Simulasi Algoritma Kuantum Grover Menggunakan Ruang Vektor Kompleks dan Transformasi Matriks Unitary

Edward David Rumahorbo - 13524036
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13524036@mahasiswa.itb.ac.id, edwardrumahorbo1@gmail.com

***Ringkasan*—Makalah ini akan menjabarkan proses simulasi untuk Algoritma Pencarian Grover, sebuah algoritma kuantum yang memberikan percepatan kuadratik $O(\sqrt{N})$ untuk masalah pencarian tak terurut. Secara spesifik, akan dibuktikan juga bahwa Operator Oracle dan Diffuser merupakan manifestasi dari Refleksi Householder, yang secara geometris merotasi vektor keadaan dalam subruang dua dimensi.**

***Kata Kunci*—Algoritma Grover, Aljabar Linear, Ruang Vektor Kompleks, Refleksi Householder, Produk Tensor.**

I. Pendahuluan

Bayangkan jika anda memiliki sebuah kumpulan bilangan acak dan sebuah fungsi yang akan mengembalikan nilai *true* hanya untuk satu bilangan spesifik dari kumpulan tersebut. Tantangannya adalah anda tidak boleh melihat isi ataupun kode dari fungsi tersebut. Dengan menggunakan komputer klasik, pendekatan terbaik yang bisa anda lakukan adalah mengecek bilangan satu-persatu. Untuk N elemen, anda perlu memeriksa rata-rata $N/2$ kali, sehingga kokmkpleksitas algoritma yang dihasilkan adalah $O(N)$. Untuk nilai N yang sangat besar, tentunya kecepatan komputasi akan menjadi hal yang sangat krusial dan membuat pendekatan ini menjadi tidak efisien.

Jenis permasalahan seperti ini dikategorikan dalam kelas malasah di mana solusi eksaknya sulit ditemukan, tetapi kebenaran dari solusi tersebut sangat mudah divalidasi, atau sering disebut sebagai *NP Problem*. Lov Grover memperkenalkan sebuah algoritma kuantum yang dapat menyelesaikan masalah pencarian pada basis data tak terurut ini dengan kompleksitas $O(\sqrt{N})$. Peningkatan kecepatan ini sangat signifikan. Sebagai contoh, untuk mencari sebuah item dalam satu juta data, komputer klasik membutuhkan rata-rata membutuhkan 500.000 langkah, sedangkan dengan Algoritma Grover, hanya dibutuhkan sekitar 1.000 langkah.

Namun, memahami bagaimana kenaikan kecepatan ini terjadi seringkali terhalang oleh notasi fisika kuantum yang tidak umum. Padahal, jika ditelusuri lebih dalam, Algoritma Grover sebenarnya adalah aplikasi dari Aljabar Linear dan Geometri. Sistem kuantum tidak lain adalah vektor di dalam Ruang Vektor Umum (khususnya ruang vektor kompleks), dan operasi komputasi yang terjadi adalah Transformasi Linear yang dilakukan dengan matriks. Algoritma ini tidak bekerja dengan menebak secara lebih cepat, tetapi dengan memanipulasi *state vector* melalui rotasi dan pencerminan terhadap vektor tersebut agar mendekati vektor solusi yang diinginkan.

Malakah ini bertujuan untuk membedah Algoritma Grover menggunakan pendekatan Aljabar Linear seperti matriks, vektor, *inner product*, dan *unitary tranformation*, serta imlementasi dari algoritma ini menggunakan bahasa pemrograman Python sebagai visualisasi mengenai bagaimana operasi matriks dapat memanipulasi probabilitas, memberikan bukti konkret atas teori yang dibahas.

Gambar 1. Komputer Kuantum

II. Dasar Teori

A. Vektor dan Ruang Vektor $\mathbb{C}^n$

Pada aljabar linear, sebuah vektor v dalam ruang dimensi n dapat direpresentasikan sebagai sebuah *array* kolom berukuran $n \times 1$. Jika pada kasus klasik vektor berada di ruang bilangan riil $\mathbb{R}^n$, pada komputasi kuantum, vektor akan bekerja di ruang bilangan kompleks $\mathbb{C}^n$.

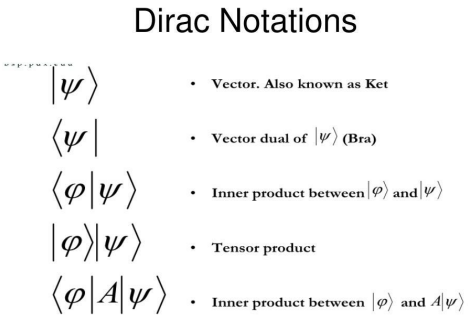
Sebuah vektor $v \in \mathbb{C}^n$ dapat ditulis sebagai

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}, \quad \text{di mana } v_i \in \mathbb{C}$$

Dua operasi fundamental pada ruang vektor ini adalah

1) Penjumlahan vektor $(u + v)$ dengan melakukan operasi elemen per elemen
2) Perkalian skalar αv, di mana α adalah bilangan kompleks

Dalam notasi fisika (notasi Dirac), vektor kolom ini ditulis sebagai *ket vector* $|v\rangle$. Hubungan ini penting untuk menerjemahkan literatur kuantum ke dalam operasi matriks standar.



Gambar 2. Notasi Dirac

B. Matriks dan Transformasi Linear

Dalam konteks ini, Matriks adalah susunan bilangan yang merepresentasikan sebuah Transformasi Linear antar ruang vektor. Jika A adalah matriks berukuran $m \times n$, maka perkalian matriks dengan vektor u, atau bisa dinotasikan sebagai $v = Au$, akan memetakan vektor u dari ruang $\mathbb{C}^n$ ke vektor v di ruang $\mathbb{C}^m$.

Sifat linearitas ini didefinisikan sebagai

$$A(\alpha u + \beta w) = \alpha(Au) + \beta(Aw)$$

Dalam algoritma Grover, semua gerbang kuantum (operasi gerbang logika dalam komputasi kuantum) adalah matriks persegi $N \times N$ yang mentransformasi *state vector* sistem. Matriks ini bertindak sebagai operator yang merotasi atau memetakan vektor ke posisi baru dalam ruang vektor.

C. Hasil Kali Dalam (Inner Product) dan Norma

Untuk mengukur panjang suatu vektor dan sudut di antara 2 buah vektor, akan digunakan perhitungan Hermitian Inner Product. Untuk dua vektor u dan v, hasil Hermitian Inner Product-nya didefinisikan sebagai

$$\langle u, v\rangle = u^\dagger v = \begin{bmatrix} u_1^* & u_2^* & \dots & u_n^* \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \sum_{i=1}^{n} u_i^* v_i$$

di mana $u^\dagger$, atau dibaca u dagger, merupakan transpos konjugat dari vektor u. Vektor u awalnya akan diubah terlebih dahulu menjadi matriks baris, kemudian setiap elemen kompleksnya di konjugat.

Norma sebuah vektor didefinisikan sebagai akar kuadrat dari hasil inner product sebuah vektor dengan dirinya sendiri. Norma bisa dinotasikan sebagai

$$\|v\| = \sqrt{\langle v, v\rangle} = \sqrt{\sum_{i=1}^{n} |v_i|^2}$$

Dalam konteks kuantum, setiap *state vector* harus merupakan sebuah vektor satuan, yang berarti norma dari vektor tersebut hasus sama dengan 1. Ini berarti, probabilitas total dari semua kemungkinan keadaan harus selalu bernilai 1.

D. Hasil Kali Luar (Outer Product) dan Proyeksi

Selain *inner product*, *outer product* juga memegang peran penting dalam algoritma ini. Jika $|u\rangle$ adalah vektor kolom berukuran $n \times 1$, maka hasil *outer product* dapat didefinisikan sebagai

$$P = |u\rangle\langle u| = uu^\dagger$$

Hasil dari *outer product* adalah sebuah matriks persegi berukuran $n \times n$. Jika $|u\rangle$ adalah sebuah vektor satuan, maka matriks P adalah sebuah Matriks Proyeksi Ortogonal. Ini penting karena operasi pada algoritma Grover akan memproyeksikan *state vektor* ke arah tertentu.

E. Matriks Unitary dan Nilai Eigen

Sebuah matriks persegi U dikatakan unitary jika invers dari matriks tersebut adalah transpos konjugat nya, atau bisa dinotasikan sebagai

$$U^{-1} = U^\dagger$$

$$U^\dagger U = I$$

Sifat penting dari sebuah matriks unitary adalah matriks ini memiliki nilai eigen dengan modulus bernilai 1, sehingga matriks ini akan mempertahankan norma vektornya. Jika sebuah vektor v dikalikan dengan matrik unitary U, panjang vektor v tidak akan berubah ($\|Uv\| = \|v\|$). Sifat ini akan menjamin bahwa selama proses komputasi kuantum terjadi, total probabilitas yang ada di dalam sistem akan selalu berjumlah 1. Dalam algoritma Grover, matriks akan direkayasa sehingga memiliki nilai eigen tertentu:

- Vektor solusi $|\omega\rangle$ adalah vektor eigen dari Matriks Oracle U_ω yang memiliki nilai eigen -1
- Vektor lain adalah vektor eigen dari Matriks Diffuser U_s yang memiliki nilai eigen 1

F. Refleksi Householder

Dalam aljabar linear, matriks Householder digunakan untuk mencerminkan sebuah vektor terhadap sebuah bidang yang tegak lurus terhadap vektor satuan u. Rumus umum dari matriks ini dapat dinotasikan sebagai

$$H_u = I - 2|u\rangle\langle u|$$

Matrik ini akan membalikkan tanda komponen vektor yang sejajar dengan u, dan tidak akan mengubah sama sekali komponen yang tegak lurus terhadap vektor u. Matriks Householder adalah matriks Unitary dan Hermitian. Pada bagian analisis matematis, akan ditunjukkan bahwa]*Oracle* dan *Difusser* hanyalah bentuk khusus dari refleksi Householder.

G. Produk Tensor (Tensor Product) dan Entanglement

untuk menggabungkan dua buah vektor yang terpisah (misalnya 2 qubit), operasi yang dilakukan adalah *tensor product*. Jika A adalah matriks $m \times n$ dan B adalah matriks $p \times q$, maka $A \otimes B$ akan menghasilkan sebuah matriks berukuran $mp \times nq$. Untuk $m = p = 2$ dan $n = q = 1$, hasil produk tensor $A \otimes B$ dapat dinotasikan sebagai

$$\begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} a \begin{bmatrix} c \\ d \end{bmatrix} \\ b \begin{bmatrix} c \\ d \end{bmatrix} \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}$$

Untuk vektor kompleks, pertumbuhan ukuran ruang vektor bisa digambarkan seperti berikut:

$$\mathcal{H} = \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2 \cong \mathbb{C}^{2^n}$$

Operasi ini akan menyebabkan matriks di ruang vektor kuantum bertumbuh secara eksponensial seiring bertambahnya jumlah qubit (2^n), dan inilah yang menjadi tantangan utama dalam simulasi klasik. Keunikan yang muncul pada proses ini adalah *entanglement*, di mana *state vector* gabungan tidak bisa difaktorkan kembali menjadi vektor-vektor individu.

III. Analisis Matematis Algoritma Grover

Algoritma Grover bekerja dengan melakukan rotasi pada vektor awal $|s\rangle$ hingga mendekati vektor solusi $|\omega\rangle$ melalui dua operasi matriks utama.

A. Menginisialisasi Superposisi Seragam

Pada mulanya, sistem akan disiapkan dalam keadaan superposisi seragam dengan menggunakan gerbang Hadamard. Vektor ini merupakan kombinasi linear dari semua basis dengan bobot yang sama.

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$$

B. Konstruksi Oracle (U_ω) dan Diffuser (U_s)

Oracle adalah sebuah fungsi untuk menandai target atau solusi $|\omega\rangle$. Secara matematis Oracle didefinisikan sebagai matriks yang membalik fase vektor solusi $|\omega\rangle$ dan membiarkan vektor ortogonalnya tetap.

$$U_\omega|\omega\rangle = -|\omega\rangle$$

$$U_\omega|x\rangle = |x\rangle \quad \text{untuk } x \neq \omega$$

Dalam bentuk matriks, Oracle adalah refleksi Householder terhadap vektor solusi.

$$U_\omega = I - 2|\omega\rangle\langle\omega|$$

Ini adalah matriks diagonal dengan entri -1 pada indeks solusi dan 1 di 1 di tempat lain

Diffuser adalah refleksi Householder terhadap vektor superposisi awal $|s\rangle$. Diffuser bertujuan untuk melakukan ”inversi terhadap rata-rata”

$$U_s = 2|s\rangle\langle s| - I = -(I - 2|s\rangle\langle s|)$$

Matriks $|s\rangle\langle s|$ adalah *outer product* yang setiap elemennya bernilai $1/N$. Matriks Diffuser padat dengan setiap elemen diagonal bernilai $2/N - 1$ dan elemen non-diagonal bernilai $2/N$.

C. Rotasi Bidang sebagai Interpretasi Geometris

Dengan meninjau bahwa bidang yang direntang oleh vektor solusi $|\omega\rangle$ dan vektor superposisi elemen selain solusi $|s'\rangle$, vektor superposisi awal $|s\rangle$ dapat ditulis sebagai

$$|s\rangle = \sin(\theta/2)|\omega\rangle + \cos(\theta/2)|s'\rangle$$

di mana $\sin(\theta/2) = 1/\sqrt{N}$. Langkah iterasi Grover adalah $G = U_s U_\omega$. Karena kita tau bahwa U_s dan U_ω adalah sebuah matriks refleksi, maka bisa dikatakan bahwa komposisi G adalah sebuah rotasi. Dalam geometri bidang, dua refleksi terhadap garis yang membentuk sudut α akan ekuivalen dengan rotasi sebesar 2α. Pada keadaan ini, sudut antara garis ortogonal $|\omega\rangle$ dan garis $|s\rangle$ adalah sebesar $\pi/2 - \theta/2$. Akibatnya, operasi Grover akan merotasi *state vektor* sebesar sudut θ mendekati $|\omega\rangle$. Dalam k iterasi, vektor akan berubah menjadi

$$G^k|s\rangle = \sin\left((2k+1)\frac{\theta}{2}\right)|\omega\rangle + \cos\left((2k+1)\frac{\theta}{2}\right)|s'\rangle$$

dan probabilitas sukses adalah kuadrat amplitudo komponen $|\omega\rangle$:

$$P(k) = \sin^2\left((2k+1)\frac{\theta}{2}\right)$$

IV. Metodologi Simulasi

Simulasi pada makalah ini akan dilakukan dengan menggunakan komputer klasik dan menggunakan pendekatan Aljabar Linear. Representasi data akan dilakukan dengan menggunkaan array NumPy complex128 (*double precision*) dengan pembagian 64-bit float untuk bagian riil dan 64-bit float untuk bagian imajiner sebagai elemen untuk *state vector* dan matriks

operator. Representasi data tersebut akan menyelesaikan permasalahan stabilitas numerik terkait *floating point*, di mana akumulasi dari galat pada operasi matriks yang iteratif bisa berujung pada jumlah galat yang signifikan. Salah satu contoh dari kasus ini adalah hilangnya sifat unitaritas dari matriks U, di mana $U^\dagger U \not\approx I$. Jika galat yang terus terakumulasi, norma *state vektor* dapat menyimpang dari 1. Selain permasalahan stabilitas numerik, ukuran matriks yang bertumbuh sebesar 2^{2n} bisa menyebabkan *memory overflow* pada RAM komputer standar, sehingga simulasi akan dibatasi hanya menggunakan $n \leq 12$ qubit.

V. Implementasi dan Eksperimen

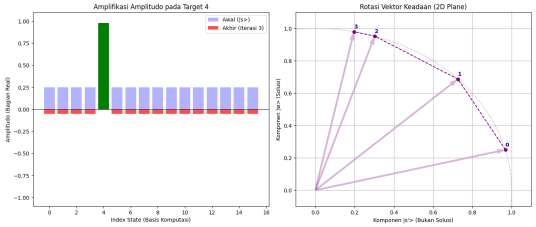
Pengimplementasian simulasi dilakukan dalam Python untuk memvisualisasikan rotasi vektor.

A. Kode Simulasi

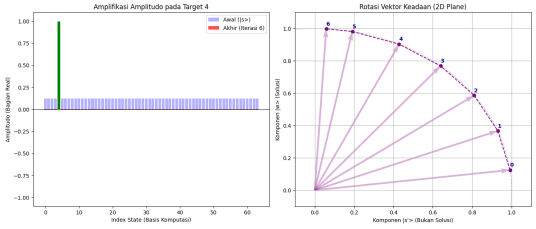
Berikut adalah kode Python dari *function* yang digunakan untuk simulasi algoritma Grover. Beberapa library yang digunakan untuk membantu implementasi simulasi algoritma Grover adalah numpy dan matplotlib (untuk mempersingkat kode, digunakan singkatan np untuk numpy dan plt untuk matplotlib). Kode lengkap untuk simulasi ini bisa dilihat melalui laman github https://github.com/bangedaw/makalah-algeo.git atau pada bagian lampiran.

B. Hasil Simulasi

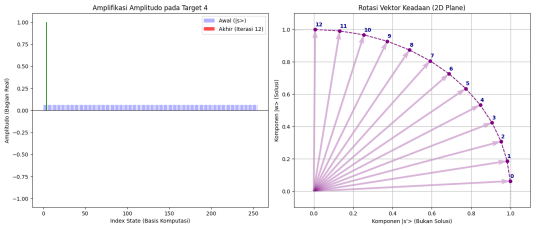
Berikut adalah hasil simulasi yang dilakukan dengan menggunakan beberapa nilai n yang berbeda



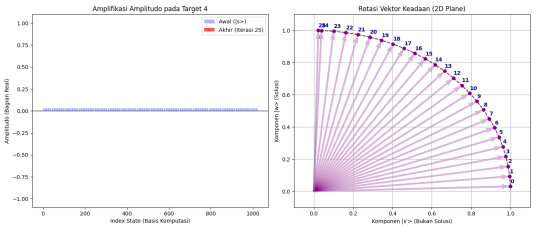
Gambar 3. Hasil simulasi untuk $n = 4$



Gambar 4. Hasil simulasi untuk $n = 6$



Gambar 5. Hasil simulasi untuk $n = 8$



Gambar 6. Hasil simulasi untuk $n = 10$

C. Analisis Hasil dan Perbandingan

Hasil simulasi dengan menggunakan nilai $n = 10$ menunjukkan beberapa perilaku yang sangat menarik, di mana

- Probabilitas meningkat secara monoton hingga melewati $k = 25$ hingga mendekati $|\omega\rangle$. Ini sangat dekat dengan prediksi teoritis $k \approx \frac{\pi}{4}\sqrt{1024} \approx 25.12$.
- Jika iterasi terus dilanjutkan hingga melewati $k = 25$, maka akan terjadi penurunan probabilitas kembali menuju ke nol. Ini membuktikan bahwa algoritma Grover bukanlah proses yang konvergen asimtotik, melainkan proses rotasi siklik. Secara geometris, *state vector* telah melewati vektor target $|\omega\rangle$, dan ini berimplikasi pada keharustauan penguji tentang kapan harus berhenti melakukan iterasi.

VI. Kesimpulan

Melalui makalah ini, algoritma Grover telah berhasil disimulasikan menggunakan komputer klasik dengna pendekatan Aljabar Linear. Kesimpulan utama dari makalah ini adalah:

- Algooritma Grover bekerja berdasarkan prinsip rotasi vektor dalam ruang hasil kali dalam kompleks, yang digerakkan oleh dua refleksi Householder.
- Simulasi memvalidasi bahwa jumlah langkah yang diperlukan sebanding dengan $\sqrt{N}$, dibuktikan oleh periode fungsi sinus probabilitas.
- Simulasi klasik menghadapi batasan memori eksponensial akibat sifat produk tensor. Selain itu, stabilitas numerik menjadi faktor kritis; simulasi jangka panjang memerlukan presisi aritmatika tinggi untuk mencegah loss of unitarity.

Lampiran

```
1    def visualize_grover_process(n_qubits,
         target_index):
2    N = 2**n_qubits
3
4    H1 = np.array([[1, 1], [1, -1]]) / np.sqrt(2)
```

```
    H_n = H1
    for _ in range(n_qubits - 1):
        H_n = np.kron(H_n, H1)

    # Inisialisasi State |0...0>
    state = np.zeros(N, dtype=np.complex128)
    state[0] = 1.0

    # Buat Superposisi Awal |s>
    state = H_n @ state
    s_vec = state.copy().reshape(-1, 1)

    # Oracle Matrix: I - 2|w><w| (Matriks Diagonal)
    Oracle = np.eye(N, dtype=np.complex128)
    Oracle[target_index, target_index] = -1

    # Diffuser Matrix: 2|s><s| - I (Refleksi Householder)
    Diffuser = 2 * (s_vec @ s_vec.conj().T) - np.eye(N)

    w_vec = np.zeros(N)
    w_vec[target_index] = 1.0

    # Vektor basis |s'> (Gram-Schmidt)
    s_prime = s_vec.flatten() - (np.vdot(w_vec, s_vec.flatten()) * w_vec)
    norm_s_prime = np.linalg.norm(s_prime)

    if norm_s_prime > 1e-10:
        s_prime = s_prime / norm_s_prime

    # Fungsi koordinat
    def get_coords(vec):
        y = np.abs(np.vdot(w_vec, vec))       # Proyeksi ke Target
        x = np.abs(np.vdot(s_prime, vec))     # Proyeksi ke Non-Target
        return x, y

    optimal_iter = int(np.pi / 4 * np.sqrt(N))
    print(f"Ruang Pencarian N={N}, Target Indeks={target_index}")
    print(f"Iterasi Optimal Teoritis: {optimal_iter}")

    amplitudes_history = []
    coords_history = []

    # Simpan kondisi awal
    amplitudes_history.append(state.real.copy())
    coords_history.append(get_coords(state))

    for i in range(optimal_iter):
        # a. Terapkan Oracle
        state = Oracle @ state

        # b. Terapkan Diffuser
        state = Diffuser @ state

        # Simpan data
        amplitudes_history.append(state.real.copy())
        coords_history.append(get_coords(state))

    fig = plt.figure(figsize=(14, 6))

    # Plot A: Evolusi Amplitudo
    ax1 = fig.add_subplot(1, 2, 1)

    indices = np.arange(N)
    ax1.bar(indices, amplitudes_history[0], color='blue', alpha=0.3, label='Awal (|s>)')
    ax1.bar(indices, amplitudes_history[-1], color='red', alpha=0.7, label=f'Akhir (Iterasi {optimal_iter})')
    ax1.bar([target_index], [amplitudes_history[-1][target_index]], color='green')

    ax1.set_xlabel('Index State (Basis Komputasi)')
    ax1.set_ylabel('Amplitudo (Bagian Real)')
    ax1.set_title(f'Amplifikasi Amplitudo pada Target {target_index}')
    ax1.set_ylim(-1.1, 1.1)
    ax1.axhline(0, color='black', linewidth=0.8)
    ax1.legend()

    # Plot B: Rotasi Geometris
    ax2 = fig.add_subplot(1, 2, 2)

    xs = [c[0] for c in coords_history]
    ys = [c[1] for c in coords_history]

    # Plot jejak rotasi
    ax2.plot(xs, ys, 'o--', color='purple', label='Lintasan Vektor')

    # Gambar panah
    for i in range(len(xs)):
        ax2.quiver(0, 0, xs[i], ys[i], angles='xy', scale_units='xy', scale=1, color='purple', alpha=0.3)
        ax2.text(xs[i], ys[i] + 0.02, f'{i}', fontsize=10, fontweight='bold', color='darkblue')

    ax2.set_xlim(-0.1, 1.1)
    ax2.set_ylim(-0.1, 1.1)
    ax2.set_xlabel("Komponen |s'> (Bukan Solusi)")
    ax2.set_ylabel("Komponen |w> (Solusi)")
    ax2.set_title("Rotasi Vektor Keadaan (2D Plane)")
    ax2.grid(True)

    circle = plt.Circle((0, 0), 1, color='gray', fill=False, linestyle=':')
    ax2.add_artist(circle)

    plt.tight_layout()
    plt.show()
```

PUSTAKA

[1] Caroline Figgatt **andothers**. **?**Complete 3-qubit Grover search on a programmable quantum computer**?** **in***Nature Communications*: 8.1 (2017), **page** 1918.

[2] Lov K Grover. **?**A fast quantum mechanical algorithm for database search**?** **in***Proceedings of the 28th Annual ACM Symposium on Theory of Computing*: ACM. 1996, **pages** 212–219.

[3] Charles R. Harris **andothers**. **?**Array programming with NumPy**?** **in***Nature*: 585.7825 (2020), **pages** 357–362. DOI: 10.1038/s41586-020-2649-2.

[4] Microsoft Azure Quantum. *Grover's search algorithm*. https://learn.microsoft.com/en-us/azure/quantum/concepts-grovers. Diakses pada 24 Desember 2025. 2024.

[5] Rinaldi Munir. *Bahan Kuliah IF2123 Aljabar Linier dan Geometri*. Program Studi Teknik Informatika, Institut Teknologi Bandung. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/. 2025.

[6] Marcel Niedermeier, Jose L Lado **and** Christian Flindt. **?**Simulating the quantum Fourier transform, Grover's algorithm, and the quantum counting algorithm with limited entanglement using tensor networks**? in***Physical Review Research*: 6.3 (2024), **page** 033325.

[7] Michael A Nielsen **and** Isaac L Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge: Cambridge University Press, 2010.

[8] Tim Dosen Algeo. *Tugas Makalah IF2123 Aljabar Linier dan Geometri*. Program Studi Teknik Informatika, Institut Teknologi Bandung. Dokumen Spesifikasi Tugas. 2025.

[9] Z. Zhao **andothers**. **?**Analysis of numerical errors in quantum circuit simulation**? in***arXiv preprint arXiv:2504.01430*: (2025).

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi. Bandung, 24 Desember 2025

Edward David Rumahorbo 13524036