat{\mathbf{L}}(i,j)\hat{\mathbf{U}} = \hat{\mathbf{I}}, \quad (3.6)$$

$$\hat{\mathbf{U}} = \hat{\mathbf{D}}^{\dagger}\prod_{i=0}^{N-1}\prod_{j=1}^{N-i}\hat{\mathbf{L}}^{\dagger}(N-i, N-i-j). \quad (3.7)$$

Esta fatoração em matrizes de dois níveis não é única, e existem métodos mais eficientes que exploram esta propriedade.

As matrizes $\hat{\mathbf{L}}(i,j)$ possuem coeficientes descritos pela equação (3.8), na qual o termo $\hat{\mathbf{P}}$ se refere ao produto parcial obtido pela multiplicação de $\hat{\mathbf{U}}$ e todas as $\hat{\mathbf{L}}(i,j)$ anteriores, e R é um fator cujo valor é dado por $R = \sqrt{|P_{ij}|^2 + |P_{jj}|^2}$.

$$\hat{\mathbf{L}}(i,j)_{mn} = \begin{cases} \delta_{mn} & \text{se } n \neq i \text{ e } n \neq j, \\ P_{ij}/R & \text{se } m = i \text{ e } n = j, \\ -P_{jj}/R & \text{se } m = i \text{ e } n = i, \\ P^*_{jj}/R & \text{se } m = j \text{ e } n = j, \\ P*_{ij}/R & \text{se } m = j \text{ e } n = i. \end{cases} \tag{3.8}$$

O número Q de fatores na equação (3.7) é dado por uma progressão aritmética $Q = 1 + 2 + ... + (N-1) = N(N-1)/2$, com $N = 2^n$. Ou seja, qualquer matriz pode ser decomposta em um produto de $O(N^2)$ matrizes de dois níveis, a assim sendo, estas últimas formam um conjunto de operações universais.

Cada uma das matrizes de dois níveis pode ser individualmente implementada através de um conjunto de $2[(log_2(i) - log_2(j)) - 1]$ portas $\mathbf{C^{n-1}\hat{X}}$ e uma porta $\mathbf{C^{n-1}\hat{U}}$, conforme ilustrado na Fig. 15, para uma matriz descrita por:

$$\hat{\mathbf{L}}(1,8) = \begin{bmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{bmatrix}. \tag{3.9}$$

As portas $\mathbf{C^{n-1}\hat{X}}$ cumprem o papel de permutar os *q-bits*, seguindo um *Gray code* (sequência de cadeias binárias na qual um elemento difere do anterior em apenas um algarismo) que conecta o j-ésimo e o i-ésimo estados-alvo da matriz de dois níveis. Uma vez que os *q-bits* foram devidamente permutados até o estado $|i-1\rangle$ através da utilização de $(log_2(i) - log_2(j) - 1)$ portas $\mathbf{C^{n-1}\hat{X}}$, a aplicação da porta $\mathbf{C^{n-1}\hat{U}}$ gera uma superposição dos estados $|i\rangle$ e $|i-1\rangle$, e a atuação das mesmas $(log_2(i) - log_2(j) - 1)$ portas $\mathbf{C^{n-1}\hat{X}}$ em ordem inversa retorna o estado $|i-1\rangle$ novamente ao estado $|j\rangle$, completando a ação equivalente à da matriz de dois níveis desejada. Para esse fim, a matriz $\hat{\mathbf{U}}$ deve possuir os coeficientes correspondentes à transformação de dois níveis desejada. Um exemplo correspondente à equação (3.9) é descrito por:

$$\hat{\mathbf{U}} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}. \tag{3.10}$$

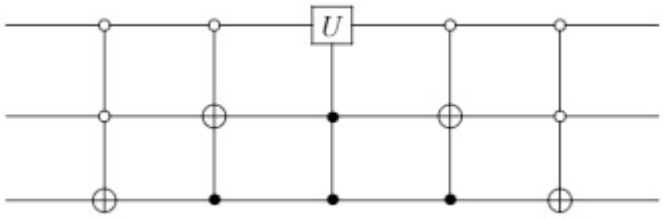

Figura 15 – Implementação de uma matriz de dois níveis. As portas de Toffoli modificam o estado inicial seguindo um *gray code* que conecta os níveis não identitários da matriz, enquanto a porta $\mathbf{C^2\hat{U}}$ aplica a transformação com os devidos coeficientes.

Um *Gray code* não é único, de tal forma que estas equivalências a matrizes de dois níveis podem ser implementadas utilizando apenas $2(n-1) = O(n)$ operações controladas, mas caso um *Gray code* mais extenso seja utilizado, a implementação pode terminar tendo complexidade maior, embora tal particularidade não refute o fato da implementação utilizar $O(n)$ portas, uma vez que estas ordens assintóticas sempre se referem ao algoritmo mais eficiente.

Em suma, através do método discutido é possível emular uma operação de *n-qbits* utilizando matrizes de dois níveis, que podem ser implementadas através de $O(n)$ portas $\mathbf{C^{n-1}\hat{X}}$ e uma $\mathbf{C^{n-1}\hat{U}}$, cada uma destas com complexidade individual de $O(n^2)$, resultando então uma complexidade total de $O(n^3)$ para a implementação de cada uma das 4^n matrizes de dois níveis necessárias. Assim, conclui-se que a complexidade de implementação de uma transformação envolvendo n *q-bits* é $O(n^3 4^n)$ (estas complexidades, assim como todas discutidas neste capítulo, adotam como métrica a quantidade de portas básicas requeridas para uma dada tarefa, sendo o tempo de execução também proporcional a esta quantidade).

3.2.2 Método de eliminação de C-NOTs e controles

Um método que possui certa similaridade com o anterior foi desenvolvido no trabalho de Vertianen, Möttönen e Saloma [30]. Este método também utiliza decomposição em matrizes de dois níveis, porém aproveita a não-unicidade desta para criar circuitos nos quais o uso das portas $\mathbf{C^{n-1}\hat{X}}$ que entremeiam a aplicação da $\mathbf{C^{n-1}\hat{U}}$ como exemplificado na figura 15 é eliminado, reduzindo a complexidade de forma a obter um ganho polinomial.

A forma mais geral de uma matriz de dois níveis que tem a propriedade de anular elementos em uma posição específica de outra através da multiplicação matricial por esta é denominada rotação de Givens [30]. Uma rotação de Givens representa um operador que age não trivialmente (isto é, de forma diferente do operador identidade) apenas em dois dos N elementos da base, $|j\rangle$ e $|k\rangle$, e além disso, possui a propriedade de anular os elementos na posição i,j de uma dada matriz após multiplicação através do devido ajuste

dos coeficientes. Denota-se tais matrizes por $\hat{\mathbf{G}}\{i\}(j,k)$. Se a matriz da qual se deseja anular um termo for $\hat{\mathbf{M}}$, então $\hat{\mathbf{G}}\{i\}(j,k)$ possui a seguinte estrutura:

$$\hat{\mathbf{G}}\{i\}(j,k)_{mn} = \begin{cases} \delta_{mn} & \text{se } n \neq j \text{ e } n \neq k, \\ -M_{ji}/R & \text{se } m = j \text{ e } n = k, \\ M^*_{ki}/R & \text{se } m = k \text{ e } n = k, \\ M_{ki}/R & \text{se } m = j \text{ e } n = j, \\ M^*_{ji}/R & \text{se } m = k \text{ e } n = j. \end{cases} \tag{3.11}$$

Através de um procedimento similar ao utilizado para escrever a equação (3.6), é possível fatorar uma matriz unitária $\hat{\mathbf{U}}$ da seguinte maneira:

$$\hat{\mathbf{D}}(\prod_{i=1}^{N-1}\prod_{j=i+1}^{N}\hat{\mathbf{G}}\{N-i\}(j,j-1))\hat{\mathbf{U}} = \hat{\mathbf{I}}. \tag{3.12}$$

Nota-se que os índices de produto seguem uma ordem específica que pode ser alterada através de uma transformação de base do tipo permutação, ou seja, alterando a ordem das cadeias de *q-bits* que compõem a base, resultando assim na seguinte transformação:

$$\hat{\mathbf{D}}(\prod_{i=1}^{N-1}\prod_{j=i+1}^{N}\hat{\mathbf{P}}\hat{\mathbf{G}}\{N-i\}(j,j-1)\hat{\mathbf{P}}^\dagger)\hat{\mathbf{P}}\hat{\mathbf{U}}\hat{\mathbf{P}}^\dagger = \hat{\mathbf{I}}. \tag{3.13}$$

A equação (3.13) corresponde à utilização de uma ordem dos vetores $|e_j\rangle$ correspondente a um *Gray code*, ao invés da usual ordem $e_j = bin(j-1)$ (a escolha deste *Gray code* será elucidada posteriormente), na qual $bin(j)$ se refere à representação binária do elemento do número j. Para tal abordagem, o *Gray code* escolhido é um tal que $e_j = bin(j)XOR(bin(j)/2$. Tal transformação de base é representada pela matriz $\hat{\mathbf{P}}$.

Caso os elementos da base sejam permutados para formar o *Gray code* desejado, a ordem da multiplicação pelas rotações de Givens passa a ser descrita pelos coeficientes resultantes da soma binária $\gamma(j) = 1 + bin(e_j)$, resultando assim em:

$$\hat{\mathbf{D}}(\prod_{i=1}^{N-1}\prod_{j=i+1}^{N}\hat{\mathbf{G}}\{\gamma(N-i)\}(\gamma(j),\gamma(j-1)))\hat{\mathbf{U}} = \hat{\mathbf{I}}. \tag{3.14}$$

Nota-se que as matrizes $\hat{\mathbf{G}}$ utilizadas na fatoração sempre atuam em elementos da base adjacentes. A escolha de um *Gray code* ao invés de ordenação numérica decimal para indexar os elementos da base se deve ao fato de, nessa situação, matrizes unitárias de dois níveis que atuam em elementos adjacentes da base ($|k\rangle$ e $|k+1\rangle$) poderem ser representadas por uma porta $\mathbf{C^{n-1}\hat{U}}$ sem utilização das $\mathbf{C^{n-1}\hat{X}}$ que permutam *q-bits*, tal

qual segue ilustrado na figuras 16 o circuito equivalente à matriz da equação (3.15):

$$\hat{\mathbf{G}}(0,1) = \begin{bmatrix} a & b & 0 & 0 & 0 & 0 & 0 & 0 \\ c & d & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}; \hat{\mathbf{U}}(\hat{\mathbf{G}}) = \begin{bmatrix} a & b \\ c & d \end{bmatrix}. \tag{3.15}$$

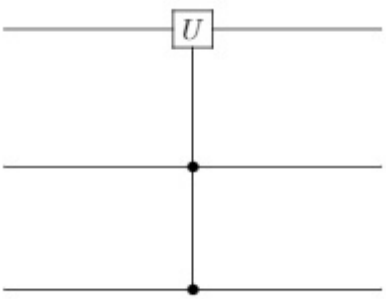

Figura 16 – Implementação de uma matriz de dois níveis com níveis adjacentes (no caso os correspondentes a $|000\rangle$ e $|001\rangle$). A adjacência poupa a necessidade de utilização de portas de Toffoli para permutar os *q-bits*.

Uma vez que a utilização de níveis sempre adjacentes e *Gray codes* possibilita remover as portas $\mathbf{C^{n-1}\hat{X}}$, cujo exemplo é apresentado na figura 15, e a quantidade necessária destas escala com $O(n)$, resultará que a quantidade de portas básicas exigidas para a implementação da porta de n *q-bits* poderá ser reduzida de $O(n^3 4^n)$ para $O(n^2 4^n)$.

Além da eliminação das portas lógicas $\mathbf{C^{n-1}\hat{X}}$, há também a possibilidade de eliminação de alguns pontos de controle. Para que um controle seja eliminado, é necessário que ainda assim os termos do produto matricial parcial não se misturem com os termos já anteriormente anulados, para que estes não adquiram novamente valor não-nulo [30]. Ao considerar este aspecto, os autores concluíram que a quantidade de controles do tipo $\mathbf{C^{n-i}\hat{U}}$ necessária é dada recursivamente pela seguinte relação:

$$g_n(n-i) = 2^{(i-1)} + \sum_{m=i+1}^{n} [g_m^0(m-i) + g_{m-1}(m-i)]. \tag{3.16}$$

O termo $g_m^0(k)$ quantifica as portas $\mathbf{C^{n-i}\hat{X}}$ necessárias para anular o quarto inferior esquerdo da matriz produto parcial, e tem valor dado por:

$$g_m^0(k) = max(2^{m-2}, 2^k) + \Theta(k-1)(2^{2m-k-2} - 2^{m-2}). \tag{3.17}$$

Os termos Θ são tais que $\Theta(x) = 1$, se $x \geq 0$ e $\Theta(x) = 0$, se $x < 0$.

A equação (3.17) garante que a quantidade de portas controladas possua um limite superior tal que $g_n(n-i) \leq 2^{n-i}$. Quando os devidos limites e valores de $g_n(n-i)$ são considerados, torna-se possível remover portas controladas de tal forma a garantir que uma dada matriz unitária 4^n-dimensional possa ser simulada por um circuito executável com complexidade $O(4^n)$.

3.2.3 Método da decomposição cosseno-seno

Este método, apresentado por Möttönen *et. al.* [31] utiliza rotações uniformemente controladas para implementar uma fatoração resultante de múltiplas decomposições cosseno-seno recursivas.

Uma rotação uniformemente controlada, denotada $F_m^k(\hat{\mathbf{R}}_\mathbf{a})$, é um tipo de operação composta por um conjunto de possíveis rotações do *q-bit* m com k *q-bits* de controle, como segue ilustrado na figura 17, para $F_4^3(\hat{\mathbf{R}}_\mathbf{a})$.

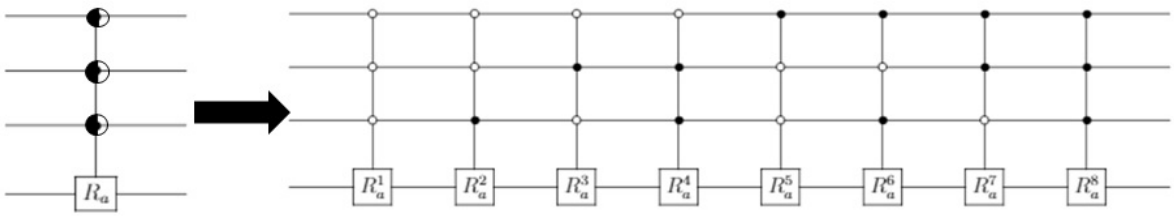

Figura 17 – Rotação uniformemente controlada. A operação, representada por um conjunto de círculos metade preto e metade branco nos pontos de controle (à esquerda), consiste no conjunto das várias rotações controladas compartilhando o mesmo eixo $\vec{a}$, mas com um ângulo diferente para cada uma das possíveis combinações de valores dos *q-bits* de controle, conforme pode ser visualizado na sequência à direita.

A representação matricial do circuito exemplificado na figura 17 é a matriz bloco-diagonal a seguir:

$$F_{k+1}^k(\hat{\mathbf{R}}_\mathbf{a}) = \begin{bmatrix} \mathbf{R_a}(\alpha_\mathbf{1}) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{R_a}(\alpha_\mathbf{N}) \end{bmatrix}; N = 2^n. \tag{3.18}$$

Os ângulos α_i podem ser arbitrariamente escolhidos. Dado um eixo $\mathbf{a}$, um ângulo α e o conjunto de matrizes de Pauli, $\hat{\sigma} = [\hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z]$, a forma geral das matrizes 2x2 de rotação $\hat{\mathbf{R}}_a(\alpha)$ é então descrita por:

$$\hat{\mathbf{R}}_a(\alpha) = exp\left(i\mathbf{a}.\sigma\frac{\alpha}{2}\right) = \hat{\mathbf{I}}cos\left(\frac{\alpha}{2}\right) + i(\mathbf{a}.\sigma)sen\left(\frac{\alpha}{2}\right). \tag{3.19}$$

Caso $F_m^k(\hat{\mathbf{R}}_\mathbf{a})$ seja uma rotação uniformemente controlada com $a_x = 0$, então é possível implementa-la através do circuito ilustrado na figura 18. Para tal propósito, utiliza-se 2^k portas $\mathbf{C}\hat{\mathbf{X}}$ e 2^k portas de rotação $\hat{\mathbf{R}}_a(\theta_i)$ agindo no *q-bit* m (logo, a complexidade temporal é de $O(2^k)$). Estrutura-se um *Gray code* binário refletido percorrendo todos os elementos de 0 a 2^{k-1}, com componentes denotados g_l, e posiciona-se os l-ésimos nós de controle de tal forma a ocuparem a posição na qual os bits de g_l e g_{l+1} diferem. Segue abaixo um exemplo de tal *Gray code*:

$$\begin{array}{cccccccc} g_0 & g_1 & g_2 & g_3 & g_4 & g_5 & g_6 & g_7 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{array} \tag{3.20}$$

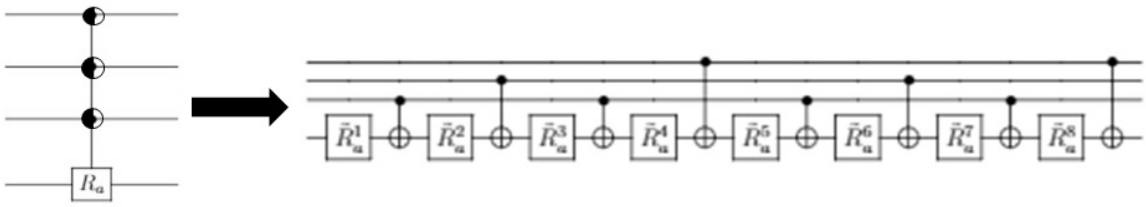

Figura 18 – Implementação da rotação uniformemente controlada.Os ângulos das operações de um *q-bit* na figura da direita se correlacionam com os ângulos das rotações através de uma matriz dependente do *Gray code* utilizado para definir as posições dos controles das C-NOT.

O fato de as rotações possuírem um eixo $a_x = 0$ implica na condição de inversão de sinal descrita a seguir:

$$\hat{\sigma}_x \hat{\mathbf{R}}_\mathbf{a}(\theta)\hat{\sigma}_x = \hat{\mathbf{R}}_\mathbf{a}(-\theta). \tag{3.21}$$

A condição supracitada torna possível escolher arbitrariamente os ângulos θ_i para estruturar o circuito da figura 18 de tal forma que os α_i da equação (3.18) possam, dada uma representação binária b_i do inteiro i, ser ajustados de acordo com a seguinte relação matricial:

$$\hat{\mathbf{T}} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_N \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}; T_{ij} = (-1)^{b_{i-1} \cdot g_{j-1}}. \tag{3.22}$$

Uma matriz $\hat{\mathbf{M}}$ N-dimensional pode ser decomposta em três fatores através do uso da decomposição cosseno-seno. Estes são descritos pela equação (3.23):

$$\hat{\mathbf{M}} = \hat{\mathbf{U}}_1 \hat{\mathbf{A}} \hat{\mathbf{U}}_2 = \begin{bmatrix} \hat{\mathbf{a}}_1 & \hat{\mathbf{0}} \\ \hat{\mathbf{0}} & \hat{\mathbf{a}}_2 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{c}} & \hat{\mathbf{s}} \\ -\hat{\mathbf{s}} & \hat{\mathbf{c}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}}_1 & \hat{\mathbf{0}} \\ \hat{\mathbf{0}} & \hat{\mathbf{b}}_2 \end{bmatrix}. \tag{3.23}$$

Os termos $\hat{\mathbf{U}}_1$ e $\hat{\mathbf{U}}_2$ são matrizes bloco-diagonais com $\hat{\mathbf{a}}$ e $\hat{\mathbf{b}}$ $N/2$-dimensionais, enquanto $\hat{\mathbf{c}}$ e $\hat{\mathbf{s}}$ são expressos como:

$$\hat{\mathbf{c}} = \begin{bmatrix} cos(\theta_1) & 0 & \cdots & 0 \\ 0 & cos(\theta_2) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & cos(\theta_{N/2}) \end{bmatrix}, \tag{3.24}$$

$$\hat{\mathbf{s}} = \begin{bmatrix} sen(\theta_1) & 0 & \cdots & 0 \\ 0 & sen(\theta_2) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & sen(\theta_{N/2}) \end{bmatrix}; N = 2^n. \tag{3.25}$$

É possível dividir $\hat{\mathbf{M}}$ em quatro blocos N/2-dimensionais, da seguinte maneira:

$$\hat{\mathbf{M}} = \begin{bmatrix} \hat{\mathbf{m}}_{11} & \hat{\mathbf{m}}_{12} \\ \hat{\mathbf{m}}_{21} & \hat{\mathbf{m}}_{22} \end{bmatrix}, \tag{3.26}$$

$$\begin{aligned} \hat{\mathbf{m}}_{11} = \hat{\mathbf{a}}_1\hat{\mathbf{c}}\hat{\mathbf{b}}_1 \quad \hat{\mathbf{m}}_{12} = \hat{\mathbf{a}}_1\hat{\mathbf{s}}\hat{\mathbf{b}}_2, \\ \hat{\mathbf{m}}_{21} = -\hat{\mathbf{a}}_2\hat{\mathbf{s}}\hat{\mathbf{b}}_1 \quad \hat{\mathbf{m}}_{22} = \hat{\mathbf{a}}_2\hat{\mathbf{c}}\hat{\mathbf{b}}_2. \end{aligned} \tag{3.27}$$

As formas que aparecem nestas últimas equações para os blocos $\hat{\mathbf{m}}_{ij}$, expressando estes como produto de matrizes trifatoriais, sugerem que a decomposição CS (cosseno-seno) possa ser feita de forma recursiva caso as matrizes sejam bloco-diagonais, pois basta utiliza-la novamente nos blocos ($\hat{\mathbf{a}}_1$ e $\hat{\mathbf{a}}_2$, por exemplo) para obter estruturas similares às que aparecem na equação (3.27), e consequentemente decompor um dos fatores $\hat{\mathbf{U}}_\mathbf{k}$ (com uma pequena modificação na forma para a matriz central $\hat{\mathbf{A}}$, a ser discutida posteriormente). Os próximos termos obtidos ao se refatorar $\hat{\mathbf{U}}_1$, por exemplo, já não conterão dois blocos N/2-dimensionais na diagonal, mas sim quatro blocos N/4-dimensionais, de tal forma que o processo pode ser repetido recursivamente até a obtenção de um produto de matrizes contendo N/2 blocos 2X2.

Para representar as matrizes que surgem após as múltiplas decomposições recursivas, utiliza-se uma notação de índices na qual o superior se refere ao nível na recursão, o inferior esquerdo se refere à ordem no produto e o inferior direito se refere à ordem em uma diagonal. A utilização correta dos índices está exemplificada a seguir:

$$\hat{\mathbf{U}} = \hat{\mathbf{U}}_1^1\hat{\mathbf{A}}_1^1\hat{\mathbf{U}}_2^1 = \begin{bmatrix} \hat{\mathbf{u}}_{11}^1 & \hat{\mathbf{0}} \\ \hat{\mathbf{0}} & \hat{\mathbf{u}}_{12}^1 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{c}}_{11}^1 & \hat{\mathbf{s}}_{11}^1 \\ -\hat{\mathbf{s}}_{11}^1 & \hat{\mathbf{c}}_{11}^1 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_{21}^1 & \hat{\mathbf{0}} \\ \hat{\mathbf{0}} & \hat{\mathbf{u}}_{22}^1 \end{bmatrix}. \tag{3.28}$$

O termo $\hat{\mathbf{u}}_{21}^1$, por exemplo, contém índices que fazem referência ao primeiro nível de recursão (primeira decomposição), segunda matriz no produto e primeiro bloco da diagonal.

O termo $diag_l(x_k)$ é utilizado para denotar uma matriz diagonal que contém os termos x_k na diagonal, com $k = 0..l$. Estes termos podem ser blocos ou um valor numérico. A matriz $\hat{\mathbf{c}}$ na equação (3.24), por exemplo, pode ser descrita como $\hat{\mathbf{c}} = diag_{N/2}(cos(\theta_k))$. Após aplicados os múltiplos níveis de recursão, as matrizes que compõem o produto serão descritas por:

$$\hat{\mathbf{U}}_j^i = diag_{2^i}(\hat{\mathbf{u}}_{jk}^i), \tag{3.29}$$

$$\hat{\mathbf{A}}_j^i = diag_{2^{i-1}}\left(\begin{bmatrix} \hat{\mathbf{c}}_{jk}^i & \hat{\mathbf{s}}_{jk}^i \\ -\hat{\mathbf{s}}_{jk}^i & \hat{\mathbf{c}}_{jk}^i \end{bmatrix}\right). \tag{3.30}$$

O termo $\hat{\mathbf{A}}$ não possui exatamente a forma descrita na equação (3.28), mas é composta por blocos assim descritos.

Para garantir a ordenação adequada dos índices, utiliza-se a função $\zeta(i,j) = 2^{n-i-1}(2j-1)$ e uma indexação para as fatorações recursivas descrita pela seguinte relação:

$$\hat{\mathbf{U}}_j^{i-1} = \hat{\mathbf{U}}_{2j-1}^i \hat{\mathbf{A}}_{\zeta(i,j)}^i \hat{\mathbf{U}}_{2j}^i. \tag{3.31}$$

O intuito de utilizar os índices dessa forma é obter uma sequência de indexação adequada, com os índices inferiores dos $\hat{\mathbf{U}}$ e $\hat{\mathbf{A}}$ aparecendo, separadamente, na ordem crescente, conforme o número de fatores no produto final aumenta com o avanço da recursão. Segue um exemplo do que ocorre, para $n = 3$:

$$\hat{\mathbf{U}} = \hat{\mathbf{U}}_1^1 \hat{\mathbf{A}}_{\zeta(1,1)}^1 \hat{\mathbf{U}}_2^1 = \hat{\mathbf{U}}_1^2 \hat{\mathbf{A}}_{\zeta(2,1)}^2 \hat{\mathbf{U}}_2^2 \hat{\mathbf{A}}_{\zeta(1,1)}^1 \hat{\mathbf{U}}_3^2 \hat{\mathbf{A}}_{\zeta(2,2)}^2 \hat{\mathbf{U}}_4^2 = \hat{\mathbf{U}}_1^2 \hat{\mathbf{A}}_1^2 \hat{\mathbf{U}}_2^2 \hat{\mathbf{A}}_2^1 \hat{\mathbf{U}}_3^2 \hat{\mathbf{A}}_3^2 \hat{\mathbf{U}}_4^2. \tag{3.32}$$

Caso denote-se como Q_U e Q_A as quantidades de matrizes $\hat{\mathbf{U}}$ e $\hat{\mathbf{A}}$ no produto, respectivamente, então estas quantidades serão dadas por:

$$Q_U = 2^{n-1} = \frac{N}{2}, \tag{3.33}$$

$$Q_A = \sum_{j=0}^{n-1} 2^j = 2^{n-1} - 1 = \frac{N}{2} - 1. \tag{3.34}$$

Enquanto as matrizes $\hat{\mathbf{U}}$ são substituídas no produto pelas duas próximas no avanço da recursão, as $\hat{\mathbf{A}}$ seguem se acumulando, de tal forma que Q_U é um termo de uma PG, enquanto Q_A é uma soma de termos de PG (mas com um termo inicial menor). O prosseguimento posterior da fatoração exemplificada na equação (3.32) resulta no produto expresso por:

$$\hat{\mathbf{U}} = \left(\prod_{i=1}^{\frac{N}{2}-1} \hat{\mathbf{U}}_i^{n-1} \hat{\mathbf{A}}_i^{\gamma(i)}\right) \hat{\mathbf{U}}_{\frac{N}{2}}^{n-1}. \tag{3.35}$$

Os índices γ devem corresponder à primeira posição, da esquerda para a direita, na qual os algarismos do vetor de representação binária de j e $j+1$ diferem entre si. Isto

se deve ao fato de os $\hat{\mathbf{A}}_j^i$ se arranjarem no produto de uma maneira similar à organização espacial dos nós de uma árvore como a ilustrada na Fig. 19, para n=3.

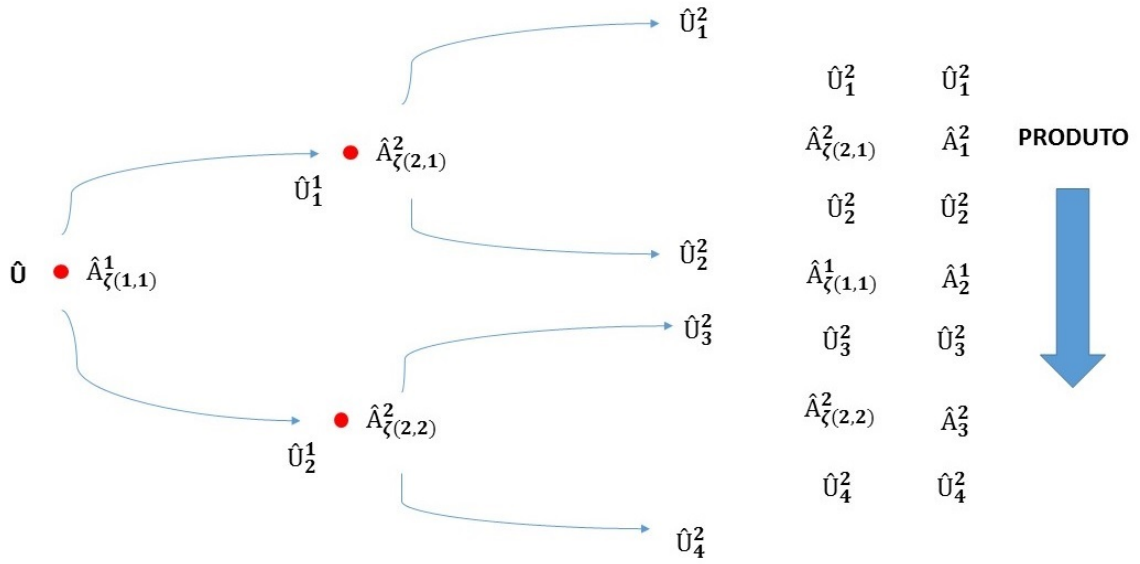

Figura 19 – Arranjo do produto de termos matriciais. A árvore à esquerda mostra como a fatoração progride, com cada matriz $\hat{\mathbf{U}}$ que surge sendo decomposta em um fator do tipo $\hat{\mathbf{U}}\hat{\mathbf{A}}\hat{\mathbf{U}}$, enquanto a seta da direita mostra a ordem correta do produto, de acordo com os componentes mais externos da árvore (de cima para baixo).

As matrizes $\hat{\mathbf{A}}_j^i$ podem ser representadas por operações do tipo $F_i^{n-1}(\hat{\mathbf{R}}_y)$, com $\hat{\mathbf{R}}_y$ dado pela equação (3.3). A ação de $F_i^{n-1}(\hat{\mathbf{R}}_y)$ em cada um dos elementos da base vetorial do espaço de Hilbert resulta sempre em uma combinação linear de dois elementos deste mesmo conjunto, os quais são determinados pelas posições dos *q-bits* alvo e dos *q-bits* de controle, com coeficientes $sen(\theta_i)$ e $cos(\theta_i)$. Nesta situação, portanto, é possível escolher a posição do *q-bit* alvo da rotação para formar os arranjos angulares descritos pela equação (3.30). As matrizes $\hat{\mathbf{U}}_j^{n-1}$ são bloco-diagonais, mas ainda não podem ser implementadas diretamente por meio de operações uniformemente controladas. Embora os resultados até então discutidos sejam suficientes para designar circuitos que possibilitam a implementação de uma dada matriz $\hat{\mathbf{U}}$, há uma maneira de reduzir a quantidade de portas necessárias para construí-la.

Define-se matrizes $\hat{\mathbf{P}}_{\mathbf{j}}^{\mathbf{i}} = diag_{2^i}(\hat{\mathbf{p}}_{j,[k/2]}^i)$, com $\hat{\mathbf{p}}_{jk}^i = diag_{2^{n-i}}(e^{i\alpha_l})$. A substituição de $\hat{\mathbf{I}} = \hat{\mathbf{P}}_{\zeta(i,j)}^i \left(\hat{\mathbf{P}}_{\zeta(i,j)}^i\right)^\dagger$ nas equações (3.31) e (3.35), em conjunto com o fato de $\hat{\mathbf{P}}_{\zeta(i,j)}^i$ e $\hat{\mathbf{A}}_{\zeta(i,j)}^i$ comutarem, resulta em:

$$\hat{\mathbf{U}}_j^{i-1} = \hat{\mathbf{U}}_{2j-1}^i \hat{\mathbf{P}}_{\zeta(i,j)}^i \hat{\mathbf{A}}_{\zeta(i,j)}^i \hat{\mathbf{U}}_{2j}^i, \tag{3.36}$$

$$\hat{\mathbf{U}} = \left(\prod_{i=1}^{\frac{N}{2}-1} \hat{\mathbf{U}}_i^{n-1} \hat{\mathbf{P}}_i^{\gamma(i)} \hat{\mathbf{A}}_i^{\gamma(i)} \right) \hat{\mathbf{U}}_{\frac{N}{2}}^{n-1}. \tag{3.37}$$

A matriz $\left(\hat{\mathbf{P}}^i_{\zeta(i,j)}\right)^\dagger$ é incorporada nos $\hat{\mathbf{U}}^i_{2j}$ na equação (3.36), pois a multiplicação por estas não compromete a estrutura bloco-diagonal.

Os ângulos $\{\alpha_l\}$ que definem $\hat{\mathbf{p}}^i_{jk}$ podem ser escolhidos de forma a possibilitar a implementação do produto $\hat{\mathbf{B}}_i = \hat{\mathbf{U}}^{n-1}_i \hat{\mathbf{P}}^{\gamma(i)}_i$ através de rotações uniformemente controladas, de acordo com a seguinte relação:

$$\hat{\mathbf{B}}_i = F^{n-1}_n(\hat{\mathbf{R}}_z) F^{n-1}_n(\hat{\mathbf{R}}_y) F^{n-1}_{\gamma(i)}(\hat{\mathbf{R}}_z). \tag{3.38}$$

Uma vez que as matrizes $\hat{\mathbf{P}}^{\gamma(i)}_i$ são determinadas pelas $\hat{\mathbf{U}}^{\mathbf{i}}_{\mathbf{j}}$, há uma ordem correta para prosseguir com a decomposição recursiva. As matrizes com índices superiores maiores tem prioridade na decomposição, e sendo estes iguais, devem ser decompostas primeiro as com índices inferiores menores. Sendo as matrizes $\hat{\mathbf{A}}^i_j$ representadas por $F^{n-1}_i(\hat{\mathbf{R}}_y)$, então os termos que compõem o produto expresso na equação (3.37) podem ser descritos por:

$$(\hat{\mathbf{B}}\hat{\mathbf{A}})_i = \hat{\mathbf{U}}^{n-1}_i \hat{\mathbf{P}}^{\gamma(i)}_i \hat{\mathbf{A}}^{\gamma(i)}_i = F^{n-1}_n(\hat{\mathbf{R}}_z) F^{n-1}_n(\hat{\mathbf{R}}_y) F^{n-1}_{\gamma(i)}(\hat{\mathbf{R}}_z) F^{n-1}_{\gamma(i)}(\hat{\mathbf{R}}_y). \tag{3.39}$$

O termo final $\hat{\mathbf{U}}^{n-1}_{N/2}$ é descrito pelo conjunto de rotações uniformemente controladas expresso por:

$$\hat{\mathbf{U}}^{n-1}_{N/2} = \left(\prod_{i=0}^{n-1} F^{n-1-i}_{n-i}(\hat{\mathbf{R}}_z) F^{n-1-i}_{n-i}(\hat{\mathbf{R}}_y) F^{n-1}_{n-i}(\hat{\mathbf{R}}_z) \right) \hat{\mathbf{\Phi}}. \tag{3.40}$$

Esgota-se a possibilidade de manipulação de graus de liberdade extras através da inserção de $\hat{\mathbf{P}}^i j$ ao se alcançar o termo final. O termo $\hat{\mathbf{\Phi}}$ corresponde a uma porta cuja finalidade é fixar a fase global que não pode ser observada, a qual pode ser implementada através de uma única operação.

A implementação de uma operação unitária $\hat{\mathbf{U}}$ através da decomposição CS e de rotações uniformemente controladas pode ser realizada através do produto descrito na equação (3.37), cujos fatores são especificados pela equação (3.39). O custo computacional em termos de quantidade de portas para tal realização é de $O(4.2^{n-1}.2^{n-1}) = O(N^2)$ (pois cada F^{n-1}_n tem complexidade $O(2^{n-1})$), enquanto a implementação de cada um dos fatores da equação (3.40) escala com $O(2^{n-1-i})$, resultando em uma complexidade total para implementação do produto completo correspondente à soma de uma PG com n termos e razão 1/2, ou seja, $O(2^{n-1}(2^n - 1)) = O(4^n) = O(N^2)$.

3.3 Considerações finais

Segue, na tabela 3, um resumo dos métodos abordados neste capítulo e suas respectivas complexidades.

A análise adequada de um circuito quântico deve sempre considerar a complexidade de implementação de cada uma das portas que de fato depende do número de *q-bits*, e não

Tabela 3 – Complexidade de operações fundamentais ($N = 2^n$).

Tipo de operação	**Descrição**	**Complexidade**
Operações controladas.	Operações de um *q-bit* controladas por n *q-bits*, implementadas através de cascatas de portas de Toffoli e alternação do uso dos *q-bits* ancilla.	$O(n^2)$, sem um *q-bit ancilla*, ou $O(n)$, com um *q-bit ancilla* ($n = log_2(N)$).
Operações gerais de n *q-bits*	Operações gerais de n *q-bits* implementadas através de fatoração em matrizes de dois níveis, que podem ser construídas, cada uma, através de portas $C^{n-1}\hat{\mathbf{X}}$ e uma $C^{n-1}\hat{\mathbf{U}}$.	$O(n^3 4^n) = O(N^2 \left(log_2(N)\right)^3)$
Operações gerais de n *q-bits*.	Operações gerais de n *q-bits*, implementadas através de permutações das bases computacionais que permitem eliminar as portas $C^{n-1}\hat{\mathbf{X}}$, e com eliminação de nós supérfluos de controle.	$O(4^n) = O(N^2)$.
Operações gerais de n *q-bits*	Operações gerais de n *q-bits* implementadas através da utilização de portas uniformemente controladas para construir as matrizes bloco-diagonais oriundas da decomposição cosseno-seno.	$O(4^n) = O(N^2)$.

somente largura e profundidade. Segue na Fig. 20 um exemplo de circuito com diversos tipos de portas diferentes, cuja análise detalhada da complexidade é exibida na tabela 4. O circuito utilizado como exemplo possui profundidade 5 e largura 4. As portas que podem ser executadas simultaneamente são exibidas verticalmente alinhadas.

Embora não seja apropriado substituir valores numéricos em ordens assintóticas de complexidade devido à necessidade de que fique explícito a função que estas representam, tal procedimento é adotado apenas para ilustrar o fato de que a complexidade pode ser analisada com base na porta mais complexa do circuito, visto que a ordem assintótica está relacionada à parcela de maior valor para números grandes quando há termos aditivos (resultando em uma complexidade total de $O(256)$ para o exemplo apresentado).

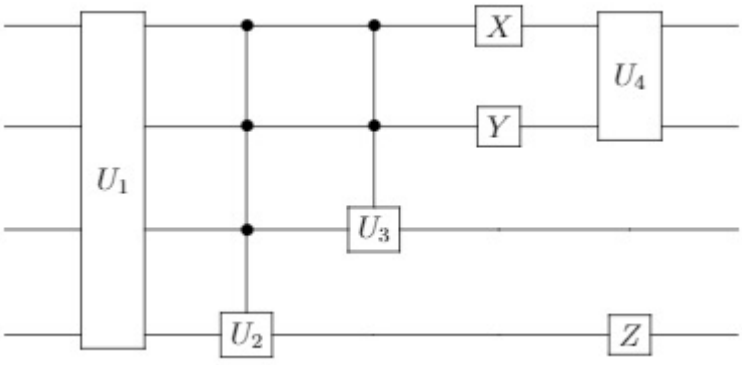

Figura 20 – Exemplo de circuito utilizado para análise de complexidade, com portas de múltiplos *q-bits* ($O(N^2)$), portas controladas ($O(n^2)$) e portas de um único *q-bit*. As portas que aparecem verticalmente alinhadas podem ser executadas simultaneamente.

Tabela 4 – Exemplo de análise de complexidade.

Portas	**Complexidade**
Porta $\hat{\mathbf{U}}_1$	$O(4^4) = O(256)$
Porta $\mathbf{C^3}\hat{\mathbf{U}}_2$	$O(3^2) = O(9)$
Porta $\mathbf{C^2}\hat{\mathbf{U}}_3$	$O(2^2) = O(4)$
Portas $\hat{\mathbf{X}}$ e $\hat{\mathbf{Y}}$	$O(1)$
Portas $\hat{\mathbf{U}}_4$ e $\hat{\mathbf{Z}}$	$O(4^2) = O(16)$ (complexidade da mais custosa)
Complexidade total:	$O(256)$

4 Algoritmos para Equações Diferenciais

O tipo de sistema mais comumente tratado na literatura é o de equações diferenciais ordinárias lineares de primeira ordem, descrito por:

$$\frac{d\vec{\mathbf{x}}}{dt} = \mathbf{A}(t)\vec{\mathbf{x}} + \vec{\mathbf{B}}(t). \tag{4.1}$$

A função $\mathbf{A}(t)$ é expressa por uma matriz NxN, enquanto $\vec{\mathbf{x}}(t)$ e $\vec{\mathbf{B}}(t)$ são representadas por vetores. O valor de $\vec{\mathbf{x}}$ é fornecido em um dado instante inicial, e objetivo do método é determina-lo em qualquer instante Ou seja, trata-se de um típico PVI (problema de Valor Inicial).

A razão do enfoque neste sistema se deve ao fato de ser possível reduzir equações diferencias parciais a sistemas de equações acopladas de primeira ordem através do método de separação de variáveis, assim como equações de ordem superior também podem ser reduzidas a tais sistemas através da introdução de variáveis extras que representam as derivadas de cada ordem envolvida na equação [34].

O primeiro algoritmo quântico proposto para solucionar um sistema de equações diferenciais lineares acopladas foi abordado de forma específica no trabalho de Berry [34], no qual utiliza-se um método multipassos que emprega o método de Euler para reduzir o problema à resolução de um sistema linear, tarefa para a qual já havia sido anteriormente proposto um algoritmo quântico [35]. Outro trabalho desenvolvido por Berry *et. al.* [36] foca em sistemas com coeficientes constantes, resolvendo-os através de um método multipassos que também permite a redução do problema a um sistema linear. Os autores ressaltam o fato de haver ganho exponencial na dependência da complexidade com relação à precisão ϵ, quando comparado ao trabalho anteriormente desenvolvido por eles próprios.

Por sua vez, o trabalho desenvolvido por Childs *et. al.* [37], utiliza métodos espectrais para resolver o sistema geral descrito pela equação (4.1) com coeficientes dependentes do tempo, alegando obter uma complexidade da ordem $O(poly(log(d), log\left(\frac{1}{\epsilon}\right))$, sendo d a dimensão do sistema e ϵ a precisão de aproximação.

Um artigo que reflete bem as generalidades dos métodos de resolução em vários aspectos e enfatiza o ganho exponencial de complexidade com relação à dimensão N do vetor $\vec{\mathbf{x}}$ (quando comparado aos algoritmos clássicos existentes) foi publicado por Tao Xin *et. al.* [23]. Devido à sua representatividade de aspectos e alegação do fato intrigante do ganho exponencial de tempo, tal trabalho foi escolhido para análise detalhada nesta

dissertação pelo autor da mesma.

Há aspectos importantes que podem ser observados nestes algoritmos. O primeiro deles é a codificação de soluções nas amplitudes de probabilidade de algum sistema quântico (muitas vezes em um subespaço do espaço de Hilbert que o descreve). Alguns algoritmos próprios para a resolução de equações diferenciais parciais, como o desenvolvido por Fillion-Gourdeau [38] para equações hiperbólicas simétricas de primeira ordem e outro para resolução de equações diferencias parciais não-lineares [39] também codificam as soluções nas amplitudes.

A codificação de soluções nas amplitudes associadas a um subespaço de um sistema quântico faz com que a probabilidade de sucesso do algoritmo esteja relacionada à dimensão desse subespaço, pois extrair as soluções dependerá da realização de medidas. Tal cenário sugere a possibilidade de utilização de rotinas de amplificação de amplitude (originalmente desenvolvidas por Brassard e Hoyer [40]) para aumentar a probabilidade de sucesso.

Outro aspecto digno de atenção é a necessidade de preparação de estados quânticos que codificam as variáveis de entrada a serem fornecidas para o algoritmo, etapa esta cuja complexidade é por vezes negligenciada nas análises feitas pelos próprios desenvolvedores.

4.1 Amplificação de amplitude

4.1.1 Estrutura matemática

O algoritmo de amplificação de amplitude, cujo procedimento original foi desenvolvido por Brassard e Hoyer [40], pode ser considerado uma generalização do algoritmo de Grover [41]. Os fundamentos foram paralelamente trabalhados por outros autores, dentre os quais se destaca o próprio Grover e seu trabalho posterior [42].

O procedimento para modificar as amplitudes do estado quântico gerado por algum algoritmo genérico que não realiza medições durante sua execução (podendo portanto ser representado por um operador $\hat{\mathbf{A}}$) faz uso da aplicação repetida do operador $\hat{\mathbf{Q}}$ descrito pela equação (4.2), agindo inicialmente no estado $\hat{\mathbf{A}}\,|0\rangle$, que pode ter qualquer dimensão N.

$$\hat{\mathbf{Q}} = -\hat{\mathbf{A}}\hat{\mathbf{S}}_0\hat{\mathbf{A}}^{\dagger}\hat{\mathbf{S}}_x. \tag{4.2}$$

O operador $\hat{\mathbf{S}}_0$ modifica o sinal do estado $|0\rangle$ e mantém todos os outros intactos,

logo deve ser descrito como $\hat{\mathbf{S}}_0 = \hat{\mathbf{I}} - 2\,|0\rangle\,\langle 0|$, e sua representação matricial é dada por:

$$\hat{\mathbf{S}}_0 = \hat{\mathbf{I}} - 2\,|0\rangle\,\langle 0| = \begin{bmatrix} -1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{4.3}$$

O estado N-dimensional produzido por uma única aplicação do algoritmo descrito por $\hat{\mathbf{A}}$ pode ser separado em duas parcelas: Uma de interesse (da qual se deseja aumentar a probabilidade de detecção) e outra de desinteresse (a qual terá sua probabilidade de detecção reduzida). O estado de interesse (denotado por $|\psi_i\rangle$) é representado por:

$$|\psi_i\rangle = \sum_{t=0}^{k} \alpha_t\,|t\rangle\,. \tag{4.4}$$

Já o estado de desinteresse é representado por:

$$|\psi_d\rangle = \sum_{t'=k+1}^{N-1} \alpha'_t\,|t'\rangle\,. \tag{4.5}$$

O algoritmo de amplificação é válido quando o estado $|\psi\rangle = \hat{\mathbf{A}}\,|0\rangle$ pode ser expresso em uma soma ortogonal dos estados de interesse e desinteresse (ou seja, $\langle\psi_i|\psi_d\rangle = 0$) como expresso a seguir:

$$|\psi\rangle = \hat{\mathbf{A}}\,|0\rangle = |\psi_i\rangle + |\psi_d\rangle\,. \tag{4.6}$$

A condição de ortogonalidade implica na existência de uma base computacional na qual inexiste intersecção entre os conjuntos dos respectivos elementos em termos dos quais $|\psi_i\rangle$ e $|\psi_d\rangle$ são individualmente expandidos.

O operador $\hat{\mathbf{S}}_x$ (também chamado de oráculo) possui a função de inverter o sinal apenas dos estados da base que expandem a componenete $|\psi_i\rangle$ de interesse, ou seja, $\hat{\mathbf{S}}_x = \hat{\mathbf{I}} - \sum_{t=0}^{k} 2\,|t\rangle\,\langle t|$. A representação matricial de um exemplo no qual o estado de interesse pode ser descrito como combinação linear dos estados $|0\rangle$ e $|1\rangle$ é descrita por:

$$\hat{\mathbf{S}}_x = \hat{\mathbf{I}} - 2\,|0\rangle\,\langle 0| - 2\,|1\rangle\,\langle 1| = \begin{bmatrix} -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{4.7}$$

O trabalho desenvolvido por Brassard *et al.* [40] não menciona diretamente o fato de $|\psi_i\rangle$ ser um estado expansível em termos de elementos da base computacional. Grover [42],

por sua vez, enfatiza esse cenário. Há diferenças sutis na abordagem e no desenvolvimento matemático adotado pelos autores, contudo, ambos induzem às mesmas conclusões.

É possível denotar o termo $a = \langle\psi_i|\psi_i\rangle$. Este representa a probabilidade de se medir o estado de interesse, caso faça-se um tipo de medição capaz de detecta-lo, ou de se medir algum dos estados que o expandem, caso sejam realizadas medições apenas de elementos da base computacional. Ambas interpretações são equivalentes, e a representa a probabilidade de ocorrência de um dentre dois eventos excludentes, então é possível parametrizar o termo como:

$$a = \sum_{t=0}^{k} |\alpha_t|^2 = sen^2(\theta_a). \tag{4.8}$$

É possível provar que o subespaço bidimensional expandido em termos dos vetores $|\psi_i\rangle$ e $|\psi_d\rangle$ é fechado sob a ação do operador $\hat{\mathbf{Q}}$, ou seja, a ação deste em um elemento do subespaço sempre resulta em outro elemento do mesmo.

É possível observar a ação de $\hat{\mathbf{Q}}$ em $|\psi\rangle$ através das equações (4.9)-(4.11):

$$\hat{\mathbf{Q}}\,|\psi\rangle = \hat{\mathbf{Q}}\hat{\mathbf{A}}\,|0\rangle = -\hat{\mathbf{A}}\hat{\mathbf{S}}_0\hat{\mathbf{A}}^\dagger\hat{\mathbf{S}}_x\hat{\mathbf{A}}\,|0\rangle = \left(-\hat{\mathbf{A}} + 2\hat{\mathbf{A}}\,|0\rangle\langle 0|\right)\left(-\hat{\mathbf{A}}^\dagger\,|\psi\rangle + \sum_{t=0}^{k} 2\hat{\mathbf{A}}^\dagger\,|t\rangle\langle t|\,|\psi\rangle\right), \tag{4.9}$$

$$\hat{\mathbf{A}}^\dagger\,|\psi\rangle = |0\rangle\,;\langle 0|\,\hat{\mathbf{A}}^\dagger\,|t\rangle = \alpha_t^* \Rightarrow \hat{\mathbf{Q}}\,|\psi\rangle = \left(1 - 4\sum_{t=0}^{k}|\alpha_t|^2\right)\hat{\mathbf{A}}\,|0\rangle + \sum_{t=0}^{k} 2\alpha_t\,|t\rangle\,, \tag{4.10}$$

$$|\psi\rangle = \hat{\mathbf{A}}\,|0\rangle = |\psi_i\rangle + |\psi_d\rangle = \sum_{t=0}^{k}\alpha_t\,|t\rangle \Rightarrow \hat{\mathbf{Q}}\,|\psi_i\rangle + \hat{\mathbf{Q}}\,|\psi_d\rangle = (1-4a)\,|\psi_d\rangle + (3-4a)\,|\psi_i\rangle\,. \tag{4.11}$$

De uma maneira similar, é notável também a ação de $\hat{\mathbf{Q}}^\dagger$ em $|\psi\rangle$:

$$\hat{\mathbf{Q}}^\dagger = -\hat{\mathbf{S}}_x\hat{\mathbf{A}}\hat{\mathbf{S}}_0\hat{\mathbf{A}}^\dagger \Rightarrow \hat{\mathbf{Q}}^\dagger\,|\psi\rangle = -\hat{\mathbf{S}}_x\hat{\mathbf{A}}\hat{\mathbf{S}}_0\,|0\rangle = \hat{\mathbf{S}}_x\,|\psi\rangle\,, \tag{4.12}$$

$$\hat{\mathbf{S}}_x\,|\psi_i\rangle = -\,|\psi_i\rangle\,;\hat{\mathbf{S}}_x\,|\psi_d\rangle = |\psi_d\rangle \Rightarrow \hat{\mathbf{Q}}^\dagger\,|\psi\rangle = |\psi_d\rangle - |\psi_i\rangle\,. \tag{4.13}$$

A multiplicação de ambos os lados da equação (4.13) por $\hat{\mathbf{Q}}$ considerando a unitariedade desse operador resulta em:

$$\hat{\mathbf{Q}}\,|\psi_d\rangle - \hat{\mathbf{Q}}\,|\psi_i\rangle = |\psi_d\rangle + |\psi_i\rangle\,. \tag{4.14}$$

As equações (4.11) e (4.14) formam um sistema que pode ser resolvido para obter a ação do operador $\hat{\mathbf{Q}}$ no subespaço bidimensional invariante expansível em termos de $|\psi_i\rangle$ e $|\psi_d\rangle$:

$$\hat{\mathbf{Q}}\,|\psi_i\rangle = (1-2a)\,|\psi_i\rangle - 2a\,|\psi_d\rangle\,, \tag{4.15}$$

$$\hat{\mathbf{Q}}\,|\psi_d\rangle = 2(1-a)\,|\psi_i\rangle + (1-2a)\,|\psi_d\rangle\,. \tag{4.16}$$

Através das duas últimas equações, é possível obter a representação matricial do operador $\hat{\mathbf{Q}}$ neste subespaço (denotado γ):

$$\hat{\mathbf{Q}} = \begin{bmatrix} (1-2a) & -2a \\ 2(1-a) & (1-2a) \end{bmatrix}. \tag{4.17}$$

Os autovetores e autovalores de $\hat{\mathbf{Q}}$ expressos em termos de sua representação no subespaço γ são dados, respectivamente, por:

$$|\psi_\pm\rangle = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a}}\,|\psi_i\rangle \pm \frac{i}{\sqrt{1-a}}\,|\psi_d\rangle\right), \tag{4.18}$$

$$\lambda_\pm = e^{i2\theta_a}. \tag{4.19}$$

Já o estado $|\psi\rangle$ pode ser descrito em termos dos autovetores de $\hat{\mathbf{Q}}$ através da equação (4.20):

$$|\psi\rangle = \hat{\mathbf{A}}\,|0\rangle = \frac{-i}{\sqrt{2}}\left(e^{i\theta_a}\,|\psi_+\rangle - e^{-i\theta_a}\,|\psi_-\rangle\right). \tag{4.20}$$

A escolha pela representação $|\pm\rangle$ (em detrimento de $|\psi_i\rangle$ e $|\psi_d\rangle$) permite acessar o estado final após a ação sequencial do operador $\hat{\mathbf{Q}}$ m vezes:

$$\hat{\mathbf{Q}}^m\,|\psi\rangle = \frac{-i}{\sqrt{2}}\left(e^{(2m+1)i\theta_a}\,|\psi_+\rangle - e^{-(2m+1)i\theta_a}\,|\psi_-\rangle\right). \tag{4.21}$$

Por sua vez, a substituição da equação (4.18) na (4.21) resulta em:

$$\hat{\mathbf{Q}}^m\,|\psi\rangle = \frac{1}{\sqrt{a}} sen((2m+1)\theta_a)\,|\psi_i\rangle + \frac{1}{\sqrt{1-a}} cos((2m+1)\theta_a)\,|\psi_d\rangle\,. \tag{4.22}$$

Através desta última equação, verifica-se que a probabilidade de medição do estado $|\psi_i\rangle$ (ou de um dos estados $|t\rangle$, caso sejam feitas medidas na base computacional) é dada por:

$$p = sen^2((2m+1)\theta_a). \tag{4.23}$$

O termo θ_a é o mesmo descrito pela equação (4.8).

A Fig. 21 mostra a variação da probabilidade p (descrita pela equação (4.23)) em função do número m de aplicações de $\hat{\mathbf{Q}}$ para quatro ordens de grandeza da probabilidade inicial a ($10^{-4}, 10^{-3}, 10^{-2}$ e 10^{-1}). Observa-se que apenas para valores muito pequenos de a (e, consequentemente, valores pequenos de θ_a) a probabilidade de sucesso poderá seguramente ser amplificada caso não a conheçamos *a priori*, para uma ampla faixa de valores de m. Quando a magnitude de a é da ordem de centésimos já é possível observar

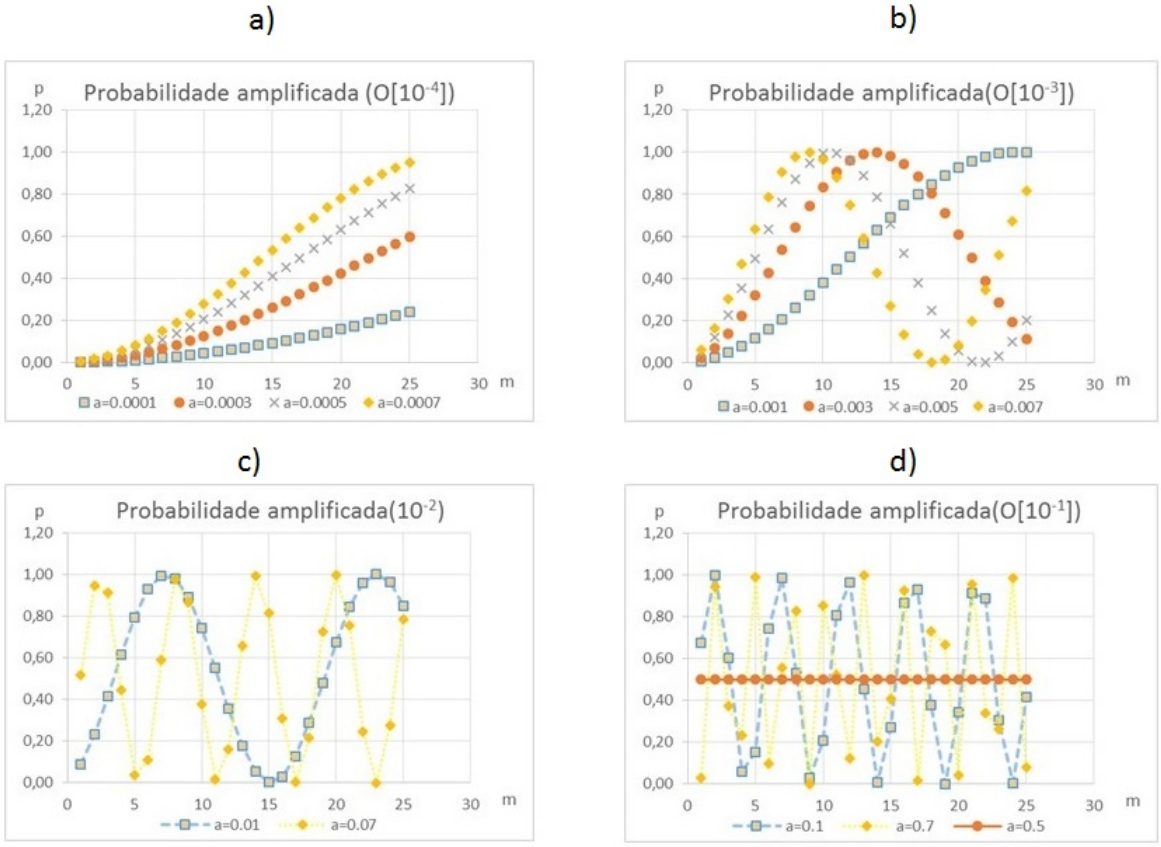

Figura 21 – Variação da probabilidade final de sucesso (eixo y) com o número de aplicações sucessivas de amplificação de amplitude (eixo x) para diferentes ordens de grandeza da probabilidade inicial de sucesso: a) Ordem de 10^{-4}. b) Ordem de 10^{-3}. c) Ordem de 10^{-2}. d) Ordem de 10^{-1}.

tanto decréscimo quanto aumento da probabilidade, mesmo para poucas repetições da aplicação de $\hat{\mathbf{Q}}$.

A ausência de monotonia crescente na variação da probabilidade evidencia que para a obtenção do efeito amplificativo desejado, existe necessidade de conhecimento prévio ao menos da ordem de grandeza das amplitudes as quais se deseja amplificar (há, inclusive, valores de a para os quais a probabilidade se mantém constante, a exemplo de $a = 0.5$).

Caso o valor de a seja realmente muito pequeno (e, de modo semelhante, $sen(\theta_a)$) é possível obter a aproximação da equação (4.23) como $\theta_a = \sqrt{a}$, o que resulta em:

$$p = (2m + 1)^2 a. \tag{4.24}$$

Denotando-se m_c como o número de repetições necessárias para alcançar 100% de probabilidade de obtenção do estado de interesse $|\psi_i\rangle$ (ou uma das bases que o compõe, dependendo do que se mede), como resultado direto da equação (4.23) obtêm-se $(2m_c+1) = \pi/\sqrt{(a)}$, ou seja, $m_c = O(1/\sqrt{(a)})$.

Para fixar rigorosamente a probabilidade final em p=1, seria necessário conhecer o

valor da probabilidade inicial a; todavia o comportamento de p é monotônico crescente tanto com m quanto com a apenas para valores pequenos desta última grandeza, conforme pode ser observado na figura 21. Tal comportamento garante que seja possível amplificar substancialmente a probabilidade caso seja conhecido ao menos um limite superior suficientemente pequeno para a.

O algoritmo de Grover utiliza um operador $\hat{\mathbf{Q}}$ no qual $\hat{\mathbf{A}}$ é a transformada de Walsh-Haddamard, a qual produz uma superposição uniforme dos N possíveis estados da base computacional utilizada para codificar os elementos relativos à busca. Nestas condições, portanto, $a = 1/N$, logo resulta o fato amplamente conhecido de ser necessário repetir o procedimento um número de vezes que escala com $O(\sqrt{N})$.

4.1.2 Análise de complexidade

Segue, na Fig. 22, uma representação do circuito utilizado para a implementação do operador $\hat{\mathbf{S}}_0$. A porta $\hat{\mathbf{Z}}$ possui o efeito de modificar apenas o sinal do estado $|1\rangle$, mantendo o coeficiente de $|0\rangle$ intacto. Logo, a ação de $\mathbf{C^{n-1}\hat{Z}}$ modificaria apenas o sinal do estado $|00..01\rangle$. A adição de portas $\hat{\mathbf{X}}$ intercaladas com a $\hat{\mathbf{Z}}$ garante o resultado discutido nos capítulos anteriores: A transferência da ação de uma operação que tem como alvo exclusivo apenas um estado específico (representado por um sequência binária) para outro, através da permutação de alguns elementos da respectiva sequência, uma vez que a ação de $\hat{\mathbf{X}}$ converte $|0\rangle$ em $|1\rangle$ (e *vice-versa*). O resultado, portanto, consiste na alteração apenas do sinal do estado $|00..00\rangle$.

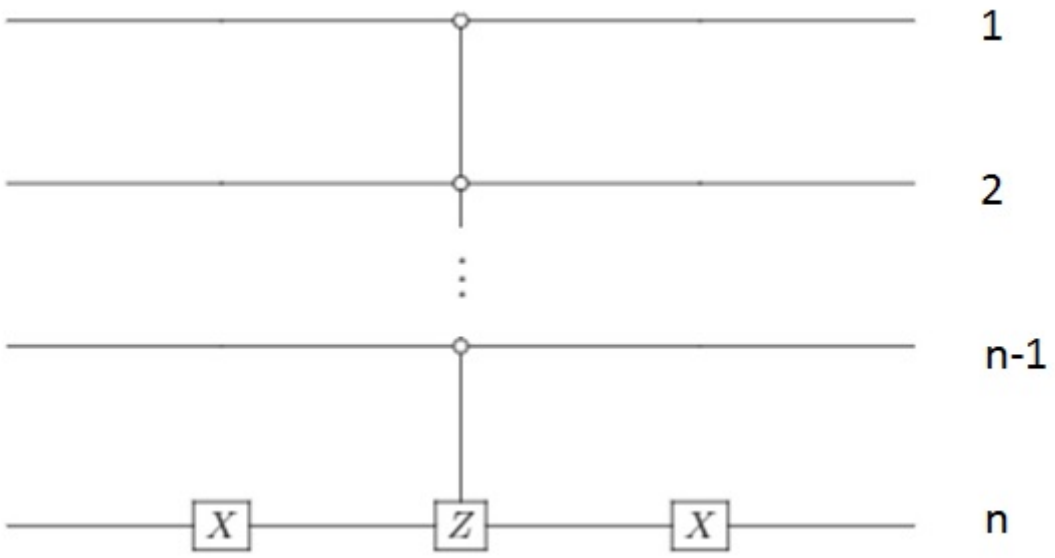

Figura 22 – Implementação de $\hat{\mathbf{S}}_0$. A porta $\hat{\mathbf{Z}}$ possui o efeito intrínseco de alterar apenas o sinal do estado $|1\rangle$, e o fato de a mesma estar intercalada por portas $\hat{\mathbf{X}}$ e com $n-1$ controles abertos (ativados somente pelo estado $|0\rangle$) resulta na mudança de sinal de $|00...0\rangle$.

Uma estratégia similar à da implementação do operador $\hat{\mathbf{S}}_0$ pode ser utilizada na construção do oráculo $\hat{\mathbf{S}}_x$, como mostra o exemplo exibido na Fig. 23 para estados de interesse $|00001\rangle$ e $|10000\rangle$ em um sistema de cinco *q-bits*. Os operadores $\hat{\mathbf{X}}$ executam a função de intercambiar os estados $|0\rangle$ e $|1\rangle$ (e *vice versa*) para fazer com que uma base específica diferente da $|11111\rangle$ sofra a ação do núcleo $\mathbf{C^{n-1}\hat{Z}}$, que consiste em alteração do sinal do coeficiente associado. Através deste artifício, é possível construir portas que modificam especificamente o sinal de um ou mais estados escolhidos da base computacional (representados por algarismos binários), deixando os demais intactos.

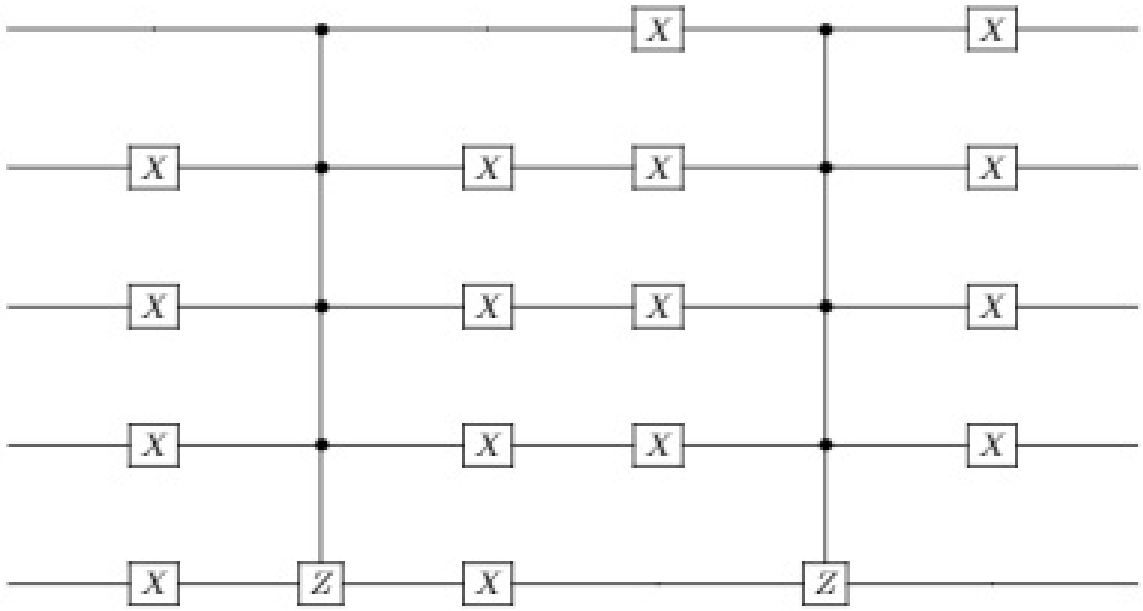

Figura 23 – Implementação de $\hat{\mathbf{S}}_x$ (sistema de cinco *q-bits* e estados alvo $|00001\rangle$ e $|10000\rangle$). A porta $\hat{\mathbf{Z}}$ controlada alteraria apenas o sinal do estado $|11111\rangle$, mas a inserção das portas $\hat{\mathbf{X}}$ na devida posição torna possível alterar o sinal de algum outro estado alvo.

É importante ressaltar que embora esse processo apresente certa similaridade com as rotações uniformemente controladas, a implementação de $\hat{\mathbf{S}}_x$ não se enquadra como um caso específico dessa categoria, pois embora esta possa ser representada por matrizes bloco-diagonais, os quatro possíveis blocos que podem vir a aparecer na matriz de $\hat{\mathbf{S}}_x$ em diferentes ordens e quantidades não podem ser descritos como uma rotação em torno do mesmo eixo $\vec{a}$ com ângulos diferentes, a qual é uma condição essencial para a implementação das rotações uniformemente controladas apresentada em [31]. Estes blocos são descritos por:

$$BL_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; BL_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}; BL_3 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}; BL_4 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{4.25}$$

Cada $\mathbf{C^{n-1}\hat{Z}}$ demanda $O(n^2)$ operações básicas, e o estado de interesse $|\psi_i\rangle$ é descrito por k estados da base computacional. Resultará, portanto, que a complexidade de

implementação do oráculo será $O(n^2k)$, pois este corresponde a uma $\mathbf{C^{n-1}\hat{Z}}$ entremeada pelas devidas $\hat{\mathbf{X}}$ para cada um dos k estados (denotados individualmente por $|t\rangle$) que expandem $|\psi_i\rangle$. Uma vez que cada fileira de portas $\hat{\mathbf{X}}$ pode ser executada através da ação simultânea de seus componentes, estas modificam a complexidade apenas em termos de uma constante aditiva, portanto não há necessidade de contabiliza-las em conjunto com as $\mathbf{C^{n-1}\hat{Z}}$. Essa complexidade pode ser reduzida a $O(nk)$ através do acréscimo de apenas um *q-bit* auxiliar, devido aos fundamentos anteriormente discutidos no capítulo 3.

Caso denotemos como $c(\hat{\mathbf{A}})$ a complexidade individual de $\hat{\mathbf{A}}$, a complexidade total da realização de m repetições do operador $\hat{\mathbf{Q}}$ será dada por $O(m\{max[c(\hat{\mathbf{A}}), nk]\})$, já que são efetuadas m repetições do oráculo e do algoritmo $\hat{\mathbf{A}}$ na sequência prevista pela equação (4.2). Ou seja, uma vez que $n = log_2(N)$, não haverá grandes perdas em termos de eficiência caso o processo de amplificação de amplitude seja necessário para melhorar o desempenho de algum algoritmo cujo tempo de execução dependa de N.

Uma precaução aconselhável é não neglicenciar o oráculo nas análises de complexidade. O algoritmo de Grover é muitas vezes referido como $O(\sqrt{N})$, porém tal classificação leva em conta somente a quantidade de repetições. Caso deseje-se uma análise mais representativa da quantidade de portas necessárias (e consequentemente do tempo de execução) é necessário contabilizar as $log_2(N)$ operações que constituem o oráculo em cada repetição. Desta maneira, conclui-se que a complexidade é, na realidade, $O(\sqrt{N}log(N))$, conforme cuidadosamente elucidado em [43].

4.2 Preparação de estados

Uma série de algoritmos, dentre os quais é possível citar o de Lloyd para resolução de sistemas de equações lineares [35], vários algoritmos para resolução de equações diferenciais [23, 34, 36, 38] e alguns protocolos de *machine learning* [44], requerem procedimentos de preparação de estados quânticos. Dada a correlação direta entre amplitudes e probabilidades, a simulação de distribuições de probabilidades também pode ser considerada como uma possível aplicação [42].

A tarefa de preparar estados consiste, em si, em um algoritmo quântico, que geralmente recebe como parâmetro um conjunto de coeficientes $\{\alpha_1, ..., \alpha_N\}$ e os codifica, a partir de algum estado inicial $|i\rangle$ (geralmente $|0\rangle$), em um estado quântico de acordo com a seguinte relação:

$$|i\rangle \Rightarrow |\psi\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle . \tag{4.26}$$

É comum utilizar uma abordagem clássico-quântica, na qual algumas etapas do processo são clássicas e outras quânticas, e exemplo do algoritmo de preparação baseado

em divisão e conquista apresentado em [45]. Tal fato se deve à dificuldade de implementar certas operações em sistemas quânticos, como o cálculo de arcos-seno [46].

Existe uma série de métodos para preparação de estados quânticos. Uma listagem descritiva pode se encontrada em [47]. Dentre estes, é possível citar o de divisão e conquista [45], a utilização do algoritmo de Grover [42], o método tradicional [48] e os métodos que utilizam uma *q-ram* (*quantum random access memory*) [49, 50]. Cada método possui vantagens e desvantagens de acordo com a respectiva aplicação em contexto específico.

O método de divisão e conquista aborda o problema dividindo-o em partes e solucionando-as separadamente. Esta abordagem utiliza $O(N)$ *q-bits* (devido à necessidade de vários *q-bits* auxiliares, além dos $log_2(N)$ de trabalho) e um tempo de execução $O(log^2(N))$. Em determinadas situações, este método apresenta uma permuta atrativa entre as complexidades espacial e temporal (já que a complexidade temporal de alguns métodos $O(N)$ ou $O(\sqrt{N})$). Todavia, a estrutura do estado de saída não é idêntica à apresentada na equação (4.26), mas sim descrita por:

$$|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i \, |i\rangle \, |\psi_i\rangle \, . \tag{4.27}$$

Devido à estrutura do estado produzido, este método não seria adequado à utilização no algoritmo para equações diferenciais que será discutido posteriormente, já que a aplicação do mesmo demanda a existência de um estado semelhante ao descrito na equação (4.26).

Uma Q-RAM é um dispositivo capaz de armazenar estados quânticos, disponibilizando-os para acesso posterior. Há métodos disponíveis para implementação de tais dispositivos através de técnicas distintas, dentre as quais é possível citar a *bucket brigade* e a *flip-flop* [49, 50].

Os métodos que utilizam Q-RAM apresentam ganho exponencial de complexidade temporal com relação ao método tradicional caso o estado seja anteriormente preparado e armazenado. Todavia, a execução desta etapa anterior tem custo temporal que escala com $O(N)$ [51, 52], equiparando-os assim ao método tradicional por rotações uniformemente controladas caso seja feita a análise do processo em sua totalidade. Além disso, há necessidade da utilização de *q-trits* (sistemas quânticos com estados tridimensionais, em contrapartida aos estados bidimensionais dos *q-bits*) no caso da BB-QRAM. Esses pormenores, além das dificuldades tecnológicas em preservar os estados sem efeitos de decoerência, acabam tornando a utilização de protocolos de preparação via *q-ram* também desvantajosos.

Os dois métodos seguintes (a forma tradicional via rotações uniformemente controladas e o método via algoritmo de Grover) serão tratados de forma detalhada.

4.2.1 Preparação via rotações uniformemente controladas

O método desenvolvido por Mottönen *et. al.* consiste em utilizar rotações uniformemente controladas para converter um estado inicial $|i\rangle$ em outro estado $|\psi\rangle$. Embora o trabalho desenvolvido [48] abranja quaisquer estados iniciais e finais, é suficiente para a classe de algoritmos analisada que o estado do sistema seja inicializado em $|0\rangle$, pois tal inicialização pode ser realizada sem grandes dificuldades.

As rotações uniformemente controladas foram abordadas detalhadamente na seção 3.2.3. O protocolo de preparação de estados utiliza rotações específicas nos eixos $\hat{\mathbf{z}}$ e $\hat{\mathbf{y}}$, descritas por:

$$\hat{\mathbf{R}}_z(\alpha) = \hat{\mathbf{I}} cos\left(\frac{\alpha}{2}\right) + i\hat{\sigma}_z sen\left(\frac{\alpha}{2}\right) = e^{-i\frac{\alpha}{2}} \begin{bmatrix} e^{i\alpha} & 0 \\ 0 & 1 \end{bmatrix}, \tag{4.28}$$

$$\hat{\mathbf{R}}_y(\alpha) = \hat{\mathbf{I}} cos\left(\frac{\alpha}{2}\right) + i\hat{\sigma}_y sen\left(\frac{\alpha}{2}\right) = \begin{bmatrix} cos\left(\frac{\alpha}{2}\right) & sen\left(\frac{\alpha}{2}\right) \\ -sen\left(\frac{\alpha}{2}\right) & cos\left(\frac{\alpha}{2}\right) \end{bmatrix}. \tag{4.29}$$

A transformação $\hat{\mathbf{R}}_z(\alpha)$ possui o efeito de multiplicar as bases computacionais por fatores de fase, enquanto $\hat{\mathbf{R}}_y(\alpha)$ permite criar qualquer superposição dos estados $|0\rangle$ e $|1\rangle$ (com coeficientes reais) através da variação do ângulo α.

Um estado quântico genérico pode ser descrito como:

$$|\psi\rangle = \sum_{i=0}^{N-1} a_i e^{\theta_i} |i\rangle . \tag{4.30}$$

Os termos a_i são expressos por coeficientes reais enquanto θ_i representam os argumentos de fase, de tal forma que as rotações $\hat{\mathbf{R}}_y(\alpha)$ e $\hat{\mathbf{R}}_z(\alpha)$ podem ser utilizadas para criar um estado arbitrário através do ajuste adequado de a_i e θ_i.

A Fig. 24 mostra o circuito utilizado para implementar o protocolo de preparação de estados. A ação das rotações uniformemente controladas $\hat{\mathbf{R}}_y$ pode ser compreendida através da árvore ilustrativa na Fig. 25 (para facilitar a compreensão, o processo foi representado com três qubits ao invés de quatro como na Fig. 24).

Para o caso mais geral, considerando como ponto de partida o estado $|0\rangle^{\oplus n}$, a primeira rotação uniformemente controlada $\hat{\mathbf{R}}_y^1$ age no primeiro *q-bit*, criando uma superposição dos estados $|1\rangle|0\rangle^{\otimes n-1}$ e $|0\rangle|0\rangle^{\otimes n-1}$. Na sequência, $\hat{\mathbf{R}}_y^2$ age nos estados $|1\rangle|0\rangle^{\otimes n-1}$ e $|0\rangle|0\rangle^{\otimes n-1}$, gerando $|1\rangle|1\rangle|0\rangle^{\otimes n-2}$ e $|1\rangle|0\rangle|0\rangle^{\otimes n-2}$ (a partir de $|1\rangle|0\rangle^{\otimes n-1}$), e também $|0\rangle|1\rangle|0\rangle^{\otimes n-2}$ e $|0\rangle|0\rangle|0\rangle^{\otimes n-2}$ (a partir de $|0\rangle|0\rangle^{\otimes n-1}$). As rotações posteriores seguem esta mesma lógica: Conforme o nível da árvore avança, elas atuam na posição correspondente ao respectivo nível no algarismo binário de cada uma das bases já geradas, cujos elementos são ramificados em pares e geram a superposição correspondente ao próximo nível (até a obtenção da superposição final de todos os N elementos de uma base computacional de n *q-bits*). Consequentemente, os coeficientes reais a_i das superposições

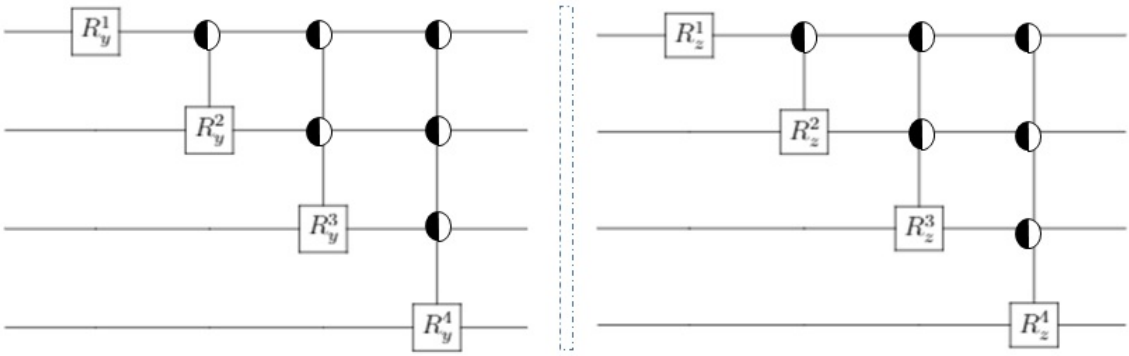

Figura 24 – . Preparação de estados por rotações uniformemente controladas (um conjunto de rotações que engloba todas possíveis combinações de estados de controle). As rotações $\hat{\mathbf{R}}_y$ ajustam as partes reais dos coeficientes, enquanto as $\hat{\mathbf{R}}_z$ ajustam as fases.

podem ser controlados arbitrariamente através da escolha adequada do conjunto de ângulos $\{\alpha_1^1, .., \alpha_N^N\}$.

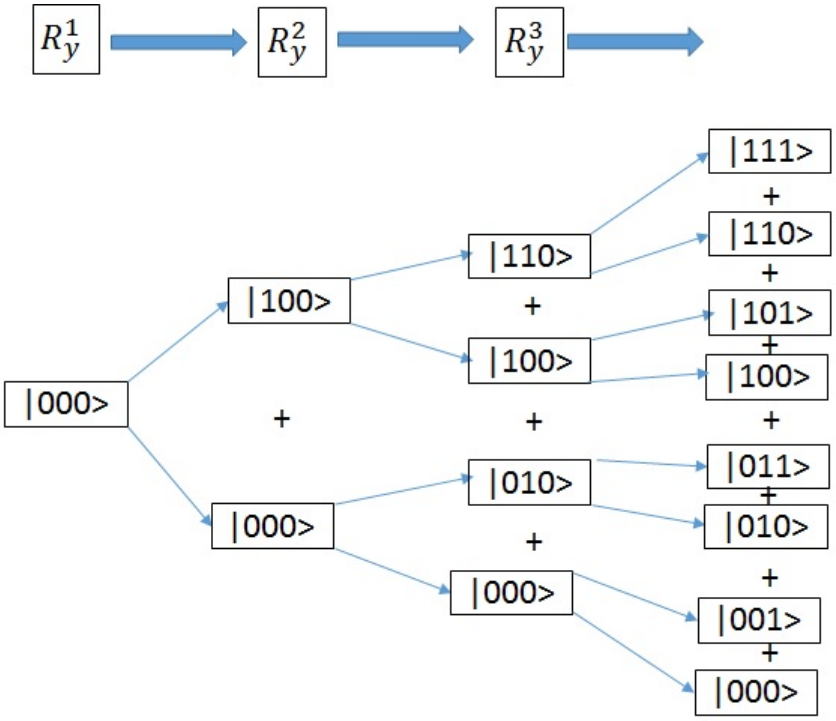

Figura 25 – Árvore ilustrativa da preparação de estados. Cada conjunto de rotações controladas $\hat{\mathbf{R}}_y$ possui como alvo uma posição específica dos algarismo de cada *q-bits*. A sequência avança da esquerda para a direita, assim como os respectivos índices das rotações.

Para obter os coeficientes reais a_i do estado desejado, os ângulos devem seguir uma relação recursiva específica. Ao converter os algarismos da representação binária para a base decimal, é possível observar que se o estado $|k\rangle$ e o índice i representam,

respectivamente, um nó e um nível da árvore, então o primeiro será transformado de acordo com a relação ilustrada na Fig. 26.

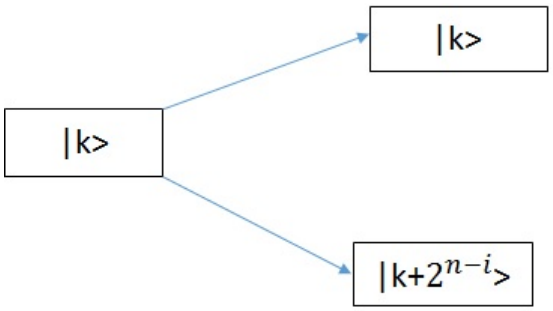

Figura 26 – Diagrama geral de transformação de uma base através de rotações $\hat{\mathbf{R}}_y$,considerando que os respectivos coeficientes estão implícitos na operação.

Caso seja denotado por $|i,k\rangle$ o estado com representação binária correspondente ao número k no i-ésimo nível da árvore, então a transformação dos coeficientes será expressa por:

$$a_{i,k}\,|i,k\rangle \Rightarrow a_{i+1,k}\,|i+1,k\rangle + a_{(i+1),(k+2^{n-i})}\left|i+1,k+2^{n-i}\right\rangle;|i+1,k\rangle = |i,k\rangle\,, \tag{4.31}$$

$$(a_{i+1,k})^2 + (a_{(i+1),(k+2^{n-i})})^2 = (a_{i,k})^2, \tag{4.32}$$

$$a_{i,k}\,|i,k\rangle \Rightarrow a_{i,k}\left(cos\left(\frac{\alpha_{i,k}}{2}\right)|i,k\rangle + sen\left(\frac{\alpha_{i,k}}{2}\right)\left|i+1,k+2^{n-i}\right\rangle\right). \tag{4.33}$$

Consequentemente, a relação de recursão que define os ângulos das rotações uniformemente controladas será dada por:

$$\alpha_{i,k} = 2arcos\left(\frac{a_{i+1,k}}{\sqrt{(a_{i+1,k})^2 + (a_{(i+1),(k+2^{n-i})})^2}}\right). \tag{4.34}$$

Desse modo, o conjunto de todos os ângulos pode ser determinado através da aplicação de (4.34), a partir do nível mais avançado da árvore (uma vez que $a_{n,k}$ são os coeficientes reais do estado preparado), e prosseguindo até a raiz.

O segundo conjunto de rotações uniformemente controladas, com eixo $\hat{\mathbf{z}}$ ao invés de $\hat{\mathbf{y}}$, é melhor compreendido através de perspectivas que não utilizam uma árvore de estados, uma vez o início da ação não ocorre a partir de algum estado específico, e sim de uma superposição que envolve todos os estados. Além disso, ao contrário dos cenários apresentados nas Figs. 25 e 26, as rotações no eixo $\hat{\mathbf{z}}$ não geram uma ramificação de novos estados.

Caso seja denotado como $\alpha_{i,j}$ o ângulo adquirido pela base $|j\rangle$ $(0 < j < 2^{n-1})$ ao passar pela i-ésima rotação uniformemente controlada, cada estado da base se transformará de acordo com a equação (4.35) após passar por todo o conjunto de rotações:

$$|j\rangle \Rightarrow \left(\prod_{i=1}^{n} e^{i\frac{\alpha_{i,j}}{2}}\right)|j\rangle\,. \tag{4.35}$$

Visto que a equação (4.30) estabelece o estado alvo que deve ser alcançado, os argumentos $\alpha_{i,j}$ devem estar de acordo com a seguinte condição:

$$\sum_{i=1}^{n} \alpha_{i,j} = 2\theta_j. \tag{4.36}$$

Nem todos os termos $\alpha_{i,j}$ são distintos entre si. A análise da Fig. 17 permite concluir que dada a representação binária de j, denotada por $B(j) = [b(j)_1, b(j)_2, ..., b(j)_n]$, a ação das rotações será tal que os índices j que compartilham a mesma sequência de algarismos binários até a i-ésima posição da representação $B(j)$ apresentam $\alpha_{i,j}$ com o mesmo valor absoluto, cujo sinal é determinado pelo algarismo binário na posição $i+1$. O sinal será negativo para os casos nos quais este algarismo é igual a 1 e positivo para os demais (isto se deve à alternância de sinais do argumento observada na equação (4.29)).

O resultado das partilhas de valores e alternância de sinais faz com que a equação (4.36) corresponda a um sistema de equações lineares que pode ser facilmente resolvido via escalonamentos sucessivos (a partir da eliminação das variáveis que se repetem menos vezes, ou seja, aquelas que correspondem às rotações com maior número de controles). Consequentemente, os argumentos das rotações (decorrentes das eliminações no escalonamento) podem ser determinados recursivamente através das seguintes relações:

$$\alpha_{i,k} = \theta_k^{i+1} - \theta_{k+2^{n-i}}^{i+1}, \tag{4.37}$$

$$\theta_k^i = \theta_k^{i+1} - \frac{\alpha_k^i}{2}. \tag{4.38}$$

O conceito da árvore de estados torna-se útil para compreender o procedimento das eliminações sucessivas ao longo do escalonamento, uma vez que os estados dos nós e dos níveis da árvore estão respectivamente correlacionados aos índices k e i na recursão do processo de eliminação de variáveis. Nas equações (4.37) e (4.38), i representa o nível da árvore e k um conjunto de índices que apresenta uma correspondência direta com os valores exibidos nos nós (Fig. 25). Desse modo, os valores sequenciais não diferem por 1 em todos os níveis de recursão, a exemplo do segundo nível imediatamente antes da raiz, no qual k assume os valores 0 e 4 (equivalentes respectivamente a 000 e 100 em notação binária).

Por sua vez, como o conjunto de todos os 2^n possíveis θ_k^n é conhecido e corresponde aos argumentos θ das equações (4.30) e (4.36), é possível determinar todos os argumentos

de todas as rotações $F_{i+1}^{i}(\hat{\mathbf{R}}_z)$ envolvidas na preparação de estados através das equações (4.37) e (4.38).

Um detalhe adicional é a existência da necessidade de realizar uma operação de deslocamento de fase no primeiro *q-bit*, já que as equações (4.37) e (4.38) geram uma aparente contradição quando a recursão (ou iteração) atinge o valor do ângulo final, correspondente à primeira $\hat{\mathbf{R}}_z$ no primeiro *q-bit*. O ângulo deveria assumir dois valores distintos para satisfazer as devidas condições. Todavia, o problema pode ser resolvido se o ângulo for especificado com um destes valores e a porta $\hat{\mathbf{R}}_z^1$ for antecedida por um deslocamento de fase que corresponde à diferença entre ambos, uma vez que esta adição não modifica a complexidade total do processo.

O circuito ilustrado na Fig. 24 apresenta um conjunto de várias $F_{i+1}^{i}(\hat{\mathbf{R}}_z)$ (nas quais i varia de 0 a $n-1$), cuja complexidade (cap. 3) é de $O(2^i)$. Logo, a complexidade total (em termos de portas C-NOT e portas de um *q-bit*) para a implementação do processo de preparação de estado é resultado da soma de uma progressão geométrica que contém n termos entre 1 e 2^{n-1}. Esta soma resulta em $2^n - 1$, ou seja, a complexidade é de $O(2^n) = O(N)$, e nenhum *q-bit* auxiliar é requerido no processo.

4.2.2 Preparação auxiliada pelo algoritmo de Grover

O método de preparação de estados auxiliado pelo algoritmo de Grover foi apresentado pelo autor homônimo em [42]. O método pode ser dividido em duas partes, e a Fig. 27 apresenta o circuito correspondente à primeira.

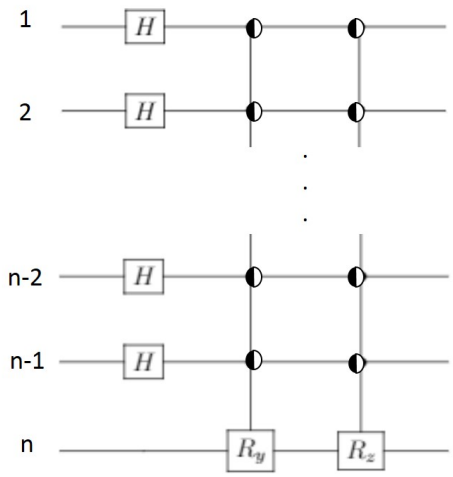

Figura 27 – Primeiro estágio da preparação auxiliada pelo algoritmo de Grover. As portas de Haddamard produzem uma superposição uniforme das bases do espaço de trabalho, enquanto as rotações seguintes ajustam os coeficientes reais e fases do estado aproveitando o grau de liberdade gerado pela presença dos *q-bits* auxiliares.

O método consiste em preparar um estado correspondente a n-1 *q-bits*, e portanto

requer a presença de um *q-bit* auxiliar (independente da dimensão do estado alvo). A primeira parte do algoritmo pode ser representada pelo seguinte operador:

$$\hat{\mathbf{U}} = \hat{\mathbf{U}}_3\hat{\mathbf{U}}_2\hat{\mathbf{U}}_1. \tag{4.39}$$

O termo $\hat{\mathbf{U}}_1 = \hat{\mathbf{H}}^{\otimes(n-1)}$ representa a transformada de Walsh-Haddamard, que produz uma superposição equiprovável dos $N/2 = 2^{n-1}$ estados da base computacional dos *q-bits* de trabalho quando atua no estado $|0\rangle^{\otimes n-1}$. Esta operação é descrita por:

$$\hat{\mathbf{U}}_1 |0\rangle^{\otimes n-1} = \frac{\sqrt{2}}{\sqrt{N}} \sum_{i=1}^{\frac{N}{2}} |i\rangle |0\rangle . \tag{4.40}$$

Na sequência, o operador $\hat{\mathbf{U}}_2$ aplica uma rotação uniformemente controlada pelos *q-bits* de trabalho ($F_n^{n-1}(\hat{\mathbf{R}}_y)$) atuando no *q-bit* auxiliar. Se a forma geral do estado-alvo for expressa pela equação (4.30), as rotações $\hat{\mathbf{R}}_y$ devem ser tais que:

$$\hat{\mathbf{R}}_y(\alpha_i) |0\rangle = a_i |0\rangle - \sqrt{1 - a_i^2} |1\rangle . \tag{4.41}$$

O intuito da transformação descrita pela equação (4.41) é que $F_n^{n-1}(\hat{\mathbf{R}}_y)$ tenha, no estado do sistema, o seguinte efeito:

$$\hat{\mathbf{U}}_2\hat{\mathbf{U}}_1 |0\rangle^{\otimes n} = \frac{\sqrt{2}}{\sqrt{N}} \sum_{i=1}^{\frac{N}{2}} a_i |i\rangle |0\rangle - \frac{\sqrt{2}}{\sqrt{N}} \sum_{i=1}^{\frac{N}{2}} \sqrt{1 - a_i^2} |i\rangle |1\rangle . \tag{4.42}$$

A condição para que isto ocorra (a qual pode ser compreendida de forma clara com auxílio da equação (4.29)) é descrita por:

$$cos(\alpha_i) = a_i. \tag{4.43}$$

Por fim, o operador $\hat{\mathbf{U}}_3$ consiste em uma sequência de rotações $F_n^{n-1}(\hat{\mathbf{R}}_z)$, tais que:

$$\hat{\mathbf{R}}_z(\theta_i) |0\rangle = e^{i\theta_i} |0\rangle . \tag{4.44}$$

O estado final, após a aplicação sucessiva dos devidos operadores, é descrito por:

$$\hat{\mathbf{U}}_3\hat{\mathbf{U}}_2\hat{\mathbf{U}}_1 |0\rangle^{\otimes n} = \frac{\sqrt{2}}{\sqrt{N}} \sum_{i=1}^{\frac{N}{2}} a_i e^{i\theta_i} |i\rangle |0\rangle - \frac{\sqrt{2}}{\sqrt{N}} \sum_{i=1}^{\frac{N}{2}} \sqrt{1 - a_i^2} e^{-i\theta_i} |i\rangle |1\rangle . \tag{4.45}$$

A complexidade do circuito (em termos de portas C-NOT e portas de um *q-bit*) ilustrado na Fig. 27 é de $O(2^n) = O(N)$, uma vez que cada rotação uniformemente controlada possui uma complexidade individual de $O(2^n)$.

Observe que o estado exibido na equação (4.45) consiste em uma superposição de estados, cuja projeção no subespaço do q-bit auxiliar quando este é descrito por $|0\rangle$ é

idêntica ao o estado o qual se deseja preparar (equação (4.30)). Logo, há duas escolhas que podem ser adotadas a partir do momento que o estado do sistema é sintetizado de acordo com o descrito pela equação (4.45).

A primeira opção consiste em aplicar a parte unitária (excetuando-se eventuais medições) do algoritmo que necessita da preparação de estados com uma modificação para que todo o processo seja controlado pelo *q-bit* auxiliar, para que este seja medido na sequência. Esta escolha faz com que o algoritmo assuma um caráter probabilístico, e reduz a probabilidade total de sucesso do protocolo caso o mesmo não seja inicialmente determinístico.

A segunda opção é utilizar o processo de amplificação de amplitude, substituindo na equação (4.2) o operador $\hat{\mathbf{U}}$ descrito pela equação (4.39). Caso o oráculo seja preparado para marcar (ou seja, mudar o sinal) de todo o conjunto de estados nos quais o *q-bit* auxiliar é descrito por $|1\rangle$ (o que corresponde a uma ordem de $O(N/2)$ estados), será necessário repetir uma ordem de $O(\sqrt{N})$ vezes o processo de amplificação para alcançar a probabilidade de 100% de obtenção do *q-bit* auxiliar neste estado em uma medição. A necessidade desta repetição implica em multiplicar a complexidade inicial do processo de preparação pelo fator $\sqrt{N}$, tal que a complexidade total é dada por $O(\sqrt{N}N)$ (conforme já discutido na seção 4.1.1).

As duas opções disponíveis não são atrativas, visto que ambas implicam em uma complexidade temporal de ao menos $O(N)$, idêntica à do processo que utiliza rotações uniformemente controladas. Além disso, apresentam várias desvantagens com relação a este, que prepara o estado de forma exata (e pronto para ser utilizado sem intervenções adicionais). Esse procedimento é discutido inclusive pelo próprio Grover em [42] com base em um enfoque matemático, o qual cita a necessidade (aparentemente atrativa) de repetir o procedimento apenas $O(\sqrt{N})$ vezes, sem enfatizar o emprego das rotações uniformemente controladas (embora estas sejam as que mais contribuem para aumentar a complexidade envolvida no processo).

4.3 Algoritmo de Tao Xin *et. al.*

Um algoritmo elaborado para a resolução de um sistema de equações diferenciais ordinárias (EDOs) lineares acopladas foi apresentado em [23]. Os autores discutem as dificuldades de implementar os esquemas apresentados em [34, 36] com a tecnologia de estado da arte disponível atualmente, destacando que sua proposta pode ser implementada em um sistema de ressonância magnética. Além disso, as generalidades sobre natureza probabilística e a necessidade de preparação de estados também estão presente neste algoritmo.

4.3.1 Detalhes sobre o método

O problema consiste em resolver um sistema descrito por:

$$\frac{d\vec{x}}{dt} = \mathbf{M}\vec{x} + \vec{b}. \tag{4.46}$$

Os termos $\mathbf{M}$ e $\vec{b}$ referem-se, respectivamente, a uma matriz e um vetor com coeficientes constantes (ambos N-dimensionais). O valor da variável $\vec{x}$ em um instante inicial ($\vec{x}(0)$) é previamente conhecido.

A integração direta de (4.46) fornece o seguinte resultado (o qual pode ser aproximado por uma série de Taylor):

$$\vec{x}(t) = e^{\mathbf{M}t}\vec{x}(0) + (e^{\mathbf{M}t} - \mathbf{I})\mathbf{M}^{-1}\vec{b}; \tag{4.47}$$

$$\vec{x}(t) \approx \sum_{m=0}^{k} \frac{\mathbf{M}^m t^m}{m!}\vec{x}(0) + \sum_{n=1}^{k} \frac{\mathbf{M}^{n-1} t^n}{n!}\vec{b}. \tag{4.48}$$

De acordo com a equação (4.48), conclui-se que um possível algoritmo clássico para resolução de tal problema teria complexidade $O(kN^3)$, uma vez que esta é determinada pela potência mais alta de $\mathbf{M}$ na série de Taylor, e que a complexidade do produto matricial é da ordem de $O(N^3)$.

Para fins de processamento em sistemas de quânticos, $\vec{x}(0)$ e $\vec{b}$ podem ser associados respectivamente aos estados $|x(0)\rangle$ e $|b\rangle$, enquanto $\mathbf{M}$ pode ser associada a um operador $\hat{\mathbf{A}}$.

4.3.2 Matriz unitária

Se a matriz $\mathbf{M}$ for unitária, ela estará diretamente correlacionada com um operador unitário $\hat{\mathbf{A}}$. Utilizando os termos $\hat{\mathbf{U}}_m = \hat{\mathbf{A}}^m$, $C_m = ||x(0)||(t)^m/m!$ e $D_n = ||b||(t)^{n-1}t/n!$, e sendo $\aleph^2 = C^2 + D^2$, com $C = \sqrt{\sum C_m}$ e $D = \sqrt{\sum D_n}$, é possível mapear o vetor descrito pela equação (4.48) no estado quântico descrito por:

$$|x(t)\rangle \approx \frac{1}{\aleph^2}\left(\sum_{m=0}^{K} C_m \hat{\mathbf{U}}_m |x(0)\rangle + \sum_{n=1}^{K} D_n \hat{\mathbf{U}}_{n-1} |b\rangle\right). \tag{4.49}$$

O desenvolvimento deste estado emprega a utilização de $T = log_2(k+1)$ *qbits* auxiliares, $n = log_2(N)$ *q-bits* de trabalho e além destes, um *q-bit* auxiliar adicional, a partir de um estado inicial no qual todos estes *q-bits* possuem valor nulo. As Figs. 28, 29 e 30 representam, respectivamente, o primeiro, segundo e terceiro estágios da execução do algoritmo.

O primeiro estágio inicia com a aplicação da porta $\hat{\mathbf{V}}$ cujos elementos matriciais são descritos por:

$$\hat{\mathbf{V}} = \frac{1}{\aleph}\begin{bmatrix} C & D \\ D & -C \end{bmatrix}. \tag{4.50}$$

A aplicação desse operador possui finalidade de criar um pré-estado no qual seja possível aplicar operadores distintos em diferentes subespaços associados aos *q-bits* auxiliares através de operações controladas. A estrutura do algoritmo é essencialmente baseada nessa técnica, e tal pré-estado é, neste primeiro caso, descrito por:

$$|\psi_0\rangle = \frac{1}{\aleph}\left(C\,|0\rangle\,|0\rangle^{\otimes t}\,|0\rangle^{\otimes n} + D\,|1\rangle\,|0\rangle^{\otimes t}\,|0\rangle^{\otimes n}\right). \tag{4.51}$$

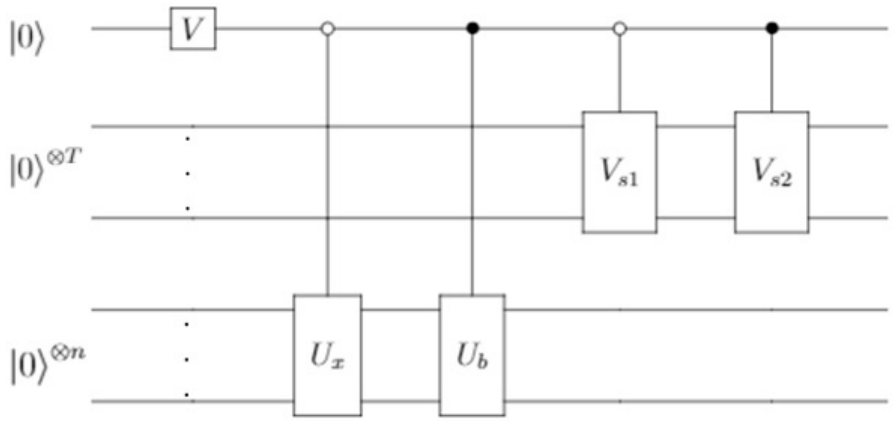

Figura 28 – Estágio inicial do algoritmo utilizado para resolução do sistema de equações diferenciais. As portas $\hat{\mathbf{U}}_x$ e $\hat{\mathbf{U}}_b$ codificam $\vec{x}(0)$ e $\vec{b}$ nos *q-bits* de trabalho, enquanto $\hat{\mathbf{V}}_{s1}$ e $\hat{\mathbf{V}}_{s2}$ preparam o espaço auxiliar para mapear corretamente as operações controladas dispostas na sequência, além de ajustar os respectivos coeficientes.

Os operadores $\hat{\mathbf{U}}_{\mathbf{x}}$ e $\hat{\mathbf{U}}_{\mathbf{b}}$ representam as operações de preparação de estado que possuem a função de criar respectivamente os estados $|x(0)\rangle$ e $|b\rangle$ para processamento posterior. Os operadores $\hat{\mathbf{V}}_{s1}$ e $\hat{\mathbf{V}}_{s2}$ possuem uma função semelhante a $\hat{\mathbf{V}}$ mas criam estados que servem como alvo de operações controladas no espaço auxiliar $(k+1)$-dimensional. Como a ação é efetuada somente no estado $|0\rangle^{\otimes T}$, apenas a primeira coluna da matriz que representa ambos operadores é relevante, sendo descrita por:

$$\hat{\mathbf{V}}_s1[:,0] = \frac{1}{C}[\sqrt{C_0}, \sqrt{C_1}, ..., \sqrt{C_k}]; \hat{\mathbf{V}}_s2[:,0] = \frac{1}{D}[\sqrt{D_0}, \sqrt{D_1}, ..., \sqrt{D_k}, 0]. \tag{4.52}$$

Ao término do primeiro estágio, o estado do sistema é descrito por:

$$|\psi_1\rangle = \frac{1}{\aleph}\left(|0\rangle \sum_{m=0}^{k} \sqrt{C_m}\,|m\rangle\,|x(0)\rangle + |1\rangle \sum_{n=1}^{k} \sqrt{D_n}\,|n-1\rangle\,|b\rangle\right). \tag{4.53}$$

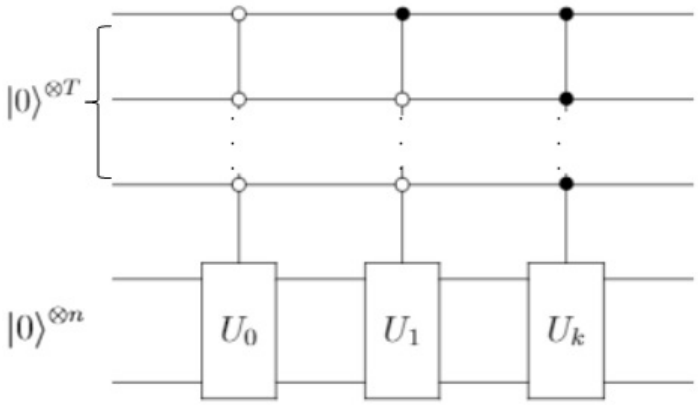

Figura 29 – Segundo estágio do algoritmo utilizado para a resolução do sistema de equações diferenciais. Os operadores $\hat{\mathbf{U}}_j$ aplicam as potências de ordem k da matriz característica, utilizando os estados superpostos no subespaço dos *q-bits* auxiliares como controle.

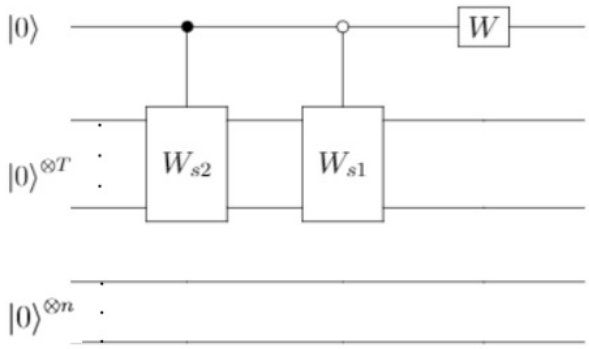

Figura 30 – Estágio final do algoritmo. Os operadores $\hat{\mathbf{W}}_{s1}$, $\hat{\mathbf{W}}_{s2}$ e $\hat{\mathbf{W}}$ equivalem respectivamente à ação inversa de $\hat{\mathbf{V}}_{s1}$, $\hat{\mathbf{V}}_{s2}$ e $\hat{\mathbf{V}}$, e terminam de codificar a solução no subespaço no qual todos os *q-bits* auxiliares se encontram no estado $|0\rangle$.

Os k estados do subespaço auxiliar servem como alvo para as operações controladas do estágio seguinte, que consiste na aplicação das potências do operador $\hat{\mathbf{U}}$ que aparecem na equação (4.49). A aplicação do segundo estágio gera emaranhamento entre os *q-bits* auxiliares e os de trabalho, resultando em:

$$|\psi_2\rangle = \frac{1}{\aleph}\left(|0\rangle \sum_{m=0}^{k} \sqrt{C_m}\,|m\rangle\,\hat{\mathbf{U}}_m\,|x(0)\rangle + |1\rangle \sum_{n=1}^{k} \sqrt{D_n}\,|n-1\rangle\,\hat{\mathbf{U}}_{n-1}\,|b\rangle\right). \tag{4.54}$$

O terceiro e último estágio consiste na aplicação dos operadores inversos de $\hat{\mathbf{V}}$, $\hat{\mathbf{V}}_{s1}$ e $\hat{\mathbf{V}}_{s2}$, denotados respectivamente por $\hat{\mathbf{W}} = \hat{\mathbf{V}}^{\dagger}$, $\hat{\mathbf{W}}_{s1} = \hat{\mathbf{V}}_{s1}^{\dagger}$ e $\hat{\mathbf{W}}_{s2} = \hat{\mathbf{V}}_{s2}^{\dagger}$. Neste caso, tanto $\hat{\mathbf{V}}_{s1}$ e $\hat{\mathbf{V}}_{s2}$ quanto $\hat{\mathbf{W}}_{s1}$ e $\hat{\mathbf{W}}_{s2}$ são operadores cuja exigência de valores específicos se aplica somente a uma única fila (a primeira coluna no caso da primeira dupla, e a primeira linha no caso da segunda). Os demais elementos podem ser escolhidos de tal forma que esses operadores sejam unitários. As operações de preparação de estado (com a devida especificação dos respectivos parâmetros) podem cumprir o papel atribuído a estes

operadores, uma vez que um operador de preparação de estados que modifica o estado $|0\rangle$, transformando-o em algum $|\psi\rangle$ específico, apresenta os coeficientes da primeira coluna de sua matriz idênticos às coordenadas de $|\psi\rangle$.

Após a aplicação de todas as operações, o sistema residirá em um estado cuja projeção no subespaço no qual todos os *q-bits* auxiliares são nulos é descrita por:

$$\langle 0|\langle 0|^{\otimes T}|\psi_3\rangle = \frac{1}{\aleph^2}\left(\sum_{m=0}^{k} C_m \hat{\mathbf{U}}_m |x(0)\rangle + \sum_{n=1}^{k} D_n \hat{\mathbf{U}}_{n-1}|b\rangle\right). \tag{4.55}$$

Caso seja realizada uma medição que resulte em um estado no qual todos os *q-bits* auxiliares possuem valor nulo, será encontrada a solução correta, codificada no subespaço associado aos *q-bits* de trabalho.

4.3.3 Análise de complexidade(caso unitário)

Primeiramente, há os operadores controlados $\hat{\mathbf{U}}_x$ e $\hat{\mathbf{U}}_b$, responsáveis por preparar os estados $|x_0\rangle$ e $|b\rangle$, respectivamente. Este operadores são expressos por matrizes NXN, controladas por um *q-bit* auxiliar. Como resultado das análises anteriormente desenvolvidas, um algoritmo de preparação de estado pode ser implementado por um conjunto de rotações uniformemente controladas com uma complexidade de $O(N)$. A adição realizada de um controle extra, com um *q-bit* auxiliar disponível apenas para tal finalidade, resultaria na multiplicação da complexidade por uma constante, sem alterar a dependência da mesma com relação a N.

Como cada uma das rotações uniformemente controladas pode ser implementada através de um conjunto de portas de um *q-bit* e C-NOT, a versão controlada por um único *q-bit* da preparação de estados pode ser construída adicionado um ponto de controle a cada uma dessas portas, e com isso, o tempo individual de execução de cada uma seria acrescido de um valor fixo. Como a estrutura do circuito permaneceria inalterada, sua respectiva complexidade seria multiplicada por uma constante. Sendo assim, a complexidade das portas $\hat{\mathbf{U}}_{\mathbf{x}}$ e $\hat{\mathbf{U}}_{\mathbf{b}}$ é também de $O(N)$. De modo semelhante, $\hat{\mathbf{V}}_{s1}$, $\hat{\mathbf{V}}_{s2}$, $\hat{\mathbf{W}}_{s1}$ e $\hat{\mathbf{W}}_{s2}$ (que atuam em estados $(k+1)$-dimensionais) podem ser implementadas com complexidade $O(k)$.

A Fig. 29 exibe o conjunto das k operações controladas $\hat{\mathbf{U}}_m$, correspondentes às potências de $\hat{\mathbf{A}}$ (dado um operador inicial $\hat{\mathbf{U}} = \hat{\mathbf{A}}$), cuja complexidade inicial seria de $O(k(N^2k^2))$ caso fosse implementada diretamente da forma ilustrada ,como visto no cap. 3 (no qual foi apresentada uma forma de implementação com complexidade de $O(4^n)$ para uma matriz geral correspondente a um operador que atua em n *q-bits*). É possível, porém, implementar esse conjunto de portas de uma forma alternativa, como ilustrado na Fig. 31.

A sequência ilustrada na Fig. 31 é equivalente à da Fig. 29, já que para uma dada representação binária *bin(m)* para um dos estados $|m\rangle$ do subespaço auxiliar, há a

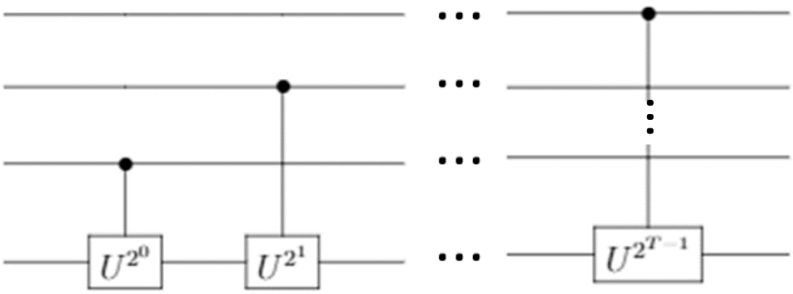

Figura 31 – Sequência equivalente de potências de $\hat{\mathbf{A}}$. O estado dos *q-bit* alvo sofre aplicações sucessivas de $\hat{\mathbf{U}}$ com potências diferentes, mas as posições dos nós de controle fazem com que o valor final do expoente seja expresso por uma soma correspondente ao algarismo binário que representa o estado $|i\rangle$ da base computacional.

garantia de que serão aplicadas as potências correspondentes a esta representação, ou seja: O circuito mapeará os algarismo binários de m a uma potência correta de $\hat{\mathbf{A}}$, uma vez que cada potência somente será aplicada se o algarismo correspondente de bin(m) for igual a um. O resultado final será dado por:

$$|m\rangle |x_0\rangle \rightarrow |m\rangle \hat{\mathbf{A}}^{\sum_{j=0}^{j=T-1} bin(m)_j 2^j} |x_0\rangle = |m\rangle \hat{\mathbf{U}}_m |x_0\rangle . \tag{4.56}$$

Para esta representação, haverá um total de T portas $\hat{\mathbf{U}}$ controladas por um único *q-bit* auxiliar, sendo cada uma delas uma operação envolvendo n *q-bits*, e portanto com complexidade de implementação de $O(N^2)$, conforme os esquemas desenvolvidos no cap. 3 (resulta da mesma lógica apresentada anteriormente para outras portas monocontroladas que a complexidade de cada uma das $\hat{\mathbf{C_1U}}$ se mantém $O(N^2)$), gerando portanto uma complexidade total de $O(TN^2) = O(N^2 log_2(k))$ para a sequência completa.

O valor final da complexidade do algoritmo resulta em $O(2N + 4k + N^2 log_2 k)$. Em [23], os autores afirmam obter um ganho exponencial com relação ao caso clássico, com complexidade de $O(log_2 N)$. Todavia, a implementação das potências $\hat{\mathbf{U}}_m$ foi tratada sob condições específicas, que serão descritas no apêndice A. Ainda assim, há um ganho em relação ao caso clássico (cuja complexidade é de $O(N^3)$).

4.3.4 Matriz não unitária

A equação (4.48) permanece válida se a matriz não for unitária. Nesse caso, porém, o estado não pode mais ser diretamente correlacionado com um operador unitário $\hat{\mathbf{A}}$ como na equação (4.49). Ao invés disso, o operador $\hat{\mathbf{A}}$ (representado por $\mathbf{M}$) pode ser escrito como uma combinação linear de operadores unitários $\hat{\mathbf{A}}_i$:

$$\hat{\mathbf{A}} = \sum_{i=1}^{L} \alpha_i \hat{\mathbf{A}}_i. \tag{4.57}$$

A substituição da equação (4.57) na equação (4.48) resulta em:

$$\vec{x}(t) \approx \sum_{m=0}^{k} \frac{||\vec{x}_0||(t)^m(\sum_{i=1}^{L}\alpha_i\hat{\mathbf{A}}_i)^m}{m!}|x_0\rangle + \sum_{n=1}^{k} \frac{||\vec{b}||(t)^n(\sum_{i=1}^{L}\alpha_i\hat{\mathbf{A}}_i)^{n-1}}{n!}|b\rangle\,. \tag{4.58}$$

Os registros necessários para resolver essa versão do problema são diferentes dos anteriores, embora uma combinação linear de operadores possa ser executada de maneira similar à soma das potências de $\hat{\mathbf{U}}$ para o caso unitário, o procedimento demanda o auxílio de subespaço L-dimensional. Como serão efetuadas potências dessa combinação, com máximo expoente k, serão necessários k subespaços L-dimensionais. Tal condição pode ser atendida adicionando um total de k sistemas de dimensão L (L *q-dits*) ao registro (tais *q-dits* podem ser obtidos através de composições de *q-bits*). Nas condições descritas, o estado inicial do sistema não será descrito por $|0\rangle\,|0\rangle^T\,|0\rangle^n$, e sim por $|0\rangle\,|0\rangle^k\,|0\rangle_L\,|0\rangle^n$, sendo $|0\rangle_L$ a representação do estado inicial nulo de um *q-dit*. As Figs. 32, 33 e 34 apresentam respectivamente as três divisões do circuito correspondente a este algoritmo.

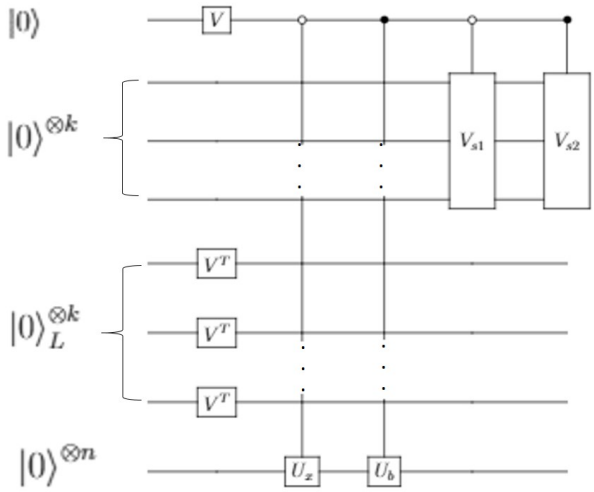

Figura 32 – Etapa inicial do circuito para o caso não-unitário. Os operadores $\hat{\mathbf{V}}^{\mathbf{T}}$ codificam os coeficientes da combinação linear de operadores unitários, enquanto os operadores $\hat{\mathbf{U}}_x$, $\hat{\mathbf{U}}_b$, $\hat{\mathbf{V}}_{s1}$ e $\hat{\mathbf{V}}_{s2}$ possuem funções semelhante às discutidas anteriormente para o caso unitário.

A etapa inicial do método consiste em aplicar os operadores $\hat{\mathbf{V}}$ no primeiro *q-bit* e os $\hat{\mathbf{V}}^T$ em cada um dos *L-q-dits* auxiliares, e também em aplicar os operadores controlados de preparação de estado, para preparar $|x_0\rangle$ e $|b\rangle$, finalizando com $\hat{\mathbf{V}}_{s1}$ e $\hat{\mathbf{V}}_{s2}$.

Os elementos matriciais de $\hat{\mathbf{V}}^T$ correspondem a:

$$\hat{\mathbf{V}}^T[j,0] = \frac{\sqrt{\alpha_j}}{\gamma}; \gamma = ||\hat{\mathbf{V}}_{s1}[..,0]|| = \sqrt{\sum_{j=0}^{L}\alpha_j}. \tag{4.59}$$

Note que $\hat{\mathbf{V}}_{s1}$ e $\hat{\mathbf{V}}_{s2}$ agem agora em k *q-bits* ao invés de $log_2(K+1)$ e, portanto, a estrutura destes deve ser alterada. Sendo $G = \sqrt{\sum_{j=0}^{k} G_j}$ e $H = \sqrt{\sum_{j=1}^{k} H_j}$, estes

operadores passam a ser descritos por:

$$\hat{\mathbf{V}}_{s1}[m,0] = \begin{cases} G_j = \frac{1}{G}\sqrt{\frac{||\vec{x_0}||t^j\gamma^j}{j!}}, & \text{se } m = 2^k - 2^{k-j}, \\ 0, & \text{para os demais casos.} \end{cases} \tag{4.60}$$

$$\hat{\mathbf{V}}_{s2}[m,0] = \begin{cases} H_j = \frac{1}{H}\sqrt{\frac{||\vec{b}||t^j\gamma^{j-1}}{j!}}, & \text{se } m = 2^k - 2^{k-j}, \\ 0, & \text{para os demais casos.} \end{cases} \tag{4.61}$$

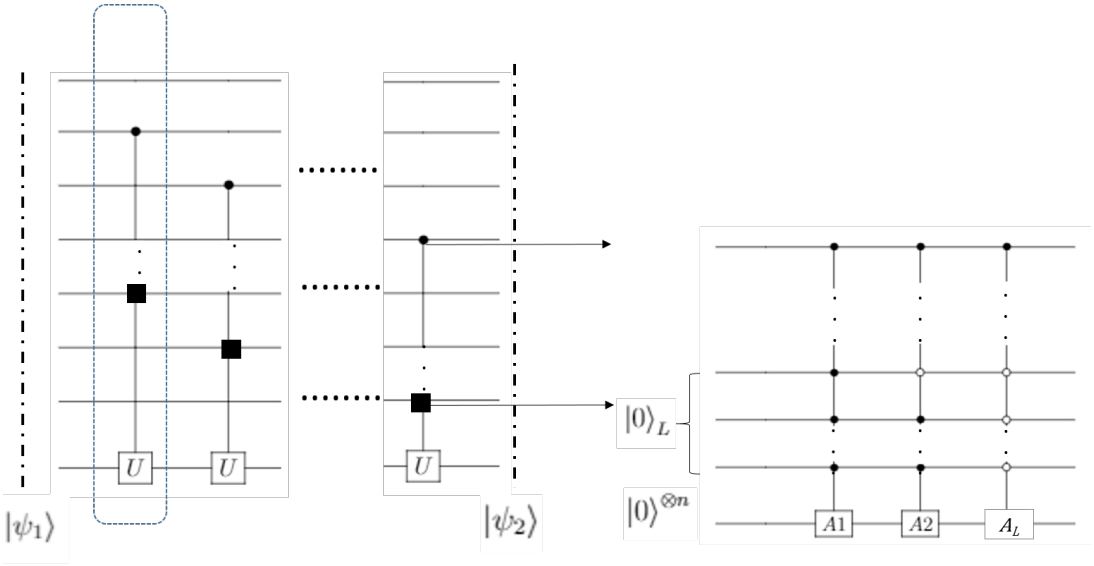

Figura 33 – Segunda parte do circuito para o caso não-unitário. O primeiro registro auxiliar regula quantos *q-dits* recebem a ação da sequência de operadores $\hat{\mathbf{A}}_i$, os quais são individualmente controlados por um dos possíveis estados dos *q-dits*. A correspondência correta entre a potência matricial de cada parcela do estado e a quantidade de *q-dits* transformados é garantida pela etapa anterior.

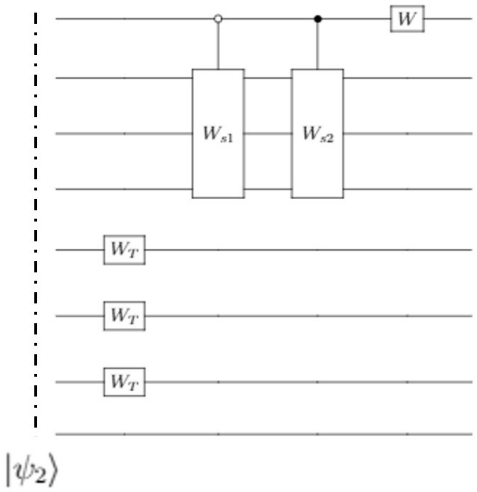

Figura 34 – Estágio final do circuito para o caso não-unitário. A aplicação dos operadores inversos aos da primeira etapa garante a solução codificada no subespaço no qual todos *q-bits* auxiliares possuem valor nulo.

A relação entre os índices m e j pode ser compreendida com base na etapa seguinte do processo. Para que seja associada o expoente matricial correto ao respectivo coeficiente, é necessário que o número de algarismos 1 na representação binária de m seja idêntico ao valor da potência j desejada. Uma vez que todos os k e $k-j$ algarismos dos respectivos números 2^k-1 e $2^{(k-j)}-1$ são todos iguais a 1, a subtração desses dois termos (2^k-2^{k-j}) resulta em um número cuja representação binária contem exatamente j algarismos 1 (a posição destes é irrelevante), satisfazendo a condição para que seja aplicado o expoente matricial correto. Já o termo γ (inserido junto às potências de t) possui a função de cancelar o divisor oriundo da equação (4.59), que ocorre devido ao fato destes operadores sempre produzirem estados normalizados, enquanto os coeficientes das combinações lineares das matrizes utilizadas para expandir $\mathbf{M}$ não atendem necessariamente a uma condição de normalização.

O primeiro operador aplicado ($\hat{\mathbf{V}}$) possui, assim como no caso unitário, a função de inserir coeficientes que possam cancelar os termos G e H que permanecem dividindo as diferentes parcelas marcadas pelos estados $|0\rangle$ e $|1\rangle$ atreladas ao primeiro *q-bit* auxiliar. Como os operadores unitários possuem restrições sobre seus coeficientes, o resultado conterá um fator multiplicativo $\frac{1}{S}$, com $S=\sqrt{G^2+H^2}$, tal que:

$$\hat{\mathbf{V}}=\frac{1}{S}\begin{bmatrix} G & H \\ H & -G \end{bmatrix}. \tag{4.62}$$

Após a aplicação desta sequência de operadores, o estado do sistema será descrito por:

$$|\psi_1\rangle=\frac{1}{S}|0\rangle\sum_{i=0}^{k}\frac{||\vec{x_0}||t^i}{i!\gamma^{k-i}}|i\rangle\,(\sum_{j=0}^{L}\sqrt{\alpha_j}\,|j\rangle)^{\otimes k}|x_0\rangle+\frac{1}{S}|1\rangle\sum_{i=1}^{k}\frac{||\vec{b}||t^i}{i!\gamma^{k-i+1}}|i\rangle\,(\sum_{j=0}^{L}\sqrt{\alpha_j}\,|j\rangle)^{\otimes k}|b\rangle\,. \tag{4.63}$$

É importante ressaltar que os operadores $\hat{\mathbf{V}}_S$ apresentam uma grande quantidade de termos nulos. Os termos γ^{k-j}, que foram originados pelos termos γ^k resultantes de k aplicações de $\hat{\mathbf{V}}^{\mathbf{T}}$ em cada um dos registros auxiliares (os quais foram apenas parcialmente cancelados pelas operações anteriores) serão cancelados posteriormente pela aplicações dos operadores inversos de $\hat{\mathbf{V}}^{\mathbf{T}}$, que também retornarão, em cada parcela, os $k-i$ *q-dits* não utilizados ao estado $|0\rangle_L$.

Após a segunda etapa (Fig. 33), o estado emaranhado do sistema será descrito por:

$$\begin{aligned}|\psi_2\rangle=&\frac{1}{S}|0\rangle\sum_{i=0}^{k}\frac{||\vec{x_0}||t^i}{i!\gamma^{k-i}}|i\rangle\,(\sum_{j=0}^{L}|j\rangle\sqrt{\alpha_j}\hat{\mathbf{A}}_j)^{\otimes i}|x_0\rangle\,(\sum_{j=0}^{L}\sqrt{\alpha_j}\,|j\rangle)^{\otimes k-i}\\ +&\frac{1}{S}|1\rangle\sum_{i=1}^{k}\frac{||\vec{b}||t^i}{i!\gamma^{k-i+1}}|i\rangle\,(\sum_{j=0}^{L}|j\rangle\sqrt{\alpha_j}\hat{\mathbf{A}}_j)^{\otimes i-1}|x_0\rangle\,(\sum_{j=0}^{L}\sqrt{\alpha_j}\,|j\rangle)^{\otimes k-i+1}.\end{aligned} \tag{4.64}$$

Na etapa final, a aplicação dos operadores inversos de todos os respectivos $\hat{\mathbf{V}}^{\mathbf{T}}$, do operador $\hat{\mathbf{V}}$ de $\hat{\mathbf{V}}_{\mathbf{s1}}$ e $\hat{\mathbf{V}}_s 2$, denotados respectivamente por $\hat{\mathbf{W}}^{\mathbf{T}}$, $\hat{\mathbf{W}}$, $\hat{\mathbf{W}}_{\mathbf{s1}}$ e $\hat{\mathbf{W}}_{\mathbf{s2}}$ (Fig. 34) gera um estado cuja projeção no subespaço onde todos os *q-bits* auxiliares possuem valor nulo é a solução multiplicada por um fator $\frac{1}{s^2}$, conforme descrito por:

$$\langle 0|\langle 0|^{\otimes k}\langle 0|_L^{\otimes k}|\psi_3\rangle = \frac{1}{S^2}\sum_{i=0}^{k}\frac{||\vec{x_0}||t^i}{i!}(\sum_{j=0}^{k}\alpha_j\hat{\mathbf{A}}_j)^{\otimes i}|x_0\rangle + \frac{1}{S^2}\sum_{i=1}^{k}\frac{||\vec{b}||t^i}{i!}(\sum_{j=0}^{k}\alpha_j\hat{\mathbf{A}}_j)^{\otimes i-1}|b\rangle\,. \tag{4.65}$$

4.3.5 Análise de complexidade(caso não-unitário)

Com os devidos detalhes elucidados, é possível analisar a complexidade temporal da execução do algoritmo para o caso não-unitário.

Nesse cenário, também há uma série de operadores para os quais apenas importa a análise da atuação no estado $|0\rangle$. Todos estes podem ser implementados como operadores de preparação de estado, desde que seus respectivos coeficientes sejam devidamente ajustados. Os $\hat{\mathbf{V}}^{\mathbf{T}}$ e seus respectivos inversos agem em estados L-dimensionais, e portanto apresentam complexidade de implementação de $O(L)$ (visto que os mesmos podem ser aplicados simultaneamente). Os pares de operadores $\{\hat{\mathbf{U}}_{\mathbf{x}},\hat{\mathbf{U}}_b\}$ e $\{\hat{\mathbf{V}}_{S1}, \hat{\mathbf{V}}_{s2}\}$ podem ser implementados com complexidades de $O(N)$ e $O(2^k)$, respectivamente.

Para aplicar as devidas potências de todas as parcelas da matriz não unitária, utiliza-se k sequências de operadores que agem em n *q-bits*, controlados por $(log_2 L + 1)$ *q-bits* (arranjo de *q-bits* que forma um L-*q-dit*). Com base nos protocolos de implementação de uma matriz geral, a complexidade será de $O(4^{(n+log_2(L)+1)}) = O(L^2N^2)$.

Ao considerar todas as etapas, a complexidade final será de $O(N+L+2^k+kL^2N^2) = O(2^k+kL^2N^2)$. Com relação à variação com N, ainda há, em certos cenários, vantagens com relação ao caso clássico. Por outro lado, a dependência com k gera desvantagens em relação ao caso unitário. O ganho obtido com relação ao caso clássico é, em diversas situações, inferior ao mencionado no trabalho desenvolvido pelos autores [23], uma vez que estes não não levam em consideração o escalamento com $O(N^2)$ resultante da implementação das matrizes $\mathbf{A}$. Este detalhe será discutido no Apêndice A.

5 Conclusão

A análise de complexidade de algoritmos quânticos requer precaução com relação a certas nuances, principalmente no que diz respeito à decomposição adequada de portas de múltiplos *q-bits*, preparação dos estados iniciais e tomografia completa do estado final.

Se todos detalhes forem considerados, é possível concluir que qualquer algoritmo que dependa de preparação de estados terá uma complexidade temporal de ao menos $O(N)$ (o termo $N = 2^n$ refere-se à dimensão do subespaço no qual se prepara o estado) caso o método tradicional seja utilizado. Uma rota aparentemente promissora para esta etapa é a utilização de uma *q-RAM*.

Segundo Tao Xin *et.al.* de [23], a complexidade associada à preparação de estados (essencial para o algoritmo apresentado) poderia ser reduzida drasticamente (para $O(logN)$) se o uso de memória quântica aleatória (q-RAM) for possível. De fato, há propostas teóricas para a construção de uma q-RAM [49], que ainda não foram implementadas experimentalmente, de modo que ainda não se sabe exatamente quais seriam as dificuldades técnicas relacionadas à construção de uma q-RAM completamente funcional. Também há uma questão que não é discutida quando citam o uso da mesma como possibilidade de preparação eficiente de estado, ao assumir que o estado quântico desejado já está preparado na q-RAM e que o acesso e transferência deste para os q-bits de trabalho possuam um custo da ordem de $O(log(N))$. Não há discussão sobre o processo pelo qual estes estados seriam preparados e o respectivo custo. Desse modo, se considerarmos que esse processo é equivalente aos protocolos mais eficientes registrados na literatura, ainda assim o custo seria da ordem de $O(N)$.

As considerações prévias referentes à preparação de estados, bem como a necessidade de implementação de matrizes $N \times N$, resultam em uma complexidade polinomial $O(N^2)$ para o algoritmo de resolução de equações diferenciais minuciosamente analisado, já que as matrizes $N \times N$ apresentam uma complexidade de implementação $O(N^2)$ (como resultado da decomposição em portas C-NOT e de um *q-bit*).

O próprio Tao XIN *et. al.* [23] afirma que a complexidade do algoritmo apresentado escala com a dimensão N da matriz do sistema por uma ordem $O(logN)$, tanto para o caso unitário quanto para o caso não-unitário. Essa divergência é oriunda do fato de assumirem que matrizes gerais (sem restrições sobre seus coeficientes) podem ser implementadas com uma complexidade cuja dependência escala logaritmicamente com N, ou seja, $O(log(N))$, o que é uma afirmação errônea (conforme a análise realizada no Apêndice A). Sendo assim, o ganho exponencial afirmado pelos autores fica restrito a situações específicas (matrizes que podem ser decompostas como um produto tensorial de elementos pertencentes a alguma

base do espaço vetorial das matrizes 2x2).

As etapas posteriores de pós-processamento aumentariam a complexidade dos métodos. Neste contexto, o processo de tomografia de estados (necessário para a reconstrução da informação completa sobre um estado quântico) pode ser realizado de diversas maneiras, cuja complexidade varia entre $O[(log^N)^2]$ e $O(N^4 logN)$ [47], (há procedimentos que possuem $O(N)$ e satisfazem as necessidades inerentes ao algoritmo de resolução de equações diferenciais). Desta maneira, o principal fator responsável pela complexidade do algoritmo é intrínseco e não se deve ao pós-processamento.

Existem plataformas de simulação (como a Qiskit, da IBM) que permitem simular algoritmos quânticos dos mais diversos tipos. O desenvolvimento e análise de simulações visando melhorias e adequações aos processos mais eficientes de preparação de estado e decomposição de portas de multiplos *q-bits* serão consideradas para os próximos trabalhos.

Bibliografia

[1] Jun John Sakurai e Jim Napolitano. *Modern quantum mechanics; 2nd ed.* San Francisco, CA: Addison-Wesley, 2011. URL: https://cds.cern.ch/record/1341875.

[2] P. Benioff. "The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines". Em: *J. Stat. Phys.* 22 (1980), pp. 563–591.

[3] John Preskill. "Quantum computing and the entanglement frontier-Rapporteur talk at the 25th Solvay Conference". Em: (mar. de 2012).

[4] Román Orús, Samuel Mugel e Enrique Lizaso. "Quantum computing for finance: Overview and prospects". Em: *Reviews in Physics* 4 (2019), p. 100028. ISSN: 2405-4283. DOI: https://doi.org/10.1016/j.revip.2019.100028. URL: https://www.sciencedirect.com/science/article/pii/S2405428318300571.

[5] Michael Li e James Muldowney. "Global stability for the SEIR model in epidemiology". Em: *Mathematical biosciences* 125 (mar. de 1995), pp. 155–64. DOI: 10.1016/0025-5564(95)92756-5.

[6] R. Naz, F.M. Mahomed e D.P. Mason. "Comparison of different approaches to conservation laws for some partial differential equations in fluid mechanics". Em: *Applied Mathematics and Computation* 205.1 (2008). Special Issue on Life System Modeling and Bio-Inspired Computing for LSMS 2007, pp. 212–230. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2008.06.042.

[7] William E. Boyce e Richard C. DiPrima. *Elementary differential equations and boundary value problems [by] William E. Boyce and Richard C. Di Prima.* English. 2d ed. Wiley New York, 1969, xiv, 533, A–43, I–8 p. ISBN: 0471093319.

[8] Michael A Nielsen e Isaac Chuang. *Quantum Computation and Quantum Information.* 2002.

[9] A. Einstein. "Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt". Em: *Annalen der Physik* 322.6 (1905), pp. 132–148. DOI: https://doi.org/10.1002/andp.19053220607.

[10] E. Schrödinger. "Quantisierung als Eigenwertproblem". Em: *Annalen der Physik* 384.4 (1926), pp. 361–376. DOI: https://doi.org/10.1002/andp.19263840404.

[11] Louis Broglie. "A Tentative Theory of Light Quanta". Em: *Philosophical Magazine Letters* 86 (jul. de 2006), pp. 411–423. DOI: 10.1080/09500830600914721.

[12] N. Bohr. "On the Constitution of Atoms and Molecules". Em: *Niels Bohr, 1913-2013: Poincaré Seminar 2013*. Ed. por Olivier Darrigol et al. Cham: Springer International Publishing, 2016, pp. 13–33. ISBN: 978-3-319-14316-3. DOI: 10.1007/978-3-319-14316-3_2.

[13] M.Planck e Verhandl. Em: *Dtsch. phys. Ges.* 2,237 (1900).

[14] John von Neumann. *Mathematical Foundations of Quantum Mechanics*. Ed. por Nicholas A. Wheeler. Princeton University Press, 2018. ISBN: 9781400889921. DOI: doi:10.1515/9781400889921.

[15] D.C. Lay, S.R. Lay e J.J. Mcdonald. *Álgebra Linear E Suas Aplicações*. LTC, 2018. ISBN: 9788521634959.

[16] R. T. Thew et al. "Qudit quantum-state tomography". Em: *Phys. Rev. A* 66 (1 jul. de 2002), p. 012303. DOI: 10.1103/PhysRevA.66.012303.

[17] David J Griffiths. *Introduction to elementary particles; 2nd rev. version*. Physics textbook. New York, NY: Wiley, 2008.

[18] G.B. Arfken, H.J. Weber e F.E. Harris. *Mathematical Methods for Physicists: A Comprehensive Guide*. Elsevier Science, 2011. ISBN: 9780123846556.

[19] Kenneth Manders e Leonard Adleman. "NP-Complete Decision Problems for Quadratic Polynomials". Em: *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*. STOC '76. Hershey, Pennsylvania, USA: Association for Computing Machinery, 1976, pp. 23–29. ISBN: 9781450374149. DOI: 10.1145/800113.803627.

[20] C. Chamon et al. "Quantum vertex model for reversible classical computing". Em: *Nature Communications* 8.1 (mai. de 2017). ISSN: 2041-1723. DOI: 10.1038/ncomms15303.

[21] C. G. Almudever et al. "The engineering challenges in quantum computing". Em: *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*. 2017, pp. 836–845. DOI: 10.23919/DATE.2017.7927104.

[22] A N de Oliveira, S P Walborn e C H Monken. "Implementing the Deutsch algorithm with polarization and transverse spatial modes". Em: *Journal of Optics B: Quantum and Semiclassical Optics* 7.9 (ago. de 2005), pp. 288–292. DOI: 10.1088/1464-4266/7/9/009.

[23] Tao Xin et al. "Quantum algorithm for solving linear differential equations: Theory and experiment". Em: *Phys. Rev. A* 101 (3 mar. de 2020), p. 032307. DOI: 10.1103/PhysRevA.101.032307. URL: https://link.aps.org/doi/10.1103/PhysRevA.101.032307.

[24] David P. DiVincenzo. "The Physical Implementation of Quantum Computation". Em: *Fortschritte der Physik* 48.9-11 (set. de 2000), pp. 771–783. ISSN: 1521-3978. DOI: 10.1002/1521-3978(200009)48:9/11<771::aid-prop771>3.0.co;2-e.

[25] Lu Jun. "Physical Realization of Harmonic Oscillator Quantum Computer". Em: *Future Communication, Computing, Control and Management: Volume 1*. Ed. por Ying Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 29–34. ISBN: 978-3-642-27311-7. DOI: 10.1007/978-3-642-27311-7_5.

[26] Ady Stern e Netanel Lindner. "Topological Quantum Computation-From Basic Concepts to First Experiments". Em: *Science (New York, N.Y.)* 339 (mar. de 2013), pp. 1179–84. DOI: 10.1126/science.1231473.

[27] Tameem Albash e Daniel A. Lidar. "Adiabatic quantum computation". Em: *Rev. Mod. Phys.* 90 (1 jan. de 2018), p. 015002. DOI: 10.1103/RevModPhys.90.015002.

[28] Laszlo Gyongyosi e Sandor Imre. "A Survey on quantum computing technology". Em: *Computer Science Review* 31 (2019), pp. 51–71. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2018.11.002.

[29] Adriano Barenco et al. "Elementary gates for quantum computation". Em: *Phys. Rev. A* 52 (5 nov. de 1995), pp. 3457–3467. DOI: 10.1103/PhysRevA.52.3457. URL: https://link.aps.org/doi/10.1103/PhysRevA.52.3457.

[30] Juha Vartiainen, Mikko Möttönen e Martti Salomaa. "Efficient Decomposition of Quantum Gates". Em: *Physical review letters* 92 (mai. de 2004), p. 177902. DOI: 10.1103/PhysRevLett.92.177902.

[31] Mikko Möttönen et al. "Quantum Circuits for General Multiqubit Gates". Em: *Physical review letters* 93 (out. de 2004), p. 130502. DOI: 10.1103/PhysRevLett.93.130502.

[32] Vivek V. Shende, Igor L. Markov e Stephen S. Bullock. "Minimal universal two-qubit controlled-NOT-based circuits". Em: *Phys. Rev. A* 69 (6 jun. de 2004), p. 062321. DOI: 10.1103/PhysRevA.69.062321.

[33] Michael Reck et al. "Experimental realization of any discrete unitary operator". Em: *Phys. Rev. Lett.* 73 (1 jul. de 1994), pp. 58–61. DOI: 10.1103/PhysRevLett.73.58.

[34] Dominic W Berry. "High-order quantum algorithm for solving linear differential equations". Em: *Journal of Physics A: Mathematical and Theoretical* 47.10 (fev. de 2014), p. 105301. ISSN: 1751-8121. DOI: 10.1088/1751-8113/47/10/105301.

[35] Aram W. Harrow, Avinatan Hassidim e Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". Em: *Phys. Rev. Lett.* 103 (15 out. de 2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502.

[36] Dominic Berry et al. "Quantum Algorithm for Linear Differential Equations with Exponentially Improved Dependence on Precision". Em: *Communications in Mathematical Physics* 356 (dez. de 2017). DOI: 10.1007/s00220-017-3002-y.

[37] Andrew M. Childs e Jin-Peng Liu. "Quantum Spectral Methods for Differential Equations". Em: *Communications in Mathematical Physics* 375 (2019), pp. 1427–1457.

[38] F. Fillion-Gourdeau e E. Lorin. *Simple digital quantum algorithm for symmetric first order linear hyperbolic systems*. 2018. arXiv: 1705.09361 [quant-ph].

[39] Juan Miguel Arrazola et al. "Quantum algorithm for nonhomogeneous linear partial differential equations". Em: *Phys. Rev. A* 100 (3 set. de 2019), p. 032306. DOI: 10.1103/PhysRevA.100.032306.

[40] Gilles Brassard et al. "Quantum amplitude amplification and estimation". Em: *Quantum Computation and Information* (2002), pp. 53–74. ISSN: 0271-4132. DOI: 10.1090/conm/305/05215.

[41] Lov K. Grover. "Quantum Mechanics Helps in Searching for a Needle in a Haystack". Em: *Phys. Rev. Lett.* 79 (2 jul. de 1997), pp. 325–328. DOI: 10.1103/PhysRevLett.79.325.

[42] Lov K. Grover. "Synthesis of Quantum Superpositions by Quantum Computation". Em: *Phys. Rev. Lett.* 85 (6 ago. de 2000), pp. 1334–1337. DOI: 10.1103/PhysRevLett.85.1334.

[43] Srinivasan Arunachalam e Ronald Wolf. "Optimizing the Number of Gates in Quantum Search". Em: *Quantum Information and Computation* 17 (dez. de 2015). DOI: 10.26421/QIC17.3-4-4.

[44] Daniel Sierra-Sosa, Michael Telahun e Adel Elmaghraby. "TensorFlow Quantum: Impacts of Quantum State Preparation on Quantum Machine Learning Performance". Em: *IEEE Access* 8 (2020), pp. 215246–215255. DOI: 10.1109/ACCESS.2020.3040798.

[45] Israel Araújo et al. "A divide-and-conquer algorithm for quantum state preparation". Em: *Scientific Reports* 11 (mar. de 2021). DOI: 10.1038/s41598-021-85474-1.

[46] Yuval Sanders et al. "Black-Box Quantum State Preparation without Arithmetic". Em: *Physical Review Letters* 122 (jan. de 2019). DOI: 10.1103/PhysRevLett.122.020502.

[47] Fernando R. Cardoso et al. *Detailed Account of Complexity for Implementation of Some Gate-Based Quantum Algorithms*. 2021. arXiv: 2106.12517 [quant-ph].

[48] Mikko Möttönen et al. "Transformation of quantum states using uniformly controlled rotations". Em: *Quantum Information Computation* 5 (set. de 2005), pp. 467–473. DOI: 10.26421/QIC5.6-5.

[49] Vittorio Giovannetti, Seth Lloyd e Lorenzo Maccone. "Architectures for a quantum random access memory". Em: *Phys. Rev. A* 78 (5 nov. de 2008), p. 052310. DOI: 10.1103/PhysRevA.78.052310. URL: https://link.aps.org/doi/10.1103/PhysRevA.78.052310.

[50] D.K. Park, F. Petruccione e J.K. Rhee. "Circuit-Based Quantum Random Access Memory for Classical Data". Em: *Sci Rep* 3949 (9 2019). DOI: 10.1038/s41598-019-40439-3. URL: https://doi.org/10.1038/s41598-019-40439-3.

[51] Dan Ventura e Tony Martinez. "Quantum associative memory". Em: *Information Sciences* 124.1 (2000), pp. 273–296. ISSN: 0020-0255. DOI: https://doi.org/10.1016/S0020-0255(99)00101-2.

[52] Frank Leymann e Johanna Barzen. "The bitter truth about gate-based quantum algorithms in the NISQ era". Em: *Quantum Science and Technology* 5.4 (set. de 2020), p. 044007. DOI: 10.1088/2058-9565/abae7d.

[53] Tao Xin et al. "Quantum simulation of quantum channels in nuclear magnetic resonance". Em: *Phys. Rev. A* 96 (6 dez. de 2017), p. 062303. DOI: 10.1103/PhysRevA.96.062303.

A Complexidade de operações não-unitárias

Os fundamentos até então discutidos e aplicados estabelecem a unitariedade das portas aplicadas na computação quântica, porém há uma maneira de emular a aplicação de portas não-unitárias [53] através da utilização de um conjunto de *q-bits* auxiliares. Este tipo de operação é utilizada como subrotina no algoritmo para resolução de equações diferenciais [23] e há alguns aspectos importantes a serem abordados sobre a complexidade destas.

O processo foi originalmente implementado como uma forma de simular canais quânticos (um tipo de mapa completamente positivo e com preservação de traço, CPTP) utilizando processadores quânticos baseados em ressonância [53].

Os canais específicos abordados descrevem a evolução de um sistema quântico aberto (que interage com o ambiente). Tais operações são descritas em termos de uma matriz de densidade ρ (uma variável descritiva de sistemas quânticos que pode, em certas circunstâncias, servir como generalização do estado quântico). O canal pode ser descrito por:

$$\epsilon(\rho) = \sum_k \hat{\mathbf{E}}_k \rho \hat{\mathbf{E}}_k^\dagger. \tag{A.1}$$

Os operadores $\hat{\mathbf{E}}_k$ são denominados operadores de Kraus, e obedecem a seguinte condição:

$$\sum_k \hat{\mathbf{E}}_k \hat{\mathbf{E}}_k^\dagger = \mathbf{I}. \tag{A.2}$$

Foram abordados $\hat{\mathbf{E}}_k$ específicos, descrevendo três tipos de canais quânticos: *Amplitude damping (AD), depolarizing(DEP) e phase damping(PD)*, porém um algoritmo geral que permite simular qualquer $\hat{\mathbf{E}}_k$ é apresentado. Para a realização do processo, é necessário uma descrição dos operadores $\hat{\mathbf{E}}_k$ em termos de uma base de operadores unitários, ou seja, expressa-los de acordo com a seguinte relação:

$$\hat{\mathbf{E}}_k = \sum_{i=1}^{L} \alpha_i \hat{\mathbf{U}}_i. \tag{A.3}$$

Embora os três canais cuja simulação foi implementada experimentalmente sejam descritos em termos da base $\{\hat{\mathbf{I}}, \hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z\}$, o algoritmo permite a implementação de operadores $\hat{\mathbf{E}}$ de qualquer dimensão, expandidos por qualquer base unitária coerente com esta.

Sendo satisfeita a condição expressa na equação (A.3), a ação do canal ϵ pode ser simulada através do circuito ilustrado na Fig. 35. A construção utiliza também um

L-qdit, com L níveis e $log_2(L)$ *q-bits*, e um conjunto de operações unitárias $\hat{\mathbf{U}}$ controladas, correspondentes às mesmas listadas na equação (A.3).

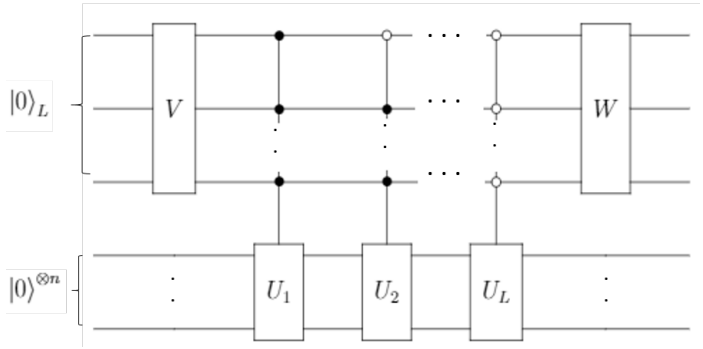

Figura 35 – Simulação geral de canais quânticos (ou qualquer operação não unitária). O operador $\hat{\mathbf{V}}$ codifica, no espaço auxiliar, os coeficientes da combinação de operadores unitários desejada. A sequência de operadores $\hat{\mathbf{U}}_j$ utiliza as bases do espaço auxiliar como controle para criar uma combinação destes operadores aplicada no estado de trabalho. Por fim, a aplicação do operador $\hat{\mathbf{W}}$ garante que a combinação de operadores desejada esteja codificada no subespaço no qual todos os *q-bits* auxiliares possuem valor nulo.

A ação de um canal quântico como o descrito pela equação (A.1) não envolve apenas um único operador não-unitário, mas vários que obedecem a equação (A.2), o que resulta no fato dos coeficientes das matrizes representativas de $\hat{\mathbf{V}}$ e $\hat{\mathbf{W}}$ obedecerem restrições que nem sempre permitem que a matriz de $\hat{\mathbf{W}}$, especificamente, possa ser simulada por um procedimento de preparação de estados. Para os propósitos inerentes à utilização no algoritmo para resolução de EDOs, porém, não há necessidade de aplicação de vários operadores não-unitários diferentes (apenas um e suas respectivas potências). Neste caso, conforme discutido no capítulo anterior, os operadores $\hat{\mathbf{V}}$ e $\hat{\mathbf{W}}$ podem ser implementados através de procedimentos de preparação de estados, pois é suficiente que os coeficientes das matrizes de $\hat{\mathbf{V}}$ e $\hat{\mathbf{W}}$ tenham restrição apenas sobre a primeira coluna, podendo o restante destas apresentar quaisquer valores que as tornem unitárias.

Dada uma expansão do operador não-unitário $\hat{\mathbf{V}}$ em termos de uma base unitária como na equação (A.3), os termos matriciais relevantes (primeira coluna) são então definidos por:

$$\hat{\mathbf{V}}[i,0] = \frac{\sqrt{\alpha_i}}{M}. \tag{A.4}$$

Consequentemente, o estado do sistema após a aplicação do operador em questão será descrito por:

$$|\psi_1\rangle = \sum_{i=0}^{L-1} \frac{\sqrt{\alpha_i}}{M} |i\rangle |0\rangle^{\otimes n}; M = \sum_{i=1}^{L} |\alpha_i|. \tag{A.5}$$

Em seguida, há a aplicação das múltiplas operações $\hat{\mathbf{U}}_i$ controladas , que levam o

sistema ao seguinte estado:

$$|\psi_2\rangle = \sum_{i=0}^{L-1} \frac{\sqrt{\alpha_i}}{M} |i\rangle \hat{\mathbf{U}}_i |0\rangle^{\otimes n} . \tag{A.6}$$

A aplicação do operador $\hat{\mathbf{W}} = \hat{\mathbf{V}}^\dagger$ leva o sistema ao seu estado final:

$$\hat{\mathbf{W}}[0,i] = \frac{\sqrt{\alpha_i}}{M} \Rightarrow |\psi_3\rangle = \sum_{i=0}^{L-1} \sqrt{\alpha_i} \sum_{j=0}^{L-1} W_{ji} |j\rangle \hat{\mathbf{U}}_i |0\rangle^{\otimes n} . \tag{A.7}$$

Este último estado pode ser separado em duas parcelas. Uma correspondente ao subespaço no qual todos os *q-bits* auxiliares são nulos e outra para os demais estados do sistema auxiliar, de acordo com a seguinte relação:

$$|\psi_3\rangle = \sum_{i=0}^{L-1} \alpha_i |0\rangle^{\otimes L} \hat{\mathbf{U}}_i |0\rangle^{\otimes n} + \sum_{i=0}^{L-1} \sqrt{\alpha_i} \sum_{j=1}^{L-1} W_{ji} |j\rangle \hat{\mathbf{U}}_i |0\rangle^{\otimes n} . \tag{A.8}$$

Se for realizada uma medição cujo resultado corresponda ao subespaço no qual todos os auxiliares são nulos, o resultado será o estado desejado, exceto por uma constante de normalização:

$$\langle 0|^{\otimes L} |\psi_3\rangle = \left(\sum_{i=0}^{L-1} \frac{\alpha_i}{M} \hat{\mathbf{U}}_i \right) |0\rangle^{\otimes n} . \tag{A.9}$$

É importante ressaltar que a aplicação de uma combinação linear de operadores unitários conforme descrita pela equação (A.3) é uma etapa importante do algoritmo apresentado em [53]. Embora o propósito de aplicação do processo descrito na sessão tenha sido, a princípio, efetuar operações não-unitárias, este pode ser usado para a realizar qualquer operação que seja descrita por uma combinação linear de operadores unitários (incluindo outros desse tipo), o que o faz adequado para efetuar as somas de potências (unitárias e não-unitárias) existentes no algoritmo para a resolução de EDOs.

Cada uma das operações controladas ($\mathbf{C}_m\hat{\mathbf{U}}$, conforme a notação apresentada nos capítulos anteriores) pode ser efetuada através de um conjunto de portas C-NOT e de um *q-bit*. O processo de decomposição é similar ao apresentado no Cap. 3. Para tal tipo de construção, não há exigência de que $\hat{\mathbf{U}}$ seja uma porta de um *q-bit*, mas há acréscimo na complexidade caso não seja. A decomposição de uma operação controlada é ilustrada na Fig. 36. O operador V é tal que $\hat{\mathbf{V}}^2 = \hat{\mathbf{U}}$

A ação do operador multicontrolado pode ser compreendida levando em conta três diferentes casos: Quando os $m-1$ primeiros *q-bits* de controle apresentam ao menos um valor diferente de um, serão ativados $\hat{\mathbf{V}}$ e $\hat{\mathbf{V}}^\dagger$ caso o último *q-bit* seja um, ou nenhum dos operadores, caso seja zero. Como $\hat{\mathbf{V}}$ é o inverso de $\hat{\mathbf{V}}^\dagger$, então ambas situações resultam na inexistência de alteração em qualquer dos *q-bits*. Se os $m-1$ primeiros *q-bits* de controle apresentarem todos valor um, e o ultimo valor zero, a porta $\hat{\mathbf{V}}^\dagger$ será ativada e a $\mathbf{C}^{m-1}\hat{\mathbf{V}}$

também, resultando novamente na aplicação do operador identidade. Por fim, se todos os *q-bits* de controle apresentarem valor um, apenas a primeira $\hat{\mathbf{V}}$ e a $\mathbf{C}^{m-1}\hat{\mathbf{V}}$ serão ativadas, e como $\hat{\mathbf{V}}^2 = \hat{\mathbf{U}}$, o resultado obtido estará de acordo com o propósito inerente à construção (aplicação de $\hat{\mathbf{U}}$ se todos os *q-bits* de controle apresentarem valor um, e inalteração no caso contrário).

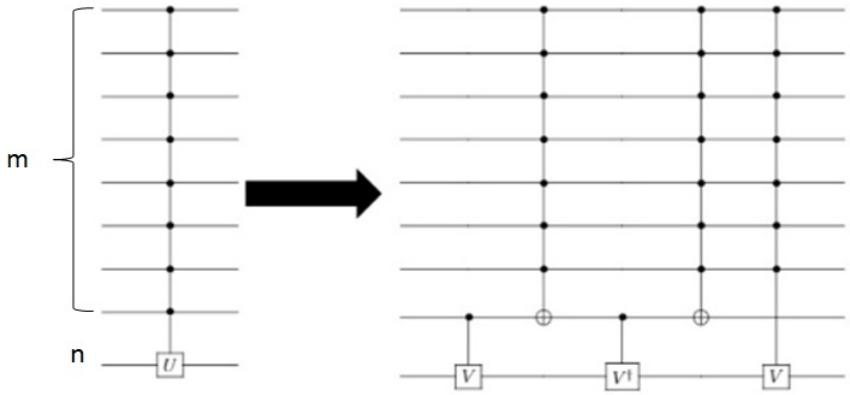

Figura 36 – Decomposição de porta multicontrolada. As portas $\hat{\mathbf{V}}$ são tais que $\hat{\mathbf{V}}^2 = \hat{\mathbf{U}}$ Este arranjo, com alternância entre as portas $\hat{\mathbf{V}}$, $\hat{\mathbf{V}}^\dagger$ e portas NOT multicontroladas garante que não haverá um resultado final equivalente à aplicação da identidade no *q-bit* alvo somente se todos os *q-bits* de controle apresentarem valor um. E somente neste caso, o resultado é a aplicação de $\hat{\mathbf{U}}$ nos *q-bits* de trabalho.

A complexidade total pode ser analisada levando em consideração o caráter recursivo da operação, através da contabilização das quatro portas utilizadas no processo. A complexidade das portas NOT controladas é de $O(n)$, e a das portas $\hat{\mathbf{V}}$ e $\hat{\mathbf{V}}^\dagger$ pode ser denotada por C(n) (visto que estas podem ser qualquer operação, a princípio). Como a implementação apresentada exige recursivamente a utilização de uma porta $\mathbf{C^{m-1}\hat{V}}$ ao final, então esta tem complexidade temporal (denotada por T) dada por:

$$T(m,n) = O(m) + C(n) + T(m-1,n). \tag{A.10}$$

A recursão expressa pela equação (A.10) deverá ser repetida m vezes, de forma tal que a complexidade final resultante pode ser descrita por:

$$T(m,n) = mC(n) + mO(m) = m^2 + mC(n). \tag{A.11}$$

A complexidade $C(n)$ associada à implementação de um operador atuando em n *q-bits*, conforme desenvolvido no Cap. 3, é de $O(4^n)$, portanto resulta que o processo de implementação de operações controladas possui uma complexidade total de $O(m^2 + m4^n) = O(m^2 + mN^2)$. Para casos específicos como o citado em [23], nos quais supõe-se como previamente conhecida uma decomposição da matriz de interesse a ser implementada em produtos tensoriais de operadores pertencentes a uma base unitária do

espaço das matrizes bidimensionais, a complexidade C(n) é reduzida para O(1). Desta maneira, para este conjunto finito de casos (produtos das possíveis permutações dos quatro operadores pertencentes à base unitária), resulta uma complexidade total de $O(m^2 + m)$ para implementação das operações multicontroladas.

B Códigos computacionais

```
from qiskit import*
import numpy as np
from numpy import linalg as la
from numpy.linalg import norm
import matplotlib.pyplot as plt
import math
import cmath

#Função que cria um vetor com os algarismos binários de 'num', usando
'dig' dígitos
def binr(dig,num):
    a=[int(x) for x in list('{0:0b}'.format(num))]
    for j in range(dig-len(a)):
        b=[0]+a
        a=b
    return a

#Função que gera uma matriz M-dimensional só com elementos nulos.
def mat_zero(M):
    aux1=[0.0]
    for i in range(M-1):
        aux1=aux1+[0.0]
    aux2=[aux1]
    for i in range (M-1):
        aux3=np.concatenate((aux2,[aux1]), axis=0)
        aux2=aux3
    return aux3

#Função que gera os ângulos a serem usados nas rotações uniformemente
controladas
def ang(vec):
    dim=len(vec)
    n=int(np.log2(dim))
    y=mat_zero(dim)
    z=mat_zero(dim)
    c=mat_zero(dim)
    a=mat_zero(dim)
    nums=[0]*dim
    #Inicializando, no último nível das matrizes, variáveis
correspondentes ao modulo e argumento do estado.
    for m in range(dim):
        c[n][m]=abs(vec[m])
        a[n][m]=cmath.phase(vec[m])
    auxn=[0]*n
    for m in range(n):
        auxn[m]=n-m-1

    #Os aux são as variáveis para percorrer a "árvore" do ultimo até o
primeiro nível.

    for aux in auxn:
```

```
        for j in range(pow(2,aux)):
            k=j*2#Como o calculo do ângulo sempre envolve dois termos,
é preciso percorrer na ordem correta
            cateto1=c[aux+1][k]
            cateto2=c[aux+1][k+1]
            hip=np.sqrt(pow(cateto1,2)+pow(cateto2,2))
            formatted_string = "{:.6f}".format(hip)
            var=float(formatted_string)
            if (var!=0.000000):
                y[aux][j]=2*np.arctan2(cateto2,cateto1)
            else:
                y[aux][j]=0.0
            c[aux][j]=hip
            #Devido à convenção do qiskit, parcelas com sinais
diferentes.
            if(aux!=0):
                z[aux][j]=-a[aux+1][k]+a[aux+1][k+1]
                a[aux][j]=a[aux+1][k]+0.5*z[aux][j]
            else:
                z[aux][j]=-2*a[aux+1][k]
                s=a[aux+1][k+1]-0.5*z[aux][j]
    return dim,n,s,y,z

#Função para criar um vetor que contem uma "contagem regressiva" de n
até n-m+1.
#Será necessário para indicar a ordem correta das portas controladas,
devido à convenção do Qiskit.
def seq(n,m):
    v=[n]
    for i in range(m-1):
        v=v+[n-i-1]
    return v

#Função com o mesmo propósito da anterior, mas cria sequência
crescente do inicial "init" até o final "fin".
def ord_seq(init,fin):
    v=[init]
    n=fin-init
    for i in range(n):
        v=v+[init+i+1]
    return v

#Função que efetivamente cria o cicrcuito de preparação de estados.
def prep(vec):
    tam,qbits,shift,ang_y,ang_z=ang(vec)
    ycircuit=QuantumCircuit(qbits)
    zcircuit=QuantumCircuit(qbits)
    ycircuit.ry(ang_y[0][0],qbits-1)
    zcircuit.rz(ang_z[0][0],qbits-1)
    indices=[0]*(qbits-1)
    for m in range(qbits-1):
```

```
            indices[m]=m+1
        for indice in indices:
            for j in range(pow(2,indice)):
                qcz=QuantumCircuit(1)
                qcy=QuantumCircuit(1)
                qcy.ry(ang_y[indice][j],0)
                qcz.rz(ang_z[indice][j],0)
                k=j*pow(2,qbits-indice)
                vec=binr(qbits,k)
                for i in range(indice):#Devido à convenção do qiskit de
mapeamento.Laço para mudar os controles de 1 para 0.
                    if(vec[i]==0):
                        ycircuit.x(qbits-i-1)
                        zcircuit.x(qbits-i-1)

                #Aplicação da rotação, com o ângulo adequado e quantidade
de controles certa, no devido q-bit.
                con_y=qcy.to_gate().control(indice)
                con_z=qcz.to_gate().control(indice)
                ycircuit.append(con_y,seq(qbits-1,indice)+[qbits-1-
indice])#Um artifício para ajustar os parâmetros.
                zcircuit.append(con_z,seq(qbits-1,indice)+[qbits-1-
indice])# [controles] +[alvos]

                for i in range(indice):
                    if(vec[i]==0):
                        ycircuit.x(qbits-i-1)
                        zcircuit.x(qbits-i-1)
        cirprep=QuantumCircuit(qbits)
        cirprep=cirprep.compose(ycircuit,range(qbits))
        cirprep.p(shift,qbits-1)#O deslocamento necessário para a correção
do ultimo ângulo na CZ.
        cirprep=cirprep.compose(zcircuit,range(qbits))
        return cirprep

#Cria matrizes garantidamente unitárias para teste do algoritmo
#...estas são puramente reais, mas para finalidade de teste é
suficiente.
def unitarygen(thetas):
    dim=len(thetas)
    produto=[1]
    for i in range(dim):
        t=thetas[i]/2
        fator=[[np.cos(t),-np.sin(t)],[np.sin(t),np.cos(t)]]
        aux=produto
        produto=np.kron(fator,aux)
    return produto

def div_mat(M,e):
    dim=len(M[0])
    A=mat_zero(dim)
```

```
    for i in range(dim):
        for j in range(dim):
            A[i][j]=M[i][j]/e
    return A

def C(M,vec,n,t):
    aux=(norm(vec)*(t)**n)/np.math.factorial(n)
    return aux
def D(M,vec,n,t):
    aux=(norm(vec)*(t)**(n-1))*t/np.math.factorial(n)
    return aux

def CG(num,vec,n,t):
    aux=(norm(vec)*((num*t)**n))/np.math.factorial(n)
    return aux
def DG(num,vec,n,t):
    aux=(norm(vec)*((num*t)**(n-1))*t)/np.math.factorial(n)
    return aux

def Solanoun(M,b,x0,t,k):#Caso com matriz unitária
    N=len(x0)
    n=math.ceil(np.log2(N))
    anc=math.ceil(np.log2(k+1))
    Cnum=0.0
    Dnum=0.0
    vc=[0]*(2**anc)
    vd=[0]*(2**anc)
    aux=0.0

    #Calculando C,D e N
    for i in range(k+1):
        aux=aux+C(M,x0,i,t)
    Cnum=np.sqrt(aux)
    aux=0.0
    for i in range(k):
        aux=aux+D(M,b,i+1,t)
    Dnum=np.sqrt(aux)
    alep=np.sqrt(Cnum**2+Dnum**2) #(alep=N)

    #Calculando a primeira coluna dos operadores Vs1 e Vs2
    for i in range(k+1):
        vc[i]=np.sqrt(C(M,x0,i,t))/Cnum
    for i in range(k):
        vd[i]=np.sqrt(D(M,b,i+1,t))/Dnum
    #As constantes C, D e N são garantidamente reais, devido à forma
como são definidas.

    #Calculando e aplicando a primeira porta V
    v=[[Cnum/alep,Dnum/alep],[Dnum/alep,-Cnum/alep]]
    circ=QuantumCircuit(n+anc+1)
    circ.unitary(v,n+anc)
```

```
    #Aplicando Ux e Ub controlados.
    conUx=prep(x0).to_gate().control(1)
    conb=prep(b).to_gate().control(1)
    circ.x(n+anc)
    circ.append(conUx,[n+anc]+ord_seq(0,n-1))#[controles] +[alvos]
    circ.x(n+anc)
    circ.append(conb,[n+anc]+ord_seq(0,n-1))

    #Aplicando as Vs1 e Vs2 controladas.
    conVs1=prep(vc).to_gate().control(1)
    conVs2=prep(vd).to_gate().control(1)
    circ.x(n+anc)
    circ.append(conVs1,[n+anc]+ord_seq(n,n+anc-1))
    circ.x(n+anc)
    circ.append(conVs2,[n+anc]+ord_seq(n,n+anc-1))

    #Aplicando as Um controladas.
    circ1=QuantumCircuit(n+anc+1)
    A=M
    for m in range(anc):
        auxc=QuantumCircuit(n)
        mpow=la.matrix_power(A,2**m)
        auxc.unitary(mpow,ord_seq(0,n-1))
        conUm=auxc.to_gate().control(1)
        circ1.append(conUm,[n+m]+ord_seq(0,n-1))

    #Aplicando as inversas de Vs2, Vs2, Ux e Ub.
    conVs1dag=prep(vc).to_gate().inverse().control(1)
    conVs2dag=prep(vd).to_gate().inverse().control(1)
    circ1.x(n+anc)
    circ1.append(conVs1dag,[n+anc]+ord_seq(n,n+anc-1))
    circ1.x(n+anc)
    circ1.append(conVs2dag,[n+anc]+ord_seq(n,n+anc-1))
    circ1.unitary(v,n+anc).inverse()
    circ=circ.compose(circ1,range(n+anc+1))
    return alep,circ

def solanogen(M,mats,coefs,b,x_0,t,k):#Caso com matriz geral.
    n=math.ceil(np.log2(len(x_0)))
    L=len(coefs)
    qdits=math.ceil(np.log2(L))
    arr=[0]*L
    soma=0.0
    for i in range(L):
        soma=soma+coefs[i]
    fator=soma

    #Calculo das constantes g1,g2
    aux=0.0
```

```
for i in range(L):
    aux=aux+coefs[i]
g1=0.0
for i in range (k+1):
    g1=g1+CG(fator,x_0,i,t)
g1=np.sqrt(g1)
g2=0.0
for i in range(k):
    g2=g2+DG(fator,b,i+1,t)
g2=np.sqrt(g2)
S=g1**2+g2**2

#Criando e aplicando a primeira porta v
c=np.sqrt(g1**2+g2**2)
v=[[g1/c,g2/c],[g2/c,-g1/c]]
reg=1+k+k*qdits+n
circ=QuantumCircuit(reg)
circ.unitary(v,reg-1)

#Criando e aplicando as VT como prepração de estados
vecV=[0]*(2**qdits)
for i in range(L):
    vecV[i]=np.sqrt(coefs[i])
gate=prep(vecV).to_gate()
for i in range(k):
    init=n+i*qdits
    fin=init+qdits-1
    circ.append(gate,ord_seq(init,fin))

#Criando e aplicando Ux e Ub.
conUx=prep(x_0).to_gate().control(1)
conb=prep(b).to_gate().control(1)
circ.x(reg-1)
circ.append(conUx,[reg-1]+ord_seq(0,n-1))
circ.x(reg-1)
circ.append(conb,[reg-1]+ord_seq(0,n-1))
circ.barrier()

#Criando e aplicando Vs1 e Vs2
vecs1=[0]*(2**k)
vecs2=[0]*(2**k)
for j in range(k+1):
    ind=2**k-2**(k-j)
    vecs1[ind]=np.sqrt(CG(fator,x_0,j,t))
for j in range (k):
    ind=2**k-2**(k-j)
    vecs2[ind]=np.sqrt(DG(fator,b,j+1,t))
a1=norm(vecs1)
a2=norm(vecs2)
for j in range(2**k):
```

```
        vecs1[j]=vecs1[j]/a1
        vecs2[j]=vecs2[j]/a2
    convs1=prep(vecs1).to_gate().control(1)
    convs2=prep(vecs2).to_gate().control(1)
    circ.x(reg-1)
    circ.append(convs1,[reg-1]+ord_seq(n+k*qdits,n+k*qdits+k-1))
    circ.x(reg-1)
    circ.append(convs2,[reg-1]+ord_seq(n+k*qdits,n+k*qdits+k-1))
    circ.barrier()

    #Criando e aplicando as multiplas U controladas
    for i in range(k):
        for j in range(L):
            vec=binr(qdits,j)
            auxc=QuantumCircuit(n)
            auxc.unitary(mats[j],ord_seq(0,n-1))
            gate=auxc.to_gate().control(qdits+1)
            fin=reg-k-2-i*qdits
            init=fin-qdits+1
            for h in range(qdits):
                if(vec[h]==0):
                    circ.x(init+h)
            circ.append(gate,[reg-2-i]+ord_seq(fin-qdits+1,fin)+[0,n-
1])
            for h in range(qdits):
                if(vec[h]==0):
                    circ.x(init+h)
            circ.barrier()

    #Aplicando os operadores inversos de Vs1 e Vs1.
    convs1dag=prep(vecs1).to_gate().inverse().control(1)
    convs2dag=prep(vecs2).to_gate().inverse().control(1)
    circ.x(reg-1)
    circ.append(convs1dag,[reg-1]+ord_seq(n+k*qdits,n+k*qdits+k-1))
    circ.x(reg-1)
    circ.append(convs2dag,[reg-1]+ord_seq(n+k*qdits,n+k*qdits+k-1))
    circ.barrier()

    #Aplicando os inversos dos VT
    gatedag=prep(vecV).to_gate().inverse()
    for i in range(k):
        init=n+i*qdits
        fin=init+qdits-1
        circ.append(gatedag,ord_seq(init,fin))

    #Aplicando a W, inversa de V.
    circ.unitary(v,reg-1).inverse()
    return S,circ
```

```
#Procedimento para extrair a solução do vetor de estado completo
gerado pelo algoritmo do Tao Xin et. al.
def getsol(vec, anc, num):
    sol=[0]
    tam=len(vec)
    dig=math.ceil(np.log2(tam))
    for i in range(tam):
        alg=binr(dig,i)
        aux=0
        for j in range(anc):
            aux=aux+alg[j]
        if (aux==0):
            sol=sol+[vec[i]*num]
    return sol

def summat(m1,m2):
    dim=len(m1)
    s=mat_zero(dim)
    for i in range(dim):
        for j in range (dim):
            s[i][j]=m1[i][j]+m2[i][j]
    return s

#Teste unitário.

M=unitarygen([math.pi/6,math.pi/4])
b=[1.9,0.1,0.8,2.6]
x0=[0.3,0.1,2.0,1.2]
N,cir=Solanoun(M,b,x0,0.2,6)
backend=BasicAer.get_backend('statevector_simulator')
result = execute(cir, backend).result()
psi  = result.get_statevector(cir)
print(getsol(psi,4,N**2))

#Teste com a matriz anterior, com n maior e t maior
M=unitarygen([math.pi/6,math.pi/4])
b=[1.9,0.1,0.8,2.6]
x0=[0.3,0.1,2.0,1.2]
N,cir=Solanoun(M,b,x0,1.3,13)
backend=BasicAer.get_backend('statevector_simulator')
result = execute(cir, backend).result()
psi  = result.get_statevector(cir)
print(getsol(psi,5,N**2))

M=unitarygen([math.pi/3,math.pi/5])
b=[0.5,0.3,0.0,1.5]
x0=[0.3,0.1,2.0,1.2]
N,cir=Solanoun(M,b,x0,0.2,3)
backend=BasicAer.get_backend('statevector_simulator')
result = execute(cir, backend).result()
```

```
psi  = result.get_statevector(cir)
print(getsol(psi,3,N**2))

#Teste não unitário.

U=unitarygen([math.pi/3,math.pi/5])

V=unitarygen([math.pi/6,math.pi/4])

W=div_mat(V,2)
M=summat(U,W)

b=[1.9,0.1,0.8,2.6]
x0=[0.3,0.1,2.0,1.2]
S,circ=solanogen(M,[U,V],[1,1/2],b,x0,0.4,4)

backend=BasicAer.get_backend('statevector_simulator')
result = execute(circ, backend).result()
psi  = result.get_statevector(circ)
print(getsol(psi,9,S))

S,circ=solanogen(M,[U,V],[1,1/2],b,x0,0.8,5)
backend=BasicAer.get_backend('statevector_simulator')
result = execute(circ, backend).result()
psi  = result.get_statevector(circ)
print(getsol(psi,11,S))

#Outra matriz, agora combinação de 3 unitárias.

A=unitarygen([math.pi/3,math.pi/5])
B=unitarygen([math.pi/6,math.pi/4])
C=unitarygen([math.pi/8,math.pi/7])
U=A
V=div_mat(B,2)
W=div_mat(C,3)
M=summat(summat(U,V),W)
S,circ=solanogen(M,[A,B,C],[1,1/2,1/3],b,x0,0.2,4)
backend=BasicAer.get_backend('statevector_simulator')
result = execute(circ, backend).result()
psi  = result.get_statevector(circ)
print(getsol(psi,13,S))
```