Analisis Kompleksitas Algoritma dalam Menentukan Balikan Matriks Persegi

Bryan Cornelius Lauwrence - 13522033[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522033@std.stei.itb.ac.id

***Abstract*—Matriks adalah salah satu komponen yang sangat krusial dalam aljabar linear. Matriks dapat dioperasikan seperti bilangan bulat dengan cara yang khusus. Salah satunya adalah balikan matriks. Balikan matriks adalah matriks yang jika dikalikan dengan matriks asalnya akan menghasilkan matriks identitas. Matriks yang tidak memiliki balikan disebut matriks singular. Balikan matriks ada jika dan hanya jika matriks tersebut bukanlah matriks singular dan matriks tersebut adalah matriks persegi. Balikan matriks memiliki banyak penerapan dalam aljbar linear, salah satunya adalah untuk mencari solusi sistem persamaan linear. Terdapat dua metode untuk mencari balikan matriks, yaitu metode Gauss-Jordan dan metode matriks adjoin. Pencarian matriks balikan dapat diotomasi dengan membuat algoritma dari kedua metode tersebut. Agar otomasi dapat dioptimalkan, diperlukan pengetahuan atas kompleksitas algoritma dari kedua metode tersebut.**

***Keywords*—Balikan matriks, kompleksitas algoritma, metode Gauss-Jordan, metode matriks adjoin.**

I. PENDAHULUAN

Matriks merupakan salah satu komponen yang sangat penting dalam aljabar linear. Matriks adalah susunan ekspresi dalam baris dan kolom sehingga membentuk suatu segi-empat. Karena membentuk suatu segi-empat, matriks dapat berbentuk persegi. Matriks disebut matriks persegi jika memiliki jumlah baris dan kolom yang sama.

Matriks persegi yang bukan singular memiliki matriks balikan. Matriks balikan adalah matriks yang jika dikalikan dengan matriks aslinya akan menghasilkan matriks identitas, yaitu matriks yang komponen diagonal utamanya bernilai satu dan komponen lainnya bernilai nol. Matriks balikan memiliki banyak manfaat dalam aljabar linear. Salah satunya adalah dalam pencarian solusi suatu persamaan linear [1].

Pada matriks berukuran 2×2, balikannya dapat dicari dengan mudah, yaitu

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\text{ad} - \text{bc}} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \quad (1)$$

Namun, rumus tersebut tidak berlaku untuk matriks persegi yang jumlah baris dan kolomnya lebih dari dua. Oleh karena itu, terdapat metode lain untuk menentukan balikan matriks persegi yang tidak tergantung pada ordonya. Terdapat dua metode, yaitu metode eliminasi Gauss-Jordan dan metode matriks adjoin [2].

Perhitungan matriks balikan dapat diterapkan pada suatu program. Dengan begitu, perhitungan balikan suatu matriks persegi dapat diotomasi. Namun, ukuran matriks persegi bisa saja sangat besar, misalnya berukuran 1000×1000. Oleh sebab itu, diperlukan pengetahuan yang memadai tentang seberapa sangkil dan mangkus kedua metode mencari balikan suatu matriks yang telah disebutkan.

Dengan adanya kompleksitas algoritma, analisis kecepatan kedua metode dapat dilakukan dengan baik. Kompleksitas algoritma bergantung pada kebutuhan waktu (kompleksitas waktu) dan memori (kompleksitas ruang) yang diperlukan suatu algoritma dengan masukan data berukuran n. Semakin kecil kebutuhan waktu dan memorinya, semakin sangkil algoritma tersebut [3].

Kompleksitas waktu berkaitan dengan banyaknya langkah yang diperlukan untuk menyelesaikan proses algoritma dengan data berukuran n. Di sisi lain, kompleksitas ruang berkaitan dengan seberapa banyak memori yang diperlukan untuk menyelesaikan balikan.

Dalam makalah ini akan dibahas analisis kompleksitas waktu dalam notasi *Big-O* dari metode menentukan balikan suatu matriks persegi dan perbandingan dari keduanya. Matriks dianggap bukan matriks singular sehingga tidak dilakukan pengecekan.

II. LANDASAN TEORI

A. Matriks Balikan

Matriks dapat dioperasikan seperti operasi yang dilakukan pada bilangan bulat, seperti penjumlahan dan perkalian. Selain itu, khusus matriks persegi, matriks dapat memiliki matriks inversnya. Matriks yang tidak memiliki balikan disebut sebagai matriks singular. Ide dari balikan matriks adalah ketika suatu matriks dikalikan dengan matriks A yang selanjutnya dikalikan lagi dengan balikan matriks A, maka akan dihasilkan matriks awal. Dalam matriks, suatu matriks yang dikalikan dengan matriks balikannya akan menghasilkan matriks identitas [4]. Matriks identitas adalah matriks persegi yang nilai diagonal utamanya adalah satu dan nilai lainnya adalah nol. Misalkan, A adalah sebuah matriks berukuran n×n dan balikannya adalah A^{-1}, maka

$$AA^{-1} = A^{-1}A = I \quad (2)$$

Ada dua Metode untuk mencari balikan suatu matriks berkururan n×n, dengan n ≥ 1. Metode pertama adalah dengan eliminasi Gauss-Jordan dan metode kedua adalah dengan menggunakan matriks adjoin [2].

B. Metode Eliminasi Gauss-Jordan

Eliminasi Gauss-Jordan merupakan penerapan dari proses operasi baris elementer, yaitu mengalikan suatu baris pada matriks dengan konstanta atau menjumlahkan baris satu dengan baris lainnya, yang akan menghasilkan matrisk eselon baris tereduksi. Matriks eselon baris adalah matriks yang memiliki satu utama pada setiap baris, kecuali baris yang seluruhnya nol. Nilai di bawah satu utama pada matriks eselon baris harus bernilai nol. Matrisk eselon baris tereduksi adalah matriks eselon baris yang nilai di atas satu utamanya juga bernilai nol [5].

$$\begin{bmatrix} 1 & 2 & 7 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Gambar 1. Contoh matriks eselon baris (kiri) dan matriks eselon baris tereduksi (kanan).
(Sumber:.[5])

Metode mencari balikan matriks dengan eliminasi Gauss-Jordan adalah memasangkan suatu matriks *A* dengan matriks identitas *I* yang seukuruan dengan *A*. Kemudian, dilakukan operasi Gauss-Jordan sehingga matriks *A* menjadi matriks eselon baris tereduksi dan matriks *I* akan menjadi matriks balikan dari *A*, yaitu A^{-1}.

$$[A|I] \overset{\text{G-J}}{\sim} [I|A^{-1}]$$

Gambar 2. Visualisasi proses mencari matriks balikan dengan metode eliminasi Gauss-Jordan.
(Sumber : [1])

C. Metode Matriks Adjoin

Matriks adjoin tidak bisa diperoleh secara langsung dari matriks yang sudah tersedia. Pertama perlu ditentukan nilai kofaktor dari setiap elemen matriks. Kemudian nilai-nilai kofaktor dibentuk kembali menjadi matriks kembali, yang disebut dengan matriks kofaktor. Matriks adjoin adalah matriks transposisi dari matriks kofaktor. Balikan matriks akan dihitung dengan cara berikut

$$A^{-1} = \frac{1}{\det(A)} \times adj\,(A) \qquad (3).$$

Cara sederhana menentukan determinan matriks n×n untuk sembarang n adalah menjadikannya suatu matriks segitiga atas dengan operasi baris elementer. Matriks segitiga atas adalah matriks yang nilai elemen di bawah diagonal utamanya nol [6].

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}$$

Gambar 3. Matriks segitiga atas.
(Sumber : dokumen pribadi)

Determinan dari matriks segitiga atas dapat dihitung dengan mudah, yaitu dengan mengalikan seluruh elemen diagonal utamanya. Jika dilakukan pertukaran baris sebanyak p pada saat operasi baris, maka nilai determinan hanya perlu dikalikan dengan -1 pangkat p.

$$\det(A) = (-1)^p \times a_{11} \times a_{22} \times \ldots \times a_{nn} \qquad (4)$$

Minor entri elemen matriks di baris ke-i dan kolom ke-j adalah nilai determinan dari matriks tersebut tanpa menyertakan baris ke-i dan kolom ke-j. Nilai kofaktor elemen di baris ke-i dan kolom ke-j diperoleh dari minor entrinya, yaitu dengan mengalikan -1 pangkat i ditambah j dengan minor entrinya [7]. Contohnya matriks A

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

maka, minor entri elemen baris kedua dan kolom pertama (M_{21}) adalah

$$M_{21} = \det\begin{pmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{pmatrix} \qquad (5)$$

dan nilai kofaktor elemen baris kedua kolom pertama (C_{21}) adalah

$$C_{21} = (-1)^{2+1} \times M_{21} \qquad (6)$$

Matriks kofaktor merupakan matriks yang tiap elemennya diambil dari nilai kofaktor elemen matriks awalnya. Sebagai contoh matriks kofaktor dari matriks A adalah

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$$

Matrisk adjoin adalah hasil transposisi dari matriks kofaktor yang sudah diperoleh [2]. Contohnya matriks adjoin dari matriks A

$$adj(A) = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}^T = \begin{bmatrix} C_{11} & C_{21} & C_{31} \\ C_{12} & C_{22} & C_{32} \\ C_{13} & C_{23} & C_{33} \end{bmatrix} \qquad (7)$$

Selanjutnya, balikan matriks dapat diperoleh dengan cara pada persamaan 3.

D. Kompleksitas Algoritma

Algoritma yang baik adalah algoritma yang sangkil dan mangkus. Algoritma yang sangkil dan mangkus adalah algoritma yang memerlukan sesedikit mungkin waktu eksekusi dan sesedikit mungkin penggunaan memori. Untuk mengukur tingkat kompleksitas suatu algoritma, diperlukan metode pengukuran yang cocok, akurat, dan universal. Dari sanalah

konsep kompleksitas algoritma berasal.

Kompleksitas algoritma adalah suatu besaran yang berfungsi untuk mengukur seberapa efisien suatu algoritma. Kompleksitas algoritma mengukur waktu dan ruang penyimpanan ketika mengeksekusi suatu algoritma. Waktu menggambarkan seberapa banyak langkah yang diperlukan untuk menyelesaikan algoritma. Ruang menggambarkan penyimpanan yang diperlukan ketika menjalankan suatu algoritma.

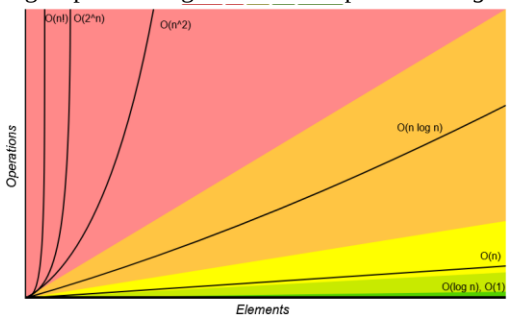
Kompleksitas waktu diukur dari banyaknya tahapan algoritma jika diberikan data berukuran n. Tahapan yang dimaksud berupa operasi matematika dasar seperti penjumlahan, pengurangan, perkalian, dan pembagian. Semakin sedikit tahapan yang harus dilakukan, semakin kecil kompleksitas waktunya.

Kompleksitas ruang diukur dari banyaknya alokasi memori yang diperlukan selama algoritma dieksekusi. Pengalokasian memori menandakan seberapa banyak data yang harus disimpan ketika program berjalan. Semakin sedikit memori yang diperlukan, semakin rendah kompleksitas ruangnya.

Kompleksitas algoritma dinotasikan secara asimptotik, yaitu notasi kompleksitas untuk algoritma dengan masukkan data yang besar. Salah satu notasi asimptotik adalah notasi *Big-O*. Untuk suatu algoritma yang membutuhkan sebanyak T(n) langkah jika menerima n masukkan, sama dengan O(f(n)) yang artinya T(n) berorde paling besar f(n) bila terdapat konstanta C sedemikian sehingga

$$T(n) \leq C \times f(n) \qquad (8)$$

untuk $n \geq n_0$ [7]. Notasi *Big-O* menggunakan fungsi-fungsi yang "standar", misalnya 1, n, n^2, dan lain-lain. Berikut adalah perbandingan-perbandingan dari beberapa notasi *Big-O*



Gambar 4. Grafik pertumbuhan beberapa notasi Big-O
(Sumber : [8])

III. Analisis Kompleksitas Algoritma Mencari Balikan Matriks

A. Metode Eliminasi Gauss-Jordan

Fungsi dan prosedur yang diperlukan dalam proses eliminasi Gauss-Jordan adalah fungsi untuk membuat matriks identitas (MakeIdentityMatrix), fungsi untuk perpangkatan (Power), prosedur pergantian baris (SwapRow), prosedur untuk mengalikan baris matriks dengan suatu konstanta (MultiplyRow), dan prosedur untuk menjumlahkan baris satu dengan baris lainnya (AddRow). Yang pertama adalah fungsi untuk membuat matriks identitas sebagai berikut

```
function MakeIdentityMatrix(n: integer) → array
        [1..n] of array[1..n] of real
KAMUS LOKAL
    M: array[1..n] of array[1..n] of integer
    i, j: integer
ALGORITMA
    i traversal [1..n]
        j traversal [1..n]
            if (i = j) then
                M[i][j] ← 1
            else
                M[i][j] ← 0
    → M
```

Gambar 5. Algoritma pembuat matriks identitas
(Sumber: dokumen pribadi)

Pada algoritma tersebut, untuk setiap nilai i dari 1 sampai n, terdapat n langkah untuk mengisi setiap elemen matriks. Jadi, kompleksitas algoritmanya adalah

$$T(n) = n^2 \qquad (9)$$

Berikutnya adalah prosedur untuk pertukaran baris sebagai berikut

```
procedure SwapRow(input/output M: array
        [1..n] of array[1..n] of real, input a,b:
integer)
KAMUS LOKAL
    Temp: array [1..n] of integer
ALGORITMA
    Temp ← M[a]
    M[a] ← M[b]
    M[b] ← Temp
```

Gambar 6. Algoritma pertukaran baris matriks
(Sumber: dokumen pribadi)

Algoritma tersebut melakukan penukaran antara baris a dan baris b. Untuk melakukannya terdapat tiga tahapan, yaitu menyimpan nilai baris a, mengganti nilai baris a menjadi baris b, dan mengganti baris b menjadi nilai yang sudah disimpan. Secara keseluruhan, prosedur tersebut melaksanakan tiga langkah. Jadi kompleksitas algoritmanya adalah

$$T(n) = 3 \qquad (10)$$

Prosedur selanjutnya adalah prosedur untuk mengalikan baris matriks dengan suatu konstanta sebagai berikut

```
procedure MultiplyRow(input/output M: array[1..n]
        of array [1..n] of real, input row:
        integer, c: real)
KAMUS LOKAL
    i: integer
ALGORITMA
    i traversal [1..n]
        M[row][i] ← M[row][i] * c
```

Gambar 7. Algoritma perkalian baris matriks dengan konstanta
(Sumber: dokumen pribadi)

Prosedur tersebut melakukan sejumlah n perkalian pada prosesnya sehingga kompleksitasnya adalah

$$T(n) = n \quad (11)$$

Prosedur yang terakhir adalah prosedur untuk menjumlahkan suatu baris matriks dengan baris lainnya yang dikalikan dengan suatu konstanta. Algoritmanya sebagai berikut

```
procedure AddRow(input/output M: array [1..n] of
        array [1..n] of real, input row1,row2:
        integer, c: real)
KAMUS LOKAL
    i: integer
ALGORITMA
    i traversal [1..n]
       M[row2][i] ← M[row2][i] + M[row1][i] * c
```

Gambar 8. Algoritma penjumlahan antarbaris pada matriks
(Sumber: dokumen pribadi)

Prosedur AddRow melakukan dua langkah pada setiap perulangan, yaitu penjumlahan dan perkalian sehingga untuk matriks persegit berukuran n×n kompleksitasnya adalah

$$T(n) = 2n \quad (12)$$

Setelah diperoleh fungsi-fungsi yang dapat digunakan, barulah proses mencari balikan matriks dapat dilakukan. Prosesnya adalah membuat matriks identitas yang seukuran, lalu menjadikan elemen diagonal utama menjadi satu dengan perkalian baris dan mengubah nilai di atas maupun di bawah diagonal utama matriks menjadi nol. Jika ditemukan elemen diagonal utama bernilai nol, akan dilakukan pertukaran baris dengan elemen di bawahnya yang bukan nol terlebih dahulu. Algoritmanya sebagai berikut

```
function InverseGaussJordan(Min: array [1..n] of
        array[1..n] of real) → array [1..n] of
        array[1..n] of real
KAMUS LOKAL
    Mout: array [1..n] of array [1..n] of real
    i, j: integer
ALGORITMA
    Mout ← MakeIdentity(n)
    i traversal [1..n]
       if (Min[i][i] = 0) then
            j ← i + 1
            while (Min[j][i] = 0) do
               j ← j + 1
            SwapRow(Min,i,j)
            SwapRow(Mout,i,j)

       MultiplyRow(Mout,i,1/Min[i][i])
       MultiplyRow(Min,i,1/Min[i][i])

       j traversal [1..n]
           if (j ≠ i) then
               AddRow(Mout,i,j,(-1)*Min[j][i])
               AddRow(Min,i,j,(-1)*Min[j][i])
    → Mout
```

Gambar 9. Algoritma mencari balikan matriks dengan metode eliminasi Gauss-Jordan
(Sumber: dokumen pribadi)

Pada algorima tersebut, langkah awalnya adalah membuat matriks identitas yang memerlukan n^2 langkah. Selanjutnya pada proses perulangan i, jika diambil kasus terburuknya, akan terjadi pertukaran baris antara baris i dengan baris terakhir sehingga terjadi n-i langkah untuk mencari baris dan enam langkah untuk pertukaran baris. Kedua hal tersebut akan diulangi sebanyak n kali. Selanjutnya, proses MultiplyRow dilakukan 2 kali 2n kali yang terjadi sebanyak n kali. Terakhir, adalah proses AddRow yang akan terjadi sebanyak n-1 kali pada setiap perulangannya. Jadi, untuk setiap langkah ke-i,

i = 1 → jumlah langkah: n-1 + 6 + 2n + n(n-1)
i = 2 → jumlah langkah: n-2 + 6 + 2n + n(n-1)
⋮
i = n → jumlah langkah: 0 + 6 + 2n + n(n-1)

sehingga kompleksitas algoritmanya

$$T(n) = n^2 + (n-1+n-2+\cdots+1+0) + n\big(6+2n+n(n-1)\big)$$
$$T(n) = n^2 + \frac{n(n-1)}{2} + 6n + 2n^2 + n^3 + n^2 \quad (13)$$

Notasi *Big-O* untuk proses pencarian balikan matriks dengan metode matriks balikan ditentukan sebagai berikut

$$T(n) = n^3 + \frac{9}{2}n^2 + \frac{11}{2}n \le n^3 + \frac{9}{2}n^3 + \frac{11}{2}n^3$$
$$T(n) \le 11n^3, \text{untuk } n \ge 1$$
$$T(n) = O(n^3) \quad (14)$$

Jadi, notasinya adalah $O(n^3)$. Artinya, metode ini memiliki tingkat kompleksitas kubik.

B. Metode Matriks Adjoin

Fungsi dan prosedur yang diperlukan untuk metode ini meliputi fungsi mencari determinan (Det) dengan membentuk matriks segitiga atas, fungsi memperoleh matriks minor (MinorMatrix), fungsi untuk mencari matriks kofaktor (Cofactor), dan fungsi transposisi matriks (Transpose). Yang pertama fungsi untuk mencari determinan sebagai berikut

```
function Det(M: array [1..n] of array[1..n] of
        real) → real
KAMUS LOKAL
    Ctr, i, j: integer
    D: real
ALGORITMA
    Ctr ← 1
    i traversal [1..n]
       if (M[i][i] = 0) then
            Ctr ← Ctr * (-1)
            j ← i+1
            while (M[j][i] ≠ 0) do
               j ← j+1
            SwapRow(M,i,j)
       j traversal [i+1..n]
           AddRow(M,i,j,(-1)*(M[j][i]/M[i][i]))
    D ← 1
    i traversal [1..n]
       D ← D * M[i][i]
    → Ctr * D
```

Gambar 10. Algoritma menghitung determinan matriks
(Sumber: dokumen pribadi)

Perhitungan kompleksitas algoritma diambil dari kasus terburuknya, yaitu jika harus dilakukan perturkaran baris dengan baris paling bawah pada setiap prosesnya. Pada awal proses, program memulai satu langkah untuk mengisi Ctr dengan satu. Selanjutnya dimulai perulangan sebanyak n. Pada proses ke-i pertukaran baris dilakukan perkalian Ctr untuk satu langkah, pencarian sebanyak n-i langkah, dan Swap sebanyak tiga langkah. Kemudian, dilanjutkan dengan proses AddRow sebanyak n-i langkah. AddRow melakukan n langkah sehingga dilakukan n(n-i) langkah. Terakhir, dibutuhkan selangkah untuk mengisi nilai D menjadi 1 dan mengalikan D sebanyak n kali. Sebelum mengembalikan nilai, dilakukan satu langkah terakhir untuk mengalikan Ctr dengan D. Jadi, untuk setiap langkah ke-i,

i = 1 → jumlah langkah: 1 + n-1 + 3 + n(n-1)
i = 2 → jumlah langkah: 1 + n-2 + 3 + n(n-2)
⋮
i = n → jumlah langkah: 1 + 0 + 3 + 0

sehingga kompleksitas algoritmanya

$$T(n) = 1 + \frac{n(n-1)}{2} + 3 + \frac{n(n+n(n-1))}{2}$$
$$T(n) = 4 + \frac{n^2}{2} - \frac{n}{2} + \frac{n^2}{2} + \frac{n^3}{2} - \frac{n^2}{2} \quad (15)$$

Berikutnya adalah fungsi untuk mencari matriks minor sebagai berikut

```
function MinorMatrix(M: array [1..n] of
       array[1..n] of real, row,col: integer) →
       array [1..n-1] of array[1..n-1] of real
KAMUS LOKAL
    Mo: array [1..n-1] of array [1..n-1] of real
    i, j, k, l: integer
ALGORITMA
    l ← 1
    i traversal [1..n]
       if (i ≠ row) then
           k ← 1
           j traversal [1..n]
              if (j ≠ col) then
                  Mo[l][k] ← M[i][j]
                  k ← k + 1
           l ← l + 1
    → Mo
```

Gambar 11. Algoritma menentukan minor matriks
(Sumber: dokumen pribadi)

Fungsi tersebut awalnya memerlukan satu langkah untuk mengisi nilai l. Selanjutnya fungsi akan melakukan n buah perulangan. Pada setiap perulangan dilakukan 1 langkah di awal dan di akhir. Selain itu, dilakukan juga 2 kali n-1 perulangan. Jadi kompleksitas algoritmanya

$$T(n) = 1 + (n-1)(2 + 2(n-1)) \quad (16)$$

Selanjutnya adalah fungsi untuk menghasilkan matriks kofaktor sebagai berikut

```
function CofactorMatrix(M: array [1..n] of
       array[1..n] of real) → array[1..n] of
       array[1..n] of real
KAMUS LOKAL
    C: array [1..n] of array [1..n] of real
    i, j: integer
    Ctr, Temp: integer
ALGORITMA
    Ctr ← -1
    i traversal [1..n]
       Ctr ← Ctr * (-1)
       Temp ← Ctr
       j traversal [1..n]
           C[i][j] ←Temp*Det(MinorMatrix(M,i,j))
           Temp ← Temp * (-1)
    → C
```

Gambar 12. Algoritma menentukan matriks kofaktor
(Sumber: dokumen pribadi)

Algoritma dimulai dengan satu langkah pengisian nilai Ctr. Selanjutnya, program melaukan perulangan sebanyak n kali. Pada perulangan tersebut dilakukan sekali perkalian dan sekali pengisian nilai Temp. Proses dilanjutkan lagi dengan perulangan sebanyak n. Untuk setiap perulangan, dilakukan 1 proses mencari minor matriks, satu proses pencarian determinan dari matriks persegi berukuran n-1, dan satu langkah perkalian. Jadi, untuk setiap langkah ke-i,

i = 1 → jumlah langkah: $2 + 2n^2 - 2n + 1 + 4 + n^3/2 + n^2/2 - n/2$
i = 2 → jumlah langkah: $2 + 2n^2 - 2n + 1 + 4 + n^3/2 + n^2/2 - n/2$
⋮
i = n → jumlah langkah: $2 + 2n^2 - 2n + 1 + 4 + n^3/2 + n^2/2 - n/2$

sehingga kompleksitas algoritmanya

$$T(n) = n(2 + 2n^2 - 2n + 1 + 4 + \frac{n^3}{2} + \frac{n^2}{2} - \frac{n}{2})$$
$$T(n) = \frac{1}{2}n^4 + \frac{5}{2}n^3 - \frac{5}{2}n^2 + 7n \quad (17)$$

Fungsi terakhir adalah fungis tranposisi matriks. Algoritmanya sebagai berikut

```
funciton Transpose(M: array [1..n] of array
       [1..n] of real) → array [1..n] of array
       [1..n] of real
KAMUS LOKAL
    i, j: integer
    Tm: array [1..n] of array [1..n] of real
ALGORITMA
    i traversal [1..n]
       j traversal [1..n]
           Tm[i][j] ← M[i][j]
    → Tm
```

Gambar 13. Algoritma tranposisi matriks
(Sumber: dokumen pribadi)

Dari algoritma tersebut, terjadi n perulangan. Pada setiap perulangan dilakukan n langkah. Oleh karena itu kompleksitas algoritmanya sebagai berikut

$$T(n) = n^2 \quad (16)$$

Setelah diperoleh fungsi dan prosedur yang diperlukan, barulah dapat dihitung balikan dari matriks yang ingin dicari. Algoritmanya sebagai berikut

```
function InverseAdjoin(Min: array [1..n] of
       array[1..n] of real) → array [1..n] of
       array[1..n] of real
KAMUS LOKAL
    Mout: array [1..n] of array [1..n] of real
    D: real
    i: integer
ALGORITMA
    Mout ← CofactorMatrix(Min)
    Mout ← Transpose(Mout)
    D ← Det(Min)
    i traversal [1..n]
       MultiplyRow(Mout,i,1/D)
    → Mout
```

Gambar 14. Algoritma mencari balikan matriks dengan metode eliminasi Gauss-Jordan
(Sumber: dokumen pribadi)

Proses pencarian balikan matriks dengan adjoin dimulai dengan menghitung matriks kofaktornya. Setelah itu, matriks kofaktor ditransposisikan agar diperoleh Kemudian, dicari nilai determinan dari matriks asal. Terakhir, setiap baris matriks adjoin dibagi dengan nilai determinan matriks asal. Jadi, kompleksitas algoritmanya

$$T(n) = \frac{1}{2}n^4 + \frac{5}{2}n^3 - \frac{5}{2}n^2 + 7n + n^2 + 4 + \frac{n^2}{2} - \frac{n}{2} + \frac{n^2}{2} + \frac{n^3}{2} - \frac{n^2}{2} + n^2$$

$$T(n) = \tfrac{1}{2}n^4 + 3n^3 + \tfrac{13}{2}n + 4 \qquad (16)$$

Notasi *Big-O* untuk proses pencarian balikan matriks dengan metode matriks adjoin dapat ditentukan sebagai berikut

$$T(n) = \frac{1}{2}n^4 + 3n^3 + \frac{13}{2}n + 4 \le \frac{1}{2}n^4 + 3n^4 + \frac{13}{2}n^4 + 4n^4$$

$$T(n) \le 14n^4, \text{ untk n} \ge 1$$

$$T(n) = O(n^4) \qquad (17)$$

Jadi, notasinya adalah $O(n^4)$. Artinya, metode ini memiliki tingkat kompleksitas pangkat empat.

IV. KESIMPULAN

Dari analisis yang telah dilakukan, dapat disimpulkan bahwa kedua metode pencarian balikan matriks memiliki tingkat kompleksitas yang berbeda. Metode Gauss-Jordan memiliki kompleksitas $O(n^3)$, sedangkan metode matriks adjoin memiliki kompleksitas $O(n^4)$. Dapat dilihat bahwa metode matriks adjoin membutuhkan waktu yang lebih lama ketimbang metode Gauss-Jordan. Hal ini dapat terjadi karena metode matriks adjoin perlu menghitung determinan, proses perhitungan determinan dilakukan dengan operasi baris elementer. Di sisi lain, metode Gauss-Jordan menghitung operasi beris elementer, tetapi metode Gauss-Jordan dapat langsung menghasilkan matriks balikannya. Bagaimanapun juga, keduanya memiliki kompleksitas yang buruk menurut Gambar 4. Jadi, keduanya masih memerlukan optimalisasi jika memungkinan supaya program dapat berjalan lebih cepat lagi.

V. LAMPIRAN

Kode program untuk mencari balikan matriks persegi dapat diperiksa pada pranala berikut:

https://github.com/BryanLauw/MakalahMatdis.git

VI. UCAPAN TERIMA KASIH

Puji syukur dan terima kasih kepada Tuhan Yang Maha Kuasa. Sebab atas bimbingan dan rahmat-Nya, makalah ini dapat diselesaikan sesuai tujuan penulis. Terima kasih juga diucapkan kepada orang tua, keluarga, dan teman-teman penulis sebab telah memberikan dukungan selama proses penulisan makalah ini. Kepada yang terhormat Dr. Nur Ulfa Maulidevi, S.T, M.Sc. selaku dosen pengajar IF2120 Matematika Diskrit kelas K1, penulis mengucapkan terima kasih atas bimbingan dan bantuannya selama makalah ini dibuat. Tidak lupa, penulis mengucapkan terima kasih kepada pihak lainnya yang mendukung terselesaikannya makalah ini dan kepada para pembaca. Terakhir, penulis dengan segenap hati mengucapkan minta maaf jika pada makalah ini terdapat salah kata yang tanpa sengaja menyakiti pihak-pihak tertentu.

REFERENSI

[1] R. Munir, "Sistem persamaan linier (Bagian 3: Metode eliminasi Gauss-Jordan)," *IF2123 Aljabar Linier dan Geometri*, 2023. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-05-Sistem-Persamaan-Linier-2023.pdf. [Diakses: 8 Desember 2023].

[2] R. Munir, "Determinan (Bagian 2)," *IF2123 Aljabar Linier dan Geometri*, 2023. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-09-Determinan-bagian2-2023.pdf. [Diakses: 8 Desember 2023].

[3] R. Munir, "Kompleksitas Algoritma (Bagian 1)," *IF2120 Matematika Diskrit*, 2023. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/24-Kompleksitas-Algoritma-Bagian1-2023.pdf. [Diakses: 8 Desember 2023].

[4] G. Strang, "Matrices and Gaussian Elimination," *Linear Algebra and Its Applications*, Fourth Edition. United States of America : Cengage Learning, 2006. pp. 45.

[5] R. Munir, "Matriks Eselon," *IF2123 Aljabar Linier dan* Geometri, 2023. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-02-Matriks-Eselon-2023.pdf. [Diakses: 8 Desember 2023].

[6] R. Munir, "Determinan (Bagian 1)," *IF2123 Aljabar Linier dan Geometri*, 2023. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-08-Determinan-bagian1-2023.pdf. {Diakses: 8 Desember 2023].

[7] R. Munir, "Kompleksitas Algoritma (Bagian 2)," *IF2120 Matematika Diskrit*, 2023. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/25-Kompleksitas-Algoritma-Bagian2-2023.pdf. [Diakses: 8 Desember 2023].

[8] Eric, "Know Thy Complexities!," *Big-O Cheat Sheet*, 2012. [Online]. Tersedia: https://www.bigocheatsheet.com/. [Diakses: 9 Desember 2023].

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, 10 Desember 2023

Bryan Cornelius Lauwrence
13522033