UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DE CIÊNCIAS EXATAS E ENGENHARIAS
ENGENHARIA ELÉTRICA

GUILHERME ADAMATTI BRIDI

SIMULAÇÃO DE CÁLCULO DE FLUXO DE POTÊNCIA UTILIZANDO COMPUTAÇÃO QUÂNTICA

CAXIAS DO SUL
2021

Guilherme Adamatti Bridi

SIMULAÇÃO DE CÁLCULO DE FLUXO DE POTÊNCIA UTILIZANDO COMPUTAÇÃO QUÂNTICA

Trabalho de Conclusão de Curso apresentado à Área de Ciências Exatas e Engenharias da Universidade de Caxias do Sul como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica.

Orientador:
Prof. Dr. Alexandre Mesquita

CAXIAS DO SUL
2021

Guilherme Adamatti Bridi

SIMULAÇÃO DE CÁLCULO DE FLUXO DE POTÊNCIA UTILIZANDO COMPUTAÇÃO QUÂNTICA

Trabalho de Conclusão de Curso apresentado à Área de Ciências Exatas e Engenharias da Universidade de Caxias do Sul como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica.

Orientador:
Prof. Dr. Alexandre Mesquita

Aprovado em 07/12/2021

Banca Examinadora

_____
Prof. Dr. Alexandre Mesquita (orientador)
Universidade de Caxias do Sul - UCS

_____
Prof. Dr. Andre Luis Martinotto
Universidade de Caxias do Sul - UCS

_____
Prof. Me. André Bernardes Michel
Universidade de Caxias do Sul - UCS

RESUMO

Em um contexto no qual as redes elétricas estão cada vez mais distribuídas e interligadas, a miríade de dados a serem processados pelas concessionárias pode estar além da tecnologia computacional. Torna-se justificável a inserção de um novo paradigma computacional, a computação quântica, que tem potencial teórico para resolver problemas específicos em um intervalo de tempo que pode ser exponencialmente menor que o tradicional. No contexto das redes elétricas, um problema que nos últimos anos vem sendo estudado é a incorporação do algoritmo quântico de Harrow, Hassidim e Lloyd (HHL) para solução de equações lineares ao problema de fluxo de potência, tanto para o tradicional por métodos iterativos, quanto para o fluxo de potência CC, um método que lineariza o problema através de algumas aproximações. Este trabalho realiza a simulação ideal de fluxo de potência CC através de um código construído no *Qiskit* baseado no algoritmo HHL para dois sistemas, de três e de cinco barras, obtendo um circuito de 4 *q-bits* com solução exata para o primeiro e um circuito de 5 *q-bits* com erro percentual absoluto médio de 6.124% para o último, além da experimentação em um computador quântico real da IBM para o sistema de três barras, com um erro de 6.935%. Realiza ainda uma comparação entre a simulação quântica e uma simulação clássica de fluxo de potência por métodos iterativos que resulta em um erro de 7.630%. Adicionalmente, um código com funcionalidade equivalente foi construído em Matlab.

Palavras-chave: Computação Quântica. Fluxo de Potência CC. Algoritmo HHL. Qiskit.

ABSTRACT

In a context in which electricity grids are increasingly distributed and interconnected, a myriad of data to be processed by utilities may be beyond computer technology. The insertion of a new computational paradigm becomes justifiable, the quantum computing, which has theoretical potential to solve specific problems in a time interval that can be exponentially smaller than the traditional. In the context of power systems, a problem that has been studied in the last two years is the incorporation of the quantum algorithm of Harrow, Hassidim and Lloyd (HHL) for solving linear equations to the power flow problem, both for the traditional by iterative methods, as for the DC power flow, a method that linearizes the problem through some approximations. This work performs the DC power flow ideal simulation of two systems, through a code built in Qiskit based on the HHL algorithm for two systems, three and five bars, obtaining a circuit of 4 q-bits with exact solution for the former and a circuit of 5 q-bits with mean absolute percentage error of 6.124% for the latter, plus experimentation on a real IBM quantum computer for the three bar system, with an error of 6.935%. It also makes a comparison between the quantum simulation with a classical power flow simulation by iterative methods that results in an error of 7.630%. Additionally, code with equivalent functionality was built in Matlab.

Keywords: Quantum Computing. DC Power Flow. HHL Algorithm. Qiskit.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

HHL	Harrow, Hassidim e Lloyd
CC	Corrente contínua
IBM	*International Business Machines*
DFT	*Discrete fourier transform*
FFT	*Fast Fourier Transform*
QFT	*Quantum Fourier Transform*
QPE	*Quantum Phase Estimation*
Qasm	*Quantum Assembly Language*
QPF	*Quantum Power Flow*
NISQ	*Noisy intermediate-scale quantum*
QUEST	*Quantum Upgraded Electric System of Tomorrow*
QIEA	*Quantum Inspired Evolutionary Algorithm*
QAOA	*Quantum Approximation Optimization Algorithm*
qULA	Unidade lógica-aritmética quântica
qRAM	*Quantum random access memory*
MAPE	*Mean absolute percentage errory*
UCS	Universidade de Caxias do Sul

LISTA DE SÍMBOLOS

i	Unidade imaginária
V	Tensao elétrica
I	Corrente elétrica
P	Potência ativa
Q	Potência reativa
pu	Sistema por unidade
y_{km}	Admitância série
g_{km}	Condutância série
b_{km}	Susceptância série
z_{km}	Impedância série
r_{km}	Resistência série
x_{km}	Reatância série
b_{km}^{sh}	Susceptância shunt
θ	Vetor dos ângulos das tensões nodais
B	Matriz do tipo admitância nodal
p	Vetor das injeções de potência ativa
PV	Barra do tipo geração
PQ	Barra do tipo carga
$\|.\rangle$	Vetor *ket*
$\langle.\|$	Vetor bra
$\mathbb{R}$	Conjunto dos números reais
$\mathbb{C}$	Conjunto dos números complexos
$\mathcal{H}$	Espaço de Hilbert
$\mathbb{I}$	Matriz identidade
λ	Autovalor
$\otimes$	Produto tensorial
$\mathbb{H}$	Hamiltiano
X	Porta Pauli-X
Y	Porta Pauli-Y
Z	Porta Pauli-Z
H	Porta Hadamard
S	Porta de Fase

T	Porta $\pi/8$
$R_x(\phi)$	Porta de rotação em torno dos eixo x
$R_y(\phi)$	Porta de rotação em torno dos eixo y
$R_z(\phi)$	Porta de rotação em torno dos eixo z
$H^{\otimes n}$	Transformação Hadamard
$CNOT$	Porta NÃO-controlado (*Controlled-NOT*)
U	Porta quântica genérica
M	Medida
$\oplus$	Operação XOR
$C^n(U)$	Operação controlada com *n q-bits* de controle
$\|\beta_{xy}\rangle$	Espaço de Bell sintetizado
FT	Operador transformada de Fourier
κ	Número de condição
ϵ	Erro
s	Esparsidade
$\mathcal{O}$	Limite superior da complexidade computacional
Ω	Limite inferior da complexidade computacional
Θ	Limite estreito da complexidade computacional

SUMÁRIO

1 INTRODUÇÃO

A quantidade de dados da rede elétrica coletados pelas concessionárias vem aumentando em um ritmo sem precedentes devido a fatores que exigem uma rede modernizada distribuída, digitalizada e integrada, como uma geração cada vez mais renovável, distribuida e com menor emissão de carbono, o aumento da intensidade e frequência de desastres naturais como consequência de mudanças climáticas, o surgimento de novas tecnologias e a eletrificação de vários setores, como o de transporte. A aquisição de mais dados permite uma melhor observabilidade da rede e uma melhor tomada de decisão, ao mesmo tempo em que exige uma robusta computação de base capaz de convertê-los em informações úteis (ESKANDARPOUR et al., 2020a,b,c).

Uma rede modernizada deve incluir sistemas de larga escala dinâmicos e não-lineares que variam temporariamente entre transitórios e objetivos mais lentos de otimização, apresentando distintas dificuldades matemáticas e computacionais. A otimização do sistema elétrico em larga escala pode exigir esforços computacionais não-triviais. Em problemas de despacho de energia, a complexidade computacional aumenta exponencialmente com o número de variáveis, conforme se observa no estudo de Cafaro e Grossmann (2014), em que os autores criam um modelo para otimizar uma rede de cadeia de abastecimento de gás de xisto com 50 mil variáveis e 50 mil restrições que demanda um tempo computacional entre 15 e 24 horas. Uma otimização que inclua transporte, produção e demanda de energia, geração de energias renováveis e fluxos de informação da internet das coisas pode exigir um esforço computacional que não pode ser suprido pela tecnologia computacional atual (CASCIO; MA; MARÉCHAL, 2020; ESKANDARPOUR et al., 2020c). O mesmo cenário pode ser observado em problemas de análise de segurança do sistema de energia, comumente avaliada considerando prováveis falhas de componentes, que podem exigir a resolução de milhares de cenários de fluxo de energia, cada um considerando uma diferente contingência (ESKANDARPOUR et al., 2020b).

De fato, existem casos em que as técnicas proeminentes desenvolvidas nas décadas anteriores para resolver vários problemas computacionais, como os métodos exatos, as técnicas de otimização numérica e os métodos estocásticos, podem não ser adequados. Uma alternativa promissora para os problemas computacionais enfrentados na construção de uma rede confiável, resiliente e segura é a de um novo paradigma de computação: a computação quântica (ESKANDARPOUR et al., 2020c)

A computação quântica tem seu início na década de 1980. Em 1982, a ideia de um computador quântico surgiu quando o físico Richard Feynman indicou que fenômenos da mecânica quântica poderiam ser aplicados para simular um sistema quântico de forma mais eficaz do que em um computador clássico, para o qual o tempo de simulação cresce exponencialmente (FEYNMAN, 1982). A primeira proposta de um computador quântico ocorreu em 1985, quando David Deutsch descreveu formalmente a primeira máquina de Turing universal quântica (DEUTSCH, 1985), que assim como a máquina de Turing clássica (TURING, 1937), não é ade-

quada para a implementação de algoritmos complexos. O modelo de circuitos e portas lógicas quânticas foi desenvolvido a partir de 1985 e culminou no surgimento do algoritmo de Shor, em 1994 (SHOR, 1994), que resolve em tempo polinomial um problema sem solução eficiente em um computador clássico, a fatoração de números inteiros, potencialmente ameaçando vários dos métodos criptográficos comumente usados, e no algoritmo de Grover em 1996 (GROVER, 1997), que realiza uma busca em uma lista desordenada com um ganho de tempo quadrático em relação à sua versão clássica. Os algoritmos mostraram que a computação quântica tem o potencial teórico para resolver problemas que nenhum computador clássico pode.

A primeira demonstração experimental de um algoritmo quântico foi realizada em 1998 em um computador quântico de dois *q-bits* (a unidade de informação quântica) (CHUANG; GERSHENFELD; KUBINEC, 1998) e desde então, nas últimas décadas, a pesquisa e o desenvolvimento produziram significativos avanços na construção de uma máquina quântica. Entretanto, a construção de computadores quânticos enfrenta enormes problemas de engenharia para correção de erros decorrentes da decoerência quântica e outras fontes de ruído quando o número de *q-bits* aumenta e por conta disso, somente a partir de 2017 e 2018, a construção de computadores quânticos universais com dezenas de *q-bits* foram anunciados por empresas como a IBM, Google e Microsoft (PORTUGAL; MARQUEZINO, 2019; ALMUDÉVER et al., 2017). Em 2017, a IBM anunciou o programa *IBM Q Experience*, uma plataforma baseada em nuvem para experimentação e aprendizado quântico compartilhado (IBM, 2018) e o *Qiskit*, um *kit* de desenvolvimento de *software* de código aberto (ABRAHAM et al., 2021; QISKIT, 2021a,b).

A era atual de computadores quânticos é nomeada NISQ (*Noisy intermedicte-scale quantum*), um termo cunhado por John Preskill para designar o estado da arte atual na fabricação de processadores quânticos (PRESKILL, 2018). Esses processadores são compostos por centenas de *q-bits* ruidosos com tempo de decoerência limitado e não suficientemente sofisticados para se implementar a correção de erro quântico, apresentando consequentemente um desempenho imperfeito em suas operações. Com efeito, a era NISQ se encontra distante do objetivo de um computador quântico universal tolerante a falhas, com milhões de *q-bits* com baixas taxas de erro e longos tempos de coerência que pode efetivamente resolver problemas como a fatoração de números inteiros. Contudo, na exploração do potencial da era NISQ, algoritmos foram propostos em aplicações como a simulação de sistemas quânticos, a simulação de química quântica, a solução de alguns problemas de otimização, a programação linear e a aprendizagem profunda quântica (CALUDE; CALUDE, 2017; PRESKILL, 2018; BHARTI et al., 2021).

Nos últimos anos, alguns trabalhos foram realizados no emergente campo da aplicação da computação quântica ao sistema elétrico e em janeiro de 2022 será iniciada a iniciativa QUEST (*Quantum Upgraded Electric System of Tomorrow*), criada pela Universidade de Denver em colaboração de diversos serviços com o objetivo de construir um consórcio entre universidade e indústria para trazer o poder da computação quântica aos desafios de construir a rede elétrica do futuro (DAVIS, 2021; JONES, 2021).

Implementações de algoritmos evolutivos com inspiração em computação quântica (*Quantum Inspired Evolutionary Algorithm* - QIEA), foram feitas para resolver alguns problemas do sistema de elétrico em Jeong et al. (2009), Lau et al. (2009) e Vlachogiannis e Lee (2008). O artigo de Ajagekar e You (2021) propõe o uso de uma estrutura de aprendizado profundo híbrido baseada em computação quântica para diagnóstico de falhas de sistemas de energia elétrica utilizando como objeto de estudo a simulação de um sistema de energia com 30 barras com grandes variações de falhas de subestação e nas linhas de transmissão, demonstrando aplicabilidade, eficiência e capacidades de generalização. Cascio, Ma e Maréchal (2020), em uma revisão sobre os paradigmas atuais das redes inteligentes (*Smart Grids*), destaca o papel que a criptografia quântica pode desempenhar na segurança cibernética de dados da rede elétrica e também que algumas provas de conceito já foram fornecidas para resolver problemas simplificados como otimização de fluxo de tráfego e otimização de rota para sistemas de transporte multimodal como D-Wave (2021).

Ajagekar e You (2019) descrevem o potencial da computação quântica para impactar o campo da otimização de sistemas de energia discutindo aplicações e técnicas para superar os problemas enfrentados pelos computadores quânticos. O estudo pauta sua discussão na arquitetura de *hardware* de dois sistemas quânticos distintos disponíveis comercialmente fornecendo um exemplo de solução de um problema de otimização com uso de ferramentas de *software* de código aberto. Seguindo esse campo de pesquisa, Koretsky et al. (????) formulam e aplicam um algoritmo híbrido que é a expansão do Algoritmo de Otimização de Aproximação Quântica (*Quantum Approximation Optimization Algorithm* - QAOA) incorporando computação clássica para resolver um problema de otimização importante na indústria de energia elétrica denominado *Unit Commitment*, que visa para satisfazer uma carga de energia alvo visando minimizar o custo operacional com uma série de unidades geradoras de energia sujeitas a restrições.

Eskandarpour et al. (2020c) discutem em profundidade o potencial da aplicação da computação quântica ao sistema elétrico de energia as áreas de otimização, planejamento e logística; *forecasting*; previsão do tempo, *design* de turbinas eólicas; segurança cibernética; segurança da rede e estabilidade da rede. Uma das aplicações abordadas é o fluxo de potência, que consiste em um método de análise numérica do sistema elétrico em condições estáticas para determinação do estado das grandezas elétricas com base na injeção e no consumo de potências ativa ou reativa e nas características física da rede, contendo informações sobre o fluxo de eletricidade em redes de transmissão e de distribuição. O fluxo de potência é de natureza não-linear e é tradicionalmente resolvido através de métodos computacionais iterativos como o método de Newton-Raphson e os métodos desacoplados (MONTICELLI, 1983).

Os artigos sobre fluxo de potência utilizam o algoritmo quântico de Harrow, Hassidim e Lloyd (HHL), introduzido em 2009 (HARROW; HASSIDIM; LLOYD, 2009) para solução de sistemas de equações lineares em tempo polilogarítmico, um aumento exponencial em relação aos melhores algoritmos clássicos conhecidos. Nos anos subsequentes, para comprovar a corretude e eficiência do algoritmo HHL, alguns artigos foram publicados com sua a realização

experimental para sistemas de duas equações, como Barz et al. (2014), Cai et al. (2013) e Pan et al. (2014). Em 2019, uma implementação experimental para um sistema de oito equações, o maior até o momento, foi desenvolvida por Wen et al. (2019).

A primeira solução de um problema de fluxo de potência utilizando computação quântica foi proposta em 2020 por Eskandarpour et al. (2020a). No artigo, os autores realizam a simulação do fluxo de potência CC de um sistema elétrico de três barras. O fluxo de potência CC, utilizado em fases preliminares de estudos nas quais se exige a análise de um grande número de casos, é um método que utiliza algumas aproximações para reduzir o problema do fluxo do potência a um sistema de equações lineares (MONTICELLI, 1983). Uma implementação experimental do fluxo de potência CC para um sistema maior, de nove barras, foi desenvolvida em Eskandarpour et al. (2021). Em 2021, a primeira arquitetura quântica para solução de fluxo de potência por métodos iterativos foi proposta por Feng, Zhou e Zhang (2021), com uma prova de conceito para um sistema de cinco barras. Por fim, Eskandarpour et al. (2020b) apresenta um estudo comparativo baseado em medidas de desempenho computacional comprovadas matematicamente para mostrar o potencial da computação quântica na melhoria de desempenho da rede elétrica, contendo como objeto de estudo o fluxo de potência CC no contexto da análise de segurança do sistema de energia.

Diante do contexto apresentado, realizou-se a simulação de fluxo de potência CC utilizando computação quântica. Dois sistemas foram simulados, um com três e outro com cinco barras, além da experimentação em um computador quântico real para o primeiro, construindo um código no *Qiskit* baseado no algoritmo HHL.

1.1 Objetivo Geral

Realizar a simulação de fluxo de potência CC utilizando computação quântica.

1.2 Objetivos Específicos

- Construir um código no *Qiskit* para solução de equações matriciais baseado no algoritmo HHL.
- Realizar a simulação de fluxo de potência CC de dois sistemas, de três e de cinco barras e a experimentação em um computador quântico real para o sistema de três barras.
- Comparar os resultados obtidos do algoritmo desenvolvido com os resultados clássicos e para o sistema de cinco barras com o fluxo de potência pelos métodos tradicionais iterativos.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será realizada a fundamentação teórica necessária para a elaboração deste trabalho. Na Seção 2.1 será apresentado o problema de fluxo de potência e a sua linearização que culminará no modelo de fluxo de potência CC. Na Seção 2.2 serão apresentados conceitos e noções de álgebra linear elementar, a matemática que fundamenta a mecânica quântica. Na Seção 2.3 serão apresentados conceitos e postulados da mecânica quântica, que por sua vez são a base do funcionamento de um computador quântico. Na Seção 2.4 serão apresentados os fundamentos da computação quântica. Na Seção 2.5 será apresentado o algoritmo quântico de HHL, utilizado para obter a solução de um sistema de equações lineares. Na Seção 2.6 será apresentado o estado da arte da computação quântica no problema do fluxo de potência.

2.1 FLUXO DE POTÊNCIA

A operação confiável de um sistema elétrico depende de muitos problemas baseados no estudo de fluxo de potência. O fluxo de potência consiste em um método de análise numérica do sistema elétrico em condições estáticas para determinação do estado das grandezas elétricas com base na injeção e no consumo de potências ativa ou reativa, contendo informações sobre o fluxo de eletricidade em redes de transmissão e de distribuição. Problemas de fluxo de potência são resolvidos em contextos diversos, como a cada poucos minutos para problemas de controle e operação ou a cada ano para problemas de manutenção e planejamento (ESKANDARPOUR et al., 2020a; MONTICELLI, 1983).

Nos problemas de fluxo de potência, a modelagem do sistema elétrico é representada por um conjunto de equações e inequações algébricas que dependem das características físicas da rede (ESKANDARPOUR et al., 2020a). A análise não leva em conta os efeitos transitórios, sendo portanto utilizada em sistemas nos quais as variações no tempo são suficientemente lentas. Os transitórios só podem ser devidamente levados em conta através de uma modelagem dinâmica envolvendo equações diferenciais (MONTICELLI, 1983).

O problema do fluxo de potência é inerentemente não-linear e costuma ser resolvido através de métodos computacionais iterativos, como o método de Newton-Raphson e os métodos desacoplados. Contudo, aproximações podem ser feitas através de um modelo simplificado chamado fluxo de potência CC, que estima a distribuição do fluxo de potência ativa em uma rede de transmissão com baixo custo computacional e precisão aceitável para determinadas aplicações (MONTICELLI, 1983).

2.1.1 Formulação básica

O fluxo de potência é formulado por um sistema de equações e inequações não-lineares algébricas, que correspondem as leis de Kirchhoff e a um conjunto de restrições operacionais da rede elétrica e de seus componentes (MONTICELLI, 1983). A formulação básica necessária para desenvolvimento do modelo de fluxo de potência CC, descrita por Monticelli (1983), será apresentada nesta subseção.

2.1.1.1 Definição do problema

A cada barra *k* da rede são associadas quatro variáveis, sendo duas delas como dados do problemas e duas como incógnitas:

- V_k - magnitude da tensão nodal
- θ_k - ângulo da tensão nodal
- P_k - geração líquida de potência ativa
- Q_k - injeção líquida de potência reativa.

São definidos três tipos de barras:

- PQ - onde são dados P_k e Q_k e são calculados V_k e θ_k
- PV - onde são dados P_k e V_k e são calculados Q_k e θ_k
- $P\theta$ ou de referência - onde são dados V_k e θ_k e são calculados P_k e Q_k

As barras PQ e PV são utilizadas, respectivamente, para barras de carga e barras de geração. A barra $P\theta$, além da função de fornecer a referência angular do sistema (a referência de tensão é o próprio nó terra), é utilizada para fechar o balanço de potência do sistema, levando em conta perdas de transmissão não conhecidas antes de se obter a solução final do problema. Cabe ressaltar que existem outros tipos de barras, como PQV, P e V, que aparecem em situações particulares, como o intercâmbio de uma área e o controle de magnitude de uma barra remota, que não são considerados na formulação básica, e consequentemente, no modelo de fluxo de potência CC.

O conjunto de equações de fluxo de potência é formado por duas equações para cada barra, cada uma respeitando a primeira lei de Kirchhoff, isto é, potências ativas e reativas são iguais à soma dos fluxos correspondentes que deixam a barra através de linhas de transmissão. Matematicamente, as equações relacionadas as potências ativa (P_k) e reativa (Q_k) de uma barra genérica *k* são:

$$\begin{aligned} P_k &= \sum_{m\in\Omega_k} = P_{km}(V_k, V_m, \theta_k, \theta_m) \\ Q_k + Qk^{sh}(V_k) &= \sum_{m\in\Omega_k} = Q_{km}(V_k, V_m, \theta_k, \theta_m) \end{aligned} \tag{2.1.1}$$

em que $(k-m)$ correspondem aos ramos ligados à barra *k*, Ω_k representa o conjunto de barras vizinhas da barra e $Q_k^{sh}(V_k)$ corresponde a componente de potência reativa devido ao elemento capacitivo em parelelo (*shunt*) da barra *k*.

Além disso, o conjunto de inequações que fazem parte do problema de fluxo de potência é formado, entre outras (que não serão discutidas), pelas restrições de magnitude das tensões nodais das barras *PQ* e pelos limites de injeção de potência reativa das barras *PV*:

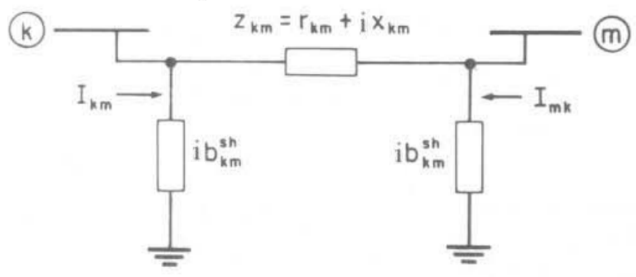
$$\begin{aligned} V_k^{min} \leq V_k \leq V_k^{max} \\ Q_k^{min} \leq Q_k \leq Q_P^{max} \end{aligned} \tag{2.1.2}$$

2.1.1.2 Modelagem de linhas e transformadores

O modelo equivalente π de uma linha de transmissão é mostrado na Figura 1 e é definido pelos parâmetros: a resistência série r_{km} e a reatância série x_{km}, que somadas formam a impedância série z_{km} e a susceptância *shunt* b_{km}^{sh}. A admitância série y_{km}, formada pela soma da condutância série g_{km} e a susceptância série b_{km}, é calculada por:

$$y_{km} = g_{km} + ib_{km} = z_{km}^{-1} = \frac{r_{km}}{r_{km}^2 + x_{km}^2} - i\frac{x_{km}}{r_{km}^2 + x_{km}^2} \tag{2.1.3}$$

Figura 1 – Modelo equivalente π de uma linha de transmissão.



Fonte: Adaptado de Monticelli (1983).

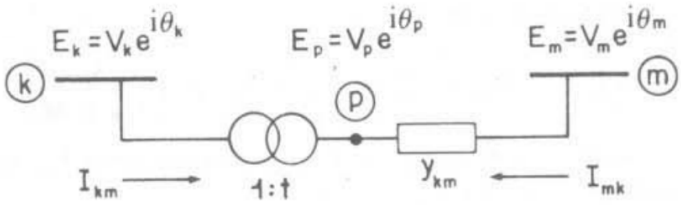
Dado que E_k e E_m sejam as tensões nas barras tais que $E_m = V_k e^{i\theta_k}$ e $E_m = V_m e^{i\theta_m}$, as correntes I_{km} e I_{mk} são calculadas pelas expressões:

$$\begin{aligned} I_{km} &= y_{km}(E_k - E_m) + ib_{km}^{sh} E_k \\ I_{mk} &= y_{km}(E_m - E_k) + ib_{km}^{sh} E_m \end{aligned} \tag{2.1.4}$$

O modelo geral para transformadores (em fase e defasadores) consiste na admitância série y_{km} e um auto-transformador ideal com relação de transformação $1:t$. Para o transformador

em fase, t é um número real $t = a$ e para o defasador, t é um número complexo $t = ae^{i\phi}$. O modelo é mostrado na Figura 2.

Figura 2 – Modelo equivalente de transformador.



Fonte: Adaptado de Monticelli (1983).

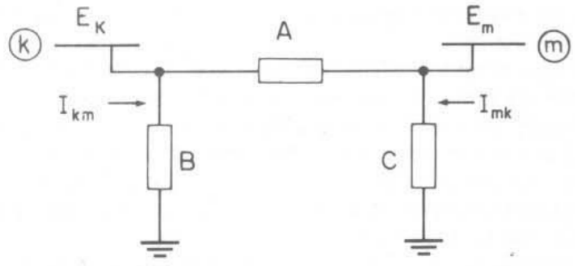
Para o transformador em fase, o circuito equivalente tem a mesma topologia que o modelo das linhas de transmissão, ilustrado na Figura 3, em que:

$$\begin{aligned} A &= ay_{km} \\ B &= a(a-1)y_{km} \\ C &= (1-a)y_{km} \end{aligned} \tag{2.1.5}$$

Para a = 1, o modelo se reduz apenas a admitância série y_{km}. As correntes I_{km} e I_{mk} são portanto:

$$\begin{aligned} I_{km} &= (A+B)E_k + (-A)E_m \\ I_{mk} &= (-A)E_k + (A+C)E_m \end{aligned} \tag{2.1.6}$$

Figura 3 – Modelo equivalente π para transformadores em fase.



Fonte: Adaptado de Monticelli (1983).

O transformador defasador é um elemento que permite que se controle o fluxo de potência ativa no ramo no qual ele está inserido. Considerando um defasador puro, isto é, que não afeta as relações entre as tensões V_k e V_m, de modo que $a = 1$. As correntes I_{km} e I_{mk} podem ser escritas, definindo $t^* = e^{-j\phi}$, como:

$$\begin{aligned} I_{km} &= -t^* y_{km}(E_m - E_p) = (y_{km})E_k + (-t^* y_{km})E_m \\ I_{mk} &= y_{km}(E_m - E_p) = (-t y_{km})E_k + (y_{km})E_m \end{aligned} \tag{2.1.7}$$

O modelo de um defasador no qual $a \neq 0$ é constituido por um transformador em fase ($t = a$) em série com um defasador puro ($t = ae^{i\phi}$).

2.1.1.3 Fluxos de potência ativa e reativa

Através de manipulação algébrica pode-se chegar em expressões gerais para os fluxos de potência ativa e reativa para linhas de transmissão, transformadores em fase e defasadores:

$$\begin{aligned} P_{km} = (a_{km}V_k)^2 g_{km} - (a_{km}V_k)V_m g_{km} cos(\theta_{km} + \phi_{km}) + \\ -(a_{km}V_k)V_m b_{km} sen(\theta_{km} + \phi_{km}) \\ Q_{km} = -(a_{km}V_k)^2 (b_{km} - b_{km}^{sh}) + (a_{km}V_k)V_m b_{km} cos(\theta_{km} - \phi_{km}) + \\ -(a_{km}V_k)V_m g_{km} sen(\theta_{km} - \phi_{km}) \end{aligned} \tag{2.1.8}$$

No caso de linhas de transmissão, $a_{km} = 1$ e $\phi_{km} = 0$, e as expressões se reduzem para:

$$\begin{aligned} P_{km} = V_k^2 g_{km} - V_k V_m g_{km} cos\ \theta_{km} - V_k V_m b_{km} sen\ \theta_{km} \\ Q_{km} = -V_k^2 (b_{km} - b_{km}^{sh}) + V_k V_m b_{km} cos\ \theta_{km} - V_k V_m g_{km} sen\ \theta_{km} \end{aligned} \tag{2.1.9}$$

Para transformadores em fase, $b_{km}^{sh} = 0$ e $\phi_{km} = 0$, e a expressão se torna:

$$\begin{aligned} P_{km} = (a_{km}V_k)^2 g_{km} - (a_{km}V_k)V_m g_{km} cos\ \theta_{km} - (a_{km}V_k)V_m b_{km} sen\ \theta_{km} \\ Q_{km} = -(a_{km}V_k)^2 b_{km} + (a_{km}V_k)V_m b_{km} cos\ \theta_{km} - (a_{km}V_k)V_m g_{km} sen\ \theta_{km} \end{aligned} \tag{2.1.10}$$

Por fim, para o defasadores puros, $a_{km} = 1$ e $b_{km}^{sh} = 0$, e as expressões se reduzem para:

$$\begin{aligned} P_{km} = V_k^2 g_{km} - V_k V_m g_{km} cos(\theta_{km} + \phi_{km}) - V_k V_m b_{km} sen(\theta_{km} + \phi_{km}) \\ Q_{km} = -V_k^2 b_{km} + V_k V_m b_{km} cos(\theta_{km} - \phi_{km}) - V_k V_m g_{km} sen(\theta_{km} - \phi_{km}) \end{aligned} \tag{2.1.11}$$

2.1.2 Fluxo de potência CC

O fluxo de potência ativa em uma linha de transmissão é aproximadamente proporcional à abertura angular na linha e se desloca no sentido dos ângulos menores para os ângulos maiores, uma relação análoga à relação entre os fluxos de corrente e a queda de tensão em um circuito em corrente contínua (lei de Ohm). Essa propriedade permite o desenvolvimento do modelo de fluxo de potência CC, descrito por Monticelli (1983) e que será apresentado nesta subseção.

O fluxo de potência CC apresenta melhores resultados para tensões maiores, não sendo aplicável em sistemas de distribuição em baixa tensão, nos quais o fluxo de potência ativa depende de forma significativa das quedas de tensão (MONTICELLI, 1983).

O modelo CC não leva em conta a magnitude das tensões nodais, as potências reativas e o

taps [1] dos transformadores, não sendo aplicável em todos os casos. Esse possui grande utilidade em fases preliminares de estudos nas quais se exige a análise de um grande número de casos, que dificilmente pode ser feito através dos métodos convencionais. Nas fases subsequentes dos estudos, pode ser necessário recorrer aos métodos clássicos de fluxo de potência, como o de Newton ou os desacoplados (MONTICELLI, 1983).

Uma importante característica do modelo linearizado é o fato dele fornecer uma solução mesmo para problemas que não poderiam ser resolvidos pelos métodos convencionais de fluxo de potência, o que ocorre frequentemente em estudos de planejamento quando testam-se acréscimos de carga ou de geração. Nesses estudos, problemas de convergência nos programas de fluxo de potência são observados e o modelo CC pode fornecer um indicativo do que está ocorrendo com a rede (MONTICELLI, 1983).

2.1.2.1 Linearização

Considerando o fluxo de potência ativo em uma linha de transmissão P_{km} dado pela Equação 2.1.9, o fluxo no extremo oposto da linha, P_{mk}, de maneira análoga, é dado por:

$$P_{mk} = V_m^2 g_{km} - V_k V_m g_{km} cos\ \theta_{km} + V_k V_m b_{km} sen\ \theta_{km} \tag{2.1.12}$$

A perda de potência ativa é dada pela soma de P_{km} e P_{mk}:

$$P_{km} + P_{mk} = g_{km}(V_m^2 + V_k^2 - V_k V_m cos\ \theta_{km}) \tag{2.1.13}$$

Se os termos das perdas forem desprezados nas Equações 2.1.9 e 2.1.12, P_{km} e P_{mk} são reduzidos para:

$$P_{km} = -P_{mk} = -V_k V_m b_{km} sen\ \theta_{km} \tag{2.1.14}$$

As seguintes aproximações ainda são introduzidas na Equação 2.1.14:

$$\begin{aligned} V_k &\cong V_m \cong 1pu \\ sen\ \theta_{km} &\cong \theta_{km} \\ b_k &\cong -\frac{1}{x_{km}} \end{aligned} \tag{2.1.15}$$

resultando em:

$$P_{km} = x_{km}^{-1}\theta_{km} = \frac{\theta_k - \theta_m}{x_{km}} \tag{2.1.16}$$

A equação tem o mesmo formato da lei de Ohm aplicada a um resistor percorrido por cor-

[1] elemento que varia a relação entre o número de enrolamentos do primário e secundário através de uma chave para controle de tensão

rente contínua, sendo P_{km} análogo à corrente, θ_k e θ_m análogos às tensões terminais e x_{km} análogo à resistência elétrica, razão pela qual o modelo é conhecido como fluxo de potência CC.

Aplicando de maneira análoga as mesmas aproximações para o caso de um transformador em fase, chega-se na expressão:

$$P_{km} = a_{km} x_{km}^{-1} \theta_{km} \tag{2.1.17}$$

sendo que a_{km} ainda pode ser aproximado por $a_{km} \cong 1$, reduzindo a expressão do fluxo no transformador para a mesma expressão válida para linhas de transmissão.

Já para o caso dos defasadores puros, chega-se na expressão:

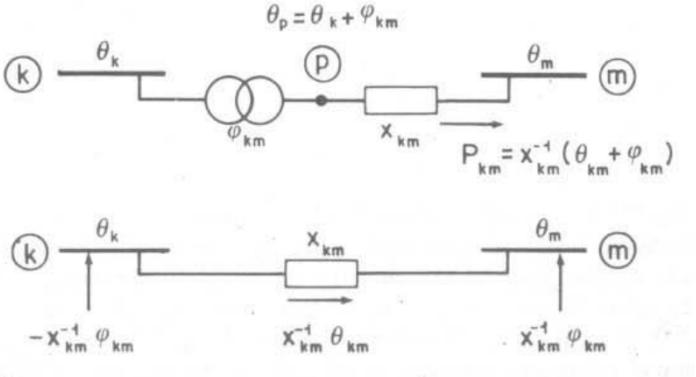
$$P_{km} = x_{km}^{-1} sen(\theta_{km} - \phi_{km}) \tag{2.1.18}$$

Os ângulos θ_{km} e ϕ_{km} podem ter, individualmente, valores elevados, entretanto, a abertura angular $(\theta_{km} + \phi_{km})$ é da mesma ordem que a abertura existente nas linhas de transmissão e transformadores em fase, de modo que pode ser feita a aproximação para:

$$P_{km} = x_{km}^{-1} (\theta_{km} - \phi_{km}) \tag{2.1.19}$$

Considerando ϕ_{km} constante, a expressão pode ser representada pelo modelo linearizado da Figura 4, onde $x_{km}^{-1}\phi_{km}$ aparece como uma carga adicional na barra *k* e uma geração adicional na barra *m* (ou vice-versa para ϕ_{km} negativo).

Figura 4 – Modelo linearizado de um defasador.



Fonte: Monticelli (1983).

2.1.2.2 Formulação matricial

O modelo linearizado, que consiste em um sistema de equações lineares, pode ser expresso por uma equação matricial. Para simplicidade de exposição, seja uma rede de transmissão sem

transformadores em fase ou defasadores com *NB* barras. Neste caso, os fluxos de potência ativa nos ramos da rede são dados por:

$$P_{km} = x_{km}^{-1}\theta_{km} \tag{2.1.20}$$

em que x_{km} é a reatância em paralelo que existe nos ramos $k-m$. A injeção de potência ativa na barra *k* é igual à soma dos fluxo que saem da barra, isto é:

$$P_k = \sum_{m\in\Omega_k} x_{km}^{-1}\theta_{km} = \left(\sum_{m\in\Omega_k} x_{km}^{-1}\right)\theta_k + \sum_{m\in\Omega_k}\left(-x_{km}^{-1}\theta_m\right) \ , \quad (k = 1...NB) \tag{2.1.21}$$

Expressando na forma matricial:

$$p = B\theta \tag{2.1.22}$$

sendo p o vetor das injeções de potência ativa, θ o vetor dos ângulos das tensões nodais θ_k e B uma matriz do tipo admitância nodal e cujos elementos são:

$$\begin{aligned} B_{km} &= x_{km}^{-1} \\ B_{kk} &= \sum_{m\in\Omega_k} x_{km}^{-1} \end{aligned} \tag{2.1.23}$$

Para resolver o problema, uma das equações do sistema deve ser eliminada adotando-se a barra correspondente como referência angular ($\theta_k = 0$), de forma que o sistema passa a ter dimensão *(NB - 1)*, e os ângulos das *(NB - 1)* barras restantes devem ser determinados resolvendo o sistema de equações lineares.

Transformadores em fase e defasadores são tratados da mesma forma na formação da matriz B e as injeções de potência ativa do modelo de desafadores, indicada na Figura 4, devem ser levadas em consideração na formação do vetor p.

2.2 Álgebra Linear

A álgebra linear é um ramo da matemática que estuda campos vetoriais e as operações lineares neles realizadas, sendo utilizada como formulação matemática para a mecânica quântica. O espaço vetorial de interesse no contexto da mecânica quântica é o $\mathbb{C}^n$, que é o conjunto de todas as *n-uplas* de números complexos. Os elementos de um espaço vetorial são chamados de vetores e são comumente representados por uma matriz-coluna, admitindo as propriedades de soma e multiplicação por escalar complexo (NIELSEN; CHUANG, 2010).

A notação padrão da mecânica quântica para a abordagem da álgebra linear é a notação de Dirac, também chamada de notação *bra-ket* , onde $|\psi\rangle$ é um rótulo para o vetor (comumente ψ ou ϕ) denominado *ket*, sendo um vetor coluna, enquanto $\langle\psi|$, o conjugado transposto de $|\psi\rangle$, é

denominado *bra*, sendo portanto, um vetor linha. O vetor *bra* também é chamado de vetor dual de *ket* (NIELSEN; CHUANG, 2010).

Nesta seção, a maioria dos conceitos e definições apresentados são desenvolvidos por Nielsen e Chuang (2010).

2.2.1 Bases e Independência linear

Um conjunto gerador de um espaço vetorial é um conjunto de vetores $|\psi_1\rangle$,...,$|\psi_n\rangle$, tal que qualquer vetor pode ser escrito como uma combinação linear:

$$|\psi\rangle = \sum_i a_i |\psi_i\rangle \tag{2.2.1}$$

em que a_1,..., $a_n \in \mathbb{C}$. Em geral, um espaço vetorial pode ter muitas bases diferentes.

Um conjunto de vetores não-nulos $|\psi_1\rangle$,...,$|\psi_n\rangle$ é dito linearmente dependente se existir um conjunto a_1,..., $a_n \in \mathbb{C}$ com pelo menos um $a_i \neq 0$ que satisfaça a equação vetorial:

$$a_1|\psi_1\rangle + a_2|\psi_2\rangle + ... + a_n|\psi_n\rangle = 0 \tag{2.2.2}$$

Os vetores do conjunto serão linearmente independentes se a equação admitir apenas a solução trivial.

Um subconjunto de vetores $|\psi_1\rangle$,...,$|\psi_n\rangle$ linearmente independentes é uma base para espaço vetorial *V*, de dimensão *n*. Todo espaço vetorial finito admite uma base.

2.2.2 Operadores lineares e matrizes

Um operador linear *A* é uma aplicação entre dois espaços vetoriais *V* e *W*, *A: V* $\rightarrow$ *W*, que preserva a propriedade de adição e multiplicação por escalar, isto é:

$$A\left(\sum_i a_i|\psi_i\rangle\right) = \sum_i a_i A|\psi_i\rangle \tag{2.2.3}$$

Um operador linear *A* é dito como sendo definido no espaço vetorial *A* quando *A: V* $\rightarrow$ *V*. Um importante operador linear é o operador identidade $\mathbb{I}_v$, definido pela equação $\mathbb{I}_v|\psi\rangle = |\psi\rangle$ para todo $|\psi\rangle$.

Operadores lineares podem ser descritos pela sua representação matricial. Seja uma matriz complexa *A*, de dimensão *m* por *n*, composta por elementos A_{ij}, que transforma vetores de $\mathbb{C}^n$ para $\mathbb{C}^m$ pela multiplicação de A por um vetor de $\mathbb{C}^n$. Supondo que *A: V* $\rightarrow$ *W*, que $|\psi_1\rangle, ..., |\psi_m\rangle$ seja uma base de *V* e que $|\phi_1\rangle, ..., |\phi_n\rangle$ seja uma base de *W*, a representação matricial de *A* é dada por números complexos A_{1j}até Anj, para cada *j (1 < j < m)*, tais que:

$$A|\psi_j\rangle = \sum_i A_{ij}|\phi_i\rangle \tag{2.2.4}$$

2.2.3 Produto interno e espaço de Hilbert

O produto interno é uma função que toma dois vetores de entrada e produz como imagem um número complexo, sendo denotado na notação de Dirac como $\langle\psi|\phi\rangle$, que para o conjunto $\mathbb{C}$ deve satisfazer os seguintes axiomas:

- linearidade pelo argumento, $\langle\psi|\sum\lambda_i\phi_i\rangle = \sum\lambda_i\,\langle\psi_i|\phi_i\rangle$
- $\langle\psi|\phi\rangle = (\langle\psi|\phi\rangle)^*$
- $\langle\psi|\phi\rangle \geq 0$, com $\langle\psi|\phi\rangle\ = 0$ se e somente se $|\psi\rangle$ = 0.

Um espaço vetorial com produto interno é dito espaço de produto interno. Em mecânica quântica, o formalismo apropriado para o estudo de seus conceitos é fornecido pelo espaço de Hilbert, que não precisa estar restrito a um número finito de dimensões. Para a computação quântica, o interesse está nos espaços vetoriais complexos de dimensão finita, nos quais o espaço de Hilbert é exatamente a mesma coisa que um espaço de produto interno. Um espaço de Hilbert, denotado por $\mathcal{H}$, é um espaço vetorial complexo normado com produto interno e munido de métrica (ZINGER, 2015).

A norma de um vetor no espaço de Hilbert provém do produto interno, sendo calculada por:

$$|||\psi\rangle|| = \sqrt{\langle\psi|\psi\rangle} \tag{2.2.5}$$

Dois vetores são ditos ortogonais se o seu produto interno for igual a zero. Um vetor unitário, também denominado normalizado, é aquele para o qual a norma é igual a um. Um conjunto de vetores é dito ortonormal se todos vetores distintos do conjunto forem simultâneamente ortogonais e normalizados.

A métrica d entre dois vetores no espaço de Hilbert, $|\psi\rangle$ e $|\phi\rangle$, é definida através da norma:

$$d(\psi,\ \phi) = |||\psi\rangle - |\phi\rangle|| = \sqrt{\langle\psi-\phi|\psi-\phi\rangle} \tag{2.2.6}$$

Por ser munido de norma, o espaço de Hilbert pode ser dito como sendo um espaço métrico. O produto interno para um espaço de Hilbert definido no espaço vetorial complexo para dois vetores genéricos $|\psi\rangle = \sum\psi_j|j\rangle$ e $|\phi\rangle = \sum\phi_k|k\rangle$, é calculado por:

$$\langle\psi|\phi\rangle = \begin{bmatrix}\psi_1^* & \cdots & \psi_n^*\end{bmatrix}\begin{bmatrix}\phi_1\\ \vdots\\ \phi_n\end{bmatrix} \tag{2.2.7}$$

Outra importante operação é a do produto externo, denotada por $|\psi\rangle\langle\phi|$, que segundo Barbosa (2005) é calculada por:

$$|\psi\rangle\langle\phi| = \begin{bmatrix} \psi_1 \\ \vdots \\ \psi_n \end{bmatrix} \begin{bmatrix} \phi_1^* & \cdots & \phi_n^* \end{bmatrix} = \begin{bmatrix} \psi_1\phi_1^* & \cdots & \psi_1\phi_n^* \\ \vdots & \ddots & \vdots \\ \psi_n\phi_1^* & \cdots & \psi_n\phi_n^* \end{bmatrix} \tag{2.2.8}$$

Adicionalmente, uma importante função matricial é o traço da matriz, definida como a soma dos elementos da sua diagonal.

$$tr(A) = \sum_i A_{ii} \tag{2.2.9}$$

2.2.4 Autovetores e autovalores

Um autovetor de um operador linear *A* em um espaço de Hilbert $\mathcal{H}$ é um vetor $|v\rangle$ tal que

$$A|v\rangle = \lambda|v\rangle \tag{2.2.10}$$

sendo λ um número complexo conhecido como autovalor de *A*, correspondente ao autovetor $|v\rangle$. Os autovalores de uma matriz *A* são determinados pela solução da equação característica:

$$det|A - \lambda\mathbb{I}| = 0 \tag{2.2.11}$$

O autoespaço que corresponde a um autovalor λ é o conjunto de todos os autovetores correspondes aos autovalores λ, sendo um subespaço vetorial do espaço vetorial onde *A* atua.

O operador *A* pode ser representado através de sua representação diagonal:

$$A = \sum_j \lambda_i |j\rangle\langle j| \tag{2.2.12}$$

em que vetores $|j\rangle$ formam um conjunto de autovetores ortogonais de *A* com os seus autovalores correspondentes λ_j. Um operador é dito diagonalizável se possuir uma representação diagonal.

2.2.5 Operador hermitiano e operador unitário

Seja *A* um operador linear em um espaço vetorial $\mathcal{H}$, existirá em $\mathcal{H}$ um único operador linear único $A^\dagger$, conhecido como adjunto ou conjugado hermitiano de *A*, tal que para todos os vetores $|\psi\rangle$, $|\phi\rangle \in \mathcal{H}$,

$$\langle A^\dagger\phi|\psi\rangle = \langle\phi|A\psi\rangle \tag{2.2.13}$$

Se $|\psi\rangle$ for um vetor em $\mathcal{H}$, então $|\psi\rangle^\dagger$ = $\langle\psi|$, e portanto, $(A|\psi\rangle)^\dagger$ = $\langle\psi|A^\dagger$.

A operação de conjugação hermitiana leva a matriz *A* para a sua conjugada transposta, ou seja, $A^\dagger = (A^*)^T$. Um operador A cujo adjunto é o próprio A é chamado de hermitiano ou auto-adjunto. Uma importante propriedade de operadores hermitianos é a de admitirem apenas autovalores reais (GRIFFITHS, 2011).

Um operador é dito normal se $AA^\dagger = A^\dagger A$. Um operador hermitiano é um operador normal. Um operador é normal se e somente se ele for diagonalizável. Por conseguinte, todo operador hermitiano é diagonalizável.

Uma classe importante de operadores normais são os chamados operadores unitários, nos quais se satisfaz a condição $U^\dagger U = UU^\dagger = \mathbb{I}$. Operadores unitários possuem a propriedade de preservar o produto interno entre vetores. Isto é, para dois vetores $|\psi\rangle$ e $|\phi\rangle$, o produto interno entre U$|\psi\rangle$ e U$|\phi\rangle$ é o mesmo que entre $|\psi\rangle$ e $|\phi\rangle$.

$$\langle\psi|U^\dagger U|\phi\rangle = \langle\psi|\mathbb{I}|\phi\rangle = \langle\psi|\phi\rangle \tag{2.2.14}$$

2.2.6 Número de condição

Segundo Zinger (2015), o número de condição de uma função mede quanto a saída de tal função pode variar a partir de uma pequena diferença na entrada. Portanto, está relacionado com a sensibilidade de certa função à entrada. O número de condição pode ou não ser finito e é comumente dado pela razão entre o maior e o menor valor singular do operador analisado. Para um operador hermitiano, o número de condição será dado por:

$$\kappa(A) = \left|\frac{\lambda_{max}(A)}{\lambda_{min}(A)}\right| \tag{2.2.15}$$

sendo $\lambda_{max}(A)$ e $\lambda_{min}(A)$, respectivamente, em módulo, o maior e o menor autovalor de A.

2.2.7 Produto tensorial

O produto tensorial é uma forma de se juntar espaços vetoriais para formar espaços vetoriais maiores, um procedimento utilizado em mecânica quântica para descrever sistemas com muitas partículas (NIELSEN; CHUANG, 2010).

Supondo que $\mathcal{H}_1$ e $\mathcal{H}_2$ sejam espaços de Hilbert de dimensão *m* e *n*, respectivamente. O produto tensorial, representado por $\mathcal{H}_1 \otimes \mathcal{H}_1$, é um espaço vetorial de dimensão mn. Os elementos deste espaço vetorial são combinações lineares de produtos tensoriais $|\psi\rangle \otimes |\phi\rangle$ dos elementos $|\psi\rangle$ de $\mathcal{H}_1$ e $|\phi\rangle$ de $\mathcal{H}_2$. Se $|i\rangle$ e $|j\rangle$ são bases ortonomais de $\mathcal{H}_1$ e $\mathcal{H}_2$, respectivamente, então o produto tensorial $|i\rangle \otimes |j\rangle$ é uma base para $\mathcal{H}_1 \otimes \mathcal{H}_2$. A notação para o produto tensorial frequentemente é abreviada para $|\psi\rangle|\phi\rangle$, $|\psi, \phi\rangle$ ou ainda $|\psi\phi\rangle$. Todas as noções sobre operador adjunto, matriz unitaria, normalidade e hermiticidade são extentidas para $\mathcal{H}_1 \otimes \mathcal{H}_2$.

A representação matricial do produto tensorial é conhecida como produto de Kronecker. Seja *A* uma matriz de *m x n*, e *B* uma matriz *p x q*, o produto tensorial $A \otimes B$ é representado por:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{21}B & A_{22}B & \cdots & A_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}B & A_{m2}B & \cdots & A_{mn}B \end{bmatrix} \tag{2.2.16}$$

A matriz resultante tem dimensões *nq x mp*. Termos como A_{11} denotam submatrizes *p x q* cujos elementos são proporcionais aos da matriz *B*. Conforme exemplifica Barbosa (2005), para o caso da multiplicação de duas matrizes 2x2, o resultado é uma matriz 4x4.

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes \begin{bmatrix} x & y \\ v & w \end{bmatrix} = \begin{bmatrix} aB & bB \\ cB & dB \end{bmatrix} = \begin{bmatrix} ax & ay & bx & by \\ av & aw & bv & bw \\ cx & cy & dx & dy \\ cv & cw & dv & dw \end{bmatrix} \tag{2.2.17}$$

2.2.8 Matrizes esparsas

Uma matriz cuja maioria dos elementos é igual zero é chamada de esparsa. A fração de elementos com valor nulo é denominada esparsidade (ZINGER, 2015). Uma matriz é chamada *s-esparsa* se tiver no máximo *s* elementos diferentes de zero por linha ou coluna (ESKANDARPOUR et al., 2020a).

A matriz a seguir, por exemplo, tem dimensão 4×4 apesar de apresentar somente 4 elementos diferentes de zero, sendo 1-esparsa.

$$\begin{bmatrix} 0 & 8 & 0 & 0 \\ 5 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix} \tag{2.2.18}$$

Armazenar e manipular matrizes esparsas em um computador pode ser feito de forma mais eficiente através de algoritmos especializados que se beneficiam da estrutura vazia da matriz (ZINGER, 2015).

2.3 MECÂNICA QUÂNTICA

A mecânica quântica é uma estrutura matemática e conceitual para o desenvolvimento de teorias físicas. Os postulados da mecânica quântica fornecem a conexão entre o mundo físico e o formalismo da mecânica quântica (NIELSEN; CHUANG, 2010). Os conceitos e postulados de mecânica quântica serão descritos nesta seção, para fornecer a base necessária para o estudo da computação quântica.

2.3.1 Princípio da superposição

Um sistema quântico pode se encontrar simultaneamente em vários estados, em um fenômeno chamado superposição. Em mecânica quântica, cada estado possível é descrito por uma função de onda ψ, que não é uma onda com significado físico, mas uma onda de probabilidade. A probabilidade do sistema estar no estado ψ é $|\psi|^2$, um valor sempre real e positivo. $|\psi|^2$ é chamado de densidade de probabilidade (NIELSEN; CHUANG, 2010; HALLIDAY; WALKER; RESNICK, 2008).

2.3.2 Espaço de estados

A qualquer sistema físico isolado existe associado um espaço vetorial complexo com produto interno (espaço de Hilbert), conhecido como espaço de estados do sistema. O sistema é complemente descrito pelo seu vetor de estado, um vetor unitário no espaço de estados (NIELSEN; CHUANG, 2010).

2.3.3 Evolução de Schrödinger

A evolução de um sistema quântico fechado é descrita por uma transformação unitária, isto é, o estado $|\psi\rangle$ do sistema no tempo t_1 está relacionado ao estado $|\psi'\rangle$ do tempo t_2 através de um operador unitário U que depende somente dos tempos t_1 e t_2, tal que $|\psi'\rangle = U|\psi\rangle$ (NIELSEN; CHUANG, 2010).

A extensão para o tempo contínuo da definição da evolução de um sistema quântico é descrita pela equação de Schrödinger:

$$\frac{id\hbar|\psi\rangle}{dt} = \mathbb{H}|\psi\rangle \tag{2.3.1}$$

sendo $\hbar$ a constante de Planck, determinada experimentalmente (cujo valor pode ser ignorado para os propósitos deste trabalho) e $\mathbb{H}$ um operador hermitiano conhecido como hamiltoniano do sistema. O operador hamiltoniano é um operador que corresponde a energia total do sistema e o espectro hamiltoniano é o conjunto de possíveis resultados quando se mede a energia total do sistema (GRIFFITHS, 2011; NIELSEN; CHUANG, 2010).

O operador hamiltoniano por ser um operador hermitiano possui sua decomposição espectral:

$$\mathbb{H} = \sum_{E} E|E\rangle\langle E| \tag{2.3.2}$$

Os estados $|E\rangle$ são autovetores normalizados convencionalmente chamados de auto-estados de energia e E são os autovalores que correspondem à energia do estado $|E\rangle$ (NIELSEN; CHUANG, 2010).

A equação de Schrödinger para um Hamiltoniano que é independente do tempo pode ser reduzida para:

$$\mathbb{H}|\psi\rangle = E|\psi\rangle \tag{2.3.3}$$

O operador hamiltoniano é um operador observável. Em mecânica quântica, um operador observável é definido como um operador que aplicado ao estado de um sistema produz como autovalores os possíveis valores de alguma grandeza passível de mensuração. Todos os operadores observáveis são hermitianos. Resolver a equação de Schrödinger consiste em encontrar os autovalores e auto-estados do operador hamiltoniano do sistema. Os autovalores de um dado operador hamiltoniano representam as grandezas físicas observáveis permitidas, e portanto conforme propriedade supracitada dos operadores hermitianos devem assumir apenas valores reais (GRIFFITHS, 2011).

2.3.4 Medidas quânticas

Um sistema quântico fechado evolui através de operadores unitários e para que uma medida seja realizada, um sistema físico externo, não necessariamente descrito por um operador unitário, deve acabar com o isolamento do sistema (NIELSEN; CHUANG, 2010).

Medições quânticas são descritas por uma coleção de operadores de medição M_m que atuam no espaço de estados sendo medido. Seja $|\psi\rangle$ o estado de um sistema quântico, a probabilidade de que o resultado *m* ocorra é $p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle$. O conjunto de operadores de medida deve satisfazer a condição $\sum p(m) = 1$ (NIELSEN; CHUANG, 2010).

Conforme explica Nielsen e Chuang (2010), uma classe de medidas de interesse em computação quântica é conhecida como medidas projetivas. Uma medida projetiva é descrita por um operador observável *M* com decomposição espectral:

$$M = \sum_{m} mP_m \tag{2.3.4}$$

em que P_m é a projeção no autoespaço de *M* associado ao autovalor *m*. Após a medição do estado $|\psi\rangle$, a probabilidade de se obter o resultado *m* será $p(m) = \langle\psi|P_m|\psi\rangle$ e o estado do sistema após a medida será:

$$\frac{Pm|\psi\rangle}{\sqrt{p(m)}} \tag{2.3.5}$$

Medições projetivas possuem o cálculo de seus valores médios e desvio padrão para uma série de medições. Utilizando o conceito estatístico de esperança [2] para o cálculo da média:

$$E(M) = \sum_m mp(m) = \sum_m \langle\psi|P_m|\psi\rangle = \sum_m \langle\psi| \left(\sum_m mPm\right) |\psi\rangle = \langle\psi|M|\psi\rangle \tag{2.3.6}$$

2.3.5 Sistemas compostos

Supondo que o interesse esteja em um sistema quântico composto formado por dois ou mais sistemas quânticos distintos, o espaço de estados desse sistema físico composto é o produto tensorial dos espaços de estados dos sistemas físicos individuais. Sejam os sistemas individuais $|\psi_1\rangle, |\psi_2\rangle, ..., |\psi_n\rangle$, decorre que o estado do sistema composto será $|\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_n\rangle$ (NIELSEN; CHUANG, 2010).

2.3.6 Emaranhamento quântico

Uma importante propriedade da mecânica quântica é o emaranhamento quântico (ou entrelaçamento quântico), um fenômeno que ocorre quando pares ou grupos de partículas interagem de tal forma que o estado quântico de cada uma não pode ser descrito independentemente, e ao invés disso, um estado quântico deve ser dado para o sistema como um todo (ZINGER, 2015).

Conforme Zinger (2015), sejam dois sistemas, *A* e *B*, de estados $|\psi\rangle_A$ e $|\psi\rangle_B$, que possuem bases $|j\rangle_A$ e $|k\rangle_B$ para os espaços de Hilbert H_A e H_B, respectivamente. O sistema composto entre *A* e *B* pode ser descrito como:

$$|\psi\rangle_{AB} = \sum_{j,k} c_{j,k}|j\rangle_A \otimes |k\rangle_B \tag{2.3.7}$$

O estado $|\psi\rangle_{AB}$ é dito separável se existir c_j^A e c_k^B de forma que $c_{j,k} = c_j^A c_k^B$, e do contrário é dito inseparável. Um estado inseparável é chamado de estado emaranhado.

2.4 COMPUTAÇÃO QUÂNTICA

Conforme define a (ACM, 2005, p.9): "De uma forma geral, podemos definir computação para significar qualquer atividade orientada a objetivos que exija, se beneficie de, ou crie computadores. Assim, a computação inclui projetar e construir sistemas de *hardware* e *software* para uma ampla gama de finalidades; processamento, estruturação e gerenciamento de vários

[2] A esperança de um conjunto de resultados equivale à soma dos produtos individuais de valor multiplicada pela probabilidade.

tipos de informações; fazer estudos científicos em computadores; fazer com que os sistemas de computador se comportem de maneira inteligente; criando e usando meios de comunicação e entretenimento; encontrar e coletar informações relevantes para qualquer propósito, e assim por diante. A lista é virtualmente infinita e as possibilidades são vastas".

A computação tradicional se utiliza de *bits* baseados na lógica binária, enquanto os computadores quânticos utilizam o análogo quântico dos *bits*, o *bit* quântico, que pode estar em um estado de superposição quântica entre zero e um e se beneficiar do chamado paralelismo quântico para fazer cálculos simultâneos. (ZINGER, 2015).

Nesta seção, a maioria dos conceitos e definições apresentados são desenvolvidos por Nielsen e Chuang (2010).

2.4.1 Bits quânticos

A unidade fundamental da computação clássica é o *bit*, também denominado digito binário, que pode assumir os estados 0 ou 1 em valores diferentes para tempos diferentes. O conceito análogo da computação quântica é o *bit* quântico, ou *q-bit* (ou ainda qubit). O *q-bit*, além dos estados 0 e 1, denotados $|0\rangle$ e $|1\rangle$, pode estar em uma superposição quântica entre os dois estados. Matematicamente, a superposição de um *q-bit* é uma combinação linear de estados:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.4.1}$$

Os estados $|0\rangle$ e $|1\rangle$ são vetores no espaço de Hilbert complexo de duas dimensões que formam uma base ortonormal para o espaço $\mathbb{C}^2$ denominada base computacional.

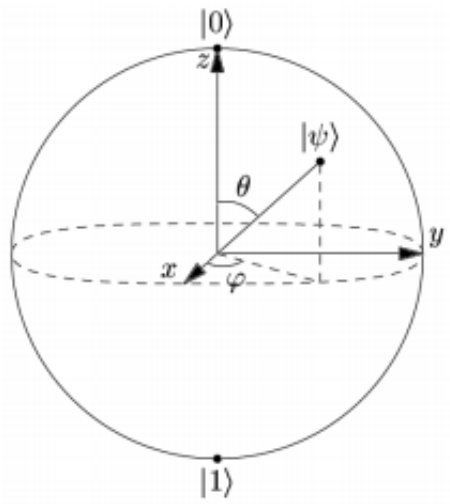
$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \; e \; |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.4.2}$$

Os números α e β são complexos e estão associados a probabilidades. Quando um *q-bit* é medido, há uma probabilidade $|\alpha|^2$ de encontrar o estado $|0\rangle$ e uma probabilidade $|\beta|^2$ de encontrar estado $|1\rangle$. Naturalmente, a soma das probabilidades deve ser igual a um, ou seja, $|\alpha|^2+|\beta|^2 = 1$. O *q-bit* é um vetor unitário em um espaço vetorial complexo de duas dimensões.

Alguns sistemas físicos podem ser utilizados para implementar um *q-bit* são: duas polarizações diferentes de um fóton, o alinhamento de um *spin* nuclear em um campo magnético uniforme e os dois estados de um elétron orbitando um núcleo atômico.

Um *q-bit* pode ser representado geometricamente. Conforme demonstra Carvalho, Lavor e Motta (2007), o *q-bit*, por pertencer ao espaço $\mathbb{C}^2$, pode ser representado no espaço $\mathbb{R}^4$, e utilizando a relação $|\alpha|^2 + |\beta|^2 = 1$ pode ser transformado para $\mathbb{R}^3$ em sua representação tridimensional na esfera de raio unitário conhecida como esfera de Bloch, ilustrada na Figura 5 e descrita pela Equação 2.4.3.

Figura 5 – Representação de um *q-bit* na esfera de Bloch.



Fonte: Nielsen e Chuang (2010).

$$|\psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}sen\frac{\theta}{2}|1\rangle) \tag{2.4.3}$$

em que θ, ϕ e γ são números reais, dados por:

$$\theta = \cos^{-1}|\alpha| = sen^{-1}|\beta|\ ,\ (0 < \theta < \pi/2) \tag{2.4.4}$$

$$\phi = tg^{-1}|\beta| - tg^{-1}|\alpha|\ ,\ (0 < \phi < 2\pi) \tag{2.4.5}$$

$$\gamma = tg^{-1}|\alpha|\ ,\ (0 < \gamma < 2\pi) \tag{2.4.6}$$

O fator $e^{i\gamma}$ pode ser ignorado por não ser observável, de modo que se pode reescrever:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}sen\frac{\theta}{2}|1\rangle \tag{2.4.7}$$

Os números θ e ϕ definem um ponto sobre a superfície da esfera de Bloch. Uma representação muito utilizada é o vetor de Bloch, definido como:

$$|\psi_B\rangle = \cos\phi\ sen\frac{\theta}{2}\begin{bmatrix}1\\0\\0\end{bmatrix} + sen\ \phi\ sen\frac{\theta}{2}\begin{bmatrix}0\\1\\0\end{bmatrix} + \cos\frac{\theta}{2}\begin{bmatrix}0\\0\\1\end{bmatrix} \tag{2.4.8}$$

2.4.2 Múltiplos *q-bits*

Em dois *bits* clássicos existem quatro estados possíveis: 00, 01, 10 e 11. De maneira análoga, um sistema de dois *q-bit* possui quatro estados na base computacional: $|00\rangle$, $|01\rangle$, $|10\rangle$

e $|11\rangle$. Segundo Santos (2018), o espaço de Hilbert para dois *q-bit*s é expandido através do produto tensorial entre eles:

$$\{|0\rangle, |1\rangle\} \otimes \{|0\rangle, |1\rangle\} = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\} \tag{2.4.9}$$

onde:

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \; e \; |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.4.10}$$

Um par de *q-bit*s pode existir em uma superposição desses quatro estados, cada um associado a uma amplitude, de modo que o vetor de estado para um sistema de dois *q-bit*s pode ser escrito como:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{2.4.11}$$

Uma medida sobre esse estado resulta em $x(= 00, 01, 10, 11)$, com probabilidade $|\alpha_x|^2$ para o sistema em $|x\rangle$ após a medida. A condição de normalização deve ser atentida conforme a equação:

$$\sum_{x \in \{0,1\}^2} |\alpha_x|^2 = 1 \tag{2.4.12}$$

Em um sistema de dois *q-bits*, poderia ser feita a medida de somente um dos estados, digamos para o primeiro *q-bit*. O resultado $|0\rangle$ é fornecido com probabilidade $|\alpha_{00}|^2 + |\alpha_{01}|^2$. O estado do sistema após a medida de $|0\rangle$ para o primeiro *q-bit* é renormalizado pelo fator $\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}$.

$$|\psi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} \tag{2.4.13}$$

2.4.3 Portas quânticas de um *q-bit*

De maneira análoga aos computadores clássicos, um computador quântico é construído a partir de um circuito quântico contendo fios e portas lógicas quânticas responsáveis por carregar e manipular a informação quântica.

As portas lógicas clássicas mais conhecidas são as portas *NOT, NAND, NOR, AND* e *OR*, que além de realizarem operações específicas, podem ser combinadas para se realizar outra tarefa mais complexa (SANTOS, 2018).

Uma porta lógica quântica é implementada através de matrizes unitárias, requeridas para se

manter a condição de normalização do estado quântico. As portas quânticas sobre um *q-bit* são matrizes 2x2, e como existem infinitas matrizes unitárias 2x2, há infinitas portas de um *q-bit*.

Algumas das portas quânticas mais relevantes são apresentadas a seguir. Quatro delas são as chamadas matrizes de Pauli:

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \; X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \; Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \; Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.4.14}$$

A porta quântica $\mathbb{I}$ é a porta identidade, que não altera o estado do *q-bit*. A porta quântica *Pauli-X* é análoga a porta clássica *NOT*, que inverte o estado do *bit*, ou seja o estado $\alpha|0\rangle + \beta|1\rangle$, a operação da porta *X* resultará em:

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \tag{2.4.15}$$

Já as operações das portas *Y* e *Z* resultarão, segundo Santos (2018), em:

$$Y \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -i\beta \\ i\alpha \end{bmatrix}, \; Z \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ -\beta \end{bmatrix} \tag{2.4.16}$$

Outras três portas importantes quânticas são as portas Hadamard (*H*), a porta de fase (*S*) e a porta $\pi/8$ (*T*).

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \; S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \; T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \tag{2.4.17}$$

que possuem duas importantes relações: H = $(X + Z)/\sqrt{2}$ e $S = T^2$. As operações das portas *S* e *T* resultarão em:

$$S \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ i\beta \end{bmatrix}, \; T \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ e^{i\pi/4}\beta \end{bmatrix} \tag{2.4.18}$$

A porta Hamadard atua mapeando os estados $|0\rangle$ e $|1\rangle$ para estados de superposição, de modo que:

$$H\,|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \; H\,|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{2.4.19}$$

As matrizes de Pauli dão origem a uma importante classe de matrizes, que são os operadores de rotação em torno dos eixos x, y e z da esfera de Bloch, definidas por:

$$R_x(\phi) = e^{-i\phi X/2} = cos\frac{\phi}{2}I - isen\frac{\phi}{2}X = \begin{pmatrix} cos\frac{\phi}{2} & -isen\frac{\phi}{2} \\ -isen\frac{\phi}{2} & cos\frac{\phi}{2} \end{pmatrix} \tag{2.4.20}$$

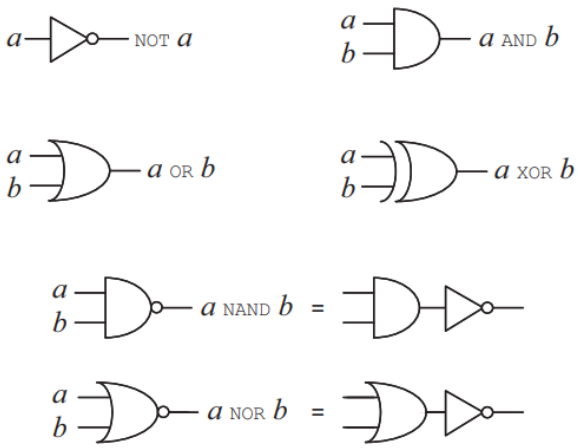
$$R_y(\phi) = e^{-i\phi Y/2} = cos\frac{\phi}{2}I - isen\frac{\phi}{2}Y = \begin{pmatrix} cos\frac{\phi}{2} & -sen\frac{\phi}{2} \\ sen\frac{\phi}{2} & cos\frac{\phi}{2} \end{pmatrix} \tag{2.4.21}$$

$$R_z(\phi) = e^{-i\phi Z/2} = cos\frac{\phi}{2}I - isen\frac{\phi}{2}Z = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix} \tag{2.4.22}$$

2.4.4 Portas quânticas de múltiplos *q-bits*

Na Figura 6 são mostradas a porta lógica clássica de um *bit NOT*, e cinco importantes portas clássicas de vários *bits*: *AND*, *OR*, *NAND*, *NOR*, *XOR*.

Figura 6 – Algumas portas lógicas clássicas de um ou mais bits.



Fonte: Adaptado de Nielsen e Chuang (2010).

Na computação clássica, qualquer função sobre *bits* pode ser computada a partir da composição de portas *NAND*, que é conhecida por ser uma porta universal. No caso quântico, o protótipo de uma porta lógica quântica de muitos bits é o *NÃO-controlado* ou porta *CNOT* (do inglês "*Controlled-NOT*").

A porta *CNOT* possui dois *q-bits* de entrada, conhecidos como *q-bit* de controle e *q-bit* alvo. Se o *q-bit* de controle estiver no estado $|0\rangle$, nada acontece com o *q-bit* alvo. Contudo, se o *q-bit* de controle estiver no estado $|1\rangle$, o *q-bit* alvo troca seu estado. Em símbolos:

$$|00\rangle \rightarrow |00\rangle; \;\; |01\rangle \rightarrow |01\rangle; \;\; |10\rangle \rightarrow |11\rangle; \;\; |11\rangle \rightarrow |10\rangle; \tag{2.4.23}$$

A porta *CNOT* pode ser interpretada como uma generalização da porta clássica *XOR*, de modo que sua ação pode ser resumida como $|A, B\rangle \rightarrow |A, B \oplus A\rangle$ (onde $\oplus$ é o símbolo da

operação *XOR*), ou seja, o resultado da operação *XOR* entre o *q-bit* de controle e o *q-bit* alvo é armazenado no *q-bit* alvo.

Outra maneira de descrever a porta *CNOT* é através de sua representação matricial, que respeita a condição de matriz unitária:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.4.24}$$

Uma extensão da porta *CNOT* é a porta *Toffoli*, aplicada em três *q-bit*. Dois *q-bit* são de controle e seus estados não são alterados pela ação da porta. O terceiro *q-bit* é o *q-bit* alvo e seu estado é invertido se e somente se os dois *q-bit*s de controle forem iguais a $|1\rangle$. Em símbolos, $|A, B, C\rangle \rightarrow |A, B, C \oplus AB\rangle$. A matriz unitária da porta *Toffoli* é a matriz de dimensão 8x8:

$$Toffoli = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.4.25}$$

2.4.5 Reversibilidade da computação quântica

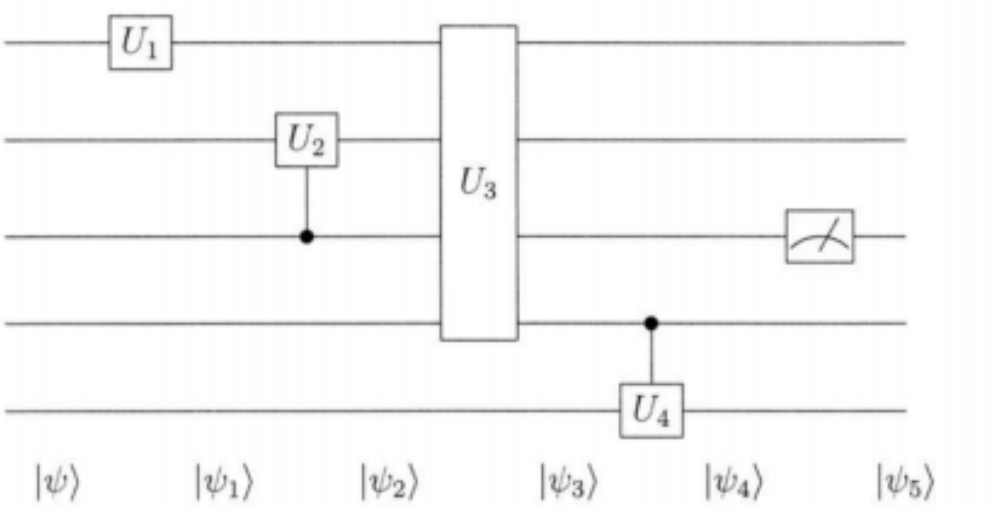
Portas clássicas como *NAND* e o *XOR* regular não podem serem vistas como portas unitárias no mesmo sentido que o *NOT* clássico está para a porta *X*. A razão é que com exceção da porta *NOT*, todas as portas clássicas são irreversíveis (ou não-reversíveis). Por exemplo, é impossível determinar as entradas *A* e *B* dada uma saída $A \oplus B$. Há uma perda de informação irrecuperável na ação da porta *XOR*. Por outro lado, portas quânticas são sempre reversíveis, o que se deve ao fato de que o inverso de uma matriz unitária é outra matriz unitária.

Se uma porta lógica é irreversível, parte da informação é apagada. O princípio de Landaver estabelece que é preciso dissipar energia para apagar informação. Em portas reversíveis, por outro lado, nenhuma informação é perdida, e em princípio, não requer gasto de energia, embora na prática dissipação seja necessária para a estabilidade e a proteção do sistema contra ruído (NIELSEN; CHUANG, 2010).

2.4.6 Circuitos quânticos

Um exemplo de circuito quântico de cinco *q-bits* é esquematizado na Figura 7. O circuito deve ser lido da esquerda para a direita. Cada linha representa um fio, que não necessariamente corresponde a um objeto físico, mas pode corresponder à passagem de tempo ou a uma partícula, como um fóton, se movendo no espaço (MONTEIRO, 2012).

Figura 7 – Exemplo de circuito quântico.

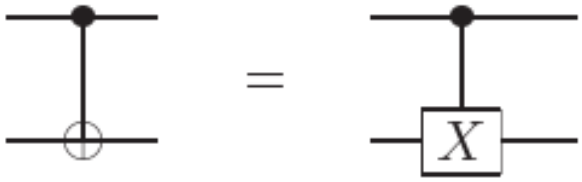


Fonte: Monteiro (2012).

O ponto de partida é o estado inicial $|\psi\rangle$, que por convenção é assumido que o estado inicial de entrada do circuito é um estado da base computacional, geralmente o estado $|0\rangle$. No circuito, as portas quânticas são portas genéricas U_1, U_2, U_3 e U_4. A porta U_1 é uma porta que opera no *q-bit* associado à sua linha, isto é, uma porta de um *q-bit*. Já a porta U_3 atua sobre três *q-bits*. As portas U_2 e U_4 são definidas como portas *U-controlada*, uma extensão natural do *NÃO-controlado*. Essa porta tem um único *q-bit* de controle, indicado pela linha com o ponto, e o n *q-bit*s alvo, indicados pelo *U* da caixa. Se o *q-bit* de controle for $|0\rangle$, nada acontece, e se for $|1\rangle$, a porta *U* é aplicada aos *q-bit*s alvos. Sendo o bit de controle representado por $|c\rangle$ e o *q-bit* alvo por $|t\rangle$, a operação *U-controlada* é descrita em símbolos por $|c\rangle|c\rangle \rightarrow |c\rangle U^c|c\rangle$ (MONTEIRO, 2012).

A porta *NÃO-controlado* é um caso particular da porta *U-controlada* no qual $U = X$ e pode ser representada de duas formas distintas, conforme ilustra a Figura 8.

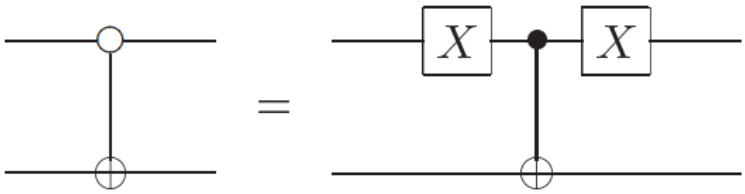
Figura 8 – Duas representações distintas da porta *NÃO-controlado*.



Fonte: Nielsen e Chuang (2010).

O estado $|0\rangle$ também pode ser utilizado como controle, o que exige a aplicação de duas portas *X* no *q-bit* de controle, conforme a notação circuital ilusrada na Figura 9. Neste circuito, a porta *X* é aplicada ao alvo se o controle estiver no estado $|0\rangle$.

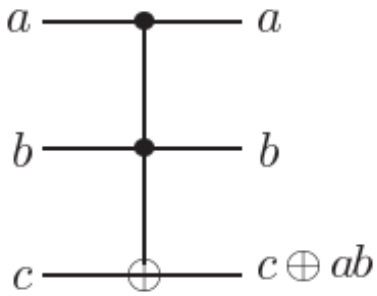
Figura 9 – Operação controlada com controle no estado $|0\rangle$.



Fonte: Nielsen e Chuang (2010).

A representação circuital da porta *Toffoli* é mostrada na Figura 10.

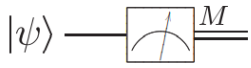
Figura 10 – Representação da porta *Toffoli*.



Fonte: Nielsen e Chuang (2010).

O símbolo ilustrado na Figura 11 indica a medida, que converte um *q-bit* em um *bit* aleatório clássico, que se distingue do *q-bit* pela linha dupla.

Figura 11 – Simbolo de medida no circuito quântico.



Fonte: Nielsen e Chuang (2010).

Um operador deverá atuar em todos os *q-bits* simultaneamente. Por conseguinte, a aplicação de um operador U em um determinado *q-bit* leva a aplicação do operador identidade $\mathbb{I}$ aos demais *q-bit*s do circuito (MONTEIRO, 2012). A sequência de operadores para a Figura 7 é:

$$|\psi_1\rangle = U_1|\psi\rangle \ , \ |\psi_2\rangle = U_2|\psi_1\rangle \ , \ |\psi_3\rangle = U_3|\psi_2\rangle \ , \ |\psi_4\rangle = U_4|\psi_3\rangle \ , \ |\psi_5\rangle = M|\psi_4\rangle \quad (2.4.26)$$

Três diferenças dos circuítos quânticos em relação aos clássicos serão destacadas:

- A proibição nos circuitos quânticos dos "laços" (*loops*), ou seja, retroalimentação (*feed-back*), que por sua vez é permitida nos circuitos clássicos. Os circuitos quânticos são portanto, acíclicos.
- Os circuitos clássicos permitem que fios sejam unidos, na operação conhecida como *FAN-IN*, com o fio resultante contendo a saída binária *OR* das entradas. A operação, por não ser reversível e consequentemente não unitária, não é permitida em circuitos quânticos
- A operação inversa *FAN-OUT,* por meio da qual cópias de bits são feitas, também não é permitida em circuítos quânticos. A propriedade de que *q-bit*s não podem ser copiados é conhecida como teorema da não-clonagem e é demonstrada por Nielsen e Chuang (2010).

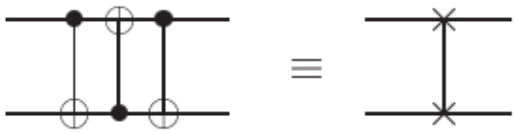
A operação clássica de troca de valores entre dois bits, conhecida como *CROSSOVER*, tem seu análogo quântico por meio da operação conhecida como *SWAP*, que troca os estados dos *q-bit*s entre si. A operação *SWAP* é implementada por meio um circuito com portas *NÃO-controlado*. O circuito e o símbolo dessa porta são ilustrados na Figura 12, e sua operação corresponde à seguinte sequência de efeitos em um estado $|a, b\rangle$:

$$|a, b\rangle \rightarrow |a, a \oplus b\rangle \rightarrow |a \oplus (a \oplus b), a \oplus b\rangle = |b, a \oplus b\rangle \rightarrow |b, (a \oplus b) \oplus b\rangle = |a, b\rangle. \quad (2.4.27)$$

A forma matricial da operação *SWAP* é:

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4.28)$$

Figura 12 – Circuito para a operação *SWAP* e seu símbolo.

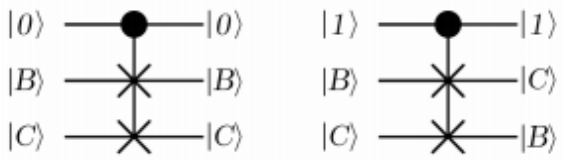


Fonte: Nielsen e Chuang (2010).

Uma extensão da operação *SWAP* é a troca controlada, implementada pela porta *Fredkin*. Segundo Zinger (2015), um *q-bit* de controle é responsável por controlar se a operação de troca entre dois outros *q-bit* ocorrerá, conforme ilustra a Figura 13. A porta *Fredkin* é representada pela matriz 8x8:

$$Fredkin = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4.29}$$

Figura 13 – Ação da porta *Fredlin*.



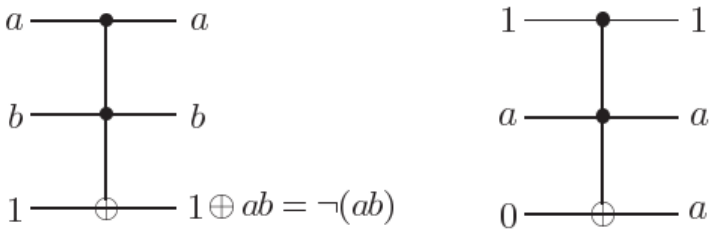
Fonte: Adaptado de Zinger (2015).

2.4.7 Computação clássica em um computador quântico

Circuitos lógicos clássicos podem ser simulados utilizando um circuito quântico. A simulação dos circuitos clássicos não pode ser feita de maneira direta pela razão de que as portas quânticas são intrinsecamente reversíveis, enquanto muitas das portas clássicas, como a *NAND*, são intrinsecamente irreversíveis. Contudo, qualquer circuito clássico pode ser substituido por um circuito equivalente fazendo o uso da porta *Toffoli*.

A porta *Toffoli* pode ser usada para simular a porta *NAND* e a operação *FAN-OUT*. Na Figura 14a, a porta *NAND* é implementada, sendo que os dois *bits* de cima são a entrada da porta, enquanto o último é fixado no estado 1. Já na Figura 14b, a porta *Toffoli* implementa a operação *FAN-OUT* com o segundo *bit* sendo a entrada, e os outros dois *bits* sendo auxiliares. A saída *FAN-OUT* aparece nos dois últimos *bits*. Em conjunto, as duas operações tornam possível a simulação de todos os elementos de um circuito clássico, e por conseguinte, circuitos clássicos podem ser simulados em circuitos reversíveis equivalentes. Em outras palavras, os computadores quânticos são capazes de realizar qualquer operação que possa ser feita em um computador clássico.

Figura 14 – (a) Porta *NAND* com a porta *Toffoli*. (b) Operação *FAN-OUT* com a porta *Toffoli*.



(a) (b)

Fonte: (a) Nielsen e Chuang (2010) e (b) Nielsen e Chuang (2010).

2.4.8 Portas quânticas universais

Conforme visto na subseção anterior, a porta *Toffoli* é universal para computação clássica, isto é, os circuitos quânticos incluem os circuitos clássicos. O conceito de universalidade existe também para a computação quântica. Um conjunto de portas é dito universal para a computação quântica se uma operação unitária qualquer puder ser construída com precisão arbitrária por um circuito quântico envolvendo somente aquelas portas.

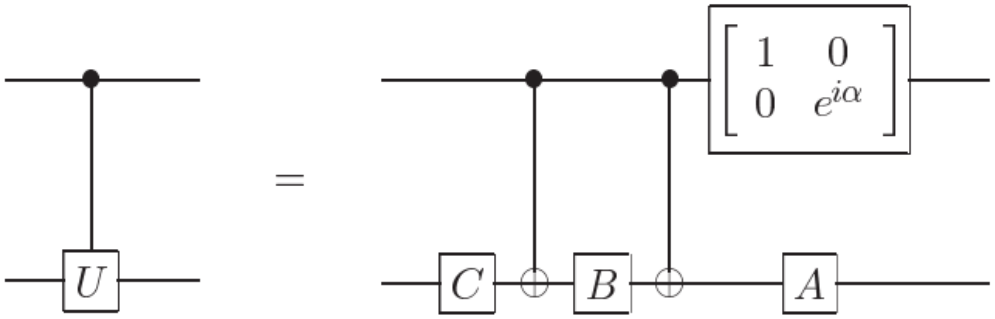
Três resultados sobre universidade das portas quânticas são descritos por Nielsen e Chuang (2010): o primeiro mostra que um operador unitário arbitrário pode ser expresso exatamente como o produto de operadores unitários, cada um atuando sobre um subespaço expandido por dois estados da base computacional. O segundo, partindo do primeiro, mostra que um operador unitário arbitrário pode ser expresso exatamente usando portas de um *q-bits* combinadas com portas *CNOT*. Por fim, o terceiro combina a construção do segundo com a prova de que operações de um *q-bit* podem ser feitas com precisão arbitrária utilizando portas *H, S* e *T*, o que culmina na conclusão de que qualquer operação unitária pode ser construida com precisão arbitrária utilizando as portas *H, S, T* e *CNOT*.

2.4.9 Operações controladas

A operação controlada é uma classe de operações extremamente importante em computação quântica. Nesta subseção será descrito como implementar operações controladas a partir de portas quânticas universais.

Seja U uma porta unitária sobre um *q-bit*, a operação controlada sobre U pode ser implementada pelo circuito da Figura 15, em que *A, B e C* são operadores unitários tais que $U = e^{i\alpha}AXBXC$ e $ABC = \mathbb{I}$.

Figura 15 – Circuito para implementar a operação *U-controlada*.



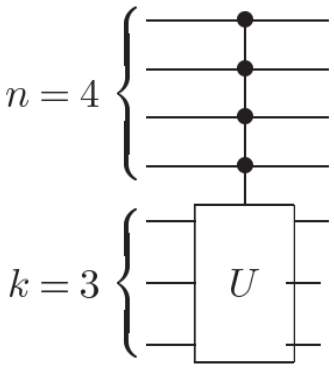
Fonte: Nielsen e Chuang (2010).

A operação controlada pode ser generalizada para *n q-bits* de controle com um operador unitário atuando sobre *k q-bits* pela equação:

$$C^n(U)|x_1x_2...x_n\rangle|\psi\rangle = |x_1x_2...x_n\rangle U^{x_1x_2...x_n}|\psi\rangle \tag{2.4.30}$$

em que $x_1x_2...x_n$ é o produto dos bits. A Figura 16 ilustra a notação circuital para a operação.

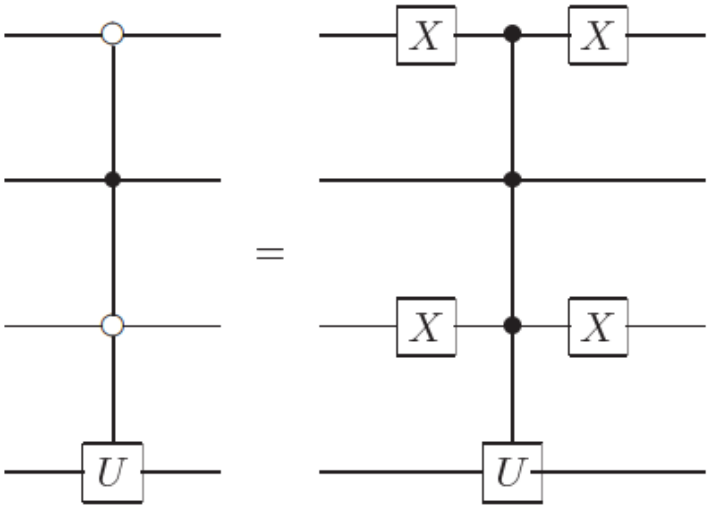
Figura 16 – Exemplo de representação da operação controlada $C^n(U)$, sendo U um operador sobre *k q-bits*, para *n = 4* e *k = 3*.



Fonte: Nielsen e Chuang (2010).

A Figura 17 ilustra um exemplo de operador $C^n(U)$ que utiliza o controle tanto pelo estado $|1\rangle$ (no segundo *q-bit*), quanto pelo estado $|0\rangle$ (no primeiro e no terceiro *q-bits*).

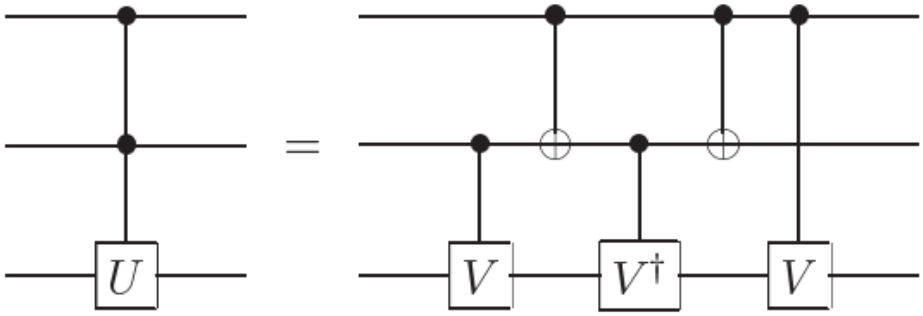
Figura 17 – Exemplo de operação controlada $C^n(U)$, com controle tanto pelo estado $|1\rangle$ quanto pelo estado $|0\rangle$.



Fonte: Nielsen e Chuang (2010).

Supondo $k = 1$, o operador $C^2(U)$ pode ser implementado pelo circuito da Figura 18, em que *V* é um operador unitário que satisfaz $V^2 = U$. As portas $C^1(U)$ podem ser implementadas pelo circuito da Figura 15.

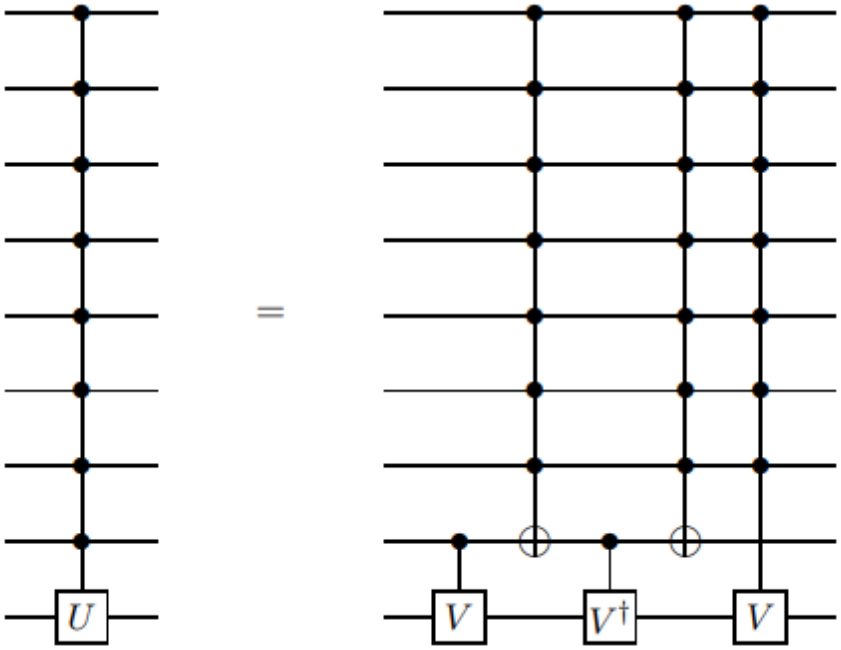
Figura 18 – Circuito para implementar a operação controlada $C^2(U)$.



Fonte: Nielsen e Chuang (2010).

Barenco et al. (1995) mostram que a operação $C^n(U)$ pode ser generalizada pela topologia do circuito da Figura 19, que mantém a relação $V^2 = U$.

Figura 19 – Circuito para implementar a operação controlada $C^8(U)$.

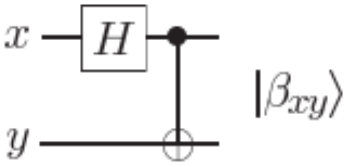


Fonte: Barenco et al. (1995).

2.4.10 Estados de Bell

O circuito quântico mostrado na Figura 20 transforma os estados da base computacional nos chamados estados de Bell, $|\beta_{xy}\rangle$, que são importantes estados emaranhados, indicados na Tabela 1.

Figura 20 – Circuito quântico para criar os estados de Bell.



Fonte: Nielsen e Chuang (2010).

Tabela 1 – Estados de Bell.

Entrada	Estados de Bell
$\|01\rangle$	$(\|00\rangle + \|11\rangle)/\sqrt{2} \equiv \|\beta_{00}\rangle$
$\|01\rangle$	$(\|01\rangle + \|10\rangle)/\sqrt{2} \equiv \|\beta_{01}\rangle$
$\|10\rangle$	$(\|00\rangle - \|11\rangle)/\sqrt{2} \equiv \|\beta_{10}\rangle$
$\|11\rangle$	$(\|01\rangle - \|10\rangle)/\sqrt{2} \equiv \|\beta_{11}\rangle$

Fonte: Adaptado de Nielsen e Chuang (2010).

Os estados de Bell podem ser sintetizados na equação:

$$|\beta_{xy}\rangle = \frac{|0, y\rangle + (-1)^x |1, \bar{y}\rangle}{\sqrt{2}} \tag{2.4.31}$$

em que $\bar{y}$ é negação de y.

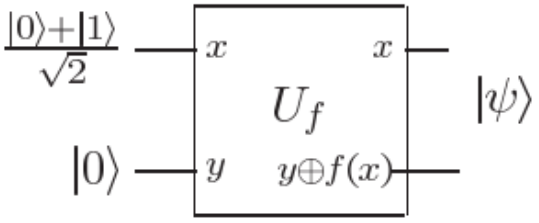
2.4.11 Paralelismo quântico

O grande poder da computação quântica está em sua capacidade de realizar calculos simultâneos para diferentes valores de uma dada função, uma propriedade conhecida como paralelismo quântico (ZINGER, 2015).

Supondo uma função $f(x) : \{0, 1\} \rightarrow \{0, 1\}$ reversível de um único *q-bit*, é possível, por meio de uma sequência apropriada de portas lógicas armazenar o valor de *x* conjuntamente com o valor de $f(x)$ no estado $|x, y \oplus f(x)\rangle$, denotando U_f a transformação unitária $|x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$. Se $y = 0$, o estado final do segundo *q-bit* é justamente $f(x)$. O circuito mostrado na Figura 21 aplica U_f em uma entrada onde o registro *x* é preparado em uma superposição $(|0\rangle + |1\rangle)/\sqrt{2}$, que pode ser criada por uma porta *H* atuando sobre o estado $|0\rangle$ e o *y* é inicializado em $|0\rangle$, resultando no estado:

$$|\psi\rangle = \frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}} \tag{2.4.32}$$

Figura 21 – Circuito para avaliar *f(0)* e *f(1)* simultaneamente.



Fonte: Nielsen e Chuang (2010).

Os termos da soma contêm informações sobre ambos $f(1)$, $f(0)$, apesar do operador U_f só ter sido aplicado uma única vez, demonstrando o paralelismo quântico. Um único circuito é empregado para avaliar múltiplos valores de *x* simultaneamente, diferentemente do paralelismo clássico, que exige o emprego de vários circuitos diferentes acionados simultaneamente.

O procedimento pode ser generalizado para funções com um número arbitrário de *bits*, utilizando a operação conhecida como transformação de Hadamard (denotada $H^{\otimes n}$), na qual *n* portas Hadamard atuam simultaneamente sobre *n q-bits*. De maneira genérica, a avaliação paralela de uma função de *n* bits, $f(x)$, pode ser feita preparando o estado de $n+1$ *q-bits*, $|0\rangle^{\otimes n}|0\rangle$, aplicando a transformação Hadamard nos primeiros *n q-bits*, seguida do circuito que implementa U_f, que resultará em um estado que contém informação sobre 2^n estados:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle|f(x)\rangle \tag{2.4.33}$$

Contudo, toda informação não é acessível, pois a medida sobre o estado forneceria $f(x)$ para apenas um valor de *x*. Porém, é possível obter informações sobre as propriedades globais de $f(x)$ devido à interferência existente entre os estados ainda emaranhados (ZINGER, 2015). Um exemplo de circuito que obtém propriedades globais é o algorítmo de Deutsch-Jozsa, demonstrado por Nielsen e Chuang (2010).

2.4.12 Transformada de Fourier quântica

A transformada de Fourier é uma ferramenta matemática utilizada em diversos campos da ciência, como por exemplo, no processamento digital de imagens e sinais. A transformada de Fourier discreta (*Discrete fourier transform* - DFT) é a versão da transformada de Fourier para dados discretos, como por exemplo, imagens digitais (BARBOSA, 2005). A DFT é um caso de transformação que se pode computar muito mais rapidamente em um computador quântico do que um computador clássico, dando origem a uma classe de algoritmos quânticos que incluem a fatoração de Shor, o logaritmo discreto, algoritmo de Deutsch-Jozsa e sendo fundamental na construção do algoritmo de resolução de sistemas lineares, foco deste trabalho.

Pela notação matemática usual, a transformada de Fourier discreta recebe como entrada um vetor de números complexos $x_0, \cdots, x_{N-1}$ de tamanho fixo *N*. Sua saída são os dados transformados, um vetor de números complexos $y_0, \cdots, y_{N-1}$, definido por:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k/N} \tag{2.4.34}$$

A transformada quântica é exatamente a mesma transformação, embora com uma notação diferente. A transformada de Fourier quântica (*Quantum Fourier Transform* - QFT) sobre uma base ortonormal $|0\rangle, ..., |N-1\rangle$ é um operador linear FT que atua nos estados da base da seguinte forma:

$$FT|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle \tag{2.4.35}$$

De forma equivalente, a ação da transformada sobre um estado arbitrário pode ser descrita como:

$$\sum_{j=0}^{N-1} x_j |k\rangle \rightarrow \sum_{k=0}^{N-1} y_k |k\rangle \tag{2.4.36}$$

As amplitudes y_k são as transformadas de Fourier discretas das amplitudes x_j (BARBOSA, 2005).

A transformação é unitária e pode ser implementada por um meio da dinâmica de um computador quântico. A obtenção do circuito quântico que realiza a transformada de Fourier é descrita por Nielsen e Chuang (2010) e Barbosa (2005):

Supondo que $N = 2^n$, em que *n* é um inteiro, e que a base $|0\rangle, ..., |2^n - 1\rangle$ é a base computacional para um computador quântico de *n q-bits*. É conveniente escrever o estado $|j\rangle$ usando a representação binária $j = j_1 j_2 j_3 \cdots j_n$, ou mais formalmente, $j = j_1 2^{n-1} + j_2 2^{n-2} + ... + j_n 2^0$. Além disso, é também conveniente adotar a notação $0.j_l j_{l+1} \cdots j_m$ para representar a fração binária $j_l/2 + j_{l+1}/4 + ... + j_m/2^{m-l+1}$. Com manipulação algébrica, a QFT pode ser escrita em sua forma conhecida como representação de produto:

$$FT|j_1...j_n\rangle = \frac{(|0\rangle + e^{2\pi i 0.j_n}|1\rangle)(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle)...(|0\rangle + e^{2\pi i 0.j_1 j_2 ... j_n}|1\rangle)}{2^{n/2}} \tag{2.4.37}$$

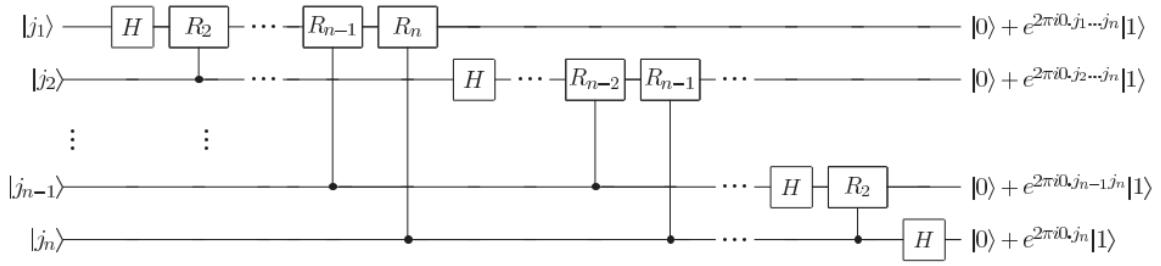
Essa representação permite escrever um circuito eficiente para a QFT, mostrado na Figura 22. Os *q-bits* devem ser invertidos por portas *SWAP* ao final da operação. A porta R_k é definida como:

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix} \tag{2.4.38}$$

O mesmo operador FT pode ser representado em forma matricial, definindo $\omega = e^{2\pi i/2^N}$:

$$FT = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{2^n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(2^n-1)} \\ 1 & \omega^3 & \omega^6 & \cdots & \omega^{3(2^n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{2^n-1} & \omega^{2(2^n-1)} & \cdots & \omega^{(2^n-1)(2^n-1)} \end{bmatrix} \tag{2.4.39}$$

Figura 22 – Circuito quântico que implementa a transformada de Fourier quântica, omitindo portas *SWAP* para reversão da ordem de *q-bits* e fatores de normalização $1/\sqrt{2}$ na saída.



Fonte: Nielsen e Chuang (2010).

Levando em consideração que o melhor algoritmo clássico para computar a transformada de Fourier em 2^n elementos, a transformada de Fourier rápida (*Fast Fourier Transform* - FFT), utiliza $\Theta(n2^n)$ portas, o algoritmo quântico precisa de um tempo exponencialmente menor para ser executado, com uma complexidade de $\Theta(n^2)$.

Por ser transformação unitária, a inversa da transformada quântica de Fourier é o adjunto hermitiano da matriz de Fourier, ou seja, $FT^{-1} = FT^{\dagger}$ (ZINGER, 2015).

2.4.13 Estimativa de fase

A transformada de Fourier é a chave para um procedimento geral conhecido como estimativa de fase (*Quantum Phase Estimation* - QPE), que por sua vez é a chave para o algoritmo de resolução de equações lineares.

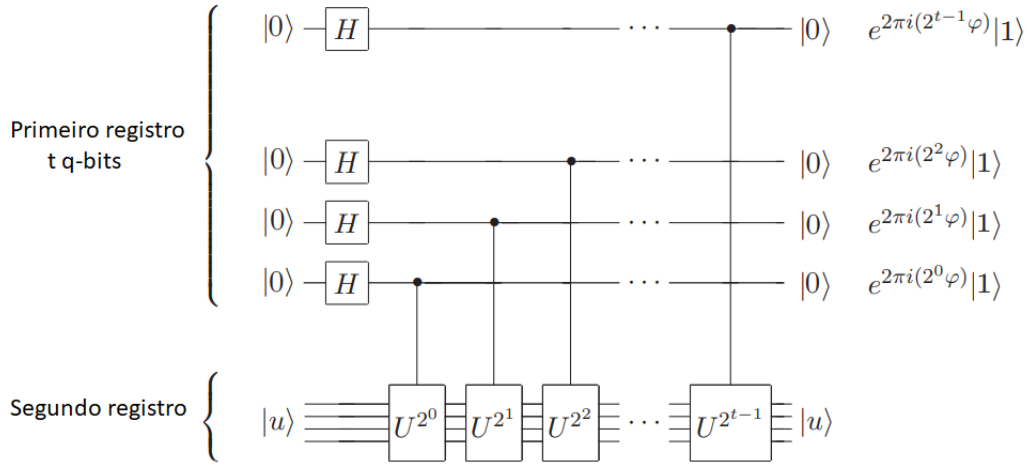
Supondo que um operador unitário U tenha autovetor $|u\rangle$ com autovalor $e^{2\pi i\phi}$, no qual o valor ϕ é desconhecido. Para estimar ϕ, supõe-se que o estado $|u\rangle$ e a operação U^{2^j}-controlada, para alguns inteiros não-negativos j, sejam preparados.

O procedimento utiliza dois registros. O primeiro contém *t q-bit*s inicialmente no estado $|0\rangle$, já o segundo inicia no estado $|u\rangle$ e contém a quantidade de *q-bit*s necessária para armazenar $|u\rangle$.

A estimativa de fase é feita em três estágios. Primeiro, o circuito da Figura 23 é aplicado. O circuito aplica uma transformação de Hadamard ao primeiro registro, seguido de aplicações de *U-controlada* no segundo registro, com *U* elevada a sucessivas potências de dois. O estado final do primeiro registro é:

$$\frac{1}{2^{t/2}}(|0\rangle + e^{2\pi i 2^{t-1}\phi}|1\rangle)(|0\rangle + e^{2\pi i 2^{t-2}\phi}|1\rangle)...(|0\rangle + e^{2\pi i 2^{0}\phi}|1\rangle) = \frac{1}{2^{t/2}}\sum_{k=0}^{2^t-1} e^{2\pi i\phi k}|k\rangle \quad (2.4.40)$$

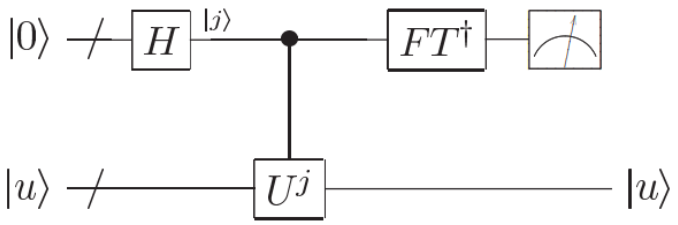
Figura 23 – Primeira etapa do procedimento de estimativa de fase.



Fonte: Adaptado de Nielsen e Chuang (2010).

O segundo estágio consiste na aplicaçao da QFT inversa ao primeiro registro e o terceiro consiste na realização de uma leitura do estado do primeiro registro da base computacional. A Figura 24 mostra um esquema geral do algoritmo.

Figura 24 – Estimativa global do procedimento de estimativa de fase. Os t *q-bits* superiores (o símbolo "/"denota um grupo de fios) formam o segundo registro e os *q-bits* inferiores o segundo registro.



Fonte: Nielsen e Chuang (2010).

Supondo que ϕ que possa ser exatamente em *t q-bits*, como $\phi = 0.\phi_1 \cdots \phi_t$, o estado da primeira etapa pode ser rescrito como:

$$\frac{1}{2^{t/2}}(|0\rangle + e^{2\pi i 0.\phi_t}|1\rangle)(|0\rangle + e^{2\pi i 0.\phi_{t-1}\phi_t}|1\rangle)...(|0\rangle + e^{2\pi i 0.\phi_1\phi_2...\phi_t}|1\rangle) \tag{2.4.41}$$

Comparando as Equações 2.4.37 e 2.4.41, pode-se concluir que o resultado da aplicação da QFT inversa é o estado-produto $|\phi_1...\phi_t\rangle$ e uma medida na base computacional fornece portanto, ϕ exatamente.

Para o caso onde ϕ não puder ser expresso exatamente com *t q-bit*s, o procedimento produzirá uma aproximação para ϕ com alta probabilidade. Supondo que se queira aproximar ϕ com precisão 2^{-n}, utilizando $t = n + p$ *q-bits*, a probabilidade de se obter a aproximação é de pelo menos $1 - 1/2(2^p - 2)$, e definindo $\epsilon = 1/2(2^p - 2)$, chega-se a conclusão de que o resultado da

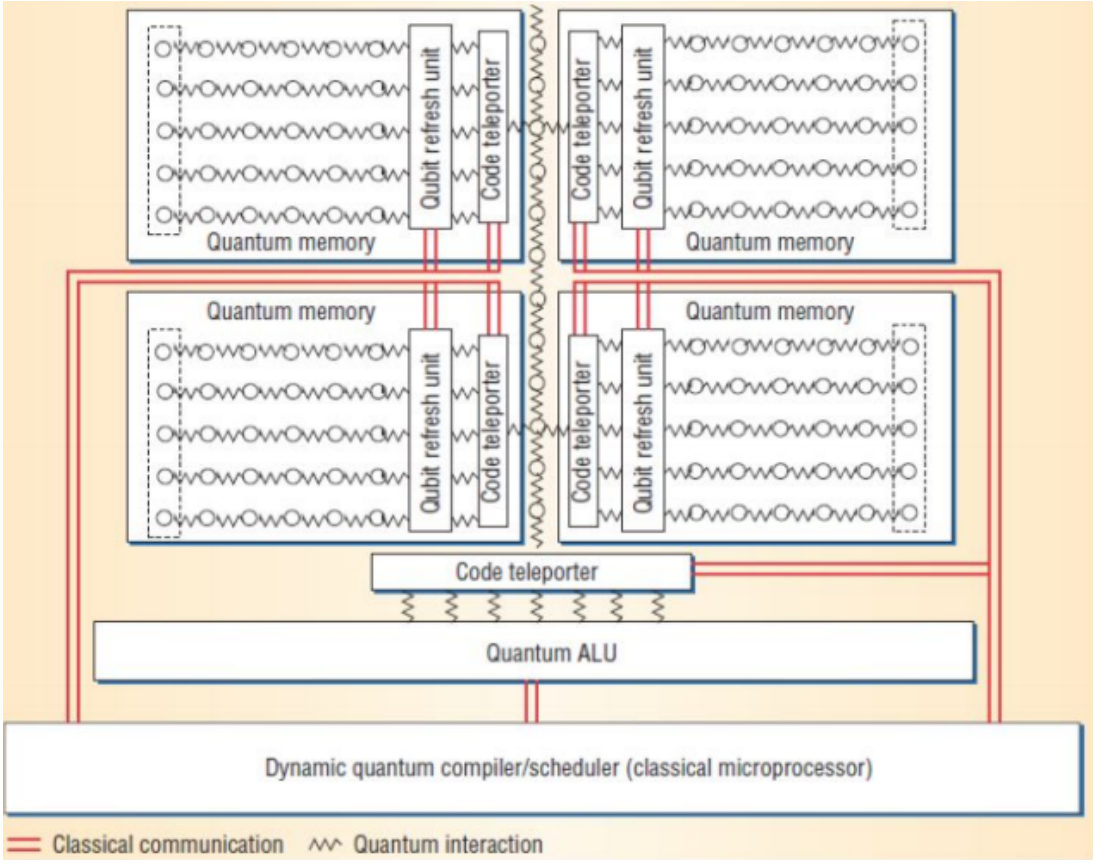
medida é uma aproximação para ϕ com precisão de $t - [log(2 + 1/2\epsilon)]$ bits, com probabilidade de sucesso de pelo menos $1 - \epsilon$.

2.4.14 Arquitetura de um computador quântico

Para construir uma arquitetura com base nas portas lógicas quânticas, defronta-se para o transporte e armazenamento de *q-bits* com o processo físico chamado de decoerência quântica, no qual o estado de um *q-bit* se perde principalmente por interação com o ambiente ou outra fonte de ruídos. Para uma arquitetura viável é necessário mecanismos de correção de erros a fim de evitar resultados incorretos decorrente da decoerência (NIELSEN; CHUANG, 2010).

Um exemplo de arquitetetura proposta por Oskin, Chong e Chuang (2002) e ilustrado na Figura 25 contém três componentes principais: a unidade lógica-aritmética quântica (qULA), a memória quântica e um escalonador dinâmico. É necessário usar a técnica de teleporte quântico, descrita em detalhes por Nielsen e Chuang (2010), no transporte da informação quântica, uma vez que um *q-bit* não pode ser clonado.

Figura 25 – Arquitetura proposta por Oskin, Chong e Chuang (2002).



Fonte: Oskin, Chong e Chuang (2002).

A construção de um mecanismo de memória confiável exige um sistema com uma taxa baixa de decoerência para os *q-bit*s estáticos, mesmo assim, esses dispositivos necessitam de unidades de *refresh*. O mecanismo da qULA, composta de um conjunto básico de portas quânticas que

formam o conjunto de portas universais, é responsável por executar as operações, tanto para computação, quanto para correção de erro. São necessários os *q-bits* auxiliares para a codificação e aplicação da correção de erro, gerados por um *hardware* específico. Um processador clássico de alta performance é previsto para o controle de escalonamento dinâmico, utilizando construções clássicas de controle de fluxo e dinamicamente traduz as operações lógicas em operações sobre *q-bits* físicos individuais, a fim de controlar a qULA, o teleporte de códigos e o *refresh* das unidades de memória quântica.

2.5 ALGORITMO DE HARROW, HASSIDIM E LLOYD

Computadores quânticos utilizam a mecânica quântica para executar certos tipos de computação de forma mais eficiente que um computador clássico. A solução de sistemas de equações lineares é um destes casos. Sistemas lineares estão presentes em todos os campos da ciência e engenharia, como por exemplo na solução de equações diferenciais parciais. Com a evolução da tecnologia e o aumento da complexidade dos sistemas utilizados, o tamanho da massa de dados comumente utilizada como entrada para estes problemas vem crescendo muito rapidamente, de forma que mesmo os computadores mais potentes levam bastante tempo para processar soluções (ESKANDARPOUR et al., 2020a; ZINGER, 2015).

O algoritmo de HHL (HARROW; HASSIDIM; LLOYD, 2009) mostra como um computador quântico é capaz de aproximar o valor de funções da solução de sistemas lineares em tempo polilogarítmico, de forma a prover um aumento exponencial de tempo sobre os melhores algoritmos clássicos conhecidos.

2.5.1 Visão geral

Conforme já apresentado, o problema de fluxo de potência CC consiste na solução do sistema de equações lineares expresso pela equação matricial $p = B\theta$. onde a matriz $B \in \mathbb{R}^{NxN}$, o vetor $p \in \mathbb{R}^N$ e a vetor solução $\theta \in \mathbb{R}^N$.

Inicialmente, p e θ devem ser normalizados e mapeados nos estados quânticos $|p\rangle$ e $|\theta\rangle$ em $log_2 N$ *q-bit*s. Para execução do algoritmo, supõe-se a hipótese de que ***B*** seja uma matriz hermitiana e esparsa. A matriz admitância, conforme Equação 2.1.23, será sempre hermitiana. Pela primeira hipótese, ***B*** pode ser representado através de sua representação diagonal:

$$B = \sum_j \lambda_j |u_j\rangle\langle u_j|, \;\; \lambda_j \in \mathbb{R} \tag{2.5.1}$$

A matriz ***B*** pode ser invertida pela inversão de seus autovalores λ_j:

$$B^{-1} = \sum_j \lambda_j^{-1} |u_j\rangle\langle u_j| \tag{2.5.2}$$

Quando $|p\rangle$ está no autoespaço de *B*, isto é:

$$|p\rangle = \sum_j p_j |u_j\rangle \tag{2.5.3}$$

o algoritmo obtém o registro do estado $|\theta\rangle$, que é a representação quântica do vetor θ desejado:

$$|\theta\rangle = B^{-1}|p\rangle = \sum_j \lambda_j^{-1} p_j |u_j\rangle \tag{2.5.4}$$

2.5.2 O algoritmo

A Figura 26 mostra o circuito correspondente ao algoritmo HHL. O primeiro registrador, denotado α, é usado para armazenar a representação binária dos autovalores de *B*. O segundo registrador, denotado β, é usado para armazenar o estado $|p\rangle$ e fornece o vetor solução. O terceiro registrador é o *q-bit* de controle denominado *ancilla*. Os registradores *ancilla* e α são inicializados em $|0\rangle$, enquanto β é inicializado através de uma rotina externa ao algoritmo em $|p\rangle$ conforme os seus respectivos coeficientes normalizados.

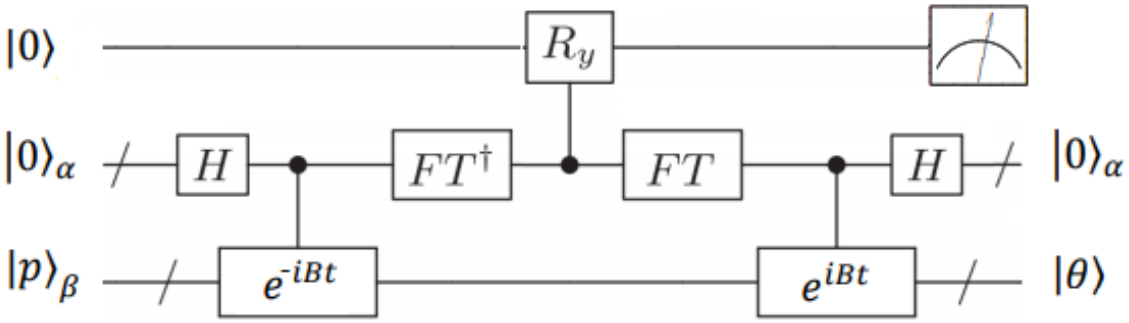
$$p = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{N-1} \end{bmatrix} \rightarrow |p\rangle_\beta = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{N-1} \end{bmatrix} \tag{2.5.5}$$

O estado inicial é, portanto:

$$|\psi_0\rangle = |0\rangle \otimes |0\rangle_\alpha \otimes |p\rangle_\beta \tag{2.5.6}$$

O algoritmo consiste em quatro etapas principais: a estimativa de fase, a rotação controlada, a computação reversa e a medida.

Figura 26 – Circuito quântico para resolução de sistemas lineares.



Fonte: Adaptado de Zinger (2015).

2.5.2.1 Simulação Hamiltoniana

Conforme visto na Seção 2.3, a evolução temporal de um sistema quântico é descrita pela equação de Schrödinger. Segundo Nielsen e Chuang (2010), um operador hamiltoniano independente do tempo fornece:

$$|\psi(t)\rangle = e^{-i\mathbb{H}t}|\psi(0)\rangle \tag{2.5.7}$$

A matriz de entrada *B* do sistema linear é s-esparsa, que é o caso de problemas de fluxo de potência CC. Para simular o hamiltoniano do sistema, a matriz *B* será decomposta na forma $B = \sum B_j$, sendo B_j 1-esparso. Uma vez com os valores das simulações de cada B_j, é possivel simular, eficientemente, a matriz *B* através do operador e^{iBt} em tempo $\mathcal{O}(log(N)s^2t)$.

A hipótese original de que *B* é hermitiana pode ser relaxada. Neste caso, é definida a matriz hermitiana:

$$B^o = \begin{pmatrix} 0 & B \\ B^\dagger & 0 \end{pmatrix} \tag{2.5.8}$$

Uma vez obtido B^o, é possível resolver a equação:

$$B^o y = \begin{pmatrix} p \\ 0 \end{pmatrix} \tag{2.5.9}$$

obtendo:

$$y = \begin{pmatrix} 0 \\ \theta \end{pmatrix} \tag{2.5.10}$$

2.5.2.2 Estimativa de fase

A primeira etapa do algoritmo é a estimativa de fase. Conforme ja descrito, a QPE é responsável por encontrar os autovalores de um autovetor um operador unitário, mapeando-os com uma mudança de base. Para o algoritmo HHL, operador *U* é igual e^{iBt}. Neste caso:

$$e^{iBt} = \sum_j e^{i\lambda_j t}|u_j\rangle\langle u_j| \tag{2.5.11}$$

Então, para cada autovetor $|u_j\rangle_\beta$ que tem como autovalor $e^{i\lambda_j t}$, a QPE resultará em $|\lambda_i^o\rangle_\alpha|u_i\rangle_\beta$, em que λ_i^o é uma representação binária em α *bits* para uma aproximação em $2^\alpha\lambda_j t/2\pi$, resultando no armazenamento dos autovetores em estados emaranhados correspondentes aos seus respectivos autovalores no registrador β. A constante *t* é calculada por:

$$t = \frac{2\pi\lambda_j^o}{2^\alpha \lambda_j} \tag{2.5.12}$$

Assumindo que cada autovalor λ_j possa ser representado corretamente em α *bits* e que $|p\rangle$ está no autoespaço de *B*, a aplicação da QPE levará o estado inicial $|\psi_0\rangle$, dado por

$$|\psi_0\rangle = |0\rangle \otimes |0\rangle_\alpha \otimes \sum_j p_j |u_j\rangle \tag{2.5.13}$$

para

$$|\psi_1\rangle = |0\rangle \otimes \sum_j p_j |\lambda_j\rangle_\alpha |u_j\rangle_\alpha \tag{2.5.14}$$

O número de bits α necessários para mapear os autovalores é uma parâmetro que depende do número de condição κ, uma vez que esse quantifica a amplitude entre o maior e o menor autovalor da matriz *B* (ZINGER, 2015).

Na simulação hamiltoniana, é possível observar o grande benefício dos operadores quânticos, uma vez que o paralelismo quântico colocará o segundo registro do estado em superposição quântica associada a todos os autovalores da matriz *B* (ZINGER, 2015).

2.5.2.3 Rotação controlada

A próxima etapa do algoritmo, fundamental para inversão dos autovalores de B, consiste na operaçao da rotação controlada aplicada pelo *q-bit ancilla*.

Conforme demonstra Zinger (2015), a rotação controlada é feita pela matriz de rotação $R_y(\phi)$, descrita pela Equação 2.4.21, de modo que:

$$|\psi_2\rangle = \sum_j \left(cos\frac{\phi}{2} I - isen\frac{\phi}{2} Y \right) |0\rangle \otimes p_j |\lambda_j\rangle_\alpha |u_j\rangle_\alpha \tag{2.5.15}$$

que resulta em:

$$|\psi_2\rangle = \sum_j \left(cos\frac{\phi}{2}|0\rangle + sen\frac{\phi}{2}|1\rangle \right) \otimes p_j |\lambda_j\rangle_\alpha |u_j\rangle_\alpha \tag{2.5.16}$$

Pela identidade:

$$cos^2\frac{\phi}{2} + sen^2\frac{\phi}{2} = 1 \;\rightarrow\; cos\frac{\phi}{2} = \sqrt{1 - sen^2\frac{\phi}{2}} \tag{2.5.17}$$

e escolhendo a constante *C* tal que:

$$\frac{C}{\lambda_j} = sen\frac{\phi_j}{2} \tag{2.5.18}$$

o estado $|\psi_2\rangle$ pode ser reescrito como:

$$|\psi_2\rangle = \sum_j \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle \right) \otimes p_j |\lambda_j\rangle_\alpha |u_j\rangle_\alpha \tag{2.5.19}$$

onde $0 < C \leq min\{\lambda_j\}$, ou seja, $C = \mathcal{O}(1/\kappa)$. Para valores de $\phi/2$ pequenos, $sen(\phi/2) \approx \phi/2$, o que implica em $C \approx \dfrac{\phi\lambda_j}{2}$.

2.5.2.4 Computação reversa

Pela reversibilidade das portas quânticas unitárias, pode-se desfazer as operações aplicadas nos passos anteriores: operadores hamiltonianos do tipo e^{iBt} e a transformada de Fourier inversa $FT^\dagger$ serão revertidas por rotações em $-\phi$, operadores hamiltonianos em e^{-iBt} e pela transformada de Fourier FT, respectivamente. Os autovalores $|\lambda_j\rangle_\alpha$ serão desemaranhados ao estado $|0\rangle_\alpha$, assim:

$$|\psi_3\rangle = \sum_j \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle \right) p_j |0\rangle_\alpha |u_j\rangle_\alpha \tag{2.5.20}$$

2.5.2.5 Medida

A última etapa consiste na medida do *q-bit ancilla* e é responsável por determinar o sucesso do algoritmo. Levando em conta apenas o *q-bit ancilla* e o registrador α e aplicando a propriedade distributiva, o estado pode ser reescrito como:

$$|\psi\rangle = \sum_j \sqrt{1 - \frac{C^2}{\lambda_j^2}} p_j |0\rangle |u_j\rangle + \sum_j \frac{C}{\lambda_j} p_j |1\rangle |u_j\rangle \tag{2.5.21}$$

Chamando de $|\Phi\rangle$ a parcela do vetor de estados que compreende o estado $|1\rangle$ do *q-bit ancilla*. Pela Equação 2.5.4:

$$|\Psi\rangle = \sum_j \frac{C}{\lambda_j} p_j |u_j\rangle = C|\theta\rangle \;\rightarrow\; |\theta\rangle = \frac{|\Psi\rangle}{C} \tag{2.5.22}$$

Com efeito, o vetor solução está contido no vetor de estados. Contudo, em um computador quântico é necessário realizar uma medida para se extrair informação, segue portanto que se for medido $|1\rangle$ *q-bit ancilla*, o estado é transformado para a expressão:

$$\sqrt{\frac{1}{\sum_j C^2 |p_j|^2 / |\lambda_j|^2}} \sum_j \frac{C}{\lambda_j} p_j |u_j\rangle \tag{2.5.23}$$

que pode ser simplificada para

$$\sqrt{\frac{1}{\sum_j |p_j|^2 / |\lambda_j|^2}} \sum_j \lambda_j^{-1} p_j |u_j\rangle \tag{2.5.24}$$

que é proporcional ao estado que se busca calcular $|\theta\rangle = \sum \lambda_j^{-1} p_j |u_j\rangle$.

A probabilidade de se obter $|1\rangle$, isto é, de sucesso do algoritmo, é:

$$p(obter|1\rangle) = \sum_j \frac{C^2}{|\lambda_j|^2} |p_j|^2 \tag{2.5.25}$$

O algoritmo deve ser executado até que a medida bem sucedida seja obtida. A probabilidade é maximizada escolhendo o maior valor para *C*, isto é, $C = \lambda_{min}$

A leitura de todas as componentes de θ, no entanto, demandaria a execução do algoritmo pelo menos *N* vezes. Contudo, se o interesse for apenas um valor médio $\theta^T M \theta$, é possível por meio de um operador observável M obter uma estimativa da média, isto é, $\langle\theta|M|\theta\rangle$. Uma grande variedade de propriedades do vetor θ pode ser extraída desta forma, incluindo a normalização, momentos e pesos em diferentes partes do espaço de estados.

2.5.3 Complexidade Computacional

Nesta subseção, será apresentada a complexidade computacional do algoritmo HHL, discutida pelos autores no artigo original (HARROW; HASSIDIM; LLOYD, 2009), comparando-a com os algoritmos clássicos e solução de equações lineares.

A simulação hamiltoniana de e^{iBt}, supondo *B* s-esparsa, pode ser feita em $\mathcal{O}(log(N)s^2t)$. Na etapa de QPE, foi assumida a hipótese de que os autovalores possam ser representados corretamente em α *bits*, todavia, para um valor não exato, a estimativa de fase introduzirá um erro de ordem $\mathcal{O}(1/t)$ enquanto estima o autovalor λ que será propagado em um erro relativo de $\mathcal{O}(1/\lambda t)$ pela inversão de λ. Como $\lambda \geq 1/\kappa$, tomando $t = \mathcal{O}(\kappa/\epsilon)$, o erro aproximado ao final do processo será da ordem $\mathcal{O}(\epsilon)$.

A probabilidade de uma medição com sucesso do *q-bit* ancilla, como $C = \mathcal{O}(1/\kappa)$, é de pelo menos $\Omega(1/\kappa^2)$. Utilizando o algoritmo de amplificação de amplitude, apresentado por Brassard et al. (2002), para diminuir o número de tentativas de obtenção do caso de sucesso, serão necessárias no máximo $\mathcal{O}(\kappa)$ repetições. A complexidade do algoritmo é portanto:

$$\mathcal{O}(log(N)s^2t)\ \mathcal{O}(\kappa) = \mathcal{O}\left(\frac{log(N)s^2\kappa}{\epsilon}\right)\mathcal{O}(\kappa) = \mathcal{O}\left(\frac{log(N)s^2\kappa^2}{\epsilon}\right) \tag{2.5.26}$$

O método clássico da eliminação de Gauss possui uma complexidade $\tilde{\mathcal{O}}(N^3\ log(||B|| + ||p||))$, em que a notação $\tilde{\mathcal{O}}$ indica alguma possível omissão de um fator logaritmo nas variáveis (EBERLY et al., 2006). Todavia, o melhor algoritmo clássico de solução de equações lineares, o método do gradiente conjugado, requer para um sistema de tamanho *N* e s-esparso $\mathcal{O}(Ns\kappa\ log(1/\epsilon))$ (SHEWCHUK, 1994). Em comparação, para complexidade de tempo, a dependência do algoritmo quântico em *N* é exponencialmente melhor enquanto a dependência em ϵ é exponencialmente pior. Trabalho posteriores melhoraram o algoritmo HHL para dependência linear de κ (AMBAINIS, 2010) e a dependência exponencialmente pior de ϵ foi eliminada

com $poly(log(1/\epsilon)$ (CHIA; LIN; WANG, 2018).

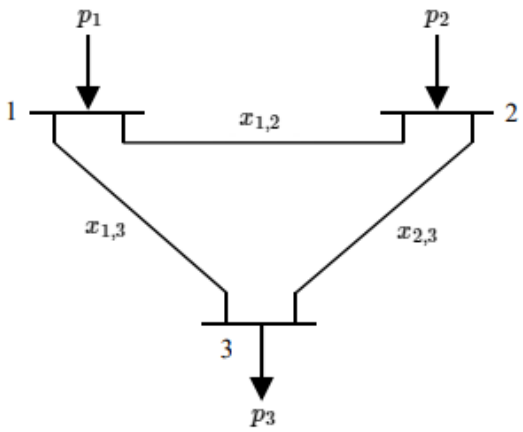
2.6 TRABALHOS CORRELATOS

Será apresentado nesta seção os trabalhos encontrados na literatura sobre a aplicação da computação quântica ao problema do fluxo de potência. Antes de prosseguir é necessário citar que as todas implementações a serem expostas foram desenvolvidas no *Qiskit*.

2.6.1 Eskandarpour et al. (2020a)

No artigo desenvolvido por Eskandarpour et al. (2020a), os autores realizam a simulação do fluxo de potência CC de um sistema elétrico de três barras, da Figura 27, que conta com reatâncias série $x_{1,2}$, $x_{1,3}$ e $x_{2,3}$, de valores $0.0125\ pu$, $0.0125\ pu$ e $0.05\ pu$, respectivamente, e potências p_1, p_2 e p_3, de $20\ MW$, $60\ MW$ e $80\ MW$ respectivamente.

Figura 27 – Sistema de estudo de Eskandarpour et al. (2020a).



Fonte: Adaptado de Eskandarpour et al. (2020a).

Ao adotar a barra 1 como referência, resulta no sistema matricial

$$\begin{bmatrix} 1 & -0.2 \\ -0.2 & 1 \end{bmatrix} \begin{bmatrix} \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix} \tag{2.6.1}$$

O sistema foi simulado no algoritmo HHL pré-programado do *Qiskit Aqua* (QISKIT, 2021c,d), obtendo os resultados da Tabela 2.

Tabela 2 – Resultados da simulação de Eskandarpour et al. (2020a).

Solução clássica	0.4583 -0.7083
Solução quântica	0.4534-0.i -0.7123+0.i
Fidelidade	0.999945
Probabilidade	0.020986

Fonte: Adaptado de Eskandarpour et al. (2020a).

Contudo, o sistema simulado possui 7 *q-bits* e um número elevado de portas lógicas, com o valor de 102 *depth*, que é um parâmetro que indica o número de portas executadas no caminho mais longo do circuito quântico, e 54 portas *CNOT*. Para implementação em computador quântico real no momento que o artigo foi desenvolvido, essa quantidade deve ser reduzida. Os autores projetaram o circuito quantico de 4 *q-bits* mostrado na Figura 28 e em uma simulação em um ambiente com ruído obtiveram um valor de 1.1810 para a propriedade $(\theta_2 - \theta_3)^2$, que possui valor exato de 1.3609.

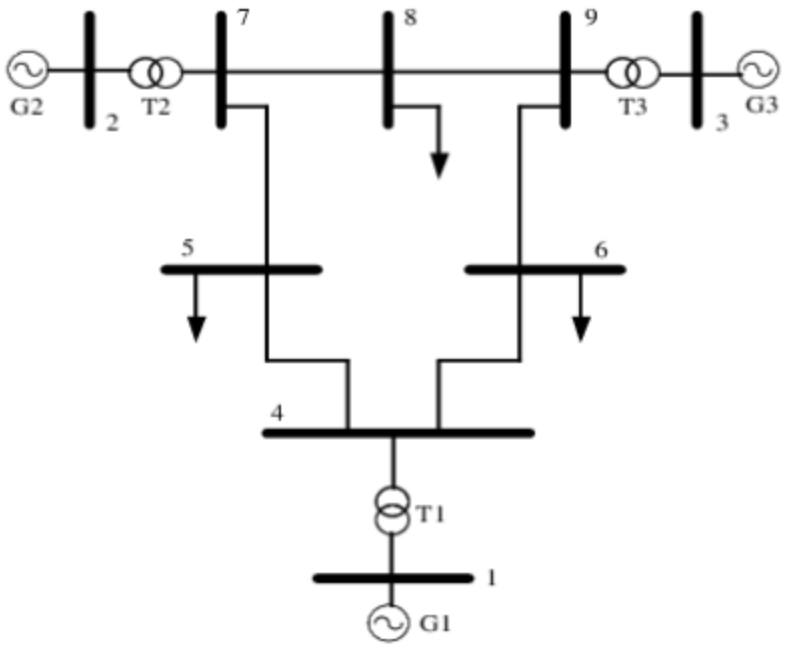
Figura 28 – Circuito quântico projetado por Eskandarpour et al. (2020a).

Fonte: Adaptado de Eskandarpour et al. (2020a).

2.6.2 Eskandarpour et al. (2021)

Eskandarpour et al. (2021) apresenta uma implementação experimental do fluxo de potência CC para o sistema de nove barras ilustrado na Figura 29, que é o sistema de teste *Western System Coordinating Council* (WSCC) (DELAVARI, 2021). Os autores obtiveram os resultados da Tabela 3, em um sistema com *depth* igual 129, 70 portas *CNOT* e 9 *q-bits*.

Figura 29 – Sistema de estudo de Eskandarpour et al. (2021).



Fonte: Eskandarpour et al. (2021).

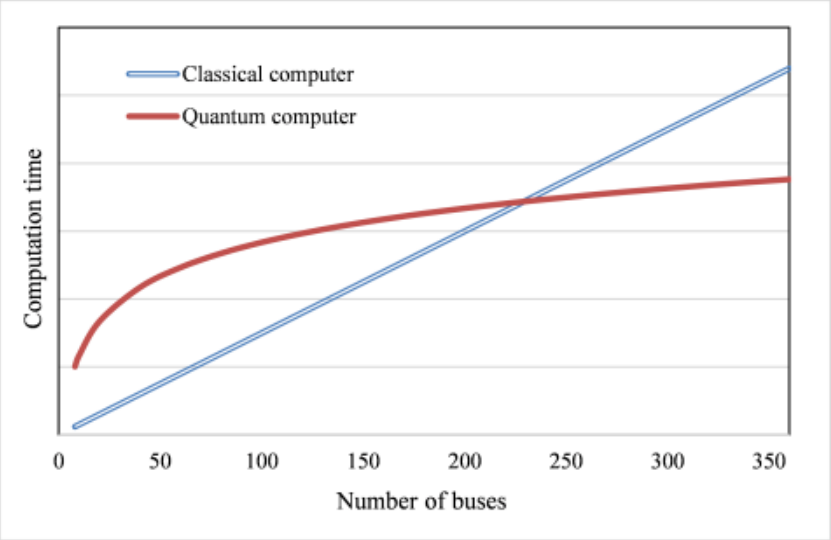
Tabela 3 – Resultados obtidos por Eskandarpour et al. (2021).

Solução clássica	0.1157 0.0948 -0.0713 -0.1316 -0.0644 0.0118 -0.0514 0.0220
Solução quântica	0.1125 0.0599 -0.0299 -0.0826 -0.0458 0.0047 -0.0858 -0.0040
Fidelidade	0.872159
Probabilidade	0.537650

Fonte: Adaptado de Eskandarpour et al. (2021).

O resultado obtido apresenta uma fidelidade que segundo os autores demonstra a promessa em encontrar soluções acuradas, apesar do ruído e da larga escala do circuito quântico. O artigo também apresenta um estudo comparativo de desempenho computacional com base na complexidade do algoritmo HHL e na do método clássico do gradiente conjugado. A Figura 30 ilustra um gráfico que compara o tempo computacional clássico e quântico em função do número de barras do sistema. A interseção ocorre em $N = 230$, acima do limiar 230, de acordo com as premissas conservadoras assumidas pelos autores, o desempenho quântico se sobressai em relação com clássico. Cabe ressaltar que se trata de uma discussão teórica e que a comparação experimental não é possível no ponto em que o artigo foi desenvolvido.

Figura 30 – Comparação entre o tempo computacional clássico e quântico em função do número de barras do sistema feita por Eskandarpour et al. (2021).

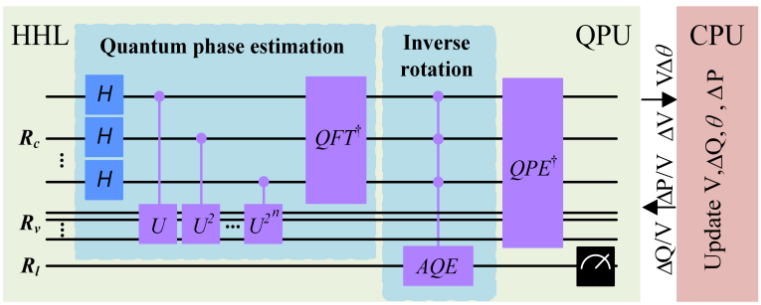


Fonte: Eskandarpour et al. (2021).

2.6.3 Feng, Zhou e Zhang (2021)

O artigo Feng, Zhou e Zhang (2021) é uma prova de conceito para a primeira arquitetura quântica para solução de fluxo de potência por métodos iterativos. O método, nomeado fluxo de potência quântico (*Quantum Power Flow* - QPE), é baseado em um dos métodos desacoplados, o desacoplado rápido, de excelente convergência e eficiência, e adota matrizes jacobianas hermitianas e constantes (o método desacoplado rápido é explicado por Monticelli (1983)). A Figura 31 mostra a arquitetura quântica do QPF. Em cada iteração as variáveis são atualizadas através do controle do fluxo quântico de forma clássica e as equações são resolvidas pelo algoritmo de HHL.

Figura 31 – Arquitetura quântica do QPF proposta por Feng, Zhou e Zhang (2021).

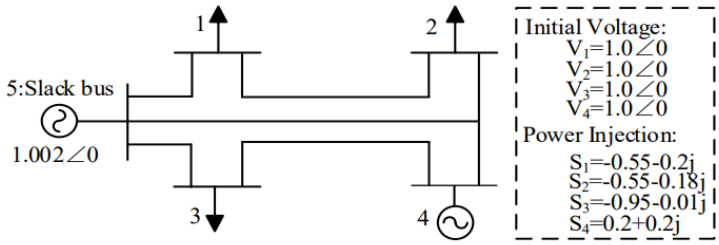


Fonte: Feng, Zhou e Zhang (2021).

O sistema de estudo é o sistema de cinco barras mostrado na Figura 32. Na Figura 33 é mostrado os resultados para as tensões e ângulos V_3, V_4, θ_3 e θ_4 com uma convergência de 10^{-5} do

QPF comparando com os métodos desacoplado rápido e o de Newton-Raphson clássicos. Com efeito, o QPF apresentou como resultado uma convergência idêntica aos métodos clássicos.

Figura 32 – Sistema de estudos de Feng, Zhou e Zhang (2021).



Fonte: Feng, Zhou e Zhang (2021).

Figura 33 – Resultados obtidos por Feng, Zhou e Zhang (2021).

Algoritmo	Iteração	$\mathbf{V}_3$	$\mathbf{V}_4$	$\boldsymbol{\theta}_3$	$\boldsymbol{\theta}_4$
QPF	1	1.0141	1.0282	-0.1143	-0.0368
	2	0.9946	1.0181	-0.1139	-0.0340
	3	0.9950	1.0183	-0.1144	-0.0393
	4	0.9948	1.0182	-0.1144	-0.0393
	5	0.9948	1.0182	-0.1144	-0.0393
	6*	0.9948	1.0182	-0.1144	-0.0393
Desacoplado rápido clássico	1	1.0141	1.0282	-0.1143	-0.0368
	2	0.9946	1.0181	-0.1139	-0.0340
	3	0.9950	1.0183	-0.1144	-0.0393
	4	0.9948	1.0182	-0.1144	-0.0393
	5	0.9948	1.0182	-0.1144	-0.0393
	6*	0.9948	1.0182	-0.1144	-0.0393
Newton-Raphson clássico	1	1.0092	1.0251	-0.1136	-0.0382
	2	0.9951	1.0183	-0.1144	-0.0392
	3*	0.9948	1.0182	-0.1144	-0.0392

Fonte: Adaptado de Feng, Zhou e Zhang (2021).

2.6.4 Eskandarpour et al. (2020b)

O artigo Eskandarpour et al. (2020b) tem como objeto de estudo, a análise de segurança do sistema de energia, que é comumente avaliada considerando prováveis falhas de componentes, que podem exigir a resolução de milhares de cenários de fluxo de energia, cada um considerando uma diferente contingência. Atualmente, a pesquisa concentra-se no desenvolvimento de novos dados estatísticos e métodos baseados em incerteza, em contraste á uma necessidade cada vez menor de estudos determinísticos, devido ao número crescente de desastres naturais que resultam na interrupção simultânea de vários componentes da rede.

Nesses estudos, baseados na formulação de fluxo de potência, sendo os cenários identificados pelo índice s, as barras por m e as linhas por mn, o tipo de estudo é indicado por $N-m$.

Em um estudo $N-1$, o mais comumente utilizado, o número de cenários é $s \in \{1, ..., N\}$, com N sendo o número de barras do sistema. Já para estudos $N-2$, $s \in \{1, ..., N(N-1)/2\}$, Com efeito, o número de cenários cresce consideravelmente, de modo que esses estudos são limitados pela falta de recursos computacionais adequados. Os autores avaliam a computação quântica como alternativa analisando o sistema padrão *IEEE 300-bus test system* (CHRISTIE, 1993), que conta com 300 barras, 69 geradores, 304 linhas de transmissão e 195 cargas, para estudos $N-m$ resolvendo o fluxo de potência com computação quântica. Em estudos $N-1$, as equações devem ser resolvidas 373 vezes, em estudos $N-2$, 69000 vezes e em $N-3$, 8.4 milhões. Considerando que cada execução dure $100\ ms$, a Tabela 4 indica o tempo computacional para os três tipos de estudos no caso clássico e o no caso quântico.

Tabela 4 – Comparativo de tempo computacional teórico feito por Eskandarpour et al. (2020b).

Tipo de contingência	N-1	N-2	N-3
Computador clássico	37 segundos	2 horas	10 dias
Computador quântico	0.3 segundos	0.5 segundos	0.7 segundos

Fonte: Adaptado de Eskandarpour et al. (2020b).

Com efeito, um estudo que leva 10 dias classicamente, teoricamente poderia ser feito em computador quântico em 0.7 segundos por conta do ganho de tempo exponencial do algoritmo HHL sobre suas contrapartes clássicas, demonstrando matematicamente o potencial teórico da computação quântica na melhoria de desempenho da rede elétrica.

3 MÉTODO PROPOSTO

O objetivo deste trabalho é a realização da simulação e experimentação de fluxo de potência CC utilizando computação quântica. O objeto de estudo são dois sistemas, um de três e outro de cinco barras, que serão apresentados na Seção 3.1. Na Seção 3.2 será introduzido o simulador, apresentando o programa *IBM Q Experience*, a linguagem Qasm e finalmente o *kit* de desenvolvimento de *software* de código aberto *Qiskit*, que será utilizado para construção do algoritmo. Na Seção 3.3 será discutido o algoritmo construído, destacando seus aspectos gerais e específicos, além de realizar comentários sobre o código equivalente construído em Matlab (MATLAB, 2018). Na Seção 3.4 será apresentado um modelo de simulação de fluxo de potência clássico tradicional por métodos iterativos para o Simulink (DOCUMENTATION, 2021), utilizado para fins comparativos com o estudo de fluxo de potência CC para o sistema de cinco barras. Na Seção 3.5 será apresentado o algoritmo HHL pré-programado disponibilizado no *Qiskit Aqua* (QISKIT, 2021c,d), utilizado para fins comparativos com os dois sistemas de estudo.

3.1 SISTEMAS PROPOSTOS

Os dois sistemas terão aspectos distintos a serem considerados. O primeiro, de três barras, terá uma matriz admitância 2x2, o caso mais simples do algoritmo de HHL. Os autovalores serão corretamente representados em dois *q-bits* e a solução quântica terá valor teórico exato. A solução numérica será apresentada com o objetivo de expor detalhadamente todas as etapas do algoritmo.

O segundo sistema, em contrapartida, terá solução teórica aproximada, como consequência da representação aproximada de seus autovalores. Por possuir cinco barras, a matriz admitância será 4x4 e consequentemente sua complexidade será maior, impossibilitando uma solução numérica aos moldes da apresentada para o sistema anterior. O objetivo é expor a versatilidade do algoritmo HHL, demonstrando que ele pode fornecer soluções aproximadas nos casos em que não é feita a representação exata dos autovalores.

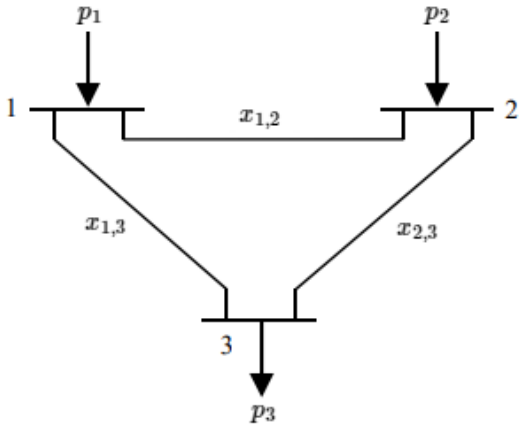
Para o primeiro sistema, além da simulação, será realizada a implementação experimental em um computador quântico real disponibilizado pela IBM, um objetivo que por outro lado atualmente é inviável para segundo sistema, devido a maior complexidade computacional de seu circuito quântico.

3.1.1 Sistema de três barras

O sistema, de mesma topologia que o sistema de estudo de Eskandarpour et al. (2020a), é mostrado na Figura 34. Este sistema, de base $100\ MW$, não possui transformadores em fase e defasadores e conta com três linhas de transmissão com reatâncias série $x_{1,2}$, $x_{1,3}$ e $x_{2,3}$, de valores $0.5\ pu$, $0.5\ pu$ e $0.5\ pu$, respectivamente. As barras 1 e 2 contam com geração de

potência ativa p_1 e p_2, de $0.2\ pu$ e $0.6\ pu$, respectivamente, e a barra 3 possui carga acoplada com potência ativa p_3 de $0.8\ pu$. A barra 1 é escolhida como referência ($\theta_1 = 0$) e as barras 2 e 3 são dos tipos *PV* e *PQ*, respectivamente.

Figura 34 – Sistema de três barras.



Fonte: Adaptado de Eskandarpour et al. (2020a).

O sistema na forma matricial $p = B\theta$ é dado por

$$\begin{bmatrix} p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} x_{1,2}^{-1} + x_{2,3}^{-1} & -x_{2,3}^{-1} \\ -x_{2,3}^{-1} & x_{1,3}^{-1} + x_{2,3}^{-1} \end{bmatrix} \begin{bmatrix} \theta_2 \\ \theta_3 \end{bmatrix}$$
$$\begin{bmatrix} 0.6 \\ -0.8 \end{bmatrix} = \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} \theta_2 \\ \theta_3 \end{bmatrix} \tag{3.1.1}$$

O problema de fluxo de potência consiste em encontrar a solução da equação matricial, o vetor θ, através do algoritmo HHL. A solução exata pode ser obtida pelos métodos clássicos, como o da eliminação gaussiana, explanado em Lay (2003).

$$\theta = \begin{bmatrix} 0.066667 \\ -0.16667 \end{bmatrix} rad \rightarrow \quad \theta = \begin{bmatrix} 3.8197^\circ \\ -9.5493^\circ \end{bmatrix} \tag{3.1.2}$$

O fluxo de potência ativa entre as barras é calculado por:

$$\begin{aligned} P_{12} &= x_{1,2}\theta_{12} = 0.5^{-1}\,(-0.066667) = -0.13333pu \\ P_{13} &= x_{1,3}\theta_{13} = 0.5^{-1}\,0.16667 = 0.33333pu \\ P_{23} &= x_{2,3}\theta_{23} = 0.5^{-1}\,0.23333 = 0.46666pu \end{aligned} \tag{3.1.3}$$

3.1.1.1 Solução numérica

A solução numérica do sistema desenvolvida a seguir foi inspirada em Hector Jose Morrell e Wong (2021) e Eskandarpour et al. (2020a).

A matriz *B* possui autovalores $\lambda_1 = 2$ e $\lambda_2 = 6$ com respectivos autovetores associados:

$$|u_1\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}-1\\-1\end{bmatrix} \; e \; |u_2\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}-1\\1\end{bmatrix} \tag{3.1.4}$$

A razão λ_1/λ_2 igual a 3, de modo que são necessários dois *q-bits* no registrador α ($\alpha = 2$), para acomodá-los nos estados $|01\rangle$ e $|11\rangle$, ou seja, $\lambda_1^o = 1$ e $\lambda_2^o = 3$, respectivamente. A constante *t* é obtida pela Equação 2.5.12.

$$t = \frac{2\pi 1}{2^2 2} = \frac{2\pi 3}{2^2 6} = \frac{\pi}{4} \tag{3.1.5}$$

O algoritmo tem seu início com:

$$|\psi_0\rangle = |0\rangle \otimes |00\rangle_\alpha \otimes |p\rangle_\beta \tag{3.1.6}$$

Escrevendo $|p\rangle_\beta$ na base de *B*, $|p\rangle_\beta = 0.6|0\rangle - 0.8|1\rangle = 0.1\sqrt{2}|u_1\rangle - 0.7\sqrt{2}|u_2\rangle$, o vetor $|\psi_0\rangle$ se torna:

$$|\psi_0\rangle = |0\rangle \otimes |00\rangle_\alpha \otimes |p\rangle_\beta = |0\rangle \otimes |00\rangle_\alpha \otimes (0.1\sqrt{2}|u_1\rangle - 0.7\sqrt{2}|u_2\rangle) \tag{3.1.7}$$

Aplicando a transformação Hadamard ao registrador α:

$$\begin{aligned} |\psi_1\rangle &= \mathbb{I} \otimes H^{\otimes 2} \otimes \mathbb{I}|\psi_0\rangle \\ &= |0\rangle \otimes 0.5(|00\rangle + |01\rangle + |00\rangle + |01\rangle) \otimes (0.1\sqrt{2}|u_1\rangle - 0.7\sqrt{2}|u_2\rangle) \end{aligned} \tag{3.1.8}$$

Para a operação controlada da etapa de estimativa de fase são necessárias duas operações, $U^{2^0} = U = e^{iBt}$ e $U^{2^1} = U^2 = e^{i2Bt}$, que são obtidas pela diagonalização da matriz *B*, $B = VB_dV^\dagger$, com

$$V = \frac{1}{\sqrt{2}}\begin{bmatrix}-1 & -1\\-1 & 1\end{bmatrix} \quad e \quad B_d = \begin{bmatrix}2 & 0\\0 & 6\end{bmatrix} \tag{3.1.9}$$

Como V é hermitiana, $V^\dagger = V$. Obtendo as matrizes diagonais de U e U^2 através da exponenciação dos elementos, de modo que:

$$U_d = \begin{bmatrix} e^{i\lambda_1 t} & 0 \\ 0 & e^{i\lambda_2 t} \end{bmatrix} = \begin{bmatrix} e^{i2\pi/4} & 0 \\ 0 & e^{i6\pi/4} \end{bmatrix} = \begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix} \quad . \tag{3.1.10}$$
$$U_d^2 = \begin{bmatrix} e^{2i\lambda_1 t} & 0 \\ 0 & e^{2i\lambda_2 t} \end{bmatrix} = \begin{bmatrix} e^{2i2\pi/4} & 0 \\ 0 & e^{2i6\pi/4} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Assim, as matrizes U e U^2 são obtidas mantendo a base original

$$U = VU_dV^\dagger = \begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix} \quad . \tag{3.1.11}$$
$$U^2 = VU_d^2V^\dagger = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Os operadores U e U^2 devem ser aplicados ao *q-bit* alvo β controlados respectivamente pelos *q-bits* α_1 e α_2. Posteriormente, a transformada inversa de Fourier, construída invertendo o circuito da Figura 22, deve ser aplicada no registrador α. O operador FT^{-1} para dois *q-bits* é

$$FT_2^{-1} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5i & -0.5 & 0.5i \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & 0.5i & -0.5 & -0.5i \end{bmatrix} \tag{3.1.12}$$

Aplicando esses operadores:

$$\begin{aligned} |\psi_2\rangle &= (\mathbb{I}^{\otimes 2} \otimes (|1\rangle\langle 1| \otimes U + |0\rangle\langle 0| \otimes \mathbb{I}) \, |\psi_1\rangle \\ |\psi_3\rangle &= (\mathbb{I} \otimes (|1\rangle\langle 1| \otimes \mathbb{I} \otimes U^2 + |0\rangle\langle 0| \otimes \mathbb{I}^{\otimes 2}) \, |\psi_2\rangle \\ |\psi_4\rangle &= \mathbb{I} \otimes FT_2^{-1} \otimes \mathbb{I}|\psi_3\rangle \\ &= |0\rangle \otimes (0.1\sqrt{2}|01\rangle|u_1\rangle - 0.7\sqrt{2}|11\rangle|u_2\rangle) \end{aligned} \tag{3.1.13}$$

A próxima etapa é a rotação controlada. O valor da constante C é escolhido como $C = \lambda_{min} = 2$ para maximizar a probabilidade de medida $|1\rangle$ no *q-bit ancilla*. A implementação pode ser feita aplicando rotações controladas de controle 01 e 11, o que exige duas operações do tipo $C^2(U)$. A aplicação da rotação apenas ao primeiro *q-bit*, além de atuar no estado $|01\rangle$, também rotaciona o estado $|11\rangle$. Contudo, o valor rotacionado poderia ser compensado no cálculo do ângulo da rotação de controle 11, exigindo operações do tipo $C^2(U)$ e $C^1(U)$, o que diminui a quantidade de portas necessárias. Uma terceira implementação pode ser realizada

rotacionando o segundo *q-bit* com controle 1, que afeta apenas o estado $|11\rangle$ e mesmo o segundo *q-bit* com controle 0, invertendo o autovalor do estado $|01\rangle$. O resultado é a necessidade apenas de operações do tipo $C^1(U)$. Assim, os ângulos de rotação são calculados isolando ϕ na Equação 2.5.18.

$$\begin{aligned}\phi_1 &= 2\ sen^{-1}(2/2) = \pi \\ \phi_2 &= 2\ sen^{-1}(2/6) = 0.6797\end{aligned} \tag{3.1.14}$$

Aplicando as rotações $R_y(\phi_1)$ e $R_y(\phi_2)$:

$$\begin{aligned}
|\psi_5\rangle &= \mathbb{I} \otimes X \otimes \mathbb{I}^{\otimes 2}\ |\psi_4\rangle \\
|\psi_6\rangle &= \ (R_y(\pi) \otimes |1\rangle\langle 1| + \mathbb{I} \otimes |0\rangle\langle 0|) \otimes \mathbb{I}^{\otimes 2}\ |\psi_5\rangle \\
|\psi_7\rangle &= \mathbb{I} \otimes X \otimes \mathbb{I}^{\otimes 2}\ |\psi_6\rangle \\
|\psi_8\rangle &= \ (R_y(0.6797) \otimes |1\rangle\langle 1| + \mathbb{I} \otimes |0\rangle\langle 0|) \otimes \mathbb{I}^{\otimes 2}\ |\psi_7\rangle \\
&= 0.1\sqrt{2}\left(\sqrt{1-\frac{2^2}{2^2}}|0\rangle + \frac{2}{2}|1\rangle\right)|01\rangle|u_1\rangle - 0.7\sqrt{2}\left(\sqrt{1-\frac{2^2}{6^2}}|0\rangle + \frac{2}{6}|1\rangle\right)|11\rangle|u_2\rangle \\
&= 0.1\sqrt{2}|1\rangle|01\rangle|u_1\rangle - 0.7\sqrt{2}\left(\frac{2\sqrt{2}}{3}|0\rangle + \frac{1}{3}|1\rangle\right)|11\rangle|u_2\rangle
\end{aligned} \tag{3.1.15}$$

A computação reversa deve ser aplicando em sequência os operadores FT, U^{-2}, U^{-1} e a transformação Hadamard.

$$\begin{aligned}
FT_2 &= \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5i & -0.5 & -0.5i \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & -0.5i & -0.5 & 0.5i \end{bmatrix}. \\
U^{-2} &= \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \\
U^{-1} &= \begin{bmatrix} 0 & -i \\ -i & 0 \end{bmatrix}
\end{aligned} \tag{3.1.16}$$

$$
\begin{aligned}
|\psi_9\rangle &= \mathbb{I} \otimes FT_2 \otimes \mathbb{I}|\psi_8\rangle \\
|\psi_{10}\rangle &= \left(\mathbb{I} \otimes (|1\rangle\langle 1| \otimes \mathbb{I} \otimes U^{-2} + |0\rangle\langle 0| \otimes \mathbb{I}^{\otimes 2}\right) |\psi_9\rangle \\
|\psi_{11}\rangle &= \left(\mathbb{I}^{\otimes 2} \otimes (|1\rangle\langle 1| \otimes U^{-1} + |0\rangle\langle 0| \otimes \mathbb{I}\right) |\psi_{10}\rangle \\
|\psi_{12}\rangle &= \mathbb{I} \otimes H^{\otimes 2} \otimes \mathbb{I}|\psi_1 1\rangle \\
&= 0.1\sqrt{2}|1\rangle|00\rangle|u_1\rangle - 0.7\sqrt{2}\left(\frac{2\sqrt{2}}{3}|0\rangle + \frac{1}{3}|1\rangle\right)|00\rangle|u_2\rangle \\
&= 0.1\sqrt{2}|1\rangle|00\rangle|u_1\rangle - \frac{7\sqrt{2}}{30}|1\rangle|00\rangle|u_2\rangle - \frac{28}{30}|0\rangle|00\rangle|u_2\rangle \\
&= |1\rangle(0.1333|000\rangle - 0.3333|001\rangle) + |0\rangle(0.66|000\rangle - 0.66|001\rangle)
\end{aligned}
\tag{3.1.17}
$$

A última etapa é a medida do *q-bit ancilla*. A probabilidade de medida $|1\rangle$ é:

$$
p(obter|1\rangle) = |0.1333|^2 + |-0.3333|^2 = 0.1289 \tag{3.1.18}
$$

resultando no estado após a medida

$$
|\psi_{13}\rangle = |00\rangle(0.3714|0\rangle - 0.9285|1\rangle) \tag{3.1.19}
$$

A solução pode ser obtida diretamente no vetor de estados $|\psi_{10}\rangle$ na parcela do *q-bit* β do estado $|1\rangle$ do *q-bit ancilla*, dividindo pela constante *C*.

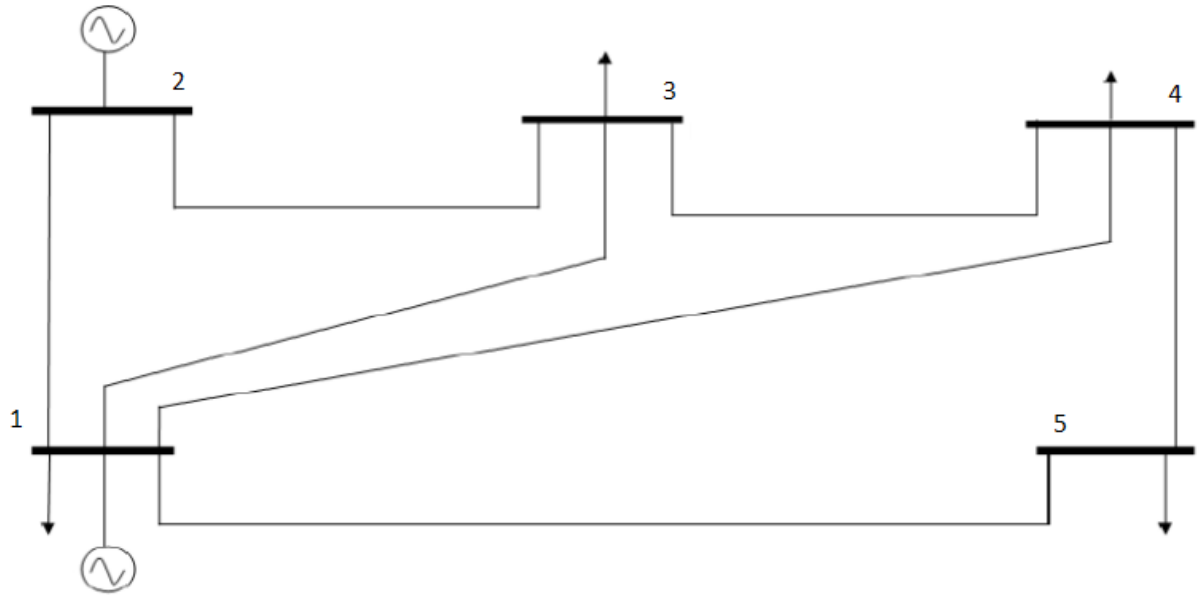
$$
\theta = \frac{1}{2}\begin{bmatrix} 0.1333 \\ -0.3333 \end{bmatrix} \rightarrow \quad \theta = \begin{bmatrix} 0.06667 \\ -0.1667 \end{bmatrix} rad \rightarrow \quad \theta = \begin{bmatrix} 3.8197^\circ \\ -9.5493^\circ \end{bmatrix} \tag{3.1.20}
$$

que é exatamente a solução clássica da Equação 3.1.2.

3.1.2 Sistema de cinco barras

O segundo sistema a ser simulado é o sistema de teste *IEEE 5-Bus System Model* (TAN, 2021), ilustrado na Figura 35. A base do sistema é $100MVA$. A Tabela 6 contém as potências injetadas e consumidas e a Tabela 5 os parâmetros das linhas de transmissão, ambas com os parâmetros arbitradores pelo autor.

Figura 35 – Sistema de cinco barras.



Fonte: Adaptado de Tan (2021).

Tabela 5 – Parâmetros das linhas de transmissão do sistema de 5 barras.

Da barra	Para barra	$r_{km}\ (pu)$	$x_{km}\ (pu)$	$b_{km}^{sh}\ (pu)$
1	2	0.03	0.1	0.01
1	3	0.08	0.25	0.025
1	4	0.08	0.25	0.025
1	5	0.03	0.1	0.01
2	3	0.15	0.5	0.05
3	4	0.05	1/6	0.02
4	5	0.15	0.5	0.05

Fonte: O autor.

Tabela 6 – Potências injetadas e consumidas no sistema de 5 barras.

Barra	MW	MVAr
2	70	0
3	-50	15
4	-50	15
5	-10	3

Fonte: O autor.

A barra 1 é a barra de referência ($\theta_1 = 0$), de modo que suas potências, ativa e reativa, devem ser determinadas no caso de um estudo tradicional de fluxo de potência. A barra 2 é de geração, *PV*, e as barras 3, 4 e 5 são do tipo carga, *PQ*. A formulação matricial do fluxo de potência CC para o sistema é:

$$\begin{bmatrix} p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix} = \begin{bmatrix} x_{1,2}^{-1} + x_{2,3}^{-1} & -x_{2,3}^{-1} & 0 & 0 \\ -x_{2,3}^{-1} & x_{2,3}^{-1} + x_{1,3}^{-1} + x_{3,4}^{-1} & -x_{3,4}^{-1} & 0 \\ 0 & -x_{3,4}^{-1} & x_{3,4}^{-1} + x_{1,4}^{-1} + x_{4,5}^{-1} & -x_{4,5}^{-1} \\ 0 & 0 & -x_{4,5}^{-1} & x_{4,5}^{-1} + x_{1,5}^{-1} \end{bmatrix} \begin{bmatrix} \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix} . \quad (3.1.21)$$

$$\begin{bmatrix} 0.7 \\ -0.5 \\ -0.5 \\ -0.1 \end{bmatrix} = \begin{bmatrix} 12 & -2 & 0 & 0 \\ -2 & 12 & -6 & 0 \\ 0 & -6 & 12 & -2 \\ 0 & 0 & -2 & 12 \end{bmatrix} \begin{bmatrix} \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix}$$

A solução clássica da equação matricial é:

$$\theta = \begin{bmatrix} 0.045727 \\ -0.075638 \\ -0.083185 \\ -0.022198 \end{bmatrix} rad \rightarrow \quad \theta = \begin{bmatrix} 2.6120^\circ \\ -4.3337^\circ \\ -4.7661^\circ \\ -1.2719^\circ \end{bmatrix} \tag{3.1.22}$$

O fluxo de potência ativa entre as barras é calculado por:

$$\begin{aligned} P_{12} &= x_{1,2}\theta_{12} = 0.1^{-1}\,(-0.045727) = -0.45727pu \\ P_{13} &= x_{1,3}\theta_{13} = 0.25^{-1}\,0.075638 = 0.30255pu \\ P_{14} &= x_{1,4}\theta_{14} = 0.25^{-1}\,0.083185 = 0.33274pu \\ P_{15} &= x_{1,5}\theta_{15} = 0.1^{-1}\,0.022198 = 0.22198pu \\ P_{23} &= x_{2,3}\theta_{23} = 0.5^{-1}\,0.12137 = 0.24273pu \\ P_{34} &= x_{3,4}\theta_{34} = (1/6)^{-1}\,0.00757 = 0.04542pu \\ P_{45} &= x_{4,5}\theta_{45} = 0.5^{-1}\,(-0.060987) = -0.12197pu \end{aligned} \tag{3.1.23}$$

3.1.2.1 Parâmetros do circuito quântico

A matriz *B* possui autovalores $\lambda_1 = 5.394$, $\lambda_2 = 11.394$, $\lambda_3 = 12.606$ e $\lambda_4 = 18.606$ com respectivos autovetores associados:

$$|u_1\rangle = \begin{bmatrix} 0.2049 \\ 0.6768 \\ 0.6768 \\ 0.2049 \end{bmatrix},\ |u_2\rangle = \begin{bmatrix} 0.6768 \\ 0.2049 \\ -0.2049 \\ -0.6768 \end{bmatrix},\ |u_3\rangle = \begin{bmatrix} 0.6768 \\ -0.2049 \\ -0.2049 \\ 0.6768 \end{bmatrix}\ e\ \ |u_4\rangle = \begin{bmatrix} -0.2049 \\ 0.6768 \\ -0.6768 \\ 0.2049 \end{bmatrix} \tag{3.1.24}$$

A representação dos autovalores não será exata. O autovalor λ_1 estará contido no estado $|01\rangle$, os autovalores λ_2 e λ_3 no estado $|10\rangle$ e o autovalor λ_4 no estado $|11\rangle$, com $\alpha = 2$. A constante $\bar{t}$ será obtida pelo valor médio para os quatro autovalores através da Equação 2.5.12.

$$\begin{aligned} t_1 = \frac{2\pi 1}{2^2 5.3940} = 0.2912\ ,\ t_2 = \frac{2\pi 2}{2^2 11.394} = 0.2757 \\ t_3 = \frac{2\pi 2}{2^2 12.606} = 0.2492\ ,\ t_4 = \frac{2\pi 3}{2^2 18.606} = 0.2533 \end{aligned}\ . \tag{3.1.25}$$

obtendo $\bar{t} = 0.26735$, com um desvio padrão relativo de 6.383%. A diagonalização da matriz, $B = VB_dV^\dagger$, é

$$
\begin{aligned}
V &= \begin{bmatrix} 0.2049 & 0.6768 & -0.6768 & -0.2049 \\ 0.6768 & 0.2049 & 0.2049 & 0.6768 \\ 0.6768 & -0.2049 & 0.2049 & -0.6768 \\ 0.2049 & -0.6768 & -0.6768 & 0.2049 \end{bmatrix} \\
V^{\dagger} &= \begin{bmatrix} 0.2049 & 0.6768 & 0.6768 & 0.2049 \\ 0.6768 & 0.2049 & -0.2049 & -0.6768 \\ -0.6768 & 0.2049 & 0.2049 & -0.6768 \\ -0.2049 & 0.6768 & -0.6768 & 0.2049 \end{bmatrix} \\
B_d &= \begin{bmatrix} 5.394 & 0 & 0 & 0 \\ 0 & 11.394 & 0 & 0 \\ 0 & 0 & 12.606 & 0 \\ 0 & 0 & 0 & 18.606 \end{bmatrix}
\end{aligned}
\tag{3.1.26}
$$

Os operadores U^1, U^2, U^{-1} e U^{-2} são obtidos através do mesmo procedimento de exponenciação apresentado a solução numérica do sistema anterior.

A etapa da rotação controlada conta com três rotações a serem a realizadas: o autovalor do estado $|01\rangle$ é invertido por controle 0 no segundo *q-bit*, o correspondente ao estado $|10\rangle$ é invertido por controle 0 no primeiro *q-bit* e por fim, para o estado $|11\rangle$ é rotacionado o segundo *q-bit* por controle 1, o que exige a compensação para o estado $|10\rangle$. Novamente, apenas de operações do tipo $C^1(U)$. Escolhendo $C = \lambda_{min} = 5.392$, os ângulos de rotação são calculados.

$$
\begin{aligned}
\phi_3 &= 2\ sen^{-1}(5.394/18.606) = 0.5883 \\
\phi_2 &= 2\ sen^{-1}(5.394/12) - \phi_3 = 0.9325 - 0.5883 = 0.3442 \\
\phi_1 &= 2\ sen^{-1}(5.394/5.394) = \pi
\end{aligned}
\tag{3.1.27}
$$

3.2 SIMULADOR

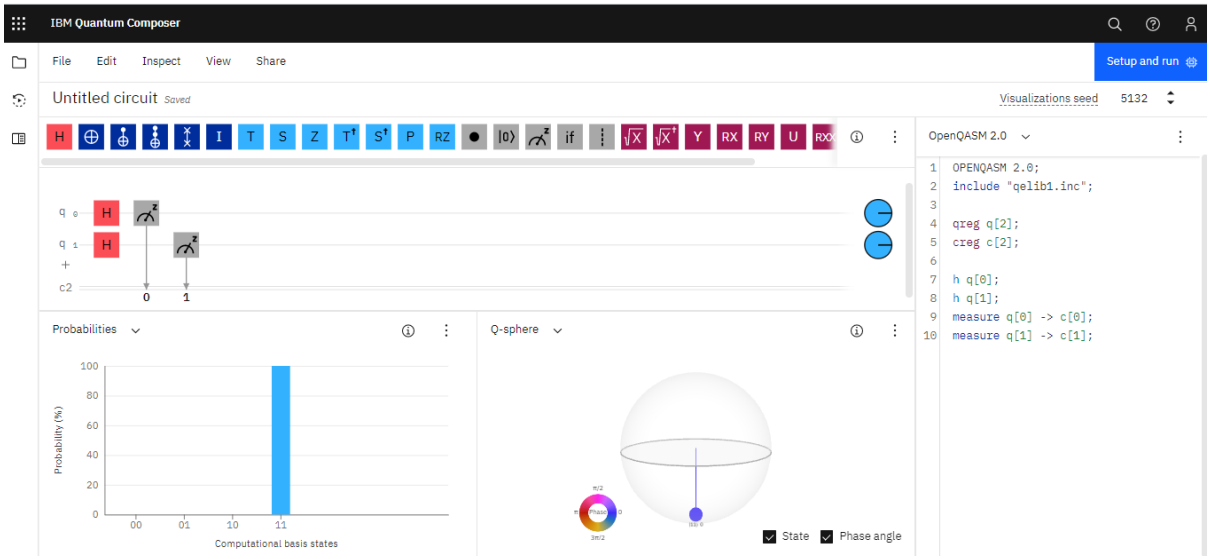
3.2.1 IBM Quantum Experience

No início de 2017, a IBM, empresa pioneira na implementação de computadores quânticos, disponibilizou em nuvem um protótipo de computador quântico com cinco *q-bits*. Posteriormente, a empresa anunciou o programa *IBM Q Experience*, uma plataforma baseada em nuvem para experimentação e aprendizado quântico compartilhado (IBM, 2018). O *composer* da página da IBM é intuitivo, pois consiste em pegar as portas quânticas disponibilizadas e arrastar para o circuito. Contudo, o *composer* pode não ser útil para programas grandes. Neste caso,

pode-se utilizar a linguagem Qasm (*Quantum Assembly Language*), que é uma linguagem de baixo nível do tipo *assembly*, introduzida em 2017 no artigo Cross et al. (2017). A linguagem foi desenvolvida com o intuito de representar os circuitos executáveis no *IBM Q Experience* e portanto é baseada no modelo de circuitos quânticos. Sempre que um circuito é criado por meio do *composer*, um código Qasm é gerado automaticamente e pode ser visualizado e editado diretamente no *browser*.

A Figura 36 mostra um exemplo de circuito no *IBM Q Experience*. O circuito contém dois *q-bits* inicializados no estado $|0\rangle$ (os *q-bits* sempre são inicializados em $|0\rangle$ no *IBM Q Experience*), ambos são colocados em superposição quântica pela porta Hadamard e então medidos. No canto inferior esquerdo é mostrado um gráfico indicando as probabilidade da base computacional, no exemplo a simulação teve a medida $|1\rangle$ para ambos os *q-bits*. Já no canto inferior direito é mostrada a representação do estado quântico na esfera de Bloch. O código Qasm é gerado no canto direito. O código deve começar com o comando `OPENQASM` seguido pela versão do Qasm que está sendo utilizada. Os comandos `qreg` e `creg` são utilizados para declarar um registrador quântico e um clássico, respectivamente (no exemplo, dois *q-bits* e dois bits clássicos). O registrador clássico armazena o resultado da medição no final do algoritmo. Para incluir uma porta lógica, o comando contempla o nome da porta e o registrador (ou registradores) em que ela deve atuar. No exemplo, os comandos para a porta Hadamard nos dois *q-bits* são `h q[0]` e `h q[1]`. Para efetuar uma medição, utiliza-se o comando `measure` seguido do registrador quântico a ser medido e do registrador clássico onde o resultado deve ser armazenado. Para o exemplo, `measure q[0] -> c[0]` e `measure q[1] -> c[1]`.

Figura 36 – Exemplo de circuito no *IBM Q Experience*.



Fonte: IBM (2018).

Conforme Cross et al. (2017), na linguagem Qasm uma porta quântica arbitrária de um *q-bit* pode ser criada por:

$$U(\theta,\phi,\lambda) = R_z(\phi)R_y(\theta)R_z(\lambda) = \begin{pmatrix} e^{-i(\phi+\lambda)/2}cos\frac{\phi}{2} & -e^{-i(\phi-\lambda)/2}sen\frac{\phi}{2} \\ e^{i(\phi-\lambda)/2}sen\frac{\phi}{2} & e^{i(\phi+\lambda)/2}cos\frac{\phi}{2} \end{pmatrix} \tag{3.2.1}$$

Os parâmetros de $U(\theta,\phi,\lambda)$ podem assumir os valores: $\theta \in [0,4\pi)$, $\phi \in [0,4\pi)$ e $\lambda \in [0,4\pi)$. Outras portas presentes na linguagem são: *H, X, Y, Z, S, T*, $S^\dagger$, $T^\dagger$ *CNOT*, *SWAP*, $\mathbb{I}$, *Toffoli*, *U-controlada*, $R_x(\phi)$, $R_y(\phi)$ e $R_z(\phi)$. O comando condicional *if* pode ser executado utilizando um argumento clássico como condição para aplicar um comando quântico. Um *q-bit* pode ser resetado para o estado $|0\rangle$ com o comando `reset`.

O circuito *IBM Q Experience* pode ser executado em um computador quântico real da IBM. Uma vez que a computação quântica é de caráter probabilístico, é possível solicitar que o código seja executado em até 8192 vezes, com o resultado correspondendo a uma distribuição de probabilidades. O circuito da Figura 36 foi executado no computador quântico *ibmq_quito* de 5 *q-bits*. O resultado está ilustrado no histograma da Figura 37. Se esperaria uma frequência igual para cada um dos quatro estados da base computacional, no entanto, devido a imprecisão inerente dos computadores quânticos reais, se observa uma pequena diferença entre os estados.

Figura 37 – Resultado em um computador quântico real.

Fonte: IBM (2018).

3.2.2 Qiskit

O *Qiskit* é um kit de desenvolvimento de *software* de código aberto, baseado na linguagem Python (PYTHON SOFTWARE FOUNDATION, 2021), desenvolvido pela IBM visando facilitar a programação dos computadores quânticos já existentes (ABRAHAM et al., 2021; QISKIT, 2021a,b). Além de permitir a execução diretamente em computadores quânticos reais, o *Qiskit* também facilita a execução de experimentos em simuladores clássicos com recursos de computação de alto desempenho. O *Qiskit* é composto por quatro elementos fundamentais:

- *Terra* - contém todos os fundamentos utilizados para escrever programas quânticos segundo o modelo de circuitos quânticos.

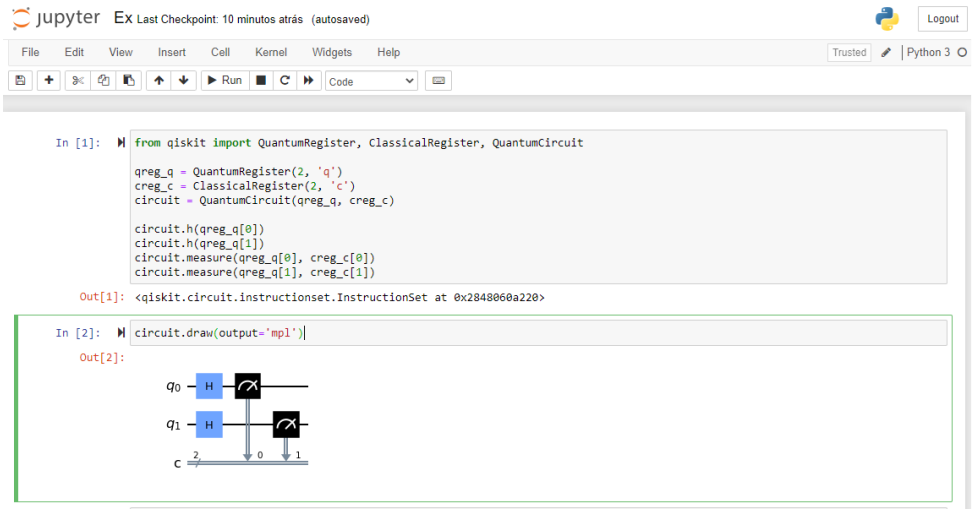
- *Aer* - possui recursos para simulação quânticas por meio de computação clássica de alto desempenho.
- *Aqua* - fornece bibliotecas contendo algoritmos pré programados em aplicações como química quântica, finanças e aprendizado de máquina quântico.
- *Ignis* - contém ferramentas para algoritmos de correção de erros que utilizam filtros, caracterização de ruídos quânticos e verificação de *hardware* quântico.

Além disso, o *IBM Q Provider*, que não é um elemento fundamental, fornece o necessário para acessar o *IBM Q Experience* e executar programas em um computador quântico real. Para isso, é necessário ao usuário importar o `IBMQ` e salvar sua conta no *IBM Q Experience* através do *token* de API, que se encontra nas configurações de *Minha Conta*. Assim, o usuário deve carregar sua conta e poderá escolher entre os processadores disponíveis, que podem ser listados através do comando `provedor.backends()`.

O *software* a ser utilizado é o Jupyter Notebook, que permite programar em Python diretamente no *browser*, com a inclusão de textos explicativos com formatação e facilidade de visualização dos resultados (KLUYVER et al., 2016).

O mesmo circuito da Figura 36 é construído em *Qiskit* utilizando o Terra. O código é ilustrado na Figura 38. O comando `circuito.draw(output='mpl')` fornece a visualização do circuito quântico gerada internamente com o *Matplotlib*. O código em Qasm pode ser gerado com o comando `print(circuit.qasm())`.

Figura 38 – Código em *Qiskit* no Jupyter Notebook.



Fonte: Kluyver et al. (2016).

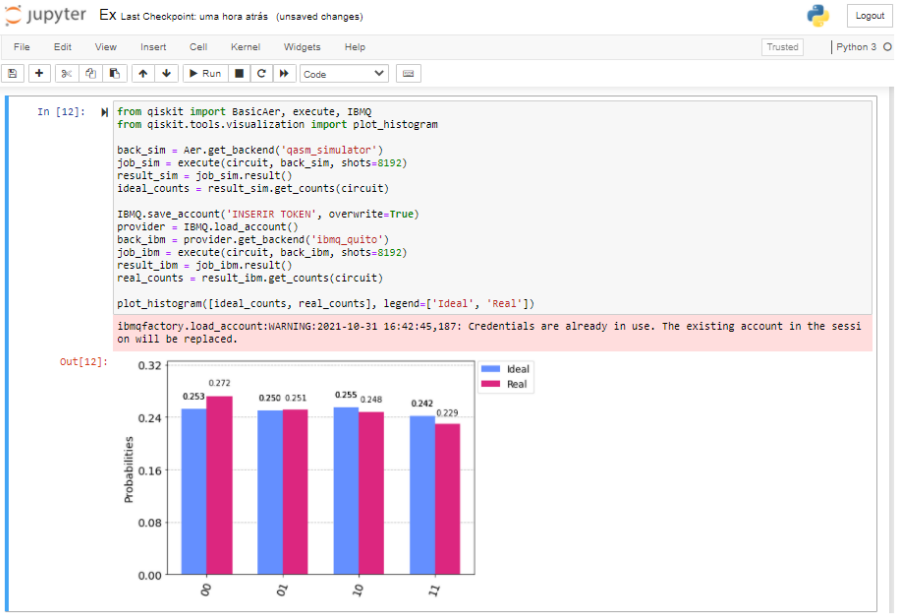
Para simular o circuito, é necessário importar o `BasicAer` e o `execute`. O tipo de simulação é escolhido através dos *backends* disponíveis no `BasicAer`. Para obter uma lista completa é utilizado o comando `BasicAer.backends()`. Atualmente três *backends* estão disponíveis no BasicAer:

- `qasm_simulator` - fornece a simulação fiel ao funcionamento de um computador quântico real.
- `statevector_simulator` - para simulação obtendo o vetor de estados do circuito, corresponde a um *array* de números complexos que podem manipulados.
- `unitary_simulator` - para obtenção da matriz unitária correspondente ao circuito, que só pode ser utilizado em um circuito sem medições, uma vez que a medida não é uma operação unitária.

Para gerar um histograma pelo `qasm_simulator` é necessário importar a ferramenta `plot_histogram`. A simulação é realizada em um processador quântico ideal, isto é, que não leva em consideração o ruído quântico de um processador quântico real, sendo portanto uma emulação numérica.

A Figura 39 ilustra a simulação ideal e a execução em um computador quântico real do circuito da Figura 38. No histograma, em azul os resultados da simulação ideal através do `qasm_simulator` e em rosa os resultados da execução no processador quântico *ibmq_quito* através do *IBM Q Provider*.

Figura 39 – Simulação ideal e a execução em um computador quântico real do circuito no *Qiskit*.



Fonte: Kluyver et al. (2016).

O *Qiskit* contém diversos outros recursos e alguns, utilizados na construção do algoritmo deste trabalho, serão apresentados na próxima seção. Maior aprofundamento sobre as ferramentas do *Qiskit* pode ser encontrado em Jesus et al. (2021). Portugal e Marquezino (2019) apresentam uma introdução à programação de computadores quânticos para o *IBM Q Experience* e o *Qiskit* em forma de tutorial utilizando, dentre outros, os algoritmos de Grover e de Deutsch como exemplo.

3.3 ALGORITMO

O algoritmo foi desenvolvido na versão *Qiskit* 0.32.0 (QISKIT, 2021a,b). O intuito foi o de abarcar o maior número de casos de equações matriciais, resolvendo os dois sistemas de estudo deste trabalho. O escopo engloba matrizes hermitianas de dimensão 2, 3 e 4 que contém apenas autovalores positivos. A fidelidade da solução depende da precisão na representação binária dos autovalores na estimativa de fase.

A primeira parte do algoritmo consiste na determinação dos parâmetros do circuito quântico, que é feita explorando os recursos do *NumPy*, uma biblioteca para Python que suporta o processamento de arranjos e matrizes, com grande coleção de funções matemáticas de alto nível (HARRIS et al., 2020). Já a segunda parte consiste na construção do circuito quântico com os parâmetros determinados. O código completo e comentado se encontra no Apêndice A.

Deve-se ressaltar que o algoritmo construído é uma demonstração e que sua complexidade computacional geral do não foi levada em consideração, especialmente no que diz respeito a rotina de preparação dos parâmetros do circuito quântico, dos quais foram obtidos com auxílio de funções da computação clássica parâmetros, como a da diagonalização da matriz B, que demanda alta complexidade computacional.

Ademais, a construção do algoritmo parte da premissa de um conhecimento prévio dos autovalores (que são calculados classicamente), que é utilizado para a escolha mais adequada da constante t e na etapa de rotação controlada. No entanto, o valor t pode ser arbitrado para um erro tal que $t = \mathcal{O}(\kappa/\epsilon)$ e uma construção para a etapa de rotação controlada que não necessita do conhecimento dos prévio dos autovalores pode ser encontrada em Cao et al. (2012).

A Subseção 3.3.1 fará uma descrição do algoritmo, enquanto a Subseção 3.3.2 fará comentários sobre um código equivalente desenvolvido no *software* Matlab (MATLAB, 2018) que gera arquivos texto com as informações sobre o circuito quântico e os resultados do algoritmo.

3.3.1 Descrição do algoritmo

Os parâmetros de entrada são os *arrays* correspondentes a matriz B e o vetor p; o parâmetro $DPtol$, relacionado a precisão da representação binária e $NQmax$, que é o número de *q-bits* admissível para o registrador α.

3.3.1.1 Rotina de preparação dos parâmetros do circuito quântico

Inicialmente, se verificam as condições, B ser hermitiana ($B = B^{\dagger}$) com apenas autovalores positivos e B e p possuírem dimensões adequadas, isto é, se o sistema for de ordem menor ou igual a 4 com B e p possuindo a mesma dimensão. O algoritmo é interrompido com mensagens de erro caso contrário. Se o sistema matricial for de dimensão 3, ele é convertido em um de dimensão 4 conforme:

$$B = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix},\ p = \begin{bmatrix} * \\ * \\ * \end{bmatrix} \rightarrow\ B = \begin{bmatrix} * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ 0 & 0 & 0 & \lambda_i \end{bmatrix},\ p = \begin{bmatrix} * \\ * \\ * \\ 0 \end{bmatrix} \tag{3.3.1}$$

em que λ_i é um autovalor arbitrário da matriz de modo a prover autovalores repetidos e diminuir o número de rotações controladas necessárias. A transformação resulta na solução do sistema original nos três primeiros elementos de θ e o valor 0 no quarto.

O sistema é normalizado pela função `linalg.norm`, da biblioteca *NumPy*, que retorna a norma de um vetor, através dos comandos

- `B = B / linalg.norm(p)`
- `p = p / linalg.norm(p)`

A diagonalização da matriz *B* é feita para posteriormente auxiliar construção das matrizes U. O primeiro parâmetro do circuito é o número de *q-bits* no registrador β, nb, que é igual a 1 para dimensão 2 e a 2 para dimensão 4. A constante C é determinada como sendo igual ao menor autovalor. O restante dos parâmetros, isto é, o número de *q-bits* do registrador α, nq, o número de rotações controladas nr com seus respectivos ângulos ϕ_i e a constante t, são determinados com base na acomodação dos autovalores na representação binária.

O procedimento é feito da seguinte forma: um laço de repetição é criado com uma variável k, correspondente ao estado em que o menor autovalor deve ser representado, iniciada em 1 e incrementada até nk, um valor correspondente ao $NQmax$, calculado pelo piso da razão entre o maior estado de $NQmax$ *q-bits* ($2^{NQmax} - 1$) e o número de condição s. A posição dos demais autovalores é calculada pelo arredondamento da multiplicação de k com razão entre o autovalor considerado e o menor autovalor. Após, o valor médio do parâmetro t é calculado de forma semelhante a Equação 3.1.25, bem como o seu desvio padrão relativo, DP. Se DP for menor ou igual ao desvio padrão tolerável $DPtol$, o laço é quebrado e os parâmetros nq, nr são calculados e armazenados. Os estados e t também são armazenados. Quando $DP = 0$, ocorre que há uma representação exata dos autovalores. Se $DPtol$ não for alcançado até o final do laço, o algoritmo é interrompido e retorna uma mensagem de erro.

O parâmetro nq depende da posição do maior autovalor, nr depende de em quantas posições diferentes os autovalores foram armazenados, uma vez que um mesmo estado pode representar mais um autovalor, como é o caso do sistema de 5 barras, visto na Subseção 3.1.2.

Por fim, o último etapa da rotina de preparação é a obtenção dos parâmetros de rotação controlada. Conforme exemplificado na solução númerica do sistema de três barras, na Subseção 3.1.1, a rotação pode ser feita de pelo menos três maneiras, que em ordem decrescente de número de portas quânticas exigidas são:

- Aplicar operações do tipo $C^{nq}(U)$ controladas por $|0\rangle$ e por $|1\rangle$ conforme a representação binária de cada estado. Esta maneira pode exigir decomposições de alta complexidade conforme a Figura 19.

- Aplicar rotações com controle apenas nos estados $|1\rangle$ com as devidas compensações de fase, diminui a complexidades das operações mas ainda pode exigir decomposições maiores.

- Aplicar rotações com controle nos estados $|0\rangle$ e $|1\rangle$ de modo a exigir apenas operações do tipo $C^1(U)$ com compensações de fase necessárias.

Neste trabalho, a terceira opção foi implementada. Três vetores de tamanho nr são construídos com o intuito de auxiliar na construção da parte do circuito quântico que corresponde a rotação controlada, são eles:

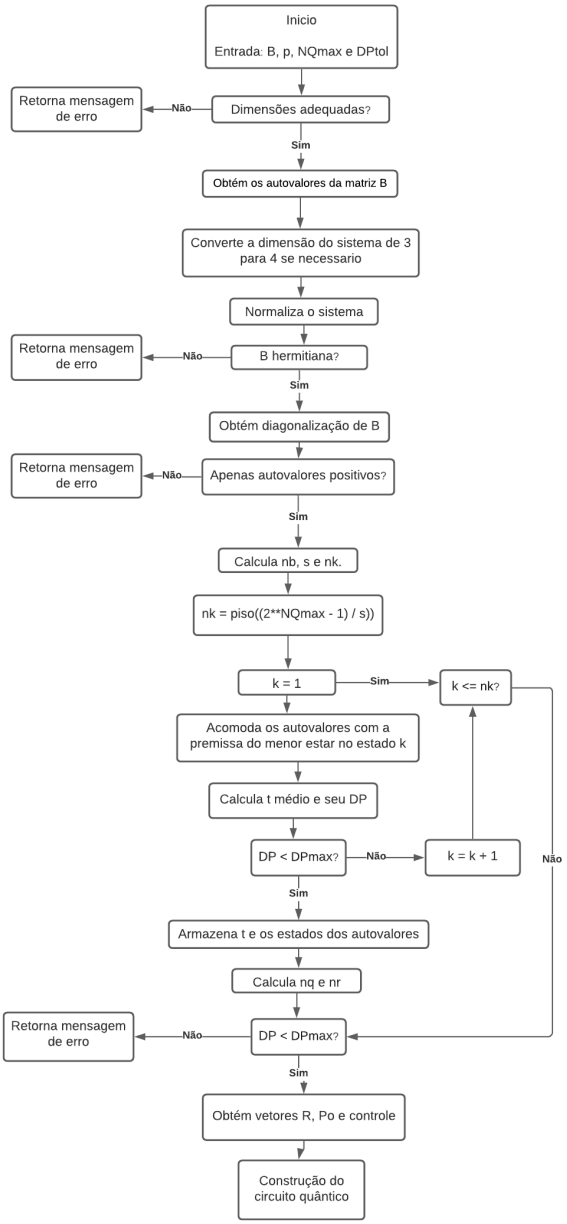
- R - contém os ângulos ϕ_i.

- P_o - contém a posição do *q-bit* a ser rotacionado.

- $controle$ - contém a informação se a operação deve ser de controle 0 ou 1.

A Figura 40 mostra um fluxograma para o processo de determinação dos parâmetros do circuito quântico.

Figura 40 – Fluxograma para o processo de determinação dos parâmetros do circuito quântico.



Fonte: O autor.

3.3.1.2 Inicialização do circuito

O circuito deve conter os três registradores quânticos, além de um registrador clássico para medida dos registradores β e *ancilla*. São eles:

- `alpha = QuantumRegister(nq, name = 'alpha')`
- `beta = QuantumRegister(nb, name = 'beta')`
- `ancilla = QuantumRegister(1, name = 'ancilla')`

- `c = ClassicalRegister(1 + nb, name = 'c')`

- `circuito = QuantumCircuit(beta, alpha, ancilla, c)`

O *Qiskit* possui uma função para inicializar qualquer vetor de estado arbitrário, que será utilizada para colocar *p* em seu estado quântico no registrador β com o seguinte comando:

- `circuito.initialize(p, beta)`

Se *p* for de um *q-bit*, é necessária uma porta como a da Equação 3.2.1 para obter o vetor de estados. Por outro lado, ao se considerar dois *q-bits*, mais de uma porta podem ser necessárias.

3.3.1.3 Estimativa de fase

A transformação Hadamard no registrador α é feita pelo comando:

- `circuito.h(alpha)`

A seguir, a simulação hamiltoniana é feita utilizando três recursos do *Qiskit*. O primeiro é a função `unitary`, importada de `Qiskit.extensions`, que converte uma matriz unitária arbitrária em uma porta lógica equivalente com determinada decomposição. A matriz U é calculada pelo procedimento de exponenciação e^{iBt} descrito na solução numérica do sistema de 3 barras. No momento, a função só é capaz de obter decomposições exatas em matrizes de no máximo dois *q-bits*, limitando o algoritmo para sistemas de até quatro equações.

O segundo recurso consiste na operação controlada da matriz unitária obtida, que é feita através da função `ControlledGate`, importada de `Qiskit.aqua.utils`. As duas operações expostas são feitas sobre um circuito auxiliar sendo, portanto, necessário uni-lo ao circuito principal, o que é feito através do terceiro recurso, a função `append`, contida em `QuantumCircuit`. O processo é repetido nq vezes para os nq *q-bits* do registrador α em um laço de repetição.

Por fim, o operador FT^{-1} é construído pelo circuito inverso ao da Figura 22. O operador R_k, descrito pela Equação 2.4.39, em *Qiskit* é implementado pelo operador $P(\lambda)$:

$$P(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix} \tag{3.3.2}$$

O operador cp realiza a operador controlada de $P(\lambda)$, pelo comando genérico:

- `circuito.cp(theta, control, target)`

em que $theta$ equivale ao parâmetro λ de $P(\lambda)$, $control$ e $target$ aos *q-bits* de controle e alvo, respectivamente. Assim, FT^{-1} para nq *q-bits* é implementado por um laço de repetição que executa as operações cp e Hadamard e outro que executa as portas *SWAP*.

3.3.1.4 Rotação controlada

Esta etapa é implementada por um laço de repetição com variável k, incrementada de 1 a nr, aplicando a operação controlada $C^1(R_y(\theta))$ pela porta cry através do comando

- `circuito.cry(R[k], alpha[Po[k]], ancilla)`

em que $R[k]$ é o ângulo de rotação θ_k , $alpha[Po[k]]$ o *q-bit* de controle na posição $Po[k]$ e $ancilla$ o *q-bit* alvo. Pela informação do vetor $controle$, portas *X* são aplicadas no *q-bit* de controle, antes e depois da aplicação de cry, caso a operação seja de controle 0.

3.3.1.5 Computação reversa

A computação reversa é feita pela inversão da etapa da estimativa de fase, construída pelo mesmo método descrito. Conforme visto na solução numérica do sistema de 3 barras, ao final da computação reversa é possível obter a solução do sistema linear no vetor de estado, que na simulação é obtido pelo *backend* `statevectorsimulator`. A resposta se encontra entre as posições $2^{(nb+nq)}$ e $2^{(nb+nq)} + 2^{nb}$ do vetor, de modo que o chamando de *vector*, o comando

- `quantica = np.array(vector[2**(nq+nb):(2**(nq+nb)+2**nb)])/C`

fornece a resposta desejada. A biblioteca *NumPy* possui a função `linalg.solve`, que retorna a solução clássica para o sistema através do comando

- `classica = np.array(linalg.solve(B,p))`

sendo possível comparar a solução quântica com a clássica. O erro percentual absoluto médio (*Mean absolute percentage error* - MAPE) (MYTTENAERE et al., 2016) pode ser calculado como:

$$\epsilon = \frac{100}{n} \sum_i \left| \frac{\theta - \theta'}{\theta} \right| \tag{3.3.3}$$

em que θ é a solução exata, θ' solução obtida e n o número de elementos de cada vetor. O cálculo é realizado com auxílio da função `mean`, que retorna a média de um vetor.

- `erro = 100*np.mean(np.abs((quantica-classica)/classica))`

A fidelidade entre dois estados quânticos arbitrários $|\psi_1\rangle$ e $|\psi_2\rangle$ pode calculada pela equação:

$$F(\rho_1, \rho_2) = tra(\sqrt{\sqrt{\rho_1}\rho_1\sqrt{\rho_1}})^2 \tag{3.3.4}$$

em que $\rho_1 = |\psi_1\rangle\langle\psi_1|$ e $\rho_2 = |\psi_2\rangle\langle\psi_2|$. O cálculo é feito pela função `state_fidelity`, importada de `qiskit.quantum_info`. Assim, normalizando a solução clássica e a quântica:

- `fidelidade = state_fidelity(quantica/ linalg.norm(quantica), classica / linalg.norm(classica))`

A probabilidade de medida $|1\rangle$ no *q-bit ancilla* é calculada pela soma de seus módulos das amplitudes ao quadrado. O comando em *Qiskit* é

- `probabilidade = 100*np.abs(sum(quantica*quantica)*C**2)`

3.3.1.6 Medida

Por fim, é feita a medida no registrador β para seus nb *q-bits* em um laço com a variável k e no registrador ancilla:

- `circuito.measure(beta[k], c[k])`
- `circuito.measure(ancilla, c[nb])`

A utilização do `qasm_simulator` fornece a simulação ideal do circuito por meio de um histograma que contém informações sobre o resultado do sistema matricial, que será exposto para os dois sistemas no Capítulo 4.

3.3.2 Código em Matlab

Um código de medologia equivalente ao do *Qiskit* foi desenvolvido no Matlab 2018a (MATLAB, 2018). Contudo, difere do algoritmo em *Qiskit* por não trabalhar com a decomposição de operadores em portas lógicas universais, permitindo qualquer operador unitário dentro dos limites de memória do *software* atuando como porta lógica, o que implica na possibilidade de solução de sistemas maiores que 4x4.

Foram criados arquivos em texto para fins de visualização. O primeiro apresenta os parâmetros do sistema, o que inclui os parâmetros de entrada, os parâmetros calculados na rotina de preparação do algoritmo e os parâmetros da construção do circuito quântico. A lista compreende:

- Parâmetros de entrada B, p, $NQmax$ e $DPtol$.
- B e p normalizados.
- Matriz B diagonalizada e seus autovalores.
- nb, nq e nr e o número total de *q-bits* e bits clássicos do sistema.
- Estados em que os autovalores estão armazenados.
- Constantes t e C.

- Desvio padrão de t.
- R, P_o e $controle$ da etapa de rotação controlada.
- Matrizes hamiltonianas e hamiltonianas inversas.
- Portas lógicas dos operadores FT^{-1} e FT.
- Matrizes da etapa de rotação controlada.

O segundo arquivo apresenta os resultados, o que inclui:

- Solução clássica.
- Solução quântica.
- Erro teórico.
- Fidelidade.
- Probabilidade de medida.
- Valores de um histograma com as medidas dos registradores β e $ancilla$.
- Vetores de estado em cada de uma das etapas do algoritmo: preparação de β, transformação Hadamard, simulação hamiltoniana, QFT inversa, rotação controlada, QFT, simulação hamiltoniana inversa e o vetor de estados final.

O código retorna mensagens de erros para os mesmo casos indicados no fluxograma da Figura 40, com a diferença que o sistema não é limitada para sistemas de dimensão 4 mas sim pela memória do Matlab. O código é exposto no Apêndice B e os arquivos texto para os dois sistemas no Apêndice C.

3.4 Modelo do Simulink

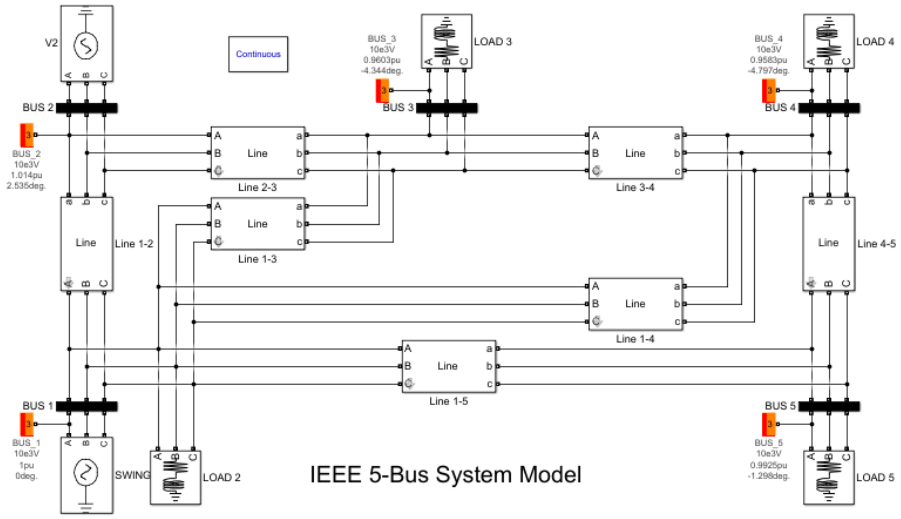
Com intuito de comparar o resultado do fluxo de potência CC com o fluxo de potência tradicional por métodos iterativos para o sistema de cinco barras, foi realizada uma simulação empregando o modelo desenvolvido em Tan (2021) para o Simulink (DOCUMENTATION, 2021) no Matlab 2018a (MATLAB, 2018). Os parâmetros utilizados foram os seguintes:

- Impedâncias das linhas de transmissão da Tabela 5, que incluem não apenas os valores de reatância série, como também o de susceptância *shunt*.
- Potências da Tabela 6, o que inclui a parcela reativa, ignorada no fluxo de potência CC.
- Barra 1 como referência, com $\theta_1 = 0$ e $V_1 = 1pu$ e com carga nula acoplada.

- Bases de $10\ kV$ e $100\ MVA$ e frequência de 60 Hz.
- Cargas do tipo potência constante.
- Número máximo de 50 iterações e tolerância de $0.00001\ pu$.

A Figura 41 indica o modelo com os parâmetros atualizados.

Figura 41 – Modelo do sistemas de 5 barras no Simulink com os parâmetros atualizados.



Fonte: Adaptado de Tan (2021).

3.5 HHL do Qiskit Aqua

No algoritmo HHL pré-programado no *Qiskit Aqua* (QISKIT, 2021c,d), o usuário deve informar a matriz e o vetor do sistema linear e o algoritmo constrói o circuito quântico. Algumas funcionalidades e recursos são expostas em Qiskit (2021d).

4 RESULTADOS

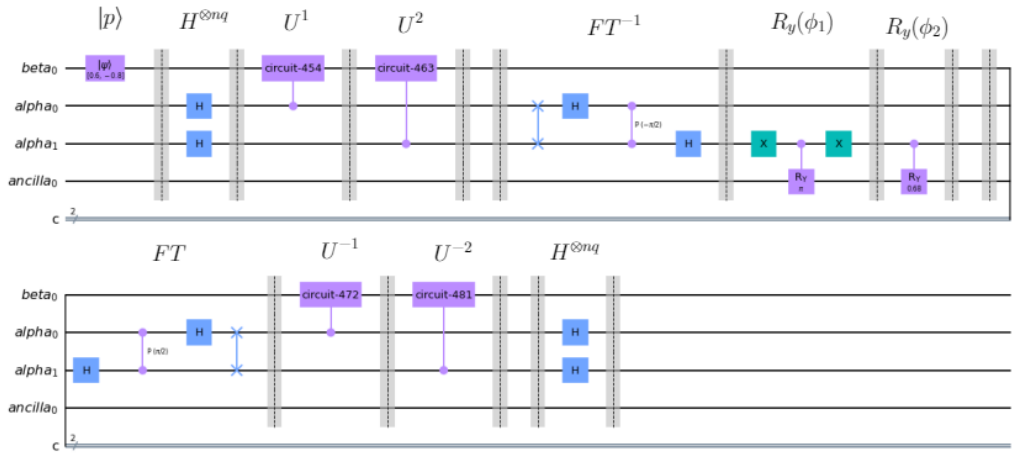
Neste capítulo serão apresentados os resultados obtidos neste trabalho. Na Seção 4.1 será exposto o circuito quântico construído e sua otimização para cada sistema com a contabilização das portas quânticas. Na Seção 4.2 serão apresentados os resultados obtidos nas simulações e na experimentação no computador quântico real para o sistema de três barras. Na Seção 4.3 será feito um comparativo dos resultados obtidos com o algoritmo HHL pré-programado do *Qiskit Aqua*.

4.1 CIRCUITOS QUÂNTICOS CONSTRUÍDOS

4.1.1 Sistema de três barras

A Figura 42 mostra circuito quântico para o sistema de três barras construído no *Qiskit*. São utilizados 4 *q-bits* (1 no registrador β, 2 no α e o *q-bit ancilla*) e 2 bits clássicos (para medida de β e *ancilla*).

Figura 42 – Circuito para o sistema de 3 barras.



Fonte: O autor.

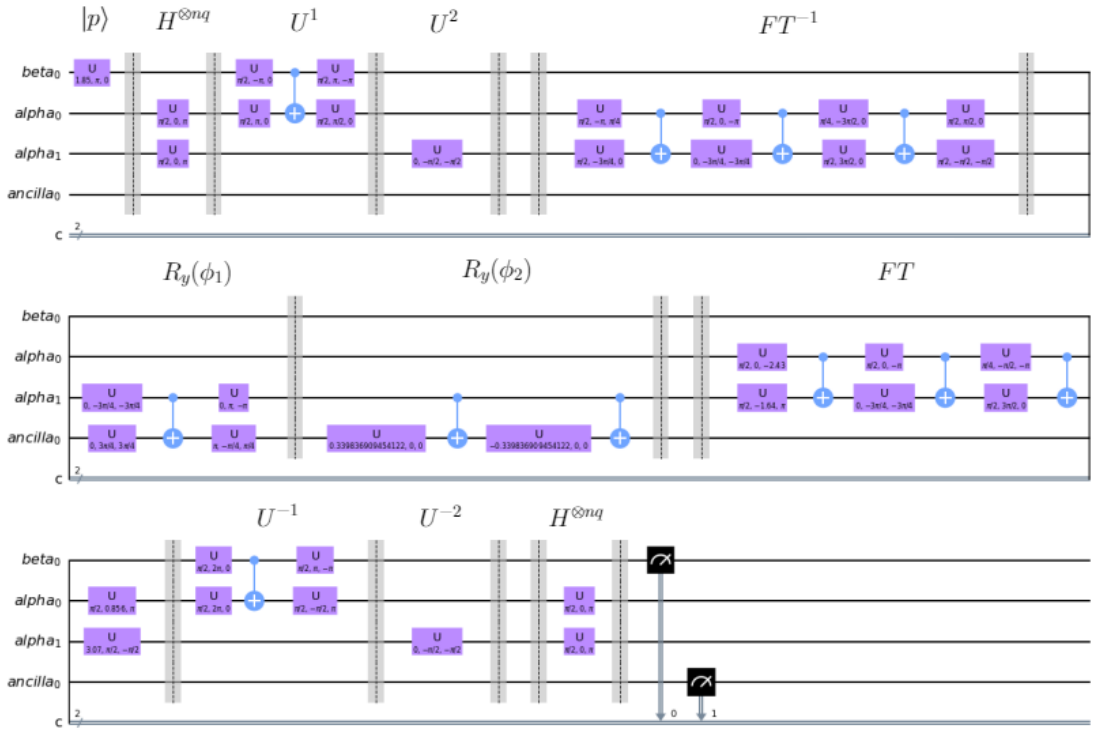
Antes de prosseguir, é necessário mencionar que em *Qiskit* os circuitos são representados de forma inversa ao que foi apresentado ao longo deste trabalho, isto é, o primeiro *q-bit* de cima para baixo aparece como o primeiro de baixo para cima e assim por diante.

As operações do circuito foram separadas por barreiras para fins de visualização pelo comando `circuito.barrier()`. Na representação da Figura 42, algumas portas devem ser decompostas em portas lógicas universais para que o circuito possa ser simulado ou executado em um computador quântico real. A função `transpile`, importada de `Qiskit.compiler`, atua como transpilador através do comando:

- `transp = transpile(circuito, basis_gates = ['u', 'cx'], optimization_level = 3)`

O primeiro parâmetro é o circuito a ser transpilado. O segundo indica o conjunto de portas quânticas de base para a construção do circuito. Foi escolhido como base as portas u e cx, que representam, respectivamente, a porta da Equação 3.2.1 e a porta *CNOT*. Por fim, o terceiro parâmetro indica o nível de otimização, que pode ser 0, 1, 2 ou 3, com ordem crescente de otimização, sendo escolhido o valor 3, que é o mais otimizado. O circuito transpilado é mostrado na Figura 43. A Tabela 7 contabiliza o número de portas do circuito considerando cada operação individualmente.

Figura 43 – Circuito transpilado para o sistema de 3 barras.



Fonte: O autor.

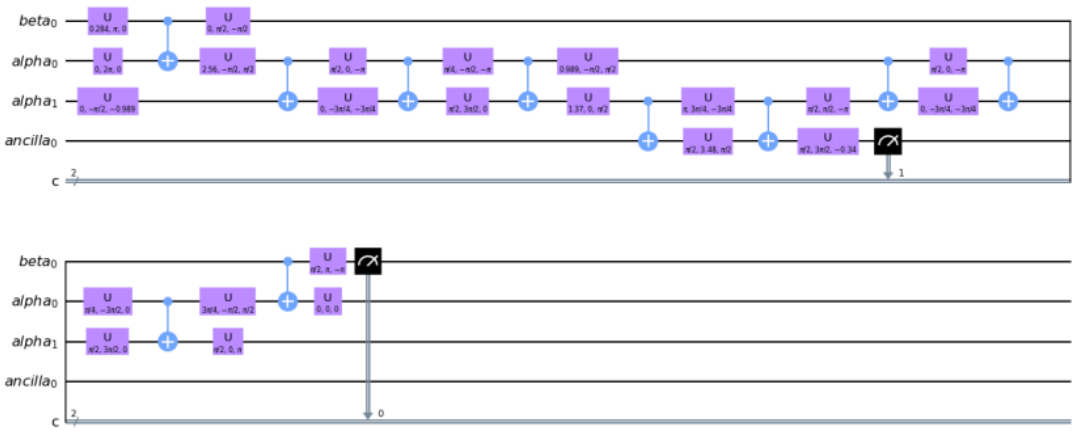
Tabela 7 – Contabilização do número de portas do sistema de 3 barras.

Operação	Portas U	Portas $CNOT$	Total de portas
$\|p\rangle$	1	0	1
$H^{\otimes nq}$	2	0	2
U^1	4	1	5
U^2	1	0	1
FT^{-1}	8	3	11
$R_y(\phi_1)$	4	1	5
$R_y(\phi_2)$	2	2	4
FT	8	3	11
U^{-1}	4	1	5
U^{-2}	1	0	1
$H^{\otimes nq}$	2	0	2
Total	37	11	48

Fonte: O autor.

É possível diminuir o número de portas ao eliminar as barreiras do circuito, permitindo que portas de operações até então separadas, possam ser unidas. O circuito é mostrado na Figura 44 e a Tabela 8 indica o número de portas do circuito, além de número de *q-bits* e bits clássicos utilizados. O parâmetro *depth*, obtido pelo comando `transp.depth()`, indica o número de portas executadas no caminho mais longo do circuito, sem incluir as medidas, sendo o parâmetro que junto ao número de *q-bits* indica se o circuito é factível para o *hardware* quântico disponível. Adicionalmente, a porta de dois *q-bit CNOT* afeta significativamente a eficiência e a acurácia em um *hardware* quântico (ESKANDARPOUR et al., 2020a).

Figura 44 – Circuito transpilado sem barreiras para o sistema de 3 barras.



Fonte: O autor.

Tabela 8 – Parâmetros do circuito quântico para o sistema de 3 barras.

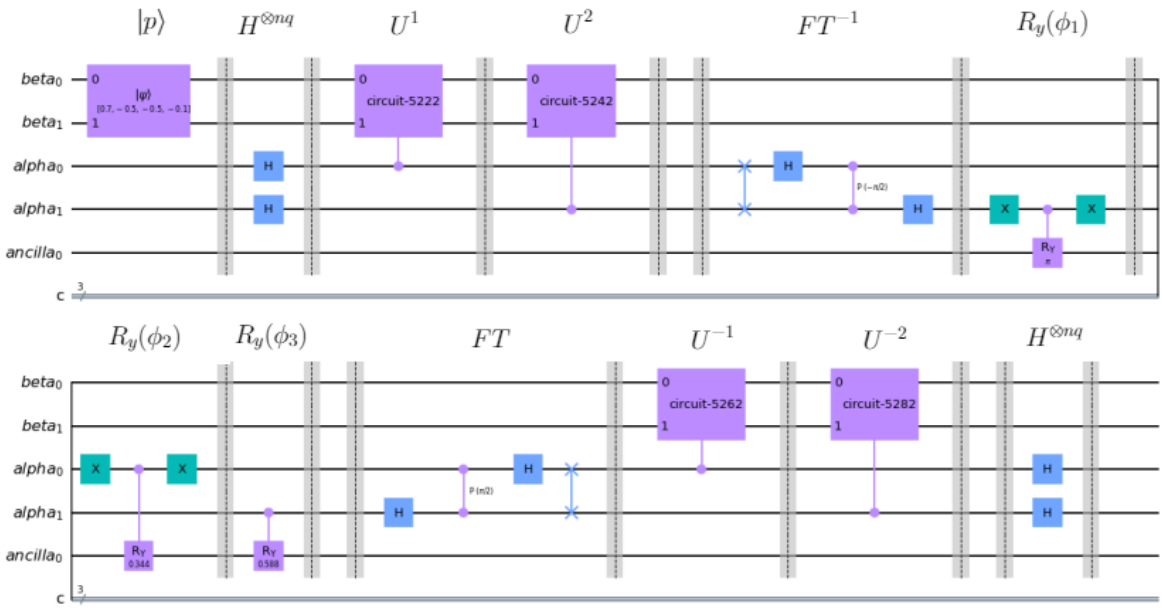
Portas U	24
Portas $CNOT$	10
Total de portas	34
Depth	21
Q-bits	4
Bits clássicos	2

Fonte: O autor.

4.1.2 Sistema de cinco barras

A Figura 45 mostra circuito quântico construído no *Qiskit*. São utilizados 5 *q-bits* (2 no registrador β, 2 no α e o *q-bit ancilla*) e 3 *bits* clássicos (para medida de β e *ancilla*).

Figura 45 – Circuito para o sistema de 5 barras.

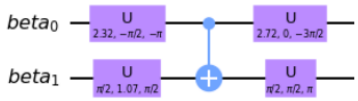


Fonte: O autor.

O mesmo procedimento para transpilar o circuito é aplicado. A Tabela 10 indica a contabilização das portas, que totaliza 250, um número muito elevado para que o circuito completo possa ser apresentado neste texto. Assim, cada etapa será comentada individualmente.

A preparação do estado $|p\rangle$ e feita pelo circuito da Figura 46, que contém 4 portas U e 1 porta $CNOT$.

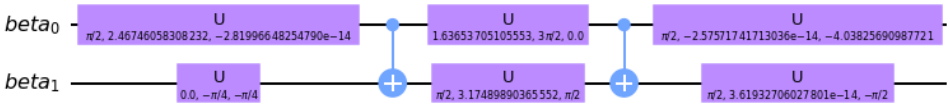
Figura 46 – Preparação do estado $|p\rangle$ para o sistema de 5 barras.



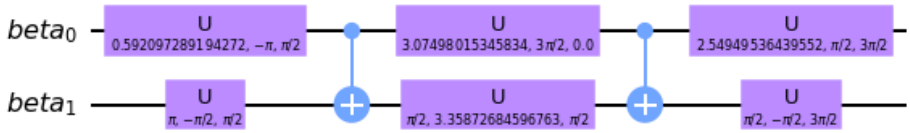
Fonte: O autor.

A transformação Hadamard é idêntica ao sistema anterior, exigindo 2 portas U tanto na estimativa como em sua computação reversa. As matrizes U^1, U^2, U^{-1} e U^{-2} atuam sobre dois *q-bits* e são decompostas pela função `unitary` conforme a Figura 47.

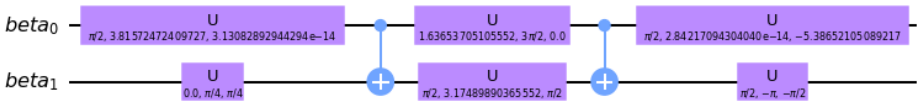
Figura 47 – (a) Decomposição de U^1. (b) Decomposição de U^2. (c) Decomposição de U^{-1}. (d) Decomposição de U^{-2}.



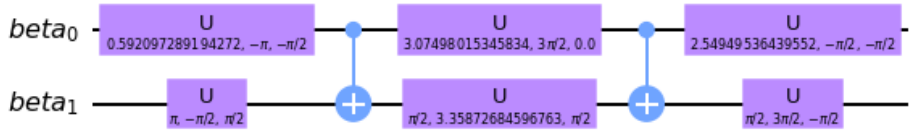
(a)



(b)



(c)

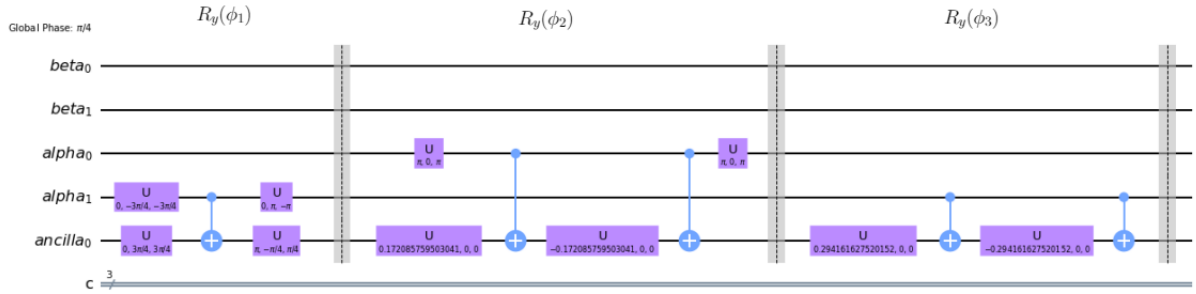


(d)

Fonte: (a) O autor, (b) O autor, (c) O autor e (d) O autor.

O número de portas de cada operação controlada das matrizes da Figura 47 é exposto na Tabela 9. As quatro operações controladas somadas compreendem 81.6% das portas lógicas do circuito. Os operadores FT^{-1} e FT que agem sobre o registrador α são os mesmos do sistema anterior, com 8 portas U e 3 porta $CNOT$ cada. Por fim, a Figura 48 mostra a decomposição das rotações controladas.

Figura 48 – Operações controladas para o sistema de 5 barras.



Fonte: O autor.

Tabela 9 – Contabilização do número de portas do sistema de 5 barras.

Operação	Portas U	Portas $CNOT$	Total de portas
$\|p\rangle$	4	1	5
$H^{\otimes nq}$	2	0	2
U^1	32	20	52
U^2	32	18	50
FT^{-1}	8	3	11
$R_y(\phi_1)$	4	1	5
$R_y(\phi_2)$	4	2	6
$R_y(\phi_3)$	2	2	4
FT	8	3	11
U^{-1}	32	20	52
U^{-2}	32	18	50
$H^{\otimes nq}$	2	0	2
Total	162	88	250

Fonte: O autor.

A Tabela 10 indica o número de portas do circuito eliminando as barreiras, o número de *q-bits* e o de *bits* clássicos.

Tabela 10 – Parâmetros do circuito quântico para o sistema de 5 barras.

Portas U	144
Portas $CNOT$	88
Total de portas	232
Depth	170
Q-bits	5
Bits clássicos	3

Fonte: O autor.

4.2 RESULTADOS OBTIDOS

O vetor de estados pode ser obtido através da função `statevectorsimulator` ao final do algoritmo e com ele o erro teórico e a probabilidade de sucesso do algoritmo podem ser calculados. Contudo, o resultado pelo vetor de estados é uma simulação numérica e a informação em um computador quântico só pode obtida por meio de medidas. Nesta seção serão expostas os resultados das simulações com medidas e experimentação em um computador quântico real e como se chegar na solução do fluxo de potência a partir destas, além dos resultados da simulação do Simulink para fins comparativos com o fluxo de potência CC.

4.2.1 Sistema de três barras

A Tabela 11 indica solução para o sistema de 3 barras obtida pelo vetor de estados, bem como o erro calculado e a probabilidade de sucesso do algoritmo. O resultado é exato, conforme o esperado.

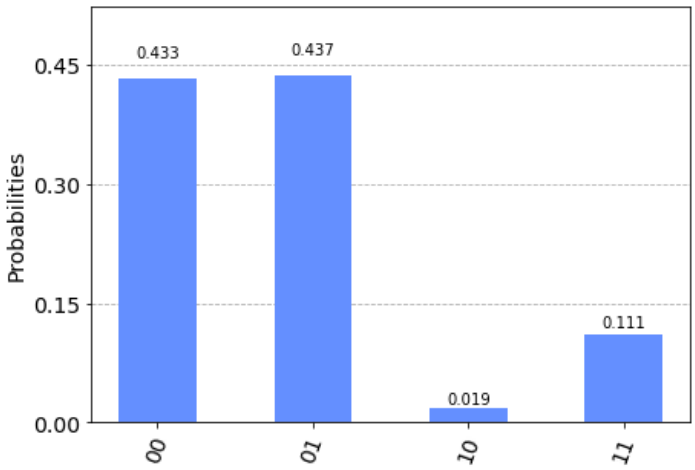
Tabela 11 – Solução do sistema de 3 barras pelo vetor de estados.

Solução clássica $[\theta_2, \theta_3]$	[0.066667, -0.16667]
Solução quântica $[\theta_2, \theta_3]$	[0.066667, -0.16667]
Erro	0
Fidelidade	1
Probabilidade	12.89%

Fonte: O autor.

A simulação através do `qasm_simulator` fornece um histograma que indica as probabilidades de medida. Aplicando operadores de medida nos registradores $ancilla$ e β e executando 8192 vezes resulta no histograma da Figura 49.

Figura 49 – Histograma do sistema de 3 barras.



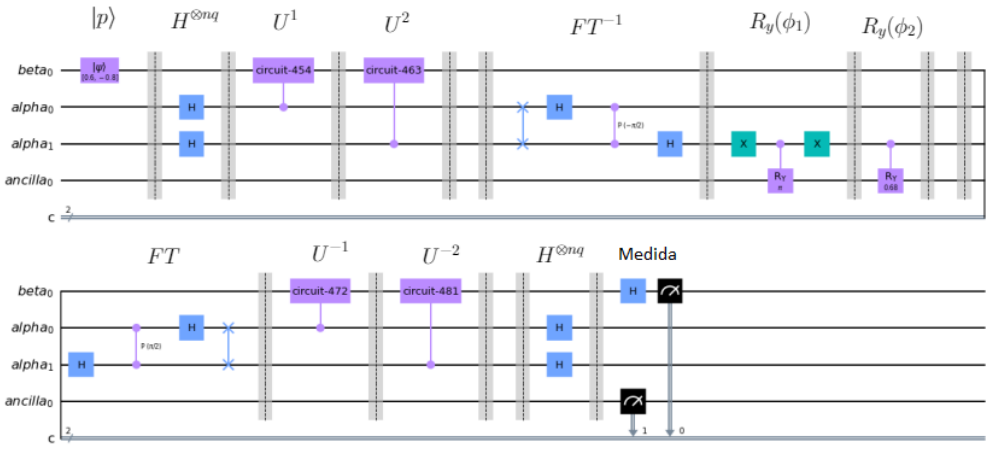
Fonte: O autor.

A probabilidade de medida de um estado é igual ao módulo da amplitude ao quadrado, o que implica na impossibilidade de saber o sinal da amplitude do estado medido (VAZQUEZ; HIPTMAIR; WOERNER, 2020). Assim, por meio do histograma da Figura 49, pode-se calcular os módulos $|\theta_2|$ e $|\theta_3|$

$$|\theta_2| = \frac{\sqrt{0.019}}{C} = 0.068920 \ , \ |\theta_3| = \frac{\sqrt{0.111}}{C} = 0.16658 \tag{4.2.1}$$

Conforme explicam Vazquez, Hiptmair e Woerner (2020), or meio da aplicação de uma porta Hadamard no registrador β, conforme mostra o circuito da Figura 50, é possível obter propriedades da solução θ que serão deduzidas a seguir.

Figura 50 – Aplicação da porta Hadamard no registrador β.



Fonte: O autor.

Seja $|\theta'\rangle$ o estado que corresponde a parcela do *q-bit* p que acompanha o estado $|1\rangle$ do *q-bit ancilla* após a etapa de computação reversa. Com efeito, $|\theta'\rangle$ é vetor solução normalizado, isto é:

$$|\theta\rangle = \theta'_2|0\rangle + \theta'_3|1\rangle = \frac{\theta_2}{C}|0\rangle + \frac{\theta_2}{C}|1\rangle \tag{4.2.2}$$

Ao medir $|1\rangle$ no *q-bit ancilla*, o estado do circuito passa a ser

$$|\psi\rangle = \frac{|\theta'\rangle}{||\theta'||} = \frac{|\theta'\rangle}{\sqrt{|\theta'_2|^2 + |\theta'_3|^2}} \tag{4.2.3}$$

em que $||\theta'||$ é a probabilidade da medida. A aplicação da porta Hadamard leva ao estado

$$|\psi_H\rangle = H|\psi\rangle = \frac{1}{\sqrt{2}}\frac{\theta'_2 + \theta'_3}{\sqrt{|\theta'_2|^2 + |\theta'_3|^2}}|0\rangle + \frac{1}{\sqrt{2}}\frac{\theta'_2 - \theta'_3}{\sqrt{|\theta'_2|^2 + |\theta'_3|^2}}|1\rangle \tag{4.2.4}$$

As probabilidades de medida $|0\rangle$ e $|1\rangle$ no *q-bit* β levando em conta a medida $|1\rangle$ no *q-bit ancilla* são calculadas por

$$\begin{aligned} p(obter|10\rangle) &= \frac{1}{2}\frac{\sqrt{|\theta'_2|^2 + |\theta'_3|^2}}{\sqrt{|\theta'_2|^2 + |\theta'_3|^2}}(\theta'_2 + \theta'_3)^2 = \frac{1}{2}(\theta'_2 + \theta'_3)^2 \\ p(obter|11\rangle) &= \frac{1}{2}\frac{\sqrt{|\theta'_2|^2 + |\theta'_3|^2}}{\sqrt{|\theta'_2|^2 + |\theta'_3|^2}}(\theta'_2 - \theta'_3)^2 = \frac{1}{2}(\theta'_2 - \theta'_3)^2 \end{aligned} \tag{4.2.5}$$

Assim, é possível determinar:

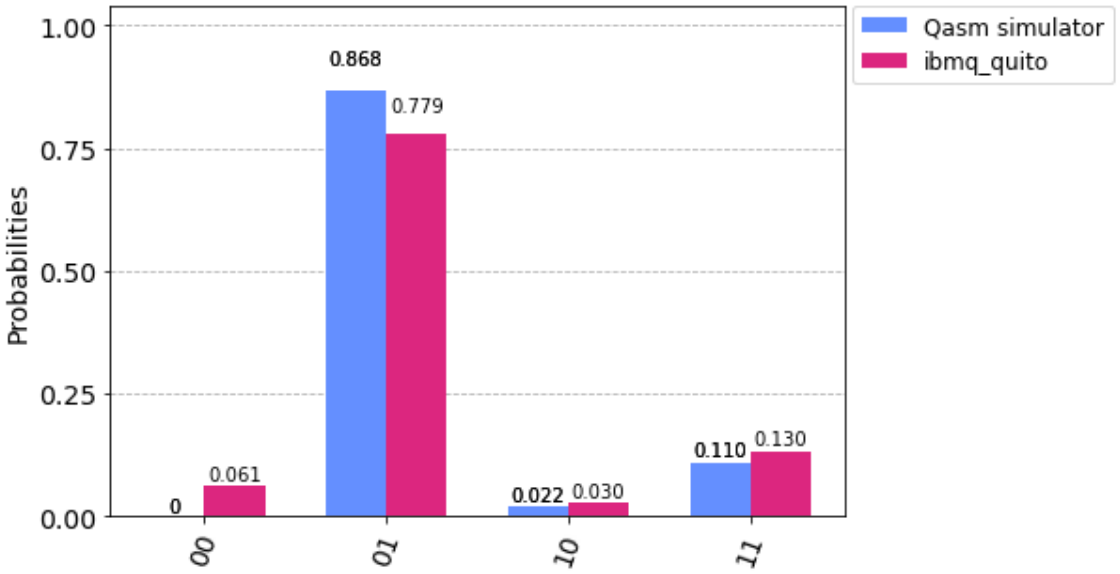
$$\begin{aligned} (\theta_2 + \theta_3)^2 &= \left(\frac{\theta'_2}{C} + \frac{\theta'_3}{C}\right)^2 = \frac{1}{C^2}(\theta'_2 + \theta'_3)^2 = \frac{2\,p(obter|10\rangle)}{C^2} \\ (\theta_2 - \theta_3)^2 &= \left(\frac{\theta'_2}{C} - \frac{\theta'_3}{C}\right)^2 = \frac{1}{C^2}(\theta'_2 - \theta'_3)^2 = \frac{2\,p(obter|11\rangle)}{C^2} \end{aligned} \tag{4.2.6}$$

Com os valores $(\theta_2 + \theta_3)^2$ e $(\theta_2 - \theta_3)^2$ é possível determinar se θ_2 e θ_3 têm sinais iguais ou distintos, de modo que:

- Se $(\theta_2 + \theta_3)^2 > (\theta_2 - \theta_3)^2$, sinais iguais
- Se $(\theta_2 + \theta_3)^2 < (\theta_2 - \theta_3)^2$, sinais distintos

Adicionalmente, $(\theta_2 - \theta_3)^2$ determina a magnitude da abertura angular θ_{23} do fluxo de potência e consequentemente θ_{23}. Foi realizada a simulação pelo `qasm_simulator` e a execução no computador quântico *ibmq_quito*, ambos para 8192 vezes, obtendo o histograma da Figura 51. É importante ressaltar que o computador *ibmq_quito* possui o conjunto de portas de base id, rz, sx, x e cx, diferente do utilizado na transpilação. Nesse conjunto, o circuito passa a ter um valor 46 de *depth*.

Figura 51 – Histograma da implementação no computador quântico *ibmq_quito*.



Fonte: O autor.

A Tabela 12 indica os resultados obtidos com base na Equação 4.2.6. Como $(\theta_2 - \theta_3)^2$ é maior que $(\theta_2 + \theta_3)^2$, se conclui que os sinais de θ_2 e θ_3 são distintos. O erro calculado é 6.935% e a fidelidade 0.9982.

Tabela 12 – Resultado obtido no computador quântico *ibmq_quito* para o sistemas de 3 barras.

	Computador real	Valor exato
$(\theta_2 + \theta_3)^2$	0.015	0.01
$(\theta_2 - \theta_3)^2$	0.065	0.054444
Solução em radianos $[\theta_2, \theta_3]$	[$\pm$0.066238, $\mp$0.188713]	[0.066667, -0.16667]
Solução em graus $[\theta_2, \theta_3]$	[$\pm$3.7952, $\mp$10.8125]	[3.8197, -9.5493]

Fonte: O autor.

4.2.2 Sistema de cinco barras

A Tabela 13 indica a solução do sistema, o erro calculado e a probabilidade de sucesso do algoritmo para o sistema de 5 barras, calculados através do vetor de estados. Com efeito, há um erro teórico devido a representação inexata dos autovalores.

Tabela 13 – Solução do sistema de 5 barras pelo vetor de estados.

Solução clássica em radianos $[\theta_2, \theta_3, \theta_4, \theta_5]$	[0.045727, -0.075638, -0.083185, -0.022198]
Solução quântica em radianos $[\theta_2, \theta_3, \theta_4, \theta_5]$	[0.047365, -0.077286, -0.082152, -0.018314]
Solução clássica em graus $[\theta_2, \theta_3, \theta_4, \theta_5]$	[2.6120, -4.3337, -4.7661, -1.2719]
Solução quântica em graus $[\theta_2, \theta_3, \theta_4, \theta_5]$	[2.7138, -4.4282, -4.7070, -1.0493]
Erro	6.124%
Fidelidade	0.9986
Probabilidade	44.526%

Fonte: O autor.

A Figura 52 mostra o histograma realizado através do `qasm_simulator` com medida nos registradores $ancilla$ e β e com 8192 execuções.

As magnitudes dos ângulos do vetor solução podem ser calculadas por

$$\begin{aligned} |\theta_2| &= \frac{\sqrt{0.069}}{C} = 0.048698\ ,\ |\theta_3| = \frac{\sqrt{0.175}}{C} = 0.077555 \\ |\theta_4| &= \frac{\sqrt{0.2}}{C} = 0.082909\ ,\ |\theta_5| = \frac{\sqrt{0.013}}{C} = 0.021138 \end{aligned} \tag{4.2.7}$$

Algumas propriedades de θ podem ser determinadas aplicando uma porta Hadamard antes da medida. Seja a aplicação feita no primeiro *q-bit*:

$$\begin{aligned} |\psi_H\rangle &= \mathbb{I} \otimes H\ |\psi\rangle \\ &= \frac{1}{\sqrt{2}\ ||\theta'||}[(\theta'_2 + \theta'_3)|00\rangle + (\theta'_2 - \theta'_3)|01\rangle + (\theta'_4 + \theta'_5)|10\rangle + (\theta'_4 - \theta'_5)|11\rangle] \end{aligned} \tag{4.2.8}$$

Com as medidas de $|100\rangle$, $|101\rangle$, $|110\rangle$ e $|111\rangle$ pode-se determinar, respectivamente, $(\theta_2 + \theta_3)^2$, $(\theta_2 - \theta_3)^2$, $(\theta_4 + \theta_5)^2$ e $(\theta_4 - \theta_5)^2$.

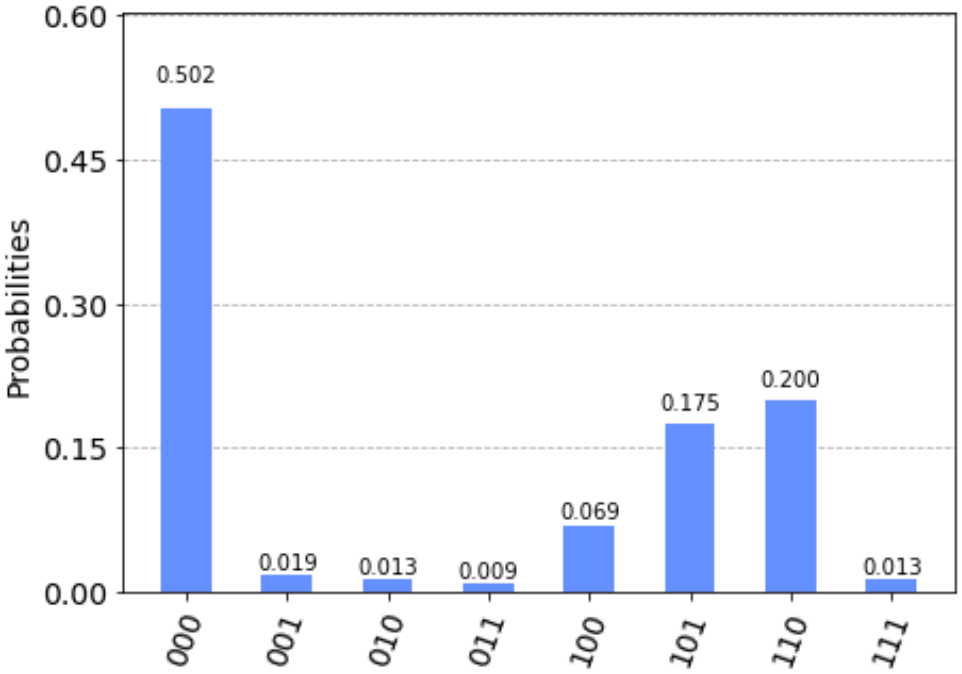
Alternativamente, aplicando a porta Hadamard no segundo *q-bit*:

$$\begin{aligned} |\psi_H\rangle &= H \otimes \mathbb{I}\ |\psi\rangle \\ &= \frac{1}{\sqrt{2}\ ||\theta'||}[(\theta'_2 + \theta'_4)|00\rangle + (\theta'_3 + \theta'_5)|01\rangle + (\theta'_2 - \theta'_4)|10\rangle + (\theta'_3 - \theta'_5)|11\rangle] \end{aligned} \tag{4.2.9}$$

É possível determinar $(\theta_2 + \theta_4)^2$, $(\theta_3 + \theta_5)^2$, $(\theta_2 - \theta_4)^2$ e $(\theta_3 - \theta_5)^2$ com as respectivas medidas de $|100\rangle$, $|101\rangle$, $|110\rangle$ e $|111\rangle$.

Como informação adicional, o circuito passa a ter o valor de *depth* igual a 323 considerando

Figura 52 – Histograma do sistema de 5 barras.



Fonte: O autor.

o conjunto de portas de base do computador *ibmq_quito*.

Por fim, foi realizada a simulação do fluxo de potência tradicional por métodos iterativos utilizando o modelo do Simulink. Os resultados obtidos são indicados na Figura 53.

Figura 53 – Resultados obtidos pelo modelo do Simulink.

	Block type	Bus type	Bus ID	Vbase (kV)	Vref (pu)	Vangle (deg)	P (MW)	Q (Mv...	Qmin (Mvar)	Qmax (Mvar)	V_LF (pu)	Vangle_LF (deg)	P_LF (MW)	Q_LF (Mvar)	Block Name
1	Vsrc	PV	BUS_2	10.00	1	0.00	70.00	0.00	0.00	0.00	1.0137	2.53	70.00	0.00	V2
2	RLC load	PQ	BUS_1	10.00	1	0.00	0.00	0.00	-Inf	Inf	1	0.00	0.00	0.00	LOAD 2
3	Vsrc	swing	BUS_1	10.00	1	0.00	40.00	30.00	-40.00	50.00	1	0.00	43.76	8.58	SWING
4	RLC load	PQ	BUS_5	10.00	1	0.00	10.00	3.00	-Inf	Inf	0.9925	-1.30	10.00	3.00	LOAD 5
5	RLC load	PQ	BUS_3	10.00	1	0.00	50.00	15.00	-Inf	Inf	0.9603	-4.34	50.00	15.00	LOAD 3
6	RLC load	PQ	BUS_4	10.00	1	0.00	50.00	15.00	-Inf	Inf	0.9583	-4.80	50.00	15.00	LOAD 4

Fonte: O autor.

A Tabela 14 mostra a comparação entre o resultado para vetor dos ângulos das tensões nodais obtido no Simulink, o fluxo de potência CC exato e o fluxo de potência CC simulado em computação quântica com o valor obtido no vetor de estados pelo `statevectorsimulator`.

Tabela 14 – Comparação de resultados entre o fluxo de potência tradicional e o fluxo de potência CC.

Fluxo de potência tradicional $[\theta_2, \theta_3, \theta_4, \theta_5]$	[2.53, -4.34, -4.80, -1.30]
Fluxo de potência CC exato $[\theta_2, \theta_3, \theta_4, \theta_5]$	[2.6120, -4.3337, -4.7661, -1.2719]
Fluxo de potência CC quântico $[\theta_2, \theta_3, \theta_4, \theta_5]$	[2.7138, -4.4282, -4.7070, -1.0493]

Fonte: O autor.

O erro calculado entre o resultado do Simulink a solução exata de fluxo de potência CC foi 1.564% enquanto em relaçao à solução quântica foi 7.630%.

4.3 Comparação com o HHL do Qiskit Aqua

A Tabela 15 e a Tabela 16 fazem a comparação do algoritmo desenvolvido neste trabalho com o algoritmo do *Aqua* para os dois sistemas de estudo considerando a solução do `statevectorsimulator` e a complexidade do circuito transpilado com otimização semelhante.

Tabela 15 – Comparação do algoritmo desenvolvido neste trabalho com o do *Aqua* para o sistema de 3 barras.

	Resultados obtidos	*Aqua*
Solução $[\theta_2, \theta_3]$	[0.066667, -0.16667]	[0.066667, -0.16667]
Erro	0	0
Fidelidade	1	1
Portas U	24	18
Portas $CNOT$	10	10
Total de portas	34	28
Depth	21	21
Q-bits	4	4

Fonte: O autor.

Tabela 16 – Comparação do algoritmo desenvolvido neste trabalho com o do *Aqua* para o sistema de 5 barras.

	Resultados obtidos	*Aqua*
Solução $[\theta_2, \theta_3, \theta_4, \theta_5]$	[0.045727, -0.075638, -0.083185, -0.022198]	[0.046300, -0.074966, -0.083456, -0.024099]
Erro	6.124%	2.758%
Fidelidade	0.9986	0.9997
Portas U	144	394
Portas $CNOT$	88	249
Total de portas	232	643
Depth	170	485
Q-bits	5	6

Fonte: O autor.

Para o primeiro sistema ambos fornecem a solução exata no vetor de estados e o algoritmo do *Aqua* tem 6 portas U a menos. Já para o segundo sistema, o *Aqua* fornece um erro menor e uma fidelidade maior com o custo de um *q-bit* adicional, uma quantidade portas *CNOT* 2.83 vezes maior e um valor de *depth* 2.85 vezes maior.

5 CONCLUSÃO

A proposta deste trabalho foi realizar a simulação de fluxo de potência CC em dois sistemas distintos utilizando computação quântica. Para chegar neste objetivo, foi necessário o conhecimento de conceitos básicos de álgebra linear e mecânica quântica para adentrar no estudo da computação quântica e posteriormente do algoritmo HHL. Ademais, foi necessário conhecer o Qiskit, sua linguagem e suas ferramentas para construção do algoritmo desenvolvido para as simulações do trabalho.

O primeiro sistema, de três barras, teve sua solução numérica desenvolvida. A simulação através do vetor de estados gerou um resultado idêntico à solução clássica em razão da representação exata dos autovalores da matriz admitância. O circuito, de 4 *q-bits*, *depth* 21 e 10 portas *CNOT*, foi executado no computador quântico real *ibmq_quito* disponibilizado pela IBM e a solução obtida apresentou um erro calculado de 6.935% e uma fidelidade de 0.9982 em relação à solução clássica.

O segundo sistema, de cinco barras, demonstrou que o algoritmo HHL pode fornecer soluções aproximadas para sistemas sem representação exata dos autovalores, uma condição que fatalmente deve ocorrer em sistemas maiores. A simulação pelo vetor de estados apresentou um erro de 6.124% e fidelidade de 0.9986. Para complementar o estudo, foi apresentado para fins comparativos, o resultado de uma simulação de fluxo de potência tradicional por métodos iterativos com valores definidos para os parâmetros desconsiderados no fluxo de potência de CC. O erro do vetor dos ângulos das tensões nodais em relação a solução exata de fluxo de potência CC foi e 1.564% e em relação a solução quântica de 7.630%, um resultado condizente para um estudo preliminar que não exige precisão. Este sistema, por conta sua complexidade elevada, contendo com 5 *q-bits*, *depth* 170 e 88 portas *CNOT*, não pôde ser executado em um computador quântico real. Sua viabilidade depende da evolução da tecnologia de computação quântica e das técnicas de otimização da decomposição de portas quânticas empregadas no *Qiskit*. A etapa de maior custo computacional foi a simulação hamiltoniana, com mais de 80% das portas lógicas do circuito. Como efeito, essa é a etapa mais dispendiosa do algoritmo HHL, uma vez que exige a exponenciação da matriz do sistema matricial e a sua decomposição em portas quânticas de base.

Como comparação em relação a solução pelo vetor de estados e complexidade do circuito quântico, foi considerado o algoritmo pré-programado *Qiskit Aqua*. Este apresenta sua solução e complexidade idênticas para o primeiro sistema. Já para o segundo, embora apresente um erro menor (2.578%) e uma fidelidade maior (0.9997), possui um circuito quântico com complexidade consideravelmente superior (6 *q-bits*, *depth* 485 e 249 portas *CNOT*).

O algoritmo desenvolvido em *Qiskit* abarca a simulação de ambos os sistemas de estudo. Indo além, ele pode ser utilizado para simular sistemas matriciais de dimensão 2, 3 e 4, o que no problema de fluxo de potência se traduz em sistemas de 3, 4 e 5 barras. É admitida apenas a entrada de matrizes hermitianas, o que não se caracteriza como uma limitação no contexto do

fluxo de potencia CC, visto que a construção da matriz admitância exige sua hermiticidade. A fidelidade da solução e a complexidade computacional do circuito quântico dependem da matriz considerada e da precisão desejada na representação dos autovalores. A decomposição das matrizes hamiltonianas é feita através das funções disponibilizadas no *Qiskit*, que na versão atual não é capaz ir além da decomposição exata de matrizes de 2 *q-bits*. Caso versões posteriores expandam a função, o algoritmo poderia com pequenas adaptações abarcar sistemas maiores. O algoritmo desenvolvido em Matlab por outro lado, por ser uma simulação numérica que não depende de decomposição de matrizes, é capaz de ir ultrapassar o limiar da dimensão 4.

A prova de conceito de fluxo de potência CC para o maior sistema até o momento foi obtida por Eskandarpour et al. (2021) para sistema de 9 barras com uma matriz admitância de dimensão 8, tamanho que é justamente o estado da arte do algoritmo HHL. A expansão desse horizonte depende da evolução da tecnologia quântica e dos estudos relacionados a decomposição das matrizes da simulação hamiltoniana e se espera que HHL tenha grande impacto futuro em aplicações, embora não é provável que seja na era NISQ, tendo em vista que o potencial do algoritmo requer arquitetura tolerante a falhas (PRESKILL, 2018; BHARTI et al., 2021). Não obstante, diante dos problemas enfrentados na construção de uma rede elétrica modernizada, é importante que o potencial de aplicações da computação quântica seja estudado desde já e que provas de conceito e medidas matemáticas teóricas de desempenho computacional sejam desenvolvidas, não apenas no contexto de fluxo de potência, mas também quaisquer outras potenciais aplicações na rede elétrica para que a evolução experimental dos computadores quânticos possa ser aproveitada da melhor maneira possível.

5.1 TRABALHOS FUTUROS

Com base no algoritmo desenvolvido em *Qiskit*, alguns pontos do código podem explorados em trabalhos futuros:

- O estudo mais aprofundado da decomposição das matrizes na etapa de simulação hamiltoniana, que neste trabalho teve o emprego de funções do *Qiskit*.
- Para a etapa de rotação controlada, três métodos foram expostos. Nos casos em que não há representação exata, a solução obtida é diferente para cada método e em geral mais precisa para o método de maior complexidade. Poder-se-ia desenvolver um uma rotina mais elaborada em que o método mais apropriado fosse escolhido com base em algum parâmetro de tolerância como o desvio padrão.
- O estudo mais aprofundado de operadores de medida para extrair informações da solução do sistema matricial.
- É possível, caso se tenham decomposições conhecidas para as matrizes hamiltonianas de um sistema específico com dimensão maior 4, adaptar o algoritmo para a sua simulação.

- Poder-se-ia adaptar o código para implementar a etapa de rotação controlada sem o conhecimento a priori dos autovalores, conforme mostram Cao et al. (2012).

Além disso, alguns estudos podem ser propostos:

- Poder-se-ia adaptar o que foi desenvolvido neste trabalho como parte do algoritmo da arquitetura QPF proposta por Feng, Zhou e Zhang (2021) para fluxo de potência por métodos iterativos.
- Poder-se-ia realizar um estudo baseado em medidas computacionais teóricas de um sistema elétrico de escala e parâmetros reais para comparação entre a complexidade quântica e clássica.

REFERÊNCIAS

ABRAHAM, H. et al. **Qiskit**: an open-source framework for quantum computing. 2021.

ACM. **Association for computing machiner**: Computing curricula 2005: The overview report. 2005.

AJAGEKAR, A.; YOU, F. Quantum computing for energy systems optimization: challenges and opportunities. **Energy**, [S.l.], v. 179, p. 76–89, jul. 2019.

AJAGEKAR, A.; YOU, F. Quantum computing based hybrid deep learning for fault diagnosis in electrical power systems. **Applied Energy**, [S.l.], v. 303, p. 117628, 2021.

ALMUDÉVER, C. G. et al. The engineering challenges in quantum computing. **Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017**, [S.l.], p. 836–845, 2017.

AMBAINIS, A. Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations. **arXiv e-prints**, [S.l.], p. arXiv:1010.4458, out. 2010.

BARBOSA, A. A. **Introdução a Circuitos Quânticos**. 2005. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Campina Grande. Campina Grande, 2005.

BARENCO, A. et al. Elementary gates for quantum computation. **Physical Review A**, [S.l.], v. 52, n. 5, p. 3457–3467, nov. 1995.

BARZ, S. et al. A two-qubit photonic quantum processor and its application to solving systems of linear equations. **Scientific Reports**, [S.l.], v. 4, n. 1, ago. 2014.

BHARTI, K. et al. Noisy intermediate-scale quantum NISQ algorithms. **arXiv e-prints**, [S.l.], p. arXiv:2101.08448, jan. 2021.

BRASSARD, G. et al. Quantum amplitude amplification and estimation. **Quantum Computation and Information**, [S.l.], p. 53–74, 2002.

CAFARO, D. C.; GROSSMANN, I. E. Strategic planning, design, and development of the shale gas supply chain network. **AIChE Journal**, [S.l.], v. 60, n. 6, p. 2122–2142, 2014.

CAI, X.-D. et al. Experimental quantum computing to solve systems of linear equations. **Phys. Rev. Lett.**, [S.l.], v. 110, p. 230501, jun. 2013.

CALUDE, C. S.; CALUDE, E. The road to quantum computational supremacy. **arXiv e-prints 1712.01356**, [S.l.], p. 15, 2017.

CAO, Y. et al. Quantum circuit design for solving linear systems of equations. **Molecular Physics**, [S.l.], v. 110, n. 15-16, p. 1675–1680, ago. 2012.

CARVALHO, L.; LAVOR, C.; MOTTA, V. Caracterização matemática e visualização da esfera de bloch: ferramentas para computação quântica. **Trends in Computational and Applied Mathematics**, [S.l.], v. 8, n. 3, p. 351–360, 2007.

CASCIO, E. L.; MA, Z.; MARÉCHAL, F. How Smart is the Grid? **arXiv e-prints**, [S.l.], p. arXiv:2006.04943, jun. 2020.

CHIA, N.-H.; LIN, H.-H.; WANG, C. Quantum-inspired sublinear classical algorithms for solving low-rank linear systems. **arXiv e-prints**, [S.l.], p. arXiv:1811.04852, nov. 2018.

CHRISTIE, R. **Power systems test case archive**: 300 Bus Power Flow Test Case. 1993. Disponível em: https://labs.ece.uw.edu/pstca/pf300/pg_tca300bus.htm. Acesso em: 10 nov. 2021.

CHUANG, I. L.; GERSHENFELD, N.; KUBINEC, M. Experimental implementation of fast quantum searching. **Phys. Rev. Lett.**, [S.l.], v. 80, p. 3408–3411, abr. 1998.

CROSS, A. W. et al. Open quantum assembly language. **arXiv e-prints 1707.03429**, [S.l.], p. 1 – 24, 2017.

D-WAVE. **The quantum computing company. Applications**. 2021. Disponível em: https://www.dwavesys.com/learn/featured-applications/. Acesso em: 18 nov. 2021.

DAVIS, M. **Quantum computing for the future grid**. 2021. Disponível em: https://www.tdworld.com/smart-utility/data-analytics/article/21169579/quantum-computing-for-the-future-grid/. Acesso em: 10 nov.
2021.

DELAVARI, A. **Matlab central file exchange**: WSCC 9-bus test system IEEE benchmark. 2021. Disponível em: https://www.mathworks.com/matlabcentral/fileexchange/66555-ieee-5-bus-system-model. Acesso em: 10 jun. 2021.

DEUTSCH, D. Quantum theory, the church–turing principle and the universal quantum computer. **Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences**, [S.l.], v. 400, p. 117 – 97, 1985.

DOCUMENTATION, S. **Mathworks**: Simulation and Model-Based Design. 2021. Disponível em: https://www.mathworks.com/products/simulink.html. Acesso em: 10 nov. 2021.

EBERLY, W. et al. Solving Sparse Integer Linear Systems. **arXiv e-prints**, [S.l.], p. cs/0603082, mar. 2006.

ESKANDARPOUR, R. et al. Quantum computing solution of dc power flow. **arXiv e-prints 2010.02442**, [S.l.], p. 19, 2020.

ESKANDARPOUR, R. et al. Quantum computing for enhancing grid security. **IEEE Transactions on Power Systems**, [S.l.], v. 35, n. 5, p. 4135–4137, 2020.

ESKANDARPOUR, R. et al. Quantum-enhanced grid of the future: a primer. **IEEE Access**, [S.l.], v. 8, p. 188993–189002, 2020.

ESKANDARPOUR, R. et al. Experimental Quantum Computing to Solve Network DC Power Flow Problem. **arXiv e-prints**, [S.l.], p. arXiv:2106.12032, jun. 2021.

FENG, F.; ZHOU, Y.; ZHANG, P. Quantum power flow. **arXiv e-prints 2104.04888**, [S.l.], p. 4, 2021.

FEYNMAN, R. P. Simulating Physics with Computers. **International Journal of Theoretical Physics**, [S.l.], v. 21, n. 6-7, p. 467–488, jun. 1982.

GRIFFITHS, D. J. **Mecânica quântica**. 2. ed. [S.l.]: Pearson Universidades, 2011. 380 p.

GROVER, L. K. Quantum mechanics helps in searching for a needle in a haystack. **Physical Review Letters**, [S.l.], v. 79, n. 2, p. 325–328, jul. 1997.

HALLIDAY, D.; WALKER, J.; RESNICK, R. **Fundamentos de física**: volume 4: óptica e física moderna. [S.l.]: LTC, 2008.

HARRIS, C. R. et al. Array programming with NumPy. **Nature**, [S.l.], v. 585, n. 7825, p. 357–362, sep. 2020.

HARROW, A. W.; HASSIDIM, A.; LLOYD, S. Quantum algorithm for linear systems of equations. **Physical Review Letters**, [S.l.], v. 103, n. 15, out. 2009.

HECTOR JOSE MORRELL, J.; WONG, H. Y. Step-by-Step HHL Algorithm Walkthrough to Enhance the Understanding of Critical Quantum Computing Concepts. **arXiv e-prints**, [S.l.], p. arXiv:2108.09004, aug. 2021.

IBM. **IBM quantum experience**. 2018. Disponível em: https://quantum-computing.ibm.com. Acesso em: 20 nov. 2021.

JEONG, Y.-W. et al. A thermal unit commitment approach using an improved quantum evolutionary algorithm. **Electric Power Components and Systems**, [S.l.], v. 37, n. 7, p. 770–786, 2009.

JESUS, G. et al. **Quantum computing**: an undergraduate approach using qiskit. 2021.

JONES, J. S. **QUEST for quantum computing in the energy sector**. 2021. Disponível em: https://www.smart-energy.com/regional-news/north-america/quest-for-quantum-computing-in-the-energy-sector/. Acesso em: 18 nov. 2021.

KLUYVER, T. et al. **Jupyter notebooks – a publishing format for reproducible computational workflows**. 2016. 87 - 90 p.

KORETSKY, S. et al. Adapting Quantum Approximation Optimization Algorithm (QAOA) for Unit Commitment. **arXiv adsurl = https://ui.adsabs.harvard.edu/abs/2021arXiv211012624K, adsnote = Provided by the SAO/NASA Astrophysics Data System**, [S.l.].

LAU, T. W. et al. Quantum-inspired evolutionary algorithm approach for unit commitment. **IEEE Transactions on Power Systems**, [S.l.], v. 24, n. 3, p. 1503–1512, 2009.

LAY, D. C. **Linear algebra and its applications**. 3rd ed. ed. New Delhi: Pearson, 2003. Includes index.

MATLAB. **version 9.4.0.813654 (R2018a)**. Natick, Massachusetts: The MathWorks Inc, 2018.

MONTEIRO, H. A. **Emulação de Circuitos Quânticos em Placa FPG**. 2012. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Campina Grande. Campina Grande, 2012.

MONTICELLI, A. J. **Fluxo de carga em redes de energia elétrica**. São Paulo: Editora Edgard Blücher LTDA, 1983. 165 p.

MYTTENAERE, A. de et al. Mean absolute percentage error for regression models. **Neurocomputing**, [S.l.], v. 192, p. 38–48, jun. 2016.

NIELSEN, M. A.; CHUANG, I. L. **Quantum computation and quantum information**: 10th anniversary edition. 10th. ed. USA: Cambridge University Press, 2010.

OSKIN, M.; CHONG, F.; CHUANG, I. A practical architecture for reliable quantum computers. **Computer**, [S.l.], v. 35, p. 79–87, 2002.

PAN, J. et al. Experimental realization of quantum algorithm for solving linear systems of equations. **Physical Review A**, [S.l.], v. 89, n. 2, fev. 2014.

PORTUGAL, R.; MARQUEZINO, F. **Introdução à Programação de Computadores Quânticos**. In CSBC 2019 – 38º JAI, pages 1–51, Belém – Pará, 2019.

PRESKILL, J. Quantum computing in the NISQ era and beyond. **Quantum**, [S.l.], v. 2, p. 79, ago. 2018.

PYTHON SOFTWARE FOUNDATION. **Python language site**: Documentation. 2021. Disponível em: https://www.python.org/doc. Acesso em: 10 nov. 2021.

QISKIT. **Qiskit 0.31.0 documentation**. 2021. Disponível em: https://qiskit.org/documentation. Acesso em: 10 nov. 2021.

QISKIT. **Qiskit textbook (beta)**. 2021. Disponível em: https://qiskit.org/textbook-beta. Acesso em: 10 nov. 2021.

QISKIT. **HHL**. 2021. Disponível em: https://qiskit.org/documentation/stubs/qiskit.aqua.algorithms.HHL.html. Acesso em: 10 nov. 2021.

QISKIT. **Solving linear systems of equations using HHL**. 2021. Disponível em: https://qiskit.org/textbook/ch-applications/hhl_tutorial.html. Acesso em: 10 nov. 2021.

SANTOS, P. S. dos. **Q-bit: um novo fundamento lógico**. 2018. Trabalho de conclusão de curso (Bacharelado em Engenharia de Computação) - Universidade Federal de Ouro Preto. Ouro Preto, 2018.

SHEWCHUK, J. R. **An introduction to the conjugate gradient method without the agonizing pain**. USA: Carnegie Mellon University, 1994.

SHOR, P. Algorithms for quantum computation: discrete logarithms and factoring. **Proceedings 35th Annual Symposium on Foundations of Computer Science**, [S.l.], p. 124–134, 1994.

TAN, R. **Matlab central file exchange**: IEEE 5-Bus System Model. 2021. Disponível em: https://www.mathworks.com/matlabcentral/fileexchange/66555-ieee-5-bus-system-model. Acesso em: 10 nov. 2021.

TURING, A. M. On computable numbers, with an application to the entscheidungsproblem. **Proceedings of the London Mathematical Society**, [S.l.], v. s2-42, n. 1, p. 230–265, 1937.

VAZQUEZ, A. C.; HIPTMAIR, R.; WOERNER, S. Enhancing the Quantum Linear Systems Algorithm using Richardson Extrapolation. **arXiv e-prints**, [S.l.], p. arXiv:2009.04484, set. 2020.

VLACHOGIANNIS, J. G.; LEE, K. Y. Quantum-inspired evolutionary algorithm for real and reactive power dispatch. **IEEE Transactions on Power Systems**, [S.l.], v. 23, n. 4, p. 1627–1636, 2008.

WEN, J. et al. Experimental realization of quantum algorithms for a linear system inspired by adiabatic quantum computing. **Physical Review A**, [S.l.], v. 99, n. 1, jan. 2019.

ZINGER, J. **Algoritmo quântico para equações lineares**. 2015. Monografia (Bacharelado em Informática) - Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2015.

APÊNDICE A - CÓDIGO EM QISKIT

HHL

November 23, 2021

[ ]:
```
#Solução de equações lineares baseada no algoritmo quântico HHL
#Trabalho de Conclusão de Curso - Engenharia Elétrica
#Universidade de Caxias do Sul (UCS)
#Autor: Guilherme Adamatti Bridi
```

[ ]:
```
from qiskit import *
from qiskit import Aer, execute, IBMQ
from qiskit import QuantumCircuit, QuantumRegister
from qiskit.tools.monitor import job_monitor
from qiskit.tools.visualization import plot_histogram
from qiskit.compiler import transpile, assemble
from qiskit.transpiler import PassManager
from qiskit.extensions import UnitaryGate
from qiskit.providers.aer import AerSimulator
from qiskit.aqua.utils.controlled_circuit import get_controlled_circuit
from qiskit.algorithms.linear_solvers.hhl import HHL
from qiskit.quantum_info import Statevector
from qiskit.algorithms.linear_solvers.numpy_linear_solver import␣
↪NumPyLinearSolver
import numpy as np
from scipy import linalg
```

[ ]:
```
#----------------------Parâmetros de entrada-----------------------------

#Sistema 2x2
#B = np.array([[4, -2],
#              [-2, 4]])

#p = np.array([0.6, -0.8])

#Sistema 4x4
B =          np.array([[12, -2, 0, 0],
                       [-2, 12, -6, 0],
                       [0, -6, 12, -2],
                       [0, 0, -2, 12]])

p = np.array([0.7, -0.5, -0.5, -0.1])
```

```
DPtol = 10 #Desvio padrão admissível na representação binária, que pode␣
 ↪introduzir erros teóricos na solução do sistema
NQmax = 6 #Número de q-bits aceitável no registrador alpha

#-----------------Preparação dos parâmetros do circuito-----------------

#Teste de validade das dimensões do sistema linear
if len(B) != len(p) or len(B) > 4:
    sys.exit("Dimensões inadequadas")

eig = linalg.eig(B) #Autovalor da matriz B original
eig = np.abs(eig[0][0]) #Autovalor para complementar matrizes de ordem 3

#Adequação de sistemas de ordem 3
if np.ceil(np.log2(len(B))) > np.log2(len(B)):
    Baux = B
    paux = p
    B = eig * np.identity(pow(2, int(np.ceil(np.log2(len(B))))))
    p = np.zeros(pow(2, int(np.ceil(np.log2(len(B))))))
    B[0: len(Baux), 0: len(Baux)] = Baux
    p[0: len(Baux)] = paux

#Normalização do sistema linear
B = B / linalg.norm(p)
p = p / linalg.norm(p)

#Teste de hermiticidade
Bo = np.transpose(B)
for k in range(len(B)):
    for l in range(len(B)):
        if abs(B[k, l] - Bo[k, l]) > 0:
            sys.exit("Matriz não hermitiana")

nb = int(np.ceil(np.log2(len(p)))) #Número de q-bits do registrador beta

eigSave = linalg.eigh(B)
eigVectors = eigSave[1] #Autovetores da matriz B
eigSave = np.sort(eigSave[0]) #Autovalores da matriz B

#Autovalores diagonalizados
eigMatrix = np.identity(len(eigSave))
for k in range(len(eigSave)):
    eigMatrix[k][k] = np.abs(eigSave[k])

#Verifica se todos os autovalores são positivos
for k in range(len(eigSave)):
    if eigSave[k] < 0:
```

```
        sys.exit("Autovalor negativo ou nulo")

s = np.max(eigSave)/np.min(eigSave) #Número de condição
nk = int(np.floor((2**NQmax - 1) / s)) #Constante nk de controle que garante␣
 ↪NQmax

#Rotina de acomodação dos autovalores
for k in range(nk):
    estadosSave = [0, 0, 0, 0]
    estadosUnique = [0, 0, 0, 0]
    eigUnique = [0, 0, 0, 0]
    estadosSave[0] = int(k + 1) #Posição do menor autovalor

    for l in range(2**nb - 1):
        estadosSave[l + 1] = int(np.around(estadosSave[0] * eigSave[l + 1] /␣
 ↪eigSave[0])) #%Posição dos demais autovalores

    estadosUn = np.unique(estadosSave) #Elimina posições repetidas

    #Elimina posições 0
    g = 0
    for l in range(len(estadosUn)):
        if int(estadosUn[l]) != 0:
            estadosUnique[g] = int(estadosUn[l])
            g += 1

    #Obtém os autovalores considerados únicos
    g = 0
    h = 1
    for l in range(2**nb):
        if estadosSave[l] == estadosUnique[g]:
            h = 1
            eigUnique[g] = eigSave[l]
            g += 1

        else:
            eigUnique[g - 1] = (h * eigUnique[g - 1] + eigSave[l]) / (h + 1)
            h += 1

    nrSave = g #Número de rotações controladas
    estadosUnique = estadosUnique[0 : nrSave] #Elimina posições 0
    eigUnique = eigUnique[0 : nrSave] #Elimina posições 0
    nqSave = int(np.floor(np.log2(estadosUnique[nrSave - 1] * 2))) #Número de␣
 ↪q-bits no registrador alpha

    #Cálculo da constante t para cada autovalor
    ti = np.zeros(2**nb)
```

```
    for l in range(2**nb):
        ti[l] = estadosSave[l] / (eigSave[l] * 2**nqSave) * 2 * np.pi

    tSave = np.mean(ti) #t médio
    dpSave = 100 * np.std(ti) / tSave #Desvio padrão de t

    #Verifica se dp é menor igual ao aceitável
    if dpSave <= DPtol:
        #Armazena os parâmetros e quebra o laço
        dp = dpSave
        nr = nrSave
        nq = nqSave
        t = tSave
        estados = estadosUnique
        eig = eigUnique
        break

#Verifica se o erro está abaixo do tolerável
if dp > DPtol:
    sys.exit("Erro não tolerável")

C = abs(np.amin(eigSave)) #C definido de forma maximizar a probabilidade de␣
 ↪leitura 1 no q-bit ancilla

R = np.zeros(nr)
for k in range(nr):
    estados[k] = bin(int(estados[k]))[2:].zfill(nq) #Estados em binário
    R[k] = 2 * np.arcsin(C / eig[k]) #Calcula ângulos de rotação original

#Matriz auxiliar de posições binárias
eigMat = np.zeros((nq, nr))
for k in range(nq):
    for l in range(nr):
        eigMat[k][l] = int(estados[l][k])

#Laço para obter R, P e controle
Po = np.zeros(nr)
controle = np.zeros(nr)
for k in range(nr):
        comp = eigMat[:, k]
        menor = 4

        for l in range(nq):
            cont = 0
            igual = np.zeros(nr)
            for g in range(nr):
                if eigMat[l][g] == comp[l] and g != k:
```

```
                    cont += 1
                    igual[g] -= R[k]

            if cont < menor:
                menor = cont
                Po[k] = nq - l - 1
                Raux = igual
                controle[k] = eigMat[l][k]

        R += Raux


#Escrita dos pârametros
print('nb =', nb)
print('nq =', nq)
print('nr =', nr)
print('t =', t)
print('Dp% =', dp, '%')
print('C =', C)
print('Autovalores =', abs(eigSave))
print('s =', s)
print('Autovalores unicos =', eig)
print('estados =', estados)
print('R =', R)
print('P =', Po)
print('controle =', controle)
```

[ ]:
```
#-------------------Construção do circuito quântico-------------------------

#Registradores
alpha = QuantumRegister(nq, name = 'alpha')
beta = QuantumRegister(nb, name = 'beta')
ancilla = QuantumRegister(1, name = 'ancilla')
c = ClassicalRegister(1 + nb, name = 'c')

#Circuito
circuito = QuantumCircuit(beta, alpha, ancilla, c)

#Vetor p
circuito.initialize(p, beta)
#circuito.barrier()

#Hadamard
circuito.h(alpha)
#circuito.barrier()

#U-controlled
```

```
for k in range(nq):
    diag = np.identity(len(eigSave), dtype=complex)
    for l in range(len(eigSave)):
        diag[l][l] = np.exp(1j * eigMatrix[l][l] * t * 2**k)

    Bex = np.matmul(np.matmul(eigVectors, diag), np.transpose(eigVectors))␣
 ↪#Matriz exponenciada

    QR = QuantumRegister(nb)
    QC = QuantumCircuit(QR)
    QC.unitary(Bex, QR) #Unitary gate
    U = QC.to_gate().control(1) #Controlled gate

    if nb == 1: #Sistema 2x2
        circuito.append(U, [alpha[k], beta[0]]) #Append

    elif nb == 2: #Sistema 4x4
        circuito.append(U, [alpha[k], beta[0], beta[1]]) #Append

    #circuito.barrier()

#circuito.barrier()

#QTF'
for k in range(nq//2):
    circuito.swap(alpha[k], alpha[nq - k - 1])

for n in range(nq):
    for k in range(n, -1, -1):
        if k != n:
            circuito.cp(-np.pi/2**(n - k), alpha[k], alpha[n])
    circuito.h(alpha[n])

#circuito.barrier()

#Rotação
for k in range(nr):
    if controle[k] == 0:
        circuito.x(alpha[int(Po[k])])

    circuito.cry(R[k], alpha[int(Po[k])], ancilla[0])

    if controle[k] == 0:
        circuito.x(alpha[int(Po[k])])

    #circuito.barrier()
```

```
#circuito.barrier()

#QTF
for n in range(nq - 1, -1, -1):
    circuito.h(alpha[n])

    for k in range(n):
        circuito.cp(np.pi/2**(n - k), alpha[k], alpha[n])

for k in range(nq//2):
    circuito.swap(alpha[k], alpha[nq - k - 1])

#circuito.barrier()

#U-controlled'
for k in range(nq):
    diag = np.identity(len(eigSave), dtype=complex)
    for l in range(len(eigSave)):
        diag[l][l] = np.exp(-1j * eigMatrix[l][l] * t * 2**k)

    Bex = np.matmul(np.matmul(eigVectors, diag), np.transpose(eigVectors))␣
 ↪#Matriz exponenciada

    QR = QuantumRegister(nb)
    QC = QuantumCircuit(QR)
    QC.unitary(Bex, QR) #Unitary gate
    U = QC.to_gate().control(1) #Controlled gate

    if nb == 1: #Sistema 2x2
        circuito.append(U, [alpha[k], beta[0]]) #Append

    elif nb == 2: #Sistema 4x4
        circuito.append(U, [alpha[k], beta[0], beta[1]]) #Append

    #circuito.barrier()

#circuito.barrier()

#Hadamard
circuito.h(alpha)
#circuito.barrier()

circuito.draw('mpl')
```

[ ]:
```
#------------------Solução pelo vetor de estado-------------------------
```

```
back_vector = Aer.get_backend('statevector_simulator') #statevector_simulator
 ↪bakend
job_vector = execute(circuito, back_vector)
resultado = job_vector.result()
vector = resultado.get_statevector(circuito, decimals=10) #Vetor de estados

classica = np.array(linalg.solve(B, p)) #Solução clássica
quantica = np.array(vector[2**(nq + nb): (2**(nq + nb) + 2**nb)]) / C #Solução
 ↪quântica

erro = 100 * np.mean(np.abs((quantica - classica) / classica)) #Erro teórico
fidelidade = state_fidelity(quantica/ linalg.norm(quantica), classica / linalg.
 ↪norm(classica)) #Fidelidade
probabilidade = 100 * np.abs(sum(quantica * quantica) * C**2) #Probabilidade de
 ↪medida

#Escreve os resultados
print('Solução clássica:', classica)
print('Solução quântica:', quantica)
print('Erro teórico:', erro, '%')
print('Fidelidade:', fidelidade)
print('p(ancilla = 1):', probabilidade, '%')
```

[ ]:
```
#----------------------Medida-----------------------------

#Se o sistema for 2x2, se aplica o operador Hadamard no registrador alpha
if nb == 1:
    circuito.h(beta[0])

for k in range(nb): #Medida do registrador alpha
    circuito.measure(beta[k], c[k])

circuito.measure(ancilla, c[nb]) #Medida do registrador ancilla

circuito.draw('mpl')
```

[ ]:
```
#----------------------Transpilação do circuito----------------------------

transp = transpile(circuito, basis_gates = ['u', 'cx'], optimization_level = 3)
transp.draw(output='mpl')
```

[ ]:
```
#----------------------Tamanho do circuito----------------------------

print('U:', transp.count_ops()['u']) #Portas U
print('CNOT:', transp.count_ops()['cx']) #Portas CNOT
print('Depth:', transp.depth()) #Depth
```

```
print('Portas:', transp.count_ops()['cx'] + transp.count_ops()['u']) #Total de␣
↪portas
print('Q-bits:', nq + nb + 1) #Número de q-bits
print('Bits clássicos:', nb + 1) #Número de bits clássicos
```

[ ]:
```
#----------------------Solução do Qiskit Aqua----------------------------

classica = np.array(linalg.solve(B, p)) #Solução clássica
aqua = HHL().solve(B, p)
aqua_vector = Statevector(aqua.state).data
nq = aqua.state.width() - nb - 1
aqua_solution = np.array(aqua_vector[2**(nq + nb): (2**(nq + nb) + 2**nb)])
aqua_solution = np.real(aqua_solution)
aqua_solution = aqua.euclidean_norm * aqua_solution / linalg.
↪norm(aqua_solution) #Solução do Qiskit Aqua
aqua_circ = transpile(aqua.state, basis_gates = ['u', 'cx'], optimization_level␣
↪= 3) #Transpilação do circuito
erro = 100 * np.mean(np.abs((aqua_solution - classica) / classica)) #Erro␣
↪teórico
fidelidade = state_fidelity(aqua_solution/ linalg.norm(aqua_solution), classica␣
↪/ linalg.norm(classica)) #Fidelidade

#Escreve os resultados do Aqua
print('Solução Aqua:', aqua_solution)
print('Solução Clássica:', classica)
print('Erro teórico:', erro)
print('Fidelidade:', fidelidade)
print('U:', aqua_circ.count_ops()['u'])
print('CNOT:', aqua_circ.count_ops()['cx'])
print('Depth:', aqua_circ.depth())
print('Portas:', aqua_circ.count_ops()['cx'] + aqua_circ.count_ops()['u'])

print(aqua.state)
```

[ ]:
```
#----------------------Simulação ideal----------------------------

back_sim = Aer.get_backend('qasm_simulator') #qasm_simulator bakend
job_sim = back_sim.run(transpile(transp, back_sim), shots=8192)
result_sim = job_sim.result()
ideal_counts = result_sim.get_counts(circuito) #Contagem
plot_histogram([ideal_counts]) #Histrograma
```

[ ]:
```
#----------------------Computador quântico real----------------------------

IBMQ.save_account('API Token', overwrite=True)
provider = IBMQ.load_account()
```

```
back_ibm = provider.get_backend('ibmq_quito')
job_ibm = back_ibm.run(transpile(transp, back_ibm), shots=8192)
result_ibm = job_ibm.result()
real_counts = result_ibm.get_counts(transp) #Contagem

#Cálculo da solução real
for k, v in real_counts.items():
    if k[0] == "1" and k[1] == "1":
        menos = 2 * v / (8192 * C**2) #(teta1 - teta2)**2
    elif k[0] == "1" and k[1] == "0":
        mais = 2 * v / (8192 * C**2) #(teta1 + teta2)**2

real = np.array([(np.sqrt(menos) - np.sqrt(mais)) / 2, (np.sqrt(menos) + np.
 ↪sqrt(mais)) / 2]) #Solução real

classica = np.abs(classica) #Solução clássica em módulo para calcular o erro
erro = 100 * np.mean(np.abs((real - classica) / classica)) #Erro teórico
fidelidade = state_fidelity(real/ linalg.norm(real), classica / linalg.
 ↪norm(classica)) #Fidelidade

#Escreve os resultados
print('Solução clássica:', classica)
print('Solução real:', real)
print('Erro teórico:', erro, '%')
print('Fidelidade:', fidelidade)

plot_histogram([ideal_counts, real_counts], legend=['Qasm simulator',␣
 ↪'ibmq_quito']) #Histrograma ideal versus real
```

APÊNDICE B - CÓDIGO EM MATLAB

```
%Solução de equações lineares baseada no algoritmo quântico HHL
%Trabalho de Conclusão de Curso - Engenharia Elétrica
%Universidade de Caxias do Sul (UCS)
%Autor: Guilherme Adamatti Bridi

%-----------------------------------------------------------------

format long
clc; close all; clear all

file = fopen('parametros.txt', 'w');
resultados = fopen('resultados.txt', 'w');

%--------------------Parâmetros de entrada----------------------

%Sistema 2x2
%B = [4, -2;
%     -2, 4];

%p = [0.6, -0.8]';

%Sistema 4x4
B = [12, -2, 0, 0;
     -2, 12, -6, 0;
     0, -6, 12, -2;
     0, 0, -2, 12];

p = [0.7, -0.5, -0.5, -0.1]';

DPtol = 10; %Desvio padrão admissível na representação binária, que
 pode introduzir erros teóricos na solução do sistema
NQmax = 9; %Número de q-bits aceitável sendo 13 o suportável pela
 memória do MATLAB

%Escrita dos parâmetros de entrada
fprintf(file, '--------------------------------ARQUIVO DOS
 PARAMETROS--------------------------------------\n\n');
fprintf(file, '------------------------------Parametros de
 entrada-------------------------------------\n\n');
escreve(B, file, 'B');
escreve(p, file, 'p');
fprintf(file, '\nNQmax = %d\n', NQmax);
fprintf(file, 'DPtol = %3.3f%%\n\n', DPtol);

%-------------Preparação dos parâmetros do circuito-----------------

%Teste de validade das dimensões do sistema linear
if length(B) ~= length(p)
    fprintf(file, 'Dimensões inadequadas');
    fclose(file);
    return
end
```

```
eigi = sort(eig(B)); %Autovalores da matriz B original
eigi = abs(eigi(1)); %Autovalor para complementar matrizes de ordem
 não múltipla de 2

%Adequação de sistemas de ordem não múltipla de 2
if ceil(log2(length(B))) > log2(length(B))
    for k = length(B): ceil(log2(length(B)))^2 - 1
        B(k, k) = eigi;
        p(k) = 0;
    end
end

%Normalização do sistema linear
B =  B / norm(p);
p =  p / norm(p);

%Teste de hermiticidade
Bo = B';
for k = 1: length(B)
    for l = 1: length(B)
        if abs(B(k, l) - Bo(k, l)) > 0.001
            fprintf(file, 'Matriz não hermitiana');
            fclose(file);
            return
        end
    end
end

nb = floor(log2(length(p))); %Número de q-bits do registrador beta

eigSave = sort(eig(B)); %Autovalores da matriz B
[eigVectors, eigMatrix] = eig(B); %Matriz diagonalizada

%Verifica se todos os autovalores são positivos
for k = 1: 2^nb
    if eigSave(k) <= 0
        fprintf(file, 'Autovalor negativo ou nulo');
        fclose(file);
        return
    end
end

s = max(eigSave) / min(eigSave); %Número de condição
nk = floor((2^(NQmax - nb - 1) - 1) / s); %Constante nk de controle
 que garante NQmax

%Rotina de acomodação dos autovalores
for k = 0: nk
    estadosSave(1) = k + 1; %Posição do menor autovalor

    for l = 0: 2^nb - 1
        estadosSave(l + 1) = round(estadosSave(1) * eigSave(l + 1) /
 eigSave(1)); %Posição dos demais autovalores
```

```
    end

    estadosUnique = unique(estadosSave); %Elimina posições repetidas
    nrSave = length(estadosUnique); %Número de rotações controladas
    nqSave = floor(log2(estadosUnique(nrSave) * 2)); %Número de q-bits
 no registrador alpha

    %Obtém os autovalores considerados únicos
    g = 1;
    h = 1;
    for l = 1: 2^nb
        if estadosSave(l) == estadosUnique(g)
            h = 1;
            eigUnique(g) = eigSave(l);
            g = g + 1;

        else
            eigUnique(g - 1) = (h * eigUnique(g - 1) + eigSave(l)) /
 (h + 1);
            h = h + 1;
        end
    end

    %Cálculo da constante t para cada autovalor
    for l = 1: 2^nb
        ti(l) = estadosSave(l) / (eigSave(l) * 2^nqSave) * 2 * pi;
    end

    tSave = mean(ti); %t médio

    %Cálculo do desvio padrão de t
    dpSave = 0;
    for l = 1: 2^nb
        dpSave = dpSave + (ti(l) - tSave)^2;
    end
    dpSave = 100 * sqrt(dpSave / 2^nb) / tSave;

    %Verifica se dp é menor igual ao aceitável
    if dpSave <= DPtol
        %Armazena os parâmetros e quebra o laço
        dp = dpSave;
        nr = nrSave;
        nq = nqSave;
        t = tSave;
        estados = estadosUnique;
        eig = eigUnique;
        break
    end
end

%Verifica se o erro está abaixo do tolerável
if dp > DPtol
    fprintf(file, 'Erro não tolerável');
    fclose(file);
```

```
    return
end

C = abs(eigSave(1)); %C definido de forma maximizar a probabilidade de
 leitura 1 no q-bit ancilla

bin = dec2bin(estados', nq); %Estados em binário

for k = 1: nr
    R(k) = 2 * asin(C / eig(k)); %Calcula ângulos de rotação original
end

%Matriz auxiliar de posições binárias
eigMat = zeros(nq, nr);
for k = 1: nq
    for l = 1: nr
        eigMat(k, l) = bin(l, k);
    end
end

%Laço para obter R, P e controle
for k = 1: nr
    comp = eigMat(:, k);
    menor = 8;

    for l = 1: nq
        cont = 0;
        igual = zeros(1, nr);

        for g = 1: nr
            if eigMat(l, g) == comp(l) && g ~= k
                    cont = cont + 1;
                    igual(g) = igual(g) - R(k);
            end
        end

        if cont < menor
             menor = cont;
             Po(k) = nq - l;
             Raux = igual;
             controle(k) = eigMat(l, k);
        end
    end

    R = R + Raux;
end

%Escrita dos parâmetros do circuito
fprintf(file, '---------------------------Preparaçao dos parametros do
 circuito----------------------------\n\n');
fprintf(file, 'Sistema quantico:\n\n');
escreve(B, file, 'B');
escreve(p, file, 'p');
escreve(eigSave, file, 'Autovalores');
```

```
fprintf(file, 's = %3.3f\n', s);
fprintf(file, 'nq = %d\n', nq);
fprintf(file, 'nb = %d\n\n', nb);
fprintf(file, '%d q-bits\n', nq + nb + 1);
fprintf(file, '%d bits classicos\n\n', nb + 1);
fprintf(file, 'nr = %d\n\n', nr);
escreve(eig', file, 'Autovalores unicos');
fprintf(file, 'Estados:\n');

for k = 1: nr
    fprintf(file, '%s <---> %d\n', string(bin(k, :)), estados(k));
end

fprintf(file, '\nDP%% = %3.3f%%', dp);
fprintf(file, '\nt = %4.4f', t);
fprintf(file, '\nC = %4.4f\n\n', C);

%-----------------------Algoritmo-------------------------------

fprintf(file, '-------------------------Matrizes do
 algoritmo-------------------------------------\n\n');

%Preparação do estado quântico p
psy0 = [1, 0]';

for k = 1: nq
    psy0 = kron([1, 0]', psy0);
end

psy0 = kron(psy0, p);

%Transformação Hadamard no registrador alpha
psy1 = kron((kron(eye(2, 2), hadamard(nq))), eye(2^nb, 2^nb)) * psy0;
fprintf(file, 'Transformacao Hadamard:\n\n');
fprintf(file, 'h([0, %d])\n\n', nq - 1);

%U-controlled para a matriz e^iBt
psy2 = psy1;

fprintf(file, 'Matrizes hamiltonianas:\n');

for k = 0: nq - 1
    diag = eye(2^nb, 2^nb);
    for l = 1: 2^nb
        diag(l, l) = exp(2^k * eigMatrix(l, l) * j * t);
    end

    psy2 = kron(eye(2^(nq - k), 2^(nq - k)), controlled1(kron(eye(2^k,
 2^k), eigVectors * diag * eigVectors'))) * psy2;

    fprintf(file, '\nQ-bit %d <---> %4.4f e^iBt\n\n', k, 2^(k));
    escreve(eigVectors * diag * eigVectors', file, 'U');

end
```

```
%Transformada de Fourier Quântica inversa no registrador alpha
fprintf(file, '\nTransformada de Fourier Quantica inversa:\n\n');

for k = 0: nq/2 - 1
     fprintf(file, 'SWAP(%d, %d)\n', k, nq - k - 1);
end

for k = 0: 1: nq - 1
    for l = 0: 1: nq - 2
        if k - l > 0
            fprintf(file, 'p(-%4.4f pi)(%d, %d)\n', (2^(l - k)), k,
 l);
        end
    end

    fprintf(file, 'h(%d)\n', k);
end

psy3 = kron(kron(eye(2, 2), QFT(2^nq)'), eye(2^nb, 2^nb)) * psy2;

%Rotação controlada no registrador ancilla
fprintf(file, '\nRotacao controlada:\n');

psy4 = psy3;

%Rotação controlada
for k = 1: nr
    X = [0, 1; 1, 0];
    Pinv = nq - Po - 1;

    fprintf(file, '\nPosiçao %d', Po(k));
    fprintf(file, '\nControle %s', string(controle(k) - 48));
    fprintf(file, '\nRotacao Ry(%5.5f)\n\n', R(k));
    escreve(Ry(R(k)), file, 'Ry');

    if controle(k) == '0'
        %Porta X na entrada
        psy4 = kron(kron(eye(2^(Pinv(k) + 1), 2^(Pinv(k) + 1)), X),
 eye(2^(nb + nq - 1 - Pinv(k)), 2^(nb + nq - 1 - Pinv(k)))) * psy4;
    end

    %Rotação controlada
    psy4 = kron(controlled2(Ry(R(k)), Pinv(k) + 2), eye(2^(nb + nq - 1
 - Pinv(k)), 2^(nb + nq - 1 - Pinv(k)))) * psy4;

    if controle(k) == '0'
         %Porta X na saída
        psy4 = kron(kron(eye(2^(Pinv(k) + 1), 2^(Pinv(k) + 1)), X),
 eye(2^(nb + nq - 1 - Pinv(k)), 2^(nb + nq - 1 - Pinv(k)))) * psy4;
    end
end

%Transformada de Fourier Quântica no registrador alpha
```

```
fprintf(file, '\nTransformada de Fourier Quantica:\n\n');

for k = nq - 1: -1: 0
    fprintf(file, 'h(%d)\n', k);
    for l = nq - 2: -1: 0
        if k - l > 0
            fprintf(file, 'p(%4.4f pi)(%d, %d)\n', (2^(l - k)), k, l);
        end
    end
end

for k = 0: nq/2 - 1
     fprintf(file, 'SWAP(%d, %d)\n', k, nq - k - 1);
end

psy5 = kron(kron(eye(2, 2), QFT(2^nq)), eye(2^nb, 2^nb)) * psy4;

%U-controlled para a matriz e^-iBt
psy6 = psy5;

fprintf(file, '\nMatrizes hamiltonianas inversas:\n');

diag = eye(2^nb, 2^nb);
for k = nq - 1: -1: 0
    for l = 1: 2^nb
        diag(l, l) = exp(-2^k * eigMatrix(l, l) * j * t);
    end

    psy6 = kron(eye(2^(nq - k), 2^(nq - k)), controlled1(kron(eye(2^k,
 2^k), eigVectors * diag * eigVectors'))) * psy6;

    fprintf(file, '\nQ-bit %d <---> %4.4f e^iBt\n\n', k, 2^(k));
    escreve(eigVectors * diag * eigVectors', file, 'U');
end

%Transformação Hadamard no registrador alpha
psy7 = kron((kron(eye(2, 2), hadamard(nq))), eye(2^nb, 2^nb)) * psy6;
fprintf(file, '\nTransformacao Hadamard:\n\n');
fprintf(file, 'h([0, %d])', nq - 1);

%-------------------Resultados----------------------
quantica = zeros(1, 2^nb)';
histograma = zeros(1, 2^(nb + 1))';
histDec = zeros(1, 2^(nb + 1))';

probabilidade = 0;
for k = 1: 2^nb
  histograma(k, 1) = psy7(k)^2; %Localiza a solução na posição
 esperada
  histograma(k + 2^nb, 1) = psy7(2^(nq + nb) + k)^2; %Localiza a
 solução na posição esperada
  probabilidade = probabilidade + histograma(k + 2^nb, 1) *
 100; %Calcula a probabilidade do estado 1, isto é, do sucesso do
 algoritmo, no q-bit ancilla
```

```
  quantica(k, 1) = psy7(2^(nq + nb) + k); %Localiza a solução na
 posição esperada
end

for k = 1: 2^(nb + 1)
  histDec(k) = k - 1; %Estados quânticos em decimal
end

histBin = dec2bin(histDec', nb + 1); %Estados quânticos em binário
eX = categorical(string(histBin)'); %Eixo x do gráfico
%grafico = bar(eX, abs(histograma)'); %Histograma
%text(1:length(abs(histograma)'),
 abs(histograma)',num2str(abs(histograma)), 'vert', 'bottom', 'horiz',
 'center'); %Mostra valores
%title('Medida') % Título do histrograma
%xlabel('Estados') % Eixo horizontal
%ylabel('Probabilidade') % Eixo vertical
%grid on

classica = linsolve(B, p); %Solução clássica
quantica = quantica / C; %Solução quântica
erro = 100 * mean(abs((quantica - classica) ./ classica)); %Erro
 teórico
classicaN = classica / norm(classica); %Solução clássica normalizada
quanticaN = quantica / norm(quantica); %Solução quântica normalizada
fidelidade = abs(trace(sqrtm(sqrtm(classicaN * classicaN')
 * (quanticaN * quanticaN') * sqrtm(classicaN *
 classicaN'))))^2; %Fidelidade

%Escrita dos resultados
fprintf(resultados, '--------------------------------ARQUIVO DOS
 RESULTADOS---------------------------------------\n\n');
fprintf(resultados, '----------------------
Solucao---------------------------\n\n');
escreve(classica, resultados, 'Classica');
escreve(quantica, resultados, 'Quantica');
fprintf(resultados, 'Erro teorico = %3.3f%%\n', erro);
fprintf(resultados, 'Fidelidade = %5.5f\n\n', fidelidade);

fprintf(resultados, '----------------------
Histograma---------------------------\n\n');
fprintf(resultados, 'p(ancilla = 1) = %3.3f%%\n\n', probabilidade);

for k = 1: 2^(nb + 1)
   fprintf(resultados, '%s <---> %4.4f\n', string(histBin(k, :)),
 histograma(k));
end

fprintf(resultados, '\n--------------------Vetores de
 estado--------------------------\n\n');
escreve(psy0, resultados, 'Estado p');
escreve(psy1, resultados, 'Hadamard');
escreve(psy2, resultados, 'U-controlled');
escreve(psy3, resultados, 'QTF-1');
```

```
escreve(psy4, resultados, 'Rotacao');
escreve(psy5, resultados, 'QTF');
escreve(psy6, resultados, 'U-controlled-1');
escreve(psy7, resultados, 'Final');

fclose(file);
fclose(resultados);

%--------------------Funções---------------------

%Transformação Hadamard
function H = hadamard(n)

H1 = [1, 1; 1, -1] / sqrt(2);

H = 1;

for k = 1 : n
    H = kron(H, H1);
end

end

%Escreve um vetor/matriz
function escreve(B, file, nome)

Dim = size(B);

fprintf(file, '%s = \n', nome);

for k = 1: Dim(1)
    fprintf(file, '          ');

    for l = 1: Dim(2)
        fprintf(file, ' ');
        fprintf(file, '%5.5f + %5.5f j ', real(B(k, l)), imag(B(k,
 l)));
    end

    fprintf(file, '\n');
end

end

%Operador Ry(teta)
function [R] = Ry(th)

R = [cos(th/2), - sin(th/2); sin(th/2), cos(th/2)];

end

%Operador transformada de Fourier
function m = QFT(n)
```

```
w = exp(2 * pi * j / n);
s = zeros(n);
row = 0: (n - 1);

for r = 1 : n
    m(r, :) = row *(r - 1);
end

m = w .^ m;
m = m ./ sqrt(n);

end

%Operação controlada tipo 1
function [C] = controlled1(U)

T = size(U);
C = eye (2 * T(1) , 2 * T(2));
k = T(1) + 1;
f = 2 * T(1);
grp = k: f;
C(grp, grp) = U;

end

%Operação controlada tipo 2
function [C] = controlled2(U, n)

N = 2^n;
C = eye(N, N);
sobe = 2^(n - 1);

for k = 2: 2: N/2;
    C(k, k) = U(1, 1);
    C(k, k + sobe) = U(1, 2);
    C(k + sobe, k) = U(2, 1);
    C(k + N/2, k + N/2) = U(2, 2);

end

end
```

Published with MATLAB® R2018a

APÊNDICE C - ARQUIVOS TEXTO GERADOS EM MATLAB

Sistema 2x2

```
--------------------------------ARQUIVO DOS PARAMETROS----------------------------------
------

------------------------------Parametros de entrada-------------------------------------
--

B =
           4.00000 + 0.00000 j  -2.00000 + 0.00000 j
           -2.00000 + 0.00000 j  4.00000 + 0.00000 j
p =
           0.60000 + 0.00000 j
           -0.80000 + 0.00000 j

NQmax = 9
DPtol = 10.000%

--------------------------Preparação dos parametros do circuito----------------------
-----

Sistema quantico:

B =
           4.00000 + 0.00000 j  -2.00000 + 0.00000 j
           -2.00000 + 0.00000 j  4.00000 + 0.00000 j
p =
           0.60000 + 0.00000 j
           -0.80000 + 0.00000 j
Autovalores =
           2.00000 + 0.00000 j
           6.00000 + 0.00000 j
s = 3.000
nq = 2
nb = 1

4 q-bits
2 bits classicos

nr = 2

Autovalores unicos =
           2.00000 + 0.00000 j
           6.00000 + 0.00000 j
Estados:
01 <---> 1
11 <---> 3

DP% = 0.000%
t = 0.7854
C = 2.0000

-------------------------Matrizes do algoritmo-------------------------------------
```

```
Transformacao Hadamard:

h([0, 1])

Matrizes hamiltonianas:

Q-bit 0 <---> 1.0000 e^iBt

U =
           -0.00000 + 0.00000 j  0.00000 + 1.00000 j
           0.00000 + 1.00000 j  -0.00000 + 0.00000 j

Q-bit 1 <---> 2.0000 e^iBt

U =
           -1.00000 + 0.00000 j  0.00000 + -0.00000 j
           0.00000 + -0.00000 j  -1.00000 + 0.00000 j

Transformada de Fourier Quantica inversa:

SWAP(0, 1)
h(0)
p(-0.5000 pi)(1, 0)
h(1)

Rotacao controlada:

Posi莽茫o 1
Controle 0
Rotacao Ry(3.14159)

Ry =
           0.00000 + 0.00000 j  -1.00000 + 0.00000 j
           1.00000 + 0.00000 j  0.00000 + 0.00000 j

Posi莽茫o 1
Controle 1
Rotacao Ry(0.67967)

Ry =
           0.94281 + 0.00000 j  -0.33333 + 0.00000 j
           0.33333 + 0.00000 j  0.94281 + 0.00000 j

Transformada de Fourier Quantica:

h(1)
p(0.5000 pi)(1, 0)
h(0)
SWAP(0, 1)

Matrizes hamiltonianas inversas:
```

```
Q-bit 1 <---> 2.0000 e^iBt

U =
           -1.00000 + -0.00000 j  0.00000 + 0.00000 j
           0.00000 + 0.00000 j  -1.00000 + -0.00000 j

Q-bit 0 <---> 1.0000 e^iBt

U =
           -0.00000 + 0.00000 j  0.00000 + -1.00000 j
           0.00000 + -1.00000 j  -0.00000 + 0.00000 j

Transformacao Hadamard:

h([0, 1])
```

```
---------------------------------ARQUIVO DOS RESULTADOS--------------------
--------------------

----------------------Solucao----------------------------

Classica =
           0.06667 + 0.00000 j
           -0.16667 + 0.00000 j
Quantica =
           0.06667 + -0.00000 j
           -0.16667 + -0.00000 j
Erro teorico = 0.000%
Fidelidade = 1.00000

----------------------Histograma----------------------------

p(ancilla = 1) = 12.889%

00 <---> 0.4356
01 <---> 0.4356
10 <---> 0.0178
11 <---> 0.1111

--------------------Vetores de estado--------------------------

Estado p =
           0.60000 + 0.00000 j
           -0.80000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
Hadamard =
           0.30000 + 0.00000 j
           -0.40000 + 0.00000 j
           0.30000 + 0.00000 j
           -0.40000 + 0.00000 j
           0.30000 + 0.00000 j
           -0.40000 + 0.00000 j
           0.30000 + 0.00000 j
           -0.40000 + 0.00000 j
           0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           0.00000 + 0.00000 j
U-controlled =
```

```
            0.30000 + 0.00000 j
            -0.40000 + 0.00000 j
            -0.00000 + -0.40000 j
            0.00000 + 0.30000 j
            -0.30000 + 0.00000 j
            0.40000 + -0.00000 j
            0.00000 + 0.40000 j
            -0.00000 + -0.30000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
QTF-1 =
            0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            -0.10000 + 0.00000 j
            -0.10000 + -0.00000 j
            0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            0.70000 + 0.00000 j
            -0.70000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
Rotacao =
            0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            -0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            0.65997 + 0.00000 j
            -0.65997 + 0.00000 j
            0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            -0.10000 + 0.00000 j
            -0.10000 + -0.00000 j
            0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            0.23333 + 0.00000 j
            -0.23333 + 0.00000 j
QTF =
            0.32998 + 0.00000 j
            -0.32998 + -0.00000 j
            -0.00000 + -0.32998 j
            0.00000 + 0.32998 j
            -0.32998 + 0.00000 j
            0.32998 + -0.00000 j
            0.00000 + 0.32998 j
            -0.00000 + -0.32998 j
            0.06667 + 0.00000 j
```

```
            -0.16667 + -0.00000 j
            0.00000 + -0.16667 j
            -0.00000 + 0.06667 j
            -0.06667 + 0.00000 j
            0.16667 + -0.00000 j
            0.00000 + 0.16667 j
            -0.00000 + -0.06667 j
U-controlled-1 =
            0.32998 + 0.00000 j
            -0.32998 + -0.00000 j
            0.32998 + -0.00000 j
            -0.32998 + -0.00000 j
            0.32998 + -0.00000 j
            -0.32998 + 0.00000 j
            0.32998 + 0.00000 j
            -0.32998 + -0.00000 j
            0.06667 + 0.00000 j
            -0.16667 + -0.00000 j
            0.06667 + 0.00000 j
            -0.16667 + -0.00000 j
            0.06667 + 0.00000 j
            -0.16667 + -0.00000 j
            0.06667 + -0.00000 j
            -0.16667 + 0.00000 j
Final =
            0.65997 + -0.00000 j
            -0.65997 + 0.00000 j
            0.00000 + 0.00000 j
            -0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            0.00000 + 0.00000 j
            0.00000 + -0.00000 j
            0.13333 + -0.00000 j
            -0.33333 + -0.00000 j
            0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            0.00000 + -0.00000 j
            -0.00000 + 0.00000 j
```

Sistema 4x4

```
-------------------------------ARQUIVO DOS PARAMETROS---------------------------------
------

------------------------------Parametros de entrada-----------------------------------
--

B =
           12.00000 + 0.00000 j  -2.00000 + 0.00000 j  0.00000 + 0.00000 j  0.00000 +
0.00000 j
           -2.00000 + 0.00000 j  12.00000 + 0.00000 j  -6.00000 + 0.00000 j  0.00000 +
0.00000 j
           0.00000 + 0.00000 j  -6.00000 + 0.00000 j  12.00000 + 0.00000 j  -2.00000 +
0.00000 j
           0.00000 + 0.00000 j  0.00000 + 0.00000 j  -2.00000 + 0.00000 j  12.00000 +
0.00000 j
p =
           0.70000 + 0.00000 j
           -0.50000 + 0.00000 j
           -0.50000 + 0.00000 j
           -0.10000 + 0.00000 j

NQmax = 9
DPtol = 10.000%

---------------------------Prepara軸o dos parametros do circuito-----------------------
-----

Sistema quantico:

B =
           12.00000 + 0.00000 j  -2.00000 + 0.00000 j  0.00000 + 0.00000 j  0.00000 +
0.00000 j
           -2.00000 + 0.00000 j  12.00000 + 0.00000 j  -6.00000 + 0.00000 j  0.00000 +
0.00000 j
           0.00000 + 0.00000 j  -6.00000 + 0.00000 j  12.00000 + 0.00000 j  -2.00000 +
0.00000 j
           0.00000 + 0.00000 j  0.00000 + 0.00000 j  -2.00000 + 0.00000 j  12.00000 +
0.00000 j
p =
           0.70000 + 0.00000 j
           -0.50000 + 0.00000 j
           -0.50000 + 0.00000 j
           -0.10000 + 0.00000 j
Autovalores =
           5.39445 + 0.00000 j
           11.39445 + 0.00000 j
           12.60555 + 0.00000 j
           18.60555 + 0.00000 j
s = 3.449
nq = 2
nb = 2
```

```
5 q-bits
3 bits classicos

nr = 3

Autovalores unicos =
           5.39445 + 0.00000 j
           12.00000 + 0.00000 j
           18.60555 + 0.00000 j
Estados:
01 <---> 1
10 <---> 2
11 <---> 3

DP% = 6.383%
t = 0.2674
C = 5.3944

-------------------------Matrizes do algoritmo-----------------------------------

Transformacao Hadamard:

h([0, 1])

Matrizes hamiltonianas:

Q-bit 0 <---> 1.0000 e^iBt

U =
           -0.88579 + -0.05909 j  -0.02109 + 0.31609 j  0.32679 + 0.02180 j  0.00434 +
-0.06513 j
           -0.02109 + 0.31609 j  0.09459 + 0.00631 j  -0.05891 + 0.88313 j  0.32679 +
0.02180 j
           0.32679 + 0.02180 j  -0.05891 + 0.88313 j  0.09459 + 0.00631 j  -0.02109 +
0.31609 j
           0.00434 + -0.06513 j  0.32679 + 0.02180 j  -0.02109 + 0.31609 j  -0.88579 +
-0.05909 j

Q-bit 1 <---> 2.0000 e^iBt

U =
           0.78376 + 0.10504 j  -0.00230 + 0.01715 j  -0.51482 + -0.06900 j  -0.04296 +
0.32053 j
           -0.00230 + 0.01715 j  -0.76069 + -0.10195 j  -0.04985 + 0.37199 j  -0.51482
+ -0.06900 j
           -0.51482 + -0.06900 j  -0.04985 + 0.37199 j  -0.76069 + -0.10195 j  -0.00230
+ 0.01715 j
           -0.04296 + 0.32053 j  -0.51482 + -0.06900 j  -0.00230 + 0.01715 j  0.78376 +
0.10504 j

Transformada de Fourier Quantica inversa:
```

```
SWAP(0, 1)
h(0)
p(-0.5000 pi)(1, 0)
h(1)

Rotacao controlada:

Posi■o 1
Controle 0
Rotacao Ry(3.14159)

Ry =
           0.00000 + 0.00000 j  -1.00000 + 0.00000 j
           1.00000 + 0.00000 j  0.00000 + 0.00000 j

Posi■o 0
Controle 0
Rotacao Ry(0.34417)

Ry =
           0.98523 + 0.00000 j  -0.17124 + 0.00000 j
           0.17124 + 0.00000 j  0.98523 + 0.00000 j

Posi■o 1
Controle 1
Rotacao Ry(0.58832)

Ry =
           0.95705 + 0.00000 j  -0.28994 + 0.00000 j
           0.28994 + 0.00000 j  0.95705 + 0.00000 j

Transformada de Fourier Quantica:

h(1)
p(0.5000 pi)(1, 0)
h(0)
SWAP(0, 1)

Matrizes hamiltonianas inversas:

Q-bit 1 <---> 2.0000 e^iBt

U =
           0.78376 + -0.10504 j  -0.00230 + -0.01715 j  -0.51482 + 0.06900 j  -0.04296
+ -0.32053 j
           -0.00230 + -0.01715 j  -0.76069 + 0.10195 j  -0.04985 + -0.37199 j  -0.51482
+ 0.06900 j
           -0.51482 + 0.06900 j  -0.04985 + -0.37199 j  -0.76069 + 0.10195 j  -0.00230
+ -0.01715 j
           -0.04296 + -0.32053 j  -0.51482 + 0.06900 j  -0.00230 + -0.01715 j  0.78376
+ -0.10504 j
```

```
Q-bit 0 <---> 1.0000 e^iBt

U =
          -0.88579 + 0.05909 j  -0.02109 + -0.31609 j  0.32679 + -0.02180 j  0.00434 +
0.06513 j
          -0.02109 + -0.31609 j  0.09459 + -0.00631 j  -0.05891 + -0.88313 j  0.32679
+ -0.02180 j
          0.32679 + -0.02180 j  -0.05891 + -0.88313 j  0.09459 + -0.00631 j  -0.02109
+ -0.31609 j
          0.00434 + 0.06513 j  0.32679 + -0.02180 j  -0.02109 + -0.31609 j  -0.88579 +
0.05909 j

Transformacao Hadamard:

h([0, 1])
```

```
--------------------------------ARQUIVO DOS RESULTADOS-------------------
--------------------

----------------------Solucao---------------------------

Classica =
           0.04573 + 0.00000 j
           -0.07564 + 0.00000 j
           -0.08319 + 0.00000 j
           -0.02220 + 0.00000 j
Quantica =
           0.04736 + -0.00000 j
           -0.07729 + -0.00000 j
           -0.08215 + -0.00000 j
           -0.01831 + 0.00000 j
Erro teorico = 6.124%
Fidelidade = 0.99860

----------------------Histograma--------------------------

p(ancilla = 1) = 44.526%

000 <---> 0.5060
001 <---> 0.0126
010 <---> 0.0123
011 <---> 0.0000
100 <---> 0.0653
101 <---> 0.1738
110 <---> 0.1964
111 <---> 0.0098

--------------------Vetores de estado--------------------------

Estado p =
           0.70000 + 0.00000 j
           -0.50000 + 0.00000 j
           -0.50000 + 0.00000 j
           -0.10000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
           0.00000 + 0.00000 j
           -0.00000 + 0.00000 j
```

```
            -0.00000 + 0.00000 j
            -0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            -0.00000 + 0.00000 j
            -0.00000 + 0.00000 j
            -0.00000 + 0.00000 j
Hadamard =
            0.35000 + 0.00000 j
            -0.25000 + 0.00000 j
            -0.25000 + 0.00000 j
            -0.05000 + 0.00000 j
            0.35000 + 0.00000 j
            -0.25000 + 0.00000 j
            -0.25000 + 0.00000 j
            -0.05000 + 0.00000 j
            0.35000 + 0.00000 j
            -0.25000 + 0.00000 j
            -0.25000 + 0.00000 j
            -0.05000 + 0.00000 j
            0.35000 + 0.00000 j
            -0.25000 + 0.00000 j
            -0.25000 + 0.00000 j
            -0.05000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
U-controlled =
            0.35000 + 0.00000 j
            -0.25000 + 0.00000 j
            -0.25000 + 0.00000 j
            -0.05000 + 0.00000 j
            -0.38667 + -0.10190 j
            -0.03264 + -0.11282 j
            0.10651 + -0.23053 j
            -0.03062 + -0.10431 j
            0.40574 + 0.03370 j
            0.22757 + -0.05806 j
            0.02256 + -0.09252 j
            0.07506 + 0.11990 j
            -0.32633 + -0.01478 j
            0.10497 + 0.18968 j
            0.13296 + 0.23684 j
            0.04896 + -0.14185 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
```

```
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
QTF-1 =
            0.02137 + -0.04149 j
            0.02495 + 0.00940 j
            0.00602 + -0.04311 j
            0.02170 + -0.06313 j
            -0.07143 + 0.01332 j
            -0.39004 + 0.09784 j
            -0.36997 + 0.05948 j
            -0.04376 + -0.02016 j
            0.73437 + 0.07519 j
            -0.04738 + -0.06746 j
            -0.23346 + -0.04941 j
            0.00336 + 0.18303 j
            0.01569 + -0.04702 j
            -0.08754 + -0.03978 j
            0.09741 + 0.03303 j
            -0.08129 + -0.09974 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
            0.00000 + 0.00000 j
Rotacao =
            -0.00366 + 0.00710 j
            -0.00427 + -0.00161 j
            -0.00103 + 0.00738 j
            -0.00372 + 0.01081 j
            -0.00000 + 0.00000 j
            -0.00000 + 0.00000 j
            -0.00000 + 0.00000 j
            -0.00000 + -0.00000 j
            0.65599 + 0.06716 j
            -0.04232 + -0.06026 j
            -0.20854 + -0.04414 j
            0.00300 + 0.16349 j
            0.01501 + -0.04500 j
            -0.08378 + -0.03807 j
```

```
            0.09322 + 0.03161 j
            -0.07780 + -0.09545 j
            0.02105 + -0.04088 j
            0.02459 + 0.00926 j
            0.00593 + -0.04247 j
            0.02138 + -0.06220 j
            -0.07143 + 0.01332 j
            -0.39004 + 0.09784 j
            -0.36997 + 0.05948 j
            -0.04376 + -0.02016 j
            0.33013 + 0.03380 j
            -0.02130 + -0.03033 j
            -0.10495 + -0.02221 j
            0.00151 + 0.08228 j
            0.00455 + -0.01363 j
            -0.02538 + -0.01153 j
            0.02824 + 0.00958 j
            -0.02357 + -0.02892 j
QTF =
            0.33367 + 0.01463 j
            -0.06519 + -0.04997 j
            -0.05817 + -0.00257 j
            -0.03926 + 0.03942 j
            -0.35232 + -0.03754 j
            -0.00001 + 0.07121 j
            0.11956 + -0.02085 j
            -0.05108 + -0.03744 j
            0.31866 + 0.05963 j
            0.01859 + -0.01190 j
            -0.15139 + -0.03419 j
            0.03854 + 0.13488 j
            -0.30732 + -0.02252 j
            0.03806 + -0.01256 j
            0.08795 + 0.07237 j
            0.04437 + -0.11524 j
            0.14215 + -0.00369 j
            -0.20607 + 0.03262 j
            -0.22037 + 0.00219 j
            -0.02222 + -0.01450 j
            -0.16801 + -0.07533 j
            -0.03174 + -0.16253 j
            0.03049 + -0.20923 j
            0.00556 + -0.08233 j
            0.20903 + -0.00338 j
            0.20935 + -0.05368 j
            0.12136 + -0.06687 j
            0.04511 + 0.03458 j
            -0.14106 + 0.00065 j
            0.07763 + 0.20212 j
            0.08039 + 0.18898 j
            0.01431 + -0.06214 j
U-controlled-1 =
            0.33367 + 0.01463 j
            -0.06519 + -0.04997 j
            -0.05817 + -0.00257 j
            -0.03926 + 0.03942 j
            0.37764 + -0.00198 j
            -0.04695 + 0.00341 j
            -0.05264 + 0.00544 j
            0.04081 + -0.00704 j
```

```
            0.37764 + 0.00198 j
            -0.04695 + -0.00341 j
            -0.05264 + -0.00544 j
            0.04081 + 0.00704 j
            0.33367 + -0.01463 j
            -0.06519 + 0.04997 j
            -0.05817 + 0.00257 j
            -0.03926 + -0.03942 j
            0.14215 + -0.00369 j
            -0.20607 + 0.03262 j
            -0.22037 + 0.00219 j
            -0.02222 + -0.01450 j
            0.11336 + 0.00122 j
            -0.21085 + -0.00210 j
            -0.22279 + -0.00335 j
            -0.07658 + 0.00434 j
            0.11336 + -0.00122 j
            -0.21085 + 0.00210 j
            -0.22279 + 0.00335 j
            -0.07658 + -0.00434 j
            0.14215 + 0.00369 j
            -0.20607 + -0.03262 j
            -0.22037 + -0.00219 j
            -0.02222 + 0.01450 j
Final =
            0.71132 + -0.00000 j
            -0.11214 + -0.00000 j
            -0.11082 + -0.00000 j
            0.00155 + 0.00000 j
            0.00000 + 0.01661 j
            0.00000 + -0.05338 j
            0.00000 + -0.00801 j
            0.00000 + 0.04647 j
            0.00000 + 0.01265 j
            0.00000 + -0.04656 j
            -0.00000 + 0.00287 j
            0.00000 + 0.03238 j
            -0.04397 + -0.00000 j
            -0.01823 + -0.00000 j
            -0.00553 + -0.00000 j
            -0.08007 + -0.00000 j
            0.25551 + -0.00000 j
            -0.41692 + -0.00000 j
            -0.44317 + -0.00000 j
            -0.09880 + 0.00000 j
            0.00000 + -0.00491 j
            0.00000 + 0.03472 j
            -0.00000 + 0.00554 j
            0.00000 + -0.01884 j
            0.00000 + -0.00247 j
            0.00000 + 0.03052 j
            0.00000 + -0.00116 j
            0.00000 + -0.01016 j
            0.02879 + 0.00000 j
            0.00479 + -0.00000 j
            0.00242 + -0.00000 j
            0.05435 + 0.00000 j
```