

Laporan

Tugas Besar 1 IF2123 Aljabar Linier dan Geometri

Sistem Persamaan Linier, Determinan, dan Aplikasinya



oleh

Andres Jerriel Sinabutar 13519218

Gde Anantha Priharsena 13519026

Shifa Salsabiila 13519106

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2020

DAFTAR ISI

DAFTAR ISI.....	ii
BAB I: DESKRIPSI MASALAH.....	1
1.1 Spesifikasi Tugas	1
1.2 Bahasa Program yang Digunakan dan Abstraksi Program.....	4
BAB II: TEORI SINGKAT	5
2.1 Matriks dan Sistem Persamaan Linear.....	5
2.1.1 Metode Eliminasi Gauss.....	6
2.1.2 Metode Eliminasi Gauss-Jordan.....	7
2.2 Determinan.....	8
2.2.1 Metode Operasi Baris Elementer.....	8
2.2.2 Metode Ekspansi Kofaktor	9
2.3 Matriks Balikan.....	10
2.4 Matriks Kofaktor.....	11
2.5 Matriks Adjoin.....	12
2.6 Kaidah Cramer	12
2.7 Interpolasi Polinom.....	13
2.8 Regresi Linier Berganda	14
BAB III: IMPLEMENTASI PROGRAM.....	16
3.1 Class MatriksInit.....	16
3.2 Class Matriks	21
3.2.1 SPLGauss	21
3.2.2 SPLGaussJordan.....	27
3.2.3 SPLCramer	29
3.2.4 DeterminanOBE	33
3.2.5 DeterminanDenganKofaktor	34
3.2.6 SolusiInterpolasi	35
3.2.7 MultipleLinearRegression	40
3.3 Class InversMatriks	44
3.4 Class TubesAlgeo	49
BAB IV: EKSPERIMEN	54
4.1 SPL.....	54
4.2 Determinan.....	58

4.3 Matriks Balikan.....	59
4.4 Interpolasi Polinom.....	61
4.5 Regresi Linier Berganda	63
BAB V: KESIMPULAN, SARAN, DAN REFLEKSI.....	64
5.1 Kesimpulan	64
5.2 Saran	64
5.3 Refleksi	65
DAFTAR REFERENSI	66

BAB I

DESKRIPSI MASALAH

1.1 Spesifikasi Tugas

Pada tugas besar ini, mahasiswa diminta untuk membuat program dalam Bahasa Java yang dapat digunakan untuk

1. Menghitung solusi SPL dengan metode eliminasi Gauss, metode Eliminasi Gauss-Jordan, metode matriks balikan, dan kaidah Cramer (kaidah Cramer khusus untuk SPL dengan n peubah dan n persamaan).
2. Menyelesaikan persoalan interpolasi dan regresi linier.
3. Menghitung matriks balikan
4. Menghitung determinan matriks dengan berbagai metode (reduksi baris dan ekspansi kofaktor).

Spesifikasi program yang dibuat sebagai berikut:

1. Program dapat menerima masukan (input) baik dari keyboard maupun membaca masukan dari file text. Untuk SPL, masukan dari keyboard adalah m , n , koefisien a_{ij} , dan b_i . Masukan dari file berbentuk matriks augmented tanpa tanda kurung, setiap elemen matriks dipisah oleh spasi. Misalnya,

```
3    4.5 2.8 10 12  
-3    7    8.3 11 -4  
0.5 -10 -9   12  0
```

2. Untuk persoalan menghitung determinan dan matriks balikan, masukan dari keyboard adalah n dan koefisien a_{ij} . Masukan dari file berbentuk matriks, setiap elemen matriks dipisah oleh spasi. Misalnya,

```
3    4.5  2.8  10  
-3    7    8.3  11  
0.5 -10  -9   12
```

3. Untuk persoalan interpolasi, masukannya jika dari keyboard adalah n , (x_0, y_0) , (x_1, y_1) , ..., (x_n, y_n) , dan nilai x yang akan ditaksir nilai fungsinya. Jika masukannya dari file, maka titik-titik dinyatakan pada setiap baris tanpa koma dan tanda kurung. Misalnya jika titik-titik datanya adalah $(8.0, 2.0794)$, $(9.0, 2.1972)$, dan $(9.5, 2.2513)$, maka di dalam file *text* ditulis sebagai berikut:

```
8.0  2.0794  
9.0  2.1972  
9.5  2.2513
```

4. Untuk persoalan regresi, masukannya jika dari keyboard adalah n (jumlah peubah x), semua nilai-nilai $x_{1i}, x_{2i}, \dots, x_{ni}$, nilai y_i , dan nilai-nilai x_k yang akan ditaksir nilai fungsinya. Jika masukannya dari file, maka titik-titik dinyatakan pada setiap baris tanpa koma dan tanda kurung.
5. Untuk persoalan SPL, luaran (output) program adalah solusi SPL. Jika solusinya tunggal, tuliskan nilainya. Jika solusinya tidak ada, tuliskan solusi tidak ada, jika solusinya banyak, maka tuliskan solusinya dalam bentuk parametrik (misalnya $x_4 = -2$, $x_3 = 2s - t$, $x_2 = s$, dan $x_1 = t$.)

6. Untuk persoalan determinan dan matriks balikan, maka luarannya sesuai dengan persoalan masing-masing
7. Untuk persoalan polinom interpolasi dan regresi, luarannya adalah persamaan polinom/regresi dan taksiran nilai fungsi pada x yang diberikan.
8. Luaran program harus dapat ditampilkan pada layar komputer dan dapat disimpan ke dalam file.
9. Bahasa program yang digunakan adalah Java.
10. Program tidak harus berbasis GUI, cukup text-based saja, namun boleh menggunakan GUI (memakai kakas Eclipse misalnya).
11. Program dapat dibuat dengan pilihan menu. Urutan menu dan isinya dipersilakan dirancang masing-masing. Misalnya, menu:

MENU

1. Sistem Persamaan Linier
2. Determinan
3. Matriks balikan
4. Interpolasi Polinom
5. Regresi linier berganda
6. Keluar

Untuk pilihan menu nomor 1 ada sub-menu lagi yaitu pilihan metode:

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer

Begitu juga untuk pilihan menu nomor 2 dan 3.

1.2 Bahasa Program yang Digunakan dan Abstraksi Program

Pada pembuatan program ini digunakan bahasa pemrograman Java dengan kakas pengembangan program menggunakan J2SE. Program untuk tugas ini sendiri tidak harus menggunakan *Graphical User Interface*, program diperbolehkan untuk dijalankan secara *text-based*, namun program dengan GUI tidak dilarang. Ketika program dijalankan, program menampilkan pilihan menu kepada pengguna, dan akan menerima masukan yang berupa pilihan dari menu tersebut. Selanjutnya program akan menjalankan perintah sesuai pilihan menu yang dimasukan oleh pengguna.

BAB II

TEORI SINGKAT

2.1 Matriks dan Sistem Persamaan Linear

Persamaan linear adalah persamaan dimana peubahnya tidak memuat eksponensial, trigonometri (seperti \sin , \cos , dll.), perkalian, pembagian dengan peubah lain atau dirinya sendiri. Jadi, sistem persamaan linear merupakan sekumpulan persamaan linear yang memuat sejumlah hingga peubah bebas yang saling terkait. Bentuk umum sistem persamaan linear :

$$a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n = b_1$$

$$a_{21} x_1 + a_{22} x_2 + \dots + a_{2n} x_n = b_2$$

$$\vdots \qquad \vdots$$

$$\vdots \qquad \vdots$$

$$a_{m1} x_1 + a_{m2} x_2 + \dots + a_{mn} x_n = b_m$$

Sistem persamaan linear diatas dapat ditulis dalam bentuk matriks yaitu:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \ddots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Menentukan solusi persamaan linear dapat dilakukan dengan menggunakan operasi baris elementer (OBE). Langkah yang pertama adalah tulis kembali sistem persamaan linear dalam bentuk matriks yang diperbesar (*augmented matrix*).

Misalkan SPL

$$3x + 2y = 5$$

$$2x - y = 3$$

dapat ditulis dalam bentuk matriks yang diperbesar:

$$\left(\begin{array}{cc|c} 3 & 2 & 5 \\ 2 & -1 & 3 \end{array} \right)$$

Selanjutnya dilakukan OBE pada matriks tersebut untuk menentukan solusinya.

2.1.1 Metode Eliminasi Gauss

Metode Eliminasi Gauss merupakan metode yang dikembangkan dari metode eliminasi, yaitu menghilangkan atau mengurangi jumlah peubah sehingga dapat diperoleh nilai dari suatu peubah bebas.

$$\left[\begin{array}{cccccc} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} & b_2 \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} & b_3 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} & b_n \end{array} \right] \rightarrow \left[\begin{array}{cccccc} c_{11} & c_{12} & c_{13} & \dots & c_{1n} & d_1 \\ 0 & c_{22} & c_{23} & \dots & c_{2n} & d_2 \\ 0 & 0 & c_{33} & \dots & c_{3n} & d_3 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & c_{mn} & d_n \end{array} \right]$$

Suatu metode dimana bentuk matriks di atas, pada bagian kiri diubah menjadi matriks segitiga atas atau segitiga bawah dengan menggunakan OBE (Operasi Baris Elementer). Sehingga penyelesaian dapat diperoleh dengan metode substitusi mundur sebagai berikut :

$$\begin{aligned} x_n &= \frac{d_n}{c_{nn}} \\ x_{n-1} &= \frac{1}{c_{n-1,n-1}} (-c_{n-1,n}x_n + d_{n-1}) \\ &\dots \\ x_2 &= \frac{1}{c_{22}} (d_2 - c_{23}x_3 - c_{24}x_4 - \dots - c_{2n}x_n) \\ x_1 &= \frac{1}{c_{11}} (d_1 - c_{12}x_2 - c_{13}x_3 - \dots - c_{1n}x_n) \end{aligned}$$

Berikut *pseudocode* untuk mengimplementasi Metode Eliminasi Gauss:

```

procedure Gauss
    i traversal [IdxBrsMin(M)..NBrsEff(M)]
        j traversal [i+1..NBrsEff(M)]
            if (Elmt(M,j,i) != 0) then
                multiplier <- Elmt(M,j,i)
                divider <- Elmt(M,i,i)
                k transversal [i..NKolEff(M)]
                    Elmt(M,j,k) -= multiplier * (Elmt(M,i,k)) / divider;
    
```

2.1.2 Metode Eliminasi Gauss-Jordan

Metode Eliminasi Gauss Jordan merupakan pengembangan metode eliminasi Gauss, hanya saja matriks *augmented* pada sebelah kiri diubah menjadi matriks eselon baris tereduksi.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \ddots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & \dots & 0 & d_1 \\ 0 & 1 & \dots & 0 & d_2 \\ \vdots & \ddots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & d_n \end{bmatrix}$$

Berikut *pseudocode* untuk mengimplementasi Metode Eliminasi Gauss-Jordan

```
Procedure Gauss_Jordan
    Gauss()
    Integer initialKol = 1;
    J traversal [IdxKolMin(M)..NKolEff(M)]
        if (isAllZeroKol(j))then
            initialKol <- j
            continue
        else then
            break
    int i <- IdxBrsMin(M)
    int j <- 1
    int k <- NBrssEff(M)
    while (i <= k)
        if (i = k) then
            i <- IdxBrsMin(M)
            j++
            k--
        else then
            found <- false;
            int c = 1
            while (not found and c <= NBrssEff(M) - i)
                if (Elmt(M,i + c,i + j) != 0) then
                    found <- true
                else then
                    c++
            int jj <- IdxKolMin(M)
            while (found AND jj < i + j) then
                if (Elmt(M,i,jj) = 0 AND Elmt(M,i + c,jj) != 0)then
                    found <- false
                else then
                    jj++
            if (found AND Elmt(M,i,i + j) != 0) then
                multiplier = Elmt(M,i,i + j)
                divider = Elmt(M,i + c,i + j)
                n traversal [IdxKolMin(M)..NKolEff(M)]
                    Elmt(M,i,n) -= multiplier * Elmt(M,i + c,n)/divider
            i++
        end if
    end while
end Gauss_Jordan
```

2.2 Determinan

Setiap matriks kuadrat/persegi mempunyai suatu nilai khusus yang disebut determinan. determinan adalah jumlah hasil kali elementer bertanda dari suatu matriks. Determinan dapat dipahami sebagai jumlah semua hasil kali entri-entri matriks yang tidak berada pada baris atau kolom yang sama. Determinan dinotasikan sebagai berikut:

Jika A adalah matriks persegi

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

Maka determinan A ditulis: $\det(A)$ atau $|A|$

$$|A| = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

Determinan dari sebuah matriks dapat dicari dengan operasi baris elementer dan ekspansi kofaktor.

2.2.1 Metode Operasi Baris Elementer

Cara mencari determinan dengan operasi baris elementer adalah sebagai berikut:

1. Lakukan operasi baris dengan memperhatikan tiga hal sebagai berikut:
 - a. Jika A' adalah matriks yang dihasilkan dari pertukaran dua baris matriks A maka $\det(A') = -\det(A)$
 - b. Jika A' adalah matriks yang dihasilkan dari perkalian matriks dengan suatu konstanta k maka $\det(A') = k\det(A)$
 - c. Jika A' adalah matriks yang dihasilkan dari penjumlahan hasil kali dari baris satu ke baris yang lain pada matriks A maka $\det(A') = \det(A)$
2. Operasi baris berhenti jika matriks yang dihasilkan adalah matriks segitiga atas atau segitiga bawah.

$$A = \begin{matrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{matrix} \quad B = \begin{matrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{matrix}$$

Matriks A adalah matriks segitiga atas dan matriks B adalah matriks segitiga bawah.

3. Determinan diperoleh dengan mengalikan semua entri pada diagonal utama dari matriks segitiga atas atau segitiga bawah

2.2.2 Metode Ekspansi Kofaktor

Suatu determinan matriks dapat dihitung dengan **ekspansi kofaktor**. Jika A adalah suatu matriks persegi ($n \times n$), maka minor entri a_{ij} dinyatakan sebagai M_{ij} dan didefinisikan menjadi determinan submatriks yang tersisa setelah baris ke i dan kolom ke j dihilangkan dari A. Bilangan $(-1)^{i+j}M_{ij}$ dinyatakan sebagai C_{ij} dan disebut sebagai **kofaktor** dari entri a_{ij} .

Kofaktor dan minor dari suatu elemen a_{ij} hanya berbeda dalam tandanya, dimana $C_{ij} = \pm M_{ij}$. Cara yang lebih baik untuk menentukan tanda yang menghubungkan C_{ij} dan M_{ij} berada pada baris ke i dan kolom ke j dari susunan berikut:

$$\begin{bmatrix} + & - & + & - & \cdots \\ - & + & - & + & \cdots \\ + & - & + & - & \cdots \\ - & + & - & + & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Berdasarkan matriks tanda diatas, maka didapatkan kofaktor:

$$C_{11} = M_{11}, C_{21} = -M_{21}, C_{43} = -M_{43}, \dots, C_{ij} = (-1)^{i+j}M_{ij}.$$

Maka secara matematis, determinan matriks dengan ordo $n \times n$ dapat dihitung dengan ekspansi kofaktor yang ditulis sebagai berikut:

$$\det(A) = \sum_{i=0}^n (-1)^{i+j} |M_{ij}| \text{ atau } |A| = \sum_{i=0}^n a_{ij} C_{ij}$$

2.3 Matriks Balikan

Suatu matriks dikatakan mempunyai matriks balikan (disebut juga invers) jika dan hanya jika matriks tersebut adalah matriks persegi (matriks yang berukuran yang berukuran $n \times n$) dan matriks tersebut non-singular (determinan $\neq 0$). Tidak semua matriks memiliki invers. Invers matriks dapat didefinisikan sebagai berikut. “Jika A adalah suatu matriks kuadrat, dan jika kita adapt mencari matriks B sehingga $AB = BA = I$, maka A dikatakan dapat dibalik (invertible) dan B dinamakan invers dari A .”

Mencari invers suatu matriks dapat dilakukan dengan dua cara, yaitu

1. Menggunakan determinan

Jika A adalah matriks yang mempunyai invers maka

$$A^{-1} = \frac{1}{\det(A)} \text{Adj}(A)$$

Jika matriks kofaktor dari matriks A yang berordo 3 adalah

$$C = \begin{matrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{matrix} \quad \text{dan} \quad C^T = \begin{matrix} C_{11} & C_{21} & C_{31} \\ C_{12} & C_{22} & C_{32} \\ C_{13} & C_{23} & C_{33} \end{matrix}$$

maka C^T adalah $\text{Adj}(A)$ untuk matriks berordo 3.

Jika matriks kofaktor dari matriks A yang berordo 2 adalah

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad \text{dan.} \quad A^T = \begin{bmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \end{bmatrix}$$

Maka C^T adalah $\text{Adj}(A)$ untuk matriks berordo 2.

2. Menggunakan Operasi Baris Elementer

Invers matriks dapat dicari menggunakan operasi baris elementer terhadap matriks $[A|I_n]$.

$$\begin{aligned} [A|I_3] &= \left[\begin{array}{ccc|ccc} a_{11} & a_{12} & a_{13} & 1 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 1 & 0 \\ a_{31} & a_{32} & a_{33} & 0 & 0 & 1 \end{array} \right] \\ [A|I_2] &= \left[\begin{array}{cc|cc} a_{11} & a_{12} & 1 & 0 \\ a_{21} & a_{22} & 0 & 1 \end{array} \right] \end{aligned}$$

2.4 Matriks Kofaktor

Jika A adalah suatu matriks persegi, maka minor dari entri a_{ij} dinyatakan sebagai M_{ij} dan didefinisikan dari submatriks yang tersisa setelah baris ke- i dan kolom ke- j dihilangkan dari A.

Bilangan $(-1)^{i+j} M_{ij}$ dinyatakan sebagai C_{ij} dan disebut kofaktor dari entri a_{ij} .

Contoh:

Misalkan,

$$S = \begin{matrix} 3 & 1 & -4 \\ 2 & 5 & 6 \\ 1 & 4 & 8 \end{matrix}$$

Minor dari entri a_{11} adalah:

$$M_{11} = \begin{vmatrix} 3 & 1 & -4 \\ 2 & 5 & 6 \\ 1 & 4 & 8 \end{vmatrix} = \begin{vmatrix} 5 & 6 \\ 4 & 8 \end{vmatrix} = 16$$

Kofaktor dari entri a_{11} adalah

$$C_{11} = (-1)^{1+1} M_{11} = M_{11} = 16$$

Minor dari entri a_{32} adalah:

$$M_{32} = \begin{vmatrix} 3 & 1 & -4 \\ 2 & 5 & 6 \\ 1 & 4 & 8 \end{vmatrix} = \begin{vmatrix} 5 & -4 \\ 2 & 6 \end{vmatrix} = 26$$

Kofaktor dari entri a_{32} adalah

$$C_{32} = (-1)^{3+2} M_{32} = M_{32} = -26$$

Perhatikan kofaktor dan minor dari suatu elemen a_{ij} hanya berbeda dalam tandanya dimana $C_{ij} = \mp M_{ij}$

Cara untuk menentukan apakah – atau + yang digunakan adalah dengan menentukan fakta bahwa tanda yang berkaitan dengan C_{ij} dan M_{ij} berada dalam baris ke- i dan kolom ke- j dari susunan “papan catur” berikut:

$$\left[\begin{array}{cccc} + & - & + & \dots \\ - & + & - & \dots \\ + & - & + & \dots \\ \vdots & \vdots & \vdots & \ddots \end{array} \right]$$

2.5 Matriks Adjoin

Jika A adalah matriks $n \times n$ sembarang dan C_{ij} adalah kofaktor dari a_{ij} , maka matriks

$$\begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix}$$

disebut matriks kofaktor dari A . Transpose dari matriks ini disebut sebagai adjoin dari A dan dinyatakan sebagai $\text{adj}(A)$.

Contoh:

Misalkan,

$$S = \begin{bmatrix} 3 & 2 & -1 \\ 1 & 6 & 3 \\ 2 & -4 & 0 \end{bmatrix}$$

Kofaktor-kofaktor dari S adalah

$$C_{11} = 12 \quad C_{12} = 6 \quad C_{13} = -16$$

$$C_{21} = 4 \quad C_{22} = 2 \quad C_{23} = 16$$

$$C_{31} = 12 \quad C_{32} = -10 \quad C_{33} = 16$$

Jadi matriks kofaktor adalah

dan adjoin dan adjoin dari S adalah

$$\begin{bmatrix} 12 & 6 & -16 \\ 4 & 2 & 16 \\ 12 & -10 & 16 \end{bmatrix}$$

$$\text{adj}(S) = \begin{bmatrix} 12 & 4 & 12 \\ 6 & 2 & -10 \\ -16 & 16 & 16 \end{bmatrix}$$

2.6 Kaidah Cramer

Jika $AX = B$ adalah sebuah sistem persamaan linier dengan A matriks persegi dan $\det(A) \neq 0$,

sedangkan $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ dan $B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$ maka sistem tersebut mempunyai penyelesaian tunggal (unik).

Untuk mencari penyelesaian menggunakan kaidah Cramer, didasarkan pada determinan matriks A . bila determinan matriks $A \neq 0$ maka akan diperoleh suatu penyelesaian, dan penyelesaian itu adalah:

$$x_1 = \frac{\det(A_1)}{\det(A)}, x_2 = \frac{\det(A_2)}{\det(A)}, x_3 = \frac{\det(A_3)}{\det(A)}, \dots, x_n = \frac{\det(A_n)}{\det(A)}$$

Dengan A_i adalah matriks yang diperoleh dengan menggantikan elemen-elemen kolom ke j dari A dengan elemen-elemen matriks B .

2.7 Interpolasi Polinom

Interpolasi polinomial digunakan untuk mencari titik-titik antara dari n buah titik $P_1(x_1, y_1), P_2(x_2, y_2), P_3(x_3, y_3), \dots, P_n(x_n, y_n)$ dengan menggunakan pendekatan fungsi polinomial pangkat $n-1$:

$$y = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

Masukkan nilai dari setiap titik ke dalam persamaan polinomial di atas dan diperoleh persamaan simultan dengan n persamaan dan n variabel bebas:

$$y_1 = a_0 + a_1x_1 + a_2x_1^2 + a_3x_1^3 + \dots + a_{n-1}x_1^{n-1}$$

$$y_2 = a_0 + a_1x_2 + a_2x_2^2 + a_3x_2^3 + \dots + a_{n-1}x_2^{n-1}$$

$$y_3 = a_0 + a_1x_3 + a_2x_3^2 + a_3x_3^3 + \dots + a_{n-1}x_3^{n-1}$$

.....

$$y_n = a_0 + a_1x_n + a_2x_n^2 + a_3x_n^3 + \dots + a_{n-1}x_n^{n-1}$$

Adapun algoritma untuk mengerjakan soal-soal dari interpolasi polinom, yaitu:

1. Menentukan jumlah titik n yang diketahui
2. Memasukkan titik-titik yang diketahui $P_i = (x_i, y_i)$ untuk $i = 1, 2, 3, \dots, n$

3. Menyusun matriks *augmented* dari titik-titik yang diketahui

$$J = \left[\begin{array}{ccccc|c} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} & y_1 \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} & y_2 \\ 1 & x_3 & x_3^2 & \dots & x_3^{n-1} & y_3 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} & y_n \end{array} \right]$$

4. Menyelesaikan persamaan simultan dengan matriks *augmented* di atas dengan menggunakan metode eliminasi Gauss-Jordan
5. Menyusun koefisien fungsi polynomial berdasarkan penyelesaian persamaan simultan di atas

$$a = \{a_i \mid a_i = J(i, n), 0 \leq i \leq n - 1\}$$

6. Memasukkan nilai x dari titik yang diketahui
7. Menghitung nilai y dari fungsi polynomial yang dihasilkan

$$y = \sum_{i=0}^{n-1} a_i x^i$$

8. Menghasilkan nilai (x, y)

2.8 Regresi Linier Berganda

Selama ini kita telah melihat konsep regresi linier sederhana dimana sebuah prediktor tunggal Variabel X digunakan untuk memodelkan variabel respon Y. Di banyak contoh lain, ada lebih dari satu faktor yang mempengaruhi respon. Model regresi linier berganda dengan demikian menjelaskan bagaimana variabel respon tunggal Y bergantung secara linier pada sejumlah variabel prediktor. Meskipun sudah ada rumus jadi untuk menghitung regresi linier sederhana, terdapat rumus umum dari regresi linear yang bisa digunakan untuk regresi linier berganda, yaitu:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \dots + \beta_k x_{ki} + \varepsilon_i$$

Untuk mendapatkan nilai dari setiap β_i dapat digunakan *Normal Estimation Equation for Multiple Linear Regression* sebagai berikut:

$$\begin{aligned}
 n\beta_0 + \beta_1 \sum_{i=1}^n x_{1i} + \beta_2 \sum_{i=1}^n x_{2i} + \dots + \beta_k \sum_{i=1}^n x_{ki} &= \sum_{i=1}^n y_i \\
 \beta_0 \sum_{i=1}^n x_{1i} + \beta_1 \sum_{i=1}^n x_{1i}^2 + \beta_2 \sum_{i=1}^n x_{1i}x_{2i} + \dots + \beta_k \sum_{i=1}^n x_{1i}x_{ki} &= \sum_{i=1}^n x_{1i}y_i \\
 \vdots &\quad \vdots & \vdots & \vdots \\
 \beta_0 \sum_{i=1}^n x_{ki} + \beta_1 \sum_{i=1}^n x_{ki}x_{1i} + \beta_2 \sum_{i=1}^n x_{ki}x_{2i} + \dots + \beta_k \sum_{i=1}^n x_{ki}^2 &= \sum_{i=1}^n x_{ki}y_i
 \end{aligned}$$

Kemudian, sistem persamaan linier tersebut dapat diselesaikan dengan metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode Matriks Balikan, maupun Kaidah Cramer. Sehingga nantinya persamaan regresi linier berganda diatas dapat digunakan untuk menentukan variable respon tunggal Y yang bergantung secara linier pada sejumlah (i) variable prediktor.

BAB III

IMPLEMENTASI PROGRAM

3.1 Class MatriksInit

```
public void openFile (String namaFile)
// I.S. Sembarang
// F.S. Jika ditemukan, namaFile siap dibaca

public void matrixInitFile ()
// I.S. Sebuah file berisi matriks sudah terbuka
// F.S. Matriks pada file berhasil dipindahkan ke dalam
// MatriksInit.matrix

public void matrixInitKeyboard ()
// I.S. Program siap menerima masukan matriks dari keyboard untuk
// penyelesaian operasi Sistem Persamaan Linier (SPL).
// F.S. this.NbrsEff dan this.NKolEff terdefinisi, this.matrix
// terdefinisi dengan ukuran NBrsEff x NKoleff

public void matrixInitKeyboard2 ()
// I.S. Program siap menerima masukan matriks dari keyboard untuk
// penyelesaian operasi pencarian Determinan dan Matriks Balikan.
// F.S. this.NbrsEff dan this.NbrsKol terdefinisi, this.matrix
// terdefinisi dengan ukuran NBrsEff x NKoleff, dengan
// NbrsEff=NKolEff

public void matrixInitKeyboard3 ()
// I.S. Program siap menerima masukan matriks dari keyboard untuk
// penyelesaian operasi Interpolasi Polinom
// F.S. this.NbrsEff dan this.NbrsKol terdefinisi, this.matrix
// terdefinisi dengan ukuran NBrsEff x NKoleff, dengan NkolEff=2

public void matrixInitKeyboard4 ()
// I.S. Program siap menerima masukan matriks dari keyboard untuk
// penyelesaian operasi Regresi Linier Berganda.
// F.S. this.NbrsEff dan this.NbrsKol terdefinisi, this.matrix
// terdefinisi dengan ukuran NbrsEff x NkolEff
```

```

import java.io.File;
import java.util.Scanner;

public class MatriksInit {
    Scanner scan;

    //Attributes
    int NBrsEff = 0; //Menyimpan jumlah baris yang terisi pada matriks
    int NKolEff = 0; //Menyimpan jumlah kolom yang terisi pada matriks
    double[][] matrix; //Matriks berukuran NBrsEff x NKolEff
    String fileName; //Menyimpan nama file untuk pembacaan matriks dari file

    //Constructor
    MatriksInit(int pilihan){

        //Tampilan pemilihan cara input
        scan = new Scanner(System.in);
        System.out.println ("#=====");
        System.out.println ("# Pilih salah satu cara input di bawah ini: #");
        System.out.println ("#=====");
        System.out.println ("# 1. Baca dari keyboard #");
        System.out.println ("# 2. Baca dari file yang sudah ada #");
        boolean inputSuccess = false;

        while (!inputSuccess) {
            int caraInput = scan.nextInt();
            //Pengguna memilih untuk memasukkan data dari keyboard
            if (caraInput == 1) {
                //Tampilan dan panduan masukan untuk operasi SPL
                if (pilihan==1) {
                    matrixInitKeyboard();
                    System.out.println ("#=====");
                    System.out.println ("# Masukkan elemen-elemen matriks A, dengan urutan: #");
                    System.out.println ("# a11 a12 a13 ... a1m #");
                    System.out.println ("# a21 a22 a23 ... a2m #");
                    System.out.println ("# ... #");
                    System.out.println ("# an1 an2 an3 ... anm #");
                    enterMatrix(scan, this.NBrsEff, this.NKolEff-1);
                    System.out.println ("#=====");
                    System.out.println ("# Masukkan elemen-elemen matriks b, dengan urutan: #");
                    System.out.println ("# b1 #");
                    System.out.println ("# b2 #");
                    System.out.println ("# ... #");
                    System.out.println ("# bn #");
                    enterSolution(scan);
                    inputSuccess = true;
                }
                //Tampilan dan panduan masukan untuk operasi determinan atau matriks balikan
                else if (pilihan==2 || pilihan==3) {
                    matrixInitKeyboard2();
                    System.out.println ("#=====");
                    System.out.println ("# Masukkan elemen-elemen matriks A, dengan urutan: #");
                    System.out.println ("# a11 a12 a13 ... a1m #");
                    System.out.println ("# a21 a22 a23 ... a2m #");
                    System.out.println ("# ... #");
                    System.out.println ("# an1 an2 an3 ... anm #");
                    enterMatrix(scan, this.NBrsEff, this.NKolEff);
                    inputSuccess = true;
                }
            }
        }
    }
}

```

```

        //Tampilan dan panduan masukan untuk operasi determinan atau matriks balikan
    else if (pilihan==2 || pilihan==3) {
        matrixInitKeyboard2();
        System.out.println ("#=====");
        System.out.println ("# Masukkan elemen-elemen matriks A, dengan urutan:      #");
        System.out.println ("# a11 a12 a13 ... a1m                                         #");
        System.out.println ("# a21 a22 a23 ... a2m                                         #");
        System.out.println ("# ...                                              #");
        System.out.println ("# an1 an2 an3 ... anm                                         #");
        enterMatrix(scan, this.NBrsEff, this.NKolEff);
        inputSuccess = true;
    }

    //Tampilan dan panduan masukan untuk operasi interpolasi polinom
    else if (pilihan==4) {
        matrixInitKeyboard3();
        System.out.println ("#=====");
        System.out.println ("# Masukkan pasangan titik x dan y sebanyak n, dengan format:      #");
        System.out.println ("# x1 y1                                         #");
        System.out.println ("# x2 y2                                         #");
        System.out.println ("# ...                                              #");
        System.out.println ("# xn yn                                         #");
        enterMatrix(scan, this.NBrsEff, this.NKolEff);
        inputSuccess = true;
    }

    //Tampilan dan panduan masukan untuk operasi regresi linier berganda
    else if (pilihan==5) {
        matrixInitKeyboard4();
        System.out.println ("#=====");
        System.out.println ("# Masukkan persamaan-persamaannya dalam bentuk:      #");
        System.out.println ("# y1 x11 x12 ... x1n                                         #");
        System.out.println ("# y2 x21 x22 ... x2n                                         #");
        System.out.println ("# ...                                              #");
        System.out.println ("# yn xn1 xn2 ... xnn                                         #");
        enterMatrix(scan, this.NBrsEff, this.NKolEff);
        inputSuccess = true;
    }

    //Pengguna memilih untuk memasukkan data dari file
    else if (caraInput == 2){
        System.out.println ("#=====");
        System.out.println ("# Masukkan nama File berisi matriks dengan format nama_folder/nama_file.txt: #");
        System.out.println ("# Contoh test_case/tc1.txt                                         #");
        Scanner fileNameScanner = new Scanner(System.in); //Pembacaan nama file
        fileName = "../"+fileNameScanner.next();
        openFile(fileName);
        matrixInitFile();
        enterMatrix(scan, this.NBrsEff, this.NKolEff);
        inputSuccess = true;
    } else {
        System.out.println ("# Pilihan tidak valid, silakan coba lagi                                         #");
    }
}

}

```

```

//Open File untuk masukan matriks dari file
public void openFile(String namaFile) {
    try {
        scan = new Scanner(new File(namaFile));
    } catch(Exception e){
        System.out.println("File not found");
    }
}

//Inisiasi matriks dari file
public void matrixInitFile() {
    while(scan.hasNextLine()) {
        this.NBrsEff += 1;
        Scanner bacaKolom = new Scanner(scan.nextLine());
        while(bacaKolom.hasNextDouble()) {
            this.NKolEff += 1;
            bacaKolom.nextDouble();
        }
    }

    this.NKolEff /= this.NBrsEff;
    this.matrix = new double[NBrsEff][NKolEff];
}

//Inisiasi matriks dari keyboard versi 1 - SPL
public void matrixInitKeyboard() {
    scan = new Scanner(System.in);
    System.out.println("=====#");
    System.out.println("# Masukkan jumlah baris matriks: #");
    this.NBrsEff = scan.nextInt(); //Inisiasi jumlah baris matriks
    System.out.println("# Masukkan jumlah kolom matriks: #");
    this.NKolEff = scan.nextInt()+1; //Inisiasi jumlah kolom matriks

    this.matrix = new double[NBrsEff][NKolEff];
}

//Inisiasi matriks dari keyboard versi 2 - determinan, invers
public void matrixInitKeyboard2() {
    scan = new Scanner(System.in);
    System.out.println("# Masukkan n: #");
    this.NBrsEff = scan.nextInt(); //Inisiasi jumlah baris matriks
    this.NKolEff = this.NBrsEff; //Inisiasi jumlah kolom matriks, dengan jumlah kolom = jumlah baris

    this.matrix = new double[NBrsEff][NKolEff];
}

//Inisiasi matriks dari keyboard versi 3 - interpolasi
public void matrixInitKeyboard3() {
    scan = new Scanner(System.in);
    System.out.println("# Masukkan n: #");
    this.NBrsEff = scan.nextInt(); //Inisiasi jumlah baris matriks
    this.NKolEff = 2; //Inisiasi jumlah kolom matriks

    this.matrix = new double[NBrsEff][NKolEff];
}

```

```

//Inisiasi matriks dari keyboard versi 4 - regresi
public void matrixInitKeyboard4() {
    scan = new Scanner(System.in);
    System.out.println("# Masukkan jumlah peubah (n):");
    this.NKolEff = scan.nextInt() + 1; //Inisiasi jumlah kolom matriks
    System.out.println("# Masukkan banyaknya data (i):");
    this.NBrsEff = scan.nextInt(); //Inisiasi jumlah baris matriks

    this.matrix = new double[NBrsEff][NKolEff];
}

//Memasukkan elemen-elemen matriks A
public void enterMatrix (Scanner scan, int n, int m) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            this.matrix[i][j] = scan.nextDouble();
        }
    }
}

//Memasukkan elemen-elemen hasil untuk SPL
public void enterSolution (Scanner scan) {
    for (int i = 0; i < this.NBrsEff; i++) {
        this.matrix[i][this.NKolEff - 1] = scan.nextDouble();
    }
}

//Mengubah MatriksInit ke Matriks
public void toMatriks(Matriks M) {
    for (int i = 0; i < this.NBrsEff; i++) {
        for (int j = 0; j < this.NKolEff; j++) {
            M.Elmt[i][j] = this.matrix[i][j];
        }
    }
}

```

3.2 Class Matriks

3.2.1 SPLGauss

```
public void SPLGauss ()
// I.S. SPL dalam bentuk matriks augmented (M)
// F.S. Solusi SPL dituliskan pada layar atau dalam sebuah file
// Proses: Mengubah matriks M ke dalam bentuk matriks eselon,
// kemudian melakukan penyulihan mundur untuk mendapatkan solusi
// SPL.

public void EliminasiGauss ()
// I.S. Matriks (M) terdefinisi dan sembarang
// F.S. Matriks (M) merupakan matriks eselon baris

public int getLastIdxBrs ()
// Mengembalikan indeks baris terakhir matriks

public int getLastIdxKol ()
// Mengembalikan indeks kolom terakhir matriks

public void TukarBrs (int idxBrs1, idxBrs2)
// I.S. Matriks (M) terdefinisi dan sembarang, idxBrs1 dan idxBrs2
// dalam range ukuran baris M
// F.S. Baris ke idxBrs1 dan baris ke idxBrs2 bertukar posisi

public boolean IsKolZeroFromCr (int cr, cc)
// Mengembalikan apakah sebuah kolom (cc) bernilai nol semua dari
// sebuah indeks baris (cr). Mengembalikan true jika benar dan
// false jika salah.

public boolean IsBrsAllZero (int Idx)
// Mengembalikan apakah sebuah baris bernilai nol semua

public boolean IsKolAllZero (int Idx)
// Mengembalikan apakah sebuah kolom bernilai nol semua

public void cleanMatriks (Matriks M, double tolerance)
// I.S. Matriks (M) terdefinisi dan sembarang
// F.S. Semua elemen M yang lebih kecil daripada tolerance akan
// diubah menjadi nol.
```

```

//EliminasiGauss
public void EliminasiGauss() {
    int cr = 0;
    int cc = 0;
    while (cr < this.NBrsEff && cc < this.NKolEff) {
        boolean colZeroFound = IsKolZeroFromCr(cr,cc);
        boolean rowZeroFound = false;

        if (colZeroFound) {
            cc++;
        } else {
            int z = cr;
            while (!rowZeroFound && z<this.NBrsEff) {
                rowZeroFound = IsBrsAllZero(z);
                if (!rowZeroFound) {
                    z++;
                }
            }
            if (rowZeroFound) {
                TukarBrs(z,this.NBrsEff-1);

            }
            //cari baris dengan this.Elmt[i][k] paling besar
            int currentP = cr;
            for (int i = cr + 1;i < this.NBrsEff; i++) {
                if (Math.abs(this.Elmt[i][cr]) > Math.abs(this.Elmt[currentP][cr])) {
                    currentP = i;
                }
            }

            //tukar baris
            TukarBrs(cr, currentP);

            //bagi baris ke k agar elemen A[k][k] = 1;
            double divider = this.Elmt[cr][cc];
            for(int i = cc; i < this.NKolEff; i++) {
                this.Elmt[cr][i] /= divider;
            }
            cleanMatriks(this, 1e-9);

            //0in nilai di bawah 1 utama currentP
            for (int i = cr + 1; i < this.NBrsEff; i++) {
                double x = -(this.Elmt[i][cc]/this.Elmt[cr][cc]);
                for (int j = cc; j < this.NKolEff; j++) {
                    this.Elmt[i][j] += this.Elmt[cr][j] * x;
                }
            }
            cleanMatriks(this, 1e-9);

            cc++;
            cr++;
        }
    }
}

```

```

//SPL Gauss
public void SPLGauss() throws IOException {
    DecimalFormat df = new DecimalFormat("#.##");
    int brsNotZero = 0;
    int cr = this.getLastIdxBrs();
    boolean hasSolution = true; //Variabel penanda apakah solusi SPL ada

    EliminasiGauss(); //Mengubah matriks ke bentuk matriks eselon
    while (hasSolution && cr>=0) {
        int cc = 0;
        boolean rowAllZero = true;

        //Proses pengecekan apakah elemen sebuah baris pada matriks A bernilai nol semua
        for (int j=0; j<this.getLastIdxKol()-1;j++) {
            if (this.Elmt[cr][j]!=0) {
                rowAllZero = false;
            }
        }

        //Mengecek elemen pada matriks b yang bersangkutan apabila ditemukan sebuah baris nol
        if (rowAllZero) {
            if (this.Elmt[cr][this.getLastIdxKol()]!=0) {
                hasSolution = false; //Deklarasi bahwa SPL tidak memiliki solusi
            }
        } else {
            brsNotZero++;
        }

        cr--;
    }

    //Kasus 1: Tidak mempunyai solusi
    if (!hasSolution) {
        String printMode = this.TulisSPLGauss();

        if (printMode.equals("2")) {
            Scanner scan = new Scanner(System.in);
            System.out.println("Masukkan nama File solusi beserta direktori dengan format nama_folder/nama_file.txt: ");
            System.out.println("Contoh: solutions/SolusiSPL.txt");
            String namafайл = "../"+scan.nextLine();

            //Tulis hasil ke file
            try (FileOutputStream file = new FileOutputStream(namafайл)) {
                byte[] b;
                String s ="SPL ini tidak memiliki solusi.\n";
                b = s.getBytes();
                file.write(b);
            }
        }
    }

    //Tulis hasil ke layar
    System.out.println("SPL ini tidak memiliki solusi.");
}

```

```

} else {
    //Kasus 2: Solusi banyak
    if (brsNotZero<this.NKolEff-1) {
        //Deklarasi array tambahan, dengan spesifikasi:
        //Array 2D dengan ukuran n x m (n: Jumlah variabel, jumlah persamaan efektif setelah eliminasi + 4)
        //4 kolumn di akhir array solusi dibuat untuk menyimpan data dengan ketentuan:
        //Kolom 1: penanda apakah sebuah variabel dijadikan variabel parametrik atau tidak (1 jika ya, 0 jika tidak)
        //Kolom 2: penanda apakah sebuah variabel memiliki solusi tunggal (1 jika ya, 0 jika tidak)
        //Kolom 3: index variabel parametrik (variabel parametrik ke berapa), jika kolom 1 bernilai 1
        //Kolom 4: nilai solusi tunggal suatu variabel apabila kolom 2 bernilai 1
        double[][] solution = new double[this.NKolEff-1][this.NKolEff-brsNotZero+3];
        //Array 1D berukuran jumlah variabel matriks, menjadi penanda apakah solusinya sudah dideklarasikan atau belum
        boolean[] solutionDeclared = new boolean[this.NKolEff-1];
        for (int i=0;i<this.getLastIdxKol()-1;i++) {
            solutionDeclared[i] = false; //Inisiasi seluruh nilai solution declared
        }

        //Menentukan variabel mana saja yang akan dijadikan variabel parametrik
        int cpar = 0; //Variabel yang menyimpan jumlah variabel yang harus dijadikan variabel parametrik
        int cr2 = brsNotZero-1;
        int cc2 = this.getLastIdxKol()-1;

        //Mencari kolumn nol untuk kemudian variabel terkait langsung dijadikan variabel parametrik
        for (int j=0;j<this.getLastIdxKol()-1;j++) {
            if (this.IsKolAllZero(j)) {
                solutionDeclared[j] = true;
                solution[j][this.NKolEff-brsNotZero-1] = 1;
                solution[j][this.NKolEff-brsNotZero+1] = cpar;
                cpar++;
            }
        }

        //Menentukan variabel lain yang akan dijadikan variabel parametrik
        while(cpar < this.NKolEff-1-brsNotZero) {
            cc2 = this.getLastIdxKol()-1;
            while(cc2>=0 && cpar < this.NKolEff-1-brsNotZero) {
                if (solutionDeclared[cc2]==false && this.Elmt[cr2][cc2]!=0) {
                    //variabel tidak akan dijadikan variabel parametrik apabila ia memiliki solusi unik atau bisa
                    //dideklarasikan dengan variabel parametrik lainnya
                    if(ItemsInRow(cr2,this.getLastIdxKol()-1)==1){

                        solutionDeclared[cc2]=true;
                        solution[cc2][this.NKolEff-brsNotZero] = 1;
                        solution[cc2][this.NKolEff-brsNotZero+2] = this.Elmt[cr2][this.getLastIdxKol()]/this.Elmt[cr2][cc2];

                    }
                    //sebuah variabel dideklarasikan sebagai variabel parametrik
                    else if(cc2!=Kolom1Utama(cr2) && IsKolZeroFromCr(cr2+1,cc2)) {
                        solutionDeclared[cc2]=true;
                        solution[cc2][this.NKolEff-brsNotZero-1] = 1;
                        solution[cc2][this.NKolEff-brsNotZero+1] = cpar;
                        cpar++;
                    }
                }
                cc2--;
            }
            cr2--;
        }
    }
}

```

```

//Mengisi array solusi sesuai dengan jenis solusi setiap variabelnya
cr2 = brsNotZero-1;
cc2 = 0;
for (int i=cr2;i>=0;i--) {
    cc2 = Kolom1Utama(i);
    if(solutionDeclared[cc2]==false) {
        solutionDeclared[cc2]=true;
        solution[cc2][this.NKolEff-brsNotZero+2] = this.Elmt[i][this.getLastIdxKol()];
        for (int k=this.getLastIdxKol()-1;k>=cc2+1;k--) {
            //Jenis solusi 1: bukan merupakan variabel parametrik ataupun solusi tunggal
            if(solution[k][this.NKolEff-brsNotZero-1]==0 && solution[k][this.NKolEff-brsNotZero]==0) {
                for (int j=0;j<cpars-1;j++) {
                    solution[cc2][j] += solution[k][j]*-(this.Elmt[i][k]);
                }
                solution[cc2][this.NKolEff-brsNotZero+2] += solution[k][this.NKolEff-brsNotZero+2]*-(this.Elmt[i][k]);
            }
            //Jenis solusi 2: Merupakan variabel parametrik
            else if (solution[k][this.NKolEff-brsNotZero-1]==1 && solution[k][this.NKolEff-brsNotZero]==0) {
                solution[cc2][(int)solution[k][this.NKolEff-brsNotZero+1]] += this.Elmt[i][k];
            }
            //Jenis solusi 3: Memiliki solusi tunggal
            else {
                solution[cc2][this.NKolEff-brsNotZero+2] -= solution[k][this.NKolEff-brsNotZero+2]*this.Elmt[i][k];
            }
        }
    }
}

//Menuliskan hasil
String[] variables = {"r","s","t","u","v","w","x","y","z"};
String printMode = this.TulisSPLGauss();

//Tulis hasil ke file
if (printMode.equals("2")) {
    Scanner scan = new Scanner(System.in);
    System.out.println("Masukkan nama File solusi beserta direktori dengan format nama_folder/nama_file.txt: ");
    System.out.println("Contoh: solutions/SolusiSPL.txt");
    String namafile = "../"+scan.nextLine();

    try (FileOutputStream file = new FileOutputStream(namafile)) {
        byte[] b;
        String s = ("SPL ini memiliki solusi banyak, yang mengikuti:\n");
        b = s.getBytes();
        file.write(b);
        for (int i = this.IdxBrsMin; i <= this.getLastIdxKol()-1; i++) {
            if(solution[i][this.NKolEff-brsNotZero-1]==0 && solution[i][this.NKolEff-brsNotZero]==0) {
                s = ("x"+(i+1)+" = ");
                b = s.getBytes();
                file.write(b);
                if (solution[i][this.NKolEff-brsNotZero+2]!=0) {
                    s = (df.format(solution[i][this.NKolEff-brsNotZero+2]));
                    b = s.getBytes();
                    file.write(b);
                }
            }
        }
    }
}

```

```

    }
    //Kasus 3: Solusi unik
    else {
        String printMode = this.TulisSPLGauss();

        //Menyimpan solusi masing-masing variabel pada sebuah array
        double[] solution = new double[this.NKolEff];
        for (int i = this.getLastIdxKol()-1; i >= 0;i--) {
            double sum = 0.0;
            for (int j = i + 1; j < this.getLastIdxKol(); j++)
                sum += this.Elmt[i][j] * solution[j];
            solution[i] = (this.Elmt[i][this.getLastIdxKol()] - sum);
        }

        //Tulis hasil ke file
        if (printMode.equals("2")) {
            Scanner scan = new Scanner(System.in);
            System.out.println("Masukkan nama File solusi beserta direktori dengan format nama_folder/nama_file.txt: ");
            System.out.println("Contoh: solutions/SolusiSPL.txt");
            String namafile = "../"+scan.nextLine();

            try (FileOutputStream file = new FileOutputStream(namafile)) {
                byte[] b;
                String s =("SPL ini memiliki solusi unik, yaitu:\n");
                b = s.getBytes();
                file.write(b);
                for (int i=0; i<this.getLastIdxKol()-1;i++) {
                    s = ("x"+(i+1)+" = "+ df.format(solution[i])+"\n");
                    b = s.getBytes();
                    file.write(b);
                }
            }
        }

        //Tulis hasil ke layar
        System.out.println("SPL ini memiliki solusi unik, yaitu:");
        for (int i=0; i<this.getLastIdxKol()-1;i++) {
            System.out.print("x"+(i+1)+" = "+ df.format(solution[i])+"\n");
        }
    }
}

```

3.2.2 SPLGaussJordan

```
public void SPLGaussJordan ()  
// I.S. SPL dalam bentuk matriks augmented (M)  
// F.S. Solusi SPL dituliskan pada layar atau dalam sebuah file  
// Proses: Mengubah matriks M ke dalam bentuk matriks eselon  
// tereduksi, kemudian melakukan penyulihan mundur untuk  
// mendapatkan solusi SPL.  
  
public void EliminasiGaussJordan ()  
// I.S. Matriks (M) terdefinisi dan sembarang  
// F.S. Matriks (M) merupakan matriks eselon baris tereduksi  
  
public void EliminasiGauss ()  
// I.S. Matriks (M) terdefinisi dan sembarang  
// F.S. Matriks (M) merupakan matriks eselon baris  
  
public int getLastIdxBrs ()  
// Mengembalikan indeks baris terakhir matriks  
  
public int getLastIdxKol ()  
// Mengembalikan indeks kolom terakhir matriks  
  
public void TukarBrs (int idxBrs1, idxBrs2)  
// I.S. Matriks (M) terdefinisi dan sembarang, idxBrs1 dan idxBrs2  
// dalam range ukuran baris M  
// F.S. Baris ke idxBrs1 dan baris ke idxBrs2 bertukar posisi  
  
public boolean IsKolZeroFromCr (int cr, cc)  
// Mengembalikan apakah sebuah kolom (cc) bernilai nol semua dari  
// sebuah indeks baris (cr). Mengembalikan true jika benar dan  
// false jika salah.  
  
public boolean IsBrsAllZero (int Idx)  
// Mengembalikan apakah sebuah baris bernilai nol semua  
  
public boolean IsKolAllZero (int Idx)  
// Mengembalikan apakah sebuah kolom bernilai nol semua  
  
public void cleanMatriks (Matriks M, double tolerance)  
// I.S. Matriks (M) terdefinisi dan sembarang  
// F.S. Semua elemen M yang lebih kecil daripada tolerance akan  
// diubah menjadi nol.
```

```

//EliminasiGaussJordan
public void EliminasiGaussJordan() {

    EliminasiGauss();
    for (int i=this.NBrsEff-1;i>0;i--) {
        boolean jFound = false;
        int j = 0;
        while(!jFound && j<=this.getLastIdxKol()) {
            if (this.Elmt[i][j]==1) {
                //0in nilai di atas 1 utama
                for (int a = i-1; a>=0; a--) {
                    double x = -(this.Elmt[a][j]/this.Elmt[1][j]);
                    for (int b = 0; b < this.NKolEff; b++) {
                        this.Elmt[a][b] += this.Elmt[i][b] * x;
                    }
                }
                jFound = true;
            }
            j++;
        }
    }
}

```

```

//SPL Gauss Jordan
public void SPLGaussJordan() throws IOException {
    DecimalFormat df = new DecimalFormat("#.##");
    int brsNotZero = 0;
    int cr = this.getLastIdxBrs();
    boolean hasSolution = true; //Variabel penanda apakah solusi SPL ada

    EliminasiGaussJordan(); //Mengubah matriks ke bentuk matriks eselon tereduksi
    while (hasSolution && cr>=0) {
        int cc = 0;
        boolean rowAllZero = true;

        //Proses pengecekan apakah elemen sebuah baris pada matriks A bernilai nol semua
        for (int j=0; j<=this.getLastIdxKol()-1;j++) {
            if (this.Elmt[cr][j]!=0) {
                rowAllZero = false;
            }
        }

        //Mengecek elemen pada matriks b yang bersangkutan apabila ditemukan sebuah baris nol
        if (rowAllZero) {
            if (this.Elmt[cr][this.getLastIdxKol()]!=0) {
                hasSolution = false; //Deklarasi bahwa SPL tidak memiliki solusi
            }
        } else {
            brsNotZero++;
        }

        cr--;
    }

    //Kasus 1: Tidak mempunyai solusi
    if (!hasSolution) {
        String printMode = this.TulisSPLGauss();

        //Tulis ke sebuah file
        if (printMode.equals("2")) {
            Scanner scan = new Scanner(System.in);
            System.out.println("Masukkan nama File solusi beserta direktori dengan format nama_folder/nama_file.txt: ");
            System.out.println("Contoh: solutions/SolusiSPL.txt");
            String namafile = "../"+scan.nextLine();

            try (FileOutputStream file = new FileOutputStream(namafile)) {
                byte[] b;
                String s =("SPL ini tidak memiliki solusi.\n");
                b = s.getBytes();
                file.write(b);
            }
        }

        //Tulis ke layar
        System.out.println("SPL ini tidak memiliki solusi.");
    }
}

```

*Selebihnya kode sama dengan yang digunakan pada metode SPLGauss
IF2123

3.2.3 SPLCramer

```
public double[] SPLCramer (boolean isMLR)
// I.S. SPL dalam bentuk matriks augmented (M)
// F.S. Jika isMLR = True maka fungsi hanya mengembalikan array
// of double yang berisi solusi SPL, jika isMLR = false fungsi
// dapat menampilkan solusi SPL ke layer atau menyimpan ke dalam
// file serta tetap mengembalikan array of double yang berisi
// solusi SPL tersebut
// Proses: Menghitung determinan matriks utama dengan ekspansi
// kofaktor, kemudian menukar kolom ke-i dengan kolom terakhir
(i
// = 0..getLastIdxKol-1) kemudian menghitung determinan matriks
// minor tersebut dan mendapatkan solusi SPL tergantung dari
// hasil perhitungan determinan-determinan tersebut.

public int getLastIdxKol ()
// Mengembalikan indeks kolom terakhir matriks

public double DeterminanDenganKofaktor ()
// I.S. Sebuah matriks (M) berukuran n x n terdefinisi dan
// sembarang.
// F.S. Mengembalikan hasil perhitungan determinan matriks M
// dengan ekspansi kofaktor

public void TukarKolCramer (int IdxKol1, int IdxKol2, Matriks
mx)
// I.S. Matriks (M) terdefinisi dan sembarang, Matriks mx
// berukuran n x n terdefinisi dan sembarang, IdxKol1 dalam
// range ukuran kolom mx dan IdxKol2 dalam range ukuran kolom M
// F.S. Kolom ke IdxKol1 pada Matriks mx dan
// Kolom ke IdxKol2 pada Matriks M bertukar posisi

public static void TulisSPLCramer (double detUtama, double[]
detminor)
// I.S. Determinan utama dari SPL matriks augmented dan
determinan
// minor dari SPL matriks augmented yang sudah bertukar kolom
// terdefinisi
// F.S. Menulis solusi SPL yang dapat ditampilkan ke layar
// atau disimpan ke dalam file dengan format .txt
```

```

//SPL CRAMER
//SPL CRAMER Dapat mengembalikan array dari detMinor/detUtama untuk digunakan dalam Multiple Linear Regression
public double[] SPLCramer(boolean isMLR) {
    // Diasumsikan SPL yang diterima hanya SPL dengan n peubah dan n persamaan
    DecimalFormat df = new DecimalFormat("#.##");
    int n = this.NBrsEff;

    // Membuat Matriks Utama yang berisi koefisien peubah pada n persamaan
    Matriks utama = new Matriks(n, n);
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            utama.Elm[i][j] = this.Elm[i][j];
        }
    }
    //Menghitung determinan matriks utama
    double detUtama = utama.DeterminanDenganKofaktor();

    //Menghitung determinan tiap matriks minor (yang kolomnya sudah ditukar)
    double[] detMinor = new double[n];
    for(int i=0; i<n; i++){
        // Menukar Kolom i dengan Kolom terakhir pada matriks utama
        this.TukarKolCramer(i, this.getLastIdxKol(), utama);
        detMinor[i] = utama.DeterminanDenganKofaktor();
        // Mengembalikan kolom i ke kolom terakhir pada matriks utama
        this.TukarKolCramer(i, this.getLastIdxKol(), utama);
    }

    if(!isMLR){
        // Untuk SPL Biasa
        try {
            // Menulis Hasil SPL dengan Kaidah Cramer
            TulisSPLCramer(detUtama, detMinor);
        }catch (IOException e) {
            // Do Nothing
        }
    }

    //Khusus Untuk Multiple Linear Regression
    double[] solusi = new double[n];
    for(int i=0; i<n; i++){
        solusi[i] = detMinor[i]/detUtama;
    }

    return solusi;
}

//TukarKolCramer
public void TukarKolCramer(int IdxKol1, int IdxKol2, Matriks mx){
    //procedure ini digunakan untuk menukar kolom dengan index IdxKol1 dengan kolom dengan IdxKol2
    double tmp;
    for (int i = this.IdxBrsMin; i <= this.getLastIdxBrs(); i++){
        //Menukar Elmt[i][IdxKol1] dengan Elmt[i][IdxKol2]
        tmp = mx.Elm[i][IdxKol1];
        mx.Elm[i][IdxKol1] = this.Elm[i][IdxKol2];
        this.Elm[i][IdxKol2] = tmp;
    }
}

```

```

public static void TulisSPLCramer(double detUtama, double[] detMinor) throws IOException{
    DecimalFormat df= new DecimalFormat("#.##");
    int n = detMinor.length;
    Scanner scan = new Scanner(System.in);
    System.out.println ("#-----#");
    System.out.println ("# PENULISAN SOLUSI SPL DENGAN KAIDAH CRAMER #");
    System.out.println ("#-----#");
    System.out.println ("# Silakan pilih salah satu dari pilihan berikut! #");
    System.out.println ("#-----#");
    System.out.println ("# 1. Tampilkan pada layar #");
    System.out.println ("# 2. Simpan dalam file #");
    System.out.println ("#-----#");
    System.out.println ("# Ketik '1' atau '2' pada keyboard: #");

    // Menerima pilihan user untuk penulisan solusi
    int pilihan = scan.nextInt();
    while (pilihan<1 || pilihan>2){
        System.out.println("Masukan Anda salah. Silakan ulangi kembali.");
        pilihan = scan.nextInt();
    }

    // Menampilkan hasil SPL ke Layar
    if (pilihan==1) {
        if(detUtama==0){
            // Jika determinan utama = 0, maka solusi SPL antara tidak memiliki solusi atau memiliki banyak solusi
            boolean isAllDetZero = true;
            int i=0;
            // Pengecekan determinan matriks minor untuk tiap peubah
            while(i<n && isAllDetZero){
                if(detMinor[i]!=0){
                    isAllDetZero = false;
                }else{
                    i++;
                }
            }
            if(isAllDetZero){
                // Jika semua determinan matriks minor = 0
                System.out.println("Solusi SPL tidak dapat dicari dengan metode cramer, karena SPL ini memiliki banyak solusi.");
            } else{
                // Jika ada determinan matriks minor != 0
                System.out.println("SPL ini tidak memiliki solusi.");
            }
        } else{
            // Jika determinan utama != 0
            System.out.println("SPL ini memiliki solusi unik, yaitu:");
            for (int i=0; i<n;i++) {
                //Menampilkan hasil bagi antara determinan minor dengan determinan utama sebagai solusi SPL untuk tiap variabel peubah
                System.out.print("x"+(i+1)+" = "+ df.format(detMinor[i]/detUtama)+"\n");
            }
        }
    } else {
        // Simpan dalam file
        Scanner sc = new Scanner(System.in);
        System.out.println("Masukkan nama File solusi beserta direktori dengan format nama_folder/nama_file.txt: ");
        System.out.println("Contoh: solutions/SolusiSPL.txt");
        String namafile = "../"+sc.nextLine();

        try (FileOutputStream file = new FileOutputStream(namafile)) {
            byte[] b;
            if(detUtama==0){
                // Jika determinan utama = 0, maka solusi SPL antara tidak memiliki solusi atau memiliki banyak solusi
                boolean isAllDetZero = true;
                int i=0;
                // Pengecekan determinan matriks minor untuk tiap peubah
                while(i<n && isAllDetZero){
                    if(detMinor[i]!=0){
                        isAllDetZero = false;
                    }else{
                        i++;
                    }
                }
            }
        }
    }
}

```

```

        }
    }

    if(isAllDetZero){
        // Jika semua determinan matriks minor = 0
        String s = "Solusi SPL tidak dapat dicari dengan metode cramer, karena SPL ini memiliki banyak solusi.\n";
        System.out.print(s);
        b = s.getBytes();
        file.write(b);
        System.out.println("Hasil perhitungan SPL dengan kaidah cramer berhasil disimpan kedalam file");
    } else{
        // Jika ada determinan matriks minor != 0
        String s = "SPL ini memiliki solusi unik, yaitu:";

        for (int i = 0; i <n; i++) {
            //Menampilkan hasil bagi antara determinan minor dengan determinan utama sebagai solusi SPL untuk tiap variabel peubah
            // Menyimpan solusi SPL kedalam file
            String s = ("x" + (i + 1) + " = " + df.format(detMinor[i]/detUtama) + "\n");
            System.out.print(s);
            b = s.getBytes();
            file.write(b);
        }
        System.out.println("Hasil perhitungan SPL dengan kaidah cramer berhasil disimpan kedalam file");
    }
}
}

```

3.2.4 DeterminanOBE

```
public void DeterminanOBE ()  
// I.S. Sebuah matriks (M) berukuran n x n terdefinisi dan  
// sembarang.  
// F.S. Determinan M dituliskan pada layar atau dalam sebuah file  
// Proses: Mengubah matriks M ke dalam bentuk matriks segitiga  
// kemudian mengalikan semua elemen pada diagonal utamanya.  
  
public double GetElmtDiagonal (int i)  
// Mengembalikan elemen matriks (M) dengan indeks baris i dan  
// indeks kolom i: M[i][i]  
  
public void TukarBrs (int idxBrs1, idxBrs2)  
// I.S. Matriks (M) terdefinisi dan sembarang, idxBrs1 dan idxBrs2  
// dalam range ukuran baris M  
// F.S. Baris ke idxBrs1 dan baris ke idxBrs2 bertukar posisi  
  
public void cleanMatriks (Matriks M, double tolerance)  
// I.S. Matriks (M) terdefinisi dan sembarang  
// F.S. Semua elemen M yang lebih kecil daripada tolerance akan  
// diubah menjadi nol.
```

```
//DETERMINAN OBE  
public double DeterminanOBE() {  
    int tukarBaris = 0; //Variabel untuk menghitung jumlah pertukaran baris yang terjadi  
    double hasil = 1;  
  
    for (int k = 0; k < this.NBrsEff; k++) {  
        //Mencari baris dengan this.Elmt[i][k] paling besar untuk dipindahkan ke paling atas  
        int currentP = k;  
        for (int i = k + 1; i < this.NBrsEff; i++) {  
            if (Math.abs(this.Elmt[i][k]) > Math.abs(this.Elmt[currentP][k])) {  
                currentP = i;  
            }  
        }  
  
        //Menukar baris sehingga baris dengan this.Elmt[i][k] terbesar dari i akan menjadi berada pada posisi i  
        if (currentP != k) {  
            tukarBaris++; //Increment tukarBaris  
            TukarBrs(k, currentP);  
        }  
  
        cleanMatriks(this, 1e-9); //Membersihkan matriks  
  
        //Menjadikan 0 semua elemen di bawah 1 utama currentP  
        for (int i = k + 1; i < this.NBrsEff; i++) {  
            double x = -(this.Elmt[i][k]/this.Elmt[k][k]); //Faktor pengali  
            for (int j = k; j < this.NKolEff; j++) {  
                this.Elmt[i][j] += this.Elmt[k][j] * x;  
            }  
        }  
  
        //Perhitungan determinan  
        for (int i=0; i<this.NBrsEff; i++) {  
            hasil *= GetElmtDiagonal(i);  
        }  
        hasil *= Math.pow(-1, tukarBaris); //Mengalikan hasil dengan -1 dipangkatkan jumlah pertukaran baris  
    }  
}
```

3.2.5 DeterminanDenganKofaktor

```
public double DeterminanDenganKofaktor ()  
// I.S. Sebuah matriks (M) berukuran n x n terdefinisi dan  
// sembarang.  
// F.S. Mengembalikan hasil perhitungan determinan matriks M  
// dengan ekspansi kofaktor  
// Proses: Melakukan perhitungan secara rekursif dari ukuran  
// matriks n x n hingga ukuran matriks mencapai 2x2, 1x1, atau 0  
// (basis).
```

```
//Determinan Kofaktor  
public double DeterminanDenganKofaktor() {  
    // Diasumsikan matriks persegi  
    int n = this.NBrsEff;  
  
    if (n<=0) {  
        // Basis  
        // Jika ukuran matriks tidak terdefinisi  
        return 0;  
    } else if (n==1){  
        // Basis  
        // Jika matriks memiliki ukuran 1x1  
        return this.Elmt[0][0];  
    } else if (n==2){  
        // Basis  
        // Jika matriks memiliki ukuran 2x2  
        return (this.Elmt[0][0] * this.Elmt[1][1]) - (this.Elmt[1][0]*this.Elmt[0][1]);  
    } else{  
        // Rekurens  
        // Jika matriks memiliki ukuran >= 3x3  
        Matriks minor = new Matriks((n-1),(n-1)); //inisialisasi matriks minor  
        int i, aj, bj, k; //indeks yang akan digunakan  
        int sign = 1;  
        double det = 0;  
  
        //Lakukan perhitungan matriks kofaktor  
        for(k=0; k<n ; k++){  
            for (i=1 ; i<n ; i++){  
                bj = 0;  
                for (aj=0; aj<n ; aj++){  
                    if (aj!=k){  
                        minor.Elmt[i-1][bj] = this.Elmt[i][aj];  
                        ++bj;  
                    }  
                }  
                det = det + (sign*this.Elmt[0][k]*minor.DeterminanDenganKofaktor());  
                // Merubah Nilai sign menjadi -sign  
                sign = -1*sign;  
            }  
            return det;  
        }  
    }  
    //Referensi : https://stackoverflow.com/questions/42802208/code-for-determinant-of-n-x-n-matrix  
}
```

3.2.6 SolusiInterpolasi

```
void MakeMatriksInterpolasi(int NBrs)
/* I.S. Matriks NBrs x 2 terdefinisi dari masukan pengguna
 F.S. Mengubah Matriks menjadi matriks yang siap diinterpolasi
 */

public void SolusiInterpolasi() throws IOException
/* I.S. Matriks interpolasi terdefinisi
 F.S. Menghitung interpolasi polinomial dan solusi yang
 dihasilkan akan ditampilkan ke layar atau disimpan ke
 dalam file .txt. Jika matriks tidak memiliki solusi interpolasi
 polinomial, akan ditampilkan "Polinom Interpolasi tidak dapat
 dihitung". */

public void TulisInterpolasi(boolean hasSolution, int brsNotZero)
/* I.S. Matriks terdefinisi dalam bentuk matriks eselon baris
 tereduksi
 F.S. Menampilkan hasil interpolasi ke layar */

public void TulisInterpolasiKeFile(boolean hasSolution, int brsNotZero) throws IOException
/* I.S. Matriks terdefinisi dalam bentuk matriks eselon baris
 tereduksi
 F.S. Menyimpan hasil interpolasi ke dalam file dan juga
 menampilkannya di layar */
public double Taksiran(double x)
/* I.S. Hasil interpolasi polinom terdefinisi dalam sebuah matriks
 F.S. Mengembalikan hasil substitusi nilai x ke dalam polinom */
```

```
void MakeMatriksInterpolasi(int NBrs) {
    /* I.S. Matriks NBrs x 2 terdefinisi dari masukan pengguna
       F.S. Mengubah Matriks menjadi matriks yang siap diinterpolasi */

    this.NBrsEff = NBrs;
    this.NKolEff = NBrs + 1;

    for (int i = this.IdxBrsMin; i <= this.getLastIdxBrs(); i++) {
        double x = this.Elmt[i][this.IdxBrsMin];
        double y = this.Elmt[i][1];
        for (int j=this.IdxKolMin;j<=this.getLastIdxKol();j++) {
            if (j != getLastIdxKol()) {
                this.Elmt[i][j] = Math.pow(x, j);
            } else {
                this.Elmt[i][j] = y;
            }
        }
    }
}

public void SolusiInterpolasi() throws IOException {
    /* I.S. Matriks interpolasi terdefinisi
       F.S. Menghitung interpolasi polinomial dan solusi yang dihasilkan akan ditampilkan ke
       layar atau disimpan
          ke dalam file .txt. Jika matriks tidak memiliki solusi interpolasi polinomial,
       akan ditampilkan
          "Polinom Interpolasi tidak dapat dihitung". */

    // Inisialisasi berbagai variabel yang dibutuhkan
```

```

Scanner scan = new Scanner(System.in);
int brsNotZero = 0;
int cr = this.getLastIdxBrs();
boolean hasSolution = true;

// Eliminasi Gauss-Jordan untuk mendapat matriks eselon baris tereduksi
this.EliminasiGaussJordan();
while (hasSolution && cr >= 0) {
    int cc = 0;
    boolean rowAllZero = true;

    for (int j = 0; j <= this.getLastIdxKol() - 1; j++) {
        if (this.Elmt[cr][j] != 0) {
            rowAllZero = false;
        }
    }

    if (rowAllZero) {
        if (this.Elmt[cr][this.getLastIdxKol()] != 0) {
            hasSolution = false;
        }
    } else {
        brsNotZero++;
    }

    cr--;
}

// Penulisan solusi ke layar atau disimpan ke file
System.out.println
("#####
");
System.out.println ("# PENULISAN SOLUSI INTERPOLASI
#");
System.out.println ("#-----#
-----#");
System.out.println ("# Silakan pilih salah satu dari pilihan berikut!
#");
System.out.println
("#####
");
System.out.println ("# 1. Tampilkan pada layar
#");
System.out.println ("# 2. Simpan dalam file
#");
System.out.println
("#####
");
System.out.println ("# Ketik '1' atau '2' pada keyboard:
#");

String pilihan = scan.nextLine();
while (!pilihan.equals("1") && !pilihan.equals("2")) {
    System.out.println("Masukan Anda salah. Silakan ulangi kembali.");
    pilihan = scan.nextLine();
}

if (pilihan.equals("1")){
    this.TulisInterpolasi(hasSolution, brsNotZero);
} else {
    this.TulisInterpolasiKeFile(hasSolution, brsNotZero);
}
}

public void TulisInterpolasi(boolean hasSolution, int brsNotZero) {
    /* I.S. Matriks terdefinisi dalam bentuk matriks eselon baris tereduksi
       F.S. Menampilkan hasil interpolasi ke layar */

    // Format penulisan desimal
    DecimalFormat df = new DecimalFormat("###,###,###");
}

```

```

// KASUS 1: SPL tidak memiliki solusi
if (!hasSolution) {
    System.out.println("Polinom Interpolasi tidak dapat dihitung");
} else {
    // KASUS 2: SPL memiliki solusi banyak
    if (brsNotZero < this.NKolEff - 1) {
        System.out.println("Polinom Interpolasi tidak dapat dihitung");
    }
    //KASUS 3: Solusi unik
    else {
        System.out.println("Hasil polinom interpolasinya adalah");
        System.out.print("P(x) = ");

        // Perulangan untuk mencetak hasil interpolasi
        for (int i = 0; i <= this.getLastIdxBrs(); i++) {
            if (i == 0) {
                if (this.Elmt[i][this.getLastIdxKol()] > 1e-9) {
                    System.out.print(df.format(this.Elmt[i][this.getLastIdxKol()]));
                } else if (this.Elmt[i][this.getLastIdxKol()] < -1e-9) {
                    System.out.print(df.format(this.Elmt[i][this.getLastIdxKol()]));
                }
            } else if (i == 1) {
                if (this.Elmt[i][this.getLastIdxKol()] > 1e-9) {
                    if (this.Elmt[i][this.getLastIdxKol()] == 1) {
                        System.out.print(" + " + "x");
                    } else {
                        System.out.print(" + " + df.format(this.Elmt[i][this.getLastIdxKol()]) +
"x");
                    }
                } else if (this.Elmt[i][this.getLastIdxKol()] < -1e-9) {
                    if (this.Elmt[i][this.getLastIdxKol()] == -1) {
                        System.out.print(" - " + "x");
                    } else {
                        System.out.print(" - " +
df.format(Math.abs(this.Elmt[i][this.getLastIdxKol()])) + "x");
                    }
                }
            } else {
                if (this.Elmt[i][this.getLastIdxKol()] > 1e-9) {
                    if (this.Elmt[i][this.getLastIdxKol()] == 1) {
                        System.out.print(" + " + "x^" + i);
                    } else {
                        System.out.print(" + " + df.format(this.Elmt[i][this.getLastIdxKol()]) +
"x^" + i);
                    }
                } else if (this.Elmt[i][this.getLastIdxKol()] < -1e-9) {
                    if (this.Elmt[i][this.getLastIdxKol()] == -1) {
                        System.out.print(" - " + "x^" + i);
                    } else {
                        System.out.print(" - " +
df.format(Math.abs(this.Elmt[i][this.getLastIdxKol()])) + "x^" + i);
                    }
                }
            }
            System.out.println();
        }
    }

    // Menghitung taksiran
    Scanner scan = new Scanner(System.in);
    double x = 0;
    do{
        System.out.println("Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999):");
        x = scan.nextDouble();
        if (x != -999) {
            double taksiran = this.Taksiran(x);
            System.out.println("P(" + x + ") ≈ " + taksiran);
        }
    }
}

```

```

        }
    } while (x != -999);
}

public void TulisInterpolasiKeFile(boolean hasSolution, int brsNotZero) throws IOException {
    /* I.S. Matriks terdefinisi dalam bentuk matriks eselon baris tereduksi
       F.S. Menyimpan hasil interpolasi ke dalam file dan juga menampilkannya di layar */

    Scanner scan = new Scanner(System.in);

    // Menerima masukan nama file dari pengguna
    System.out.println("Masukkan nama File solusi beserta direktori dengan format
nama_folder/nama_file.txt: ");
    System.out.println("Contoh: solutions/SolusiSPL.txt");
    String namafile = "../"+scan.nextLine();

    // Format penulisan desimal
    DecimalFormat df = new DecimalFormat("#.#####");

    // Penulisan solusi ke dalam file
    try (FileOutputStream file = new FileOutputStream(namafile)) {
        byte[] b;
        //KASUS 1: HAS NO SOLUTION
        if (!hasSolution) {
            String sa = "Polinom Interpolasi tidak dapat dihitung\n";
            System.out.print(sa);
            b = sa.getBytes();
            file.write(b);
        } else {
            //KASUS 2: INFINITELY MANY SOLUTIONS
            if (brsNotZero < this.NKolEff - 1) {
                String sb = "Polinom Interpolasi tidak dapat dihitung\n";
                System.out.print(sb);
                b = sb.getBytes();
                file.write(b);
            }
            //KASUS 3: UNIQUE SOLUTION
            else {
                String s1 = "Hasil polinom interpolasinya adalah\n";
                System.out.print(s1);
                b = s1.getBytes();
                file.write(b);
                String s2 = "P(x) = ";
                System.out.print(s2);
                b = s2.getBytes();
                file.write(b);

                for (int i = 0; i <= this.getLastIdxBrs(); i++) {
                    if (i == 0) {
                        if (this.Elmt[i][this.getLastIdxKol()] > 1e-9) {
                            String sal = df.format(this.Elmt[i][this.getLastIdxKol()]);
                            System.out.print(sal);
                            b = sal.getBytes();
                            file.write(b);
                        } else if (this.Elmt[i][this.getLastIdxKol()] < -1e-9) {
                            String sa2 = df.format(this.Elmt[i][this.getLastIdxKol()]);
                            System.out.print(sa2);
                            b = sa2.getBytes();
                            file.write(b);
                        }
                    } else if (i == 1) {
                        if (this.Elmt[i][this.getLastIdxKol()] > 1e-9) {
                            String sb1;
                            if (this.Elmt[i][this.getLastIdxKol()] == 1) {
                                sb1 = " + " + "x";
                            } else {
                                sb1 = " + " + df.format(this.Elmt[i][this.getLastIdxKol()]) + "x";
                            }
                            System.out.print(sb1);
                            b = sb1.getBytes();
                        }
                    }
                }
            }
        }
    }
}

```

```

        file.write(b);
    } else if (this.Elmt[i][this.getLastIdxKol()] < -1e-9) {
        String sb2;
        if (this.Elmt[i][this.getLastIdxKol()] == -1) {
            sb2 = " - " + "x";
        } else {
            sb2 = " - " + df.format(Math.abs(this.Elmt[i][this.getLastIdxKol()])) +
"x";
        }
        System.out.print(sb2);
        b = sb2.getBytes();
        file.write(b);
    }
} else {
    if (this.Elmt[i][this.getLastIdxKol()] > 1e-9) {
        String scl;
        if (this.Elmt[i][this.getLastIdxKol()] == 1) {
            scl = " + " + "x^" + i;
        } else {
            scl = " + " + df.format(this.Elmt[i][this.getLastIdxKol()]) + "x^" + i;
        }
        System.out.print(scl);
        b = scl.getBytes();
        file.write(b);
    } else if (this.Elmt[i][this.getLastIdxKol()] < -1e-9) {
        String sc2;
        if (this.Elmt[i][this.getLastIdxKol()] == -1) {
            sc2 = " - " + "x^" + i;
        } else {
            sc2 = " - " + df.format(Math.abs(this.Elmt[i][this.getLastIdxKol()])) +
"x^" + i;
        }
        System.out.print(sc2);
        b = sc2.getBytes();
        file.write(b);
    }
}
System.out.println();
}
}

// Menghitung taksiran dan menyimpan hasilnya ke dalam file
double x = 0;
do {
    System.out.println("Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu
utama, masukkan -999):");
    x = scan.nextDouble();
    if (x != -999) {
        double taksiran = this.Taksiran(x);
        String sd = "\nP(" + x + ") ≈ " + taksiran;
        System.out.println("P(" + x + ") ≈ " + taksiran);
        b = sd.getBytes();
        file.write(b);
    }
} while (x != -999);
}

public double Taksiran(double x){
    /* I.S. Hasil interpolasi polinom terdefinisi dalam sebuah matriks
       F.S. Mengembalikan hasil substitusi nilai x ke dalam polinom */

    double result = 0;
    for (int i = 0; i <= this.getLastIdxBrs(); i++) {
        result += Math.pow(x, i) * this.Elmt[i][this.getLastIdxKol()];
    }
    return result;
}

```

3.2.7 MultipleLinearRegression

```
public void MultipleLinearRegression()
// I.S. Matriks augmented yang berisi beberapa nilai variabel
// peubah dan nilai variabel respon terdefinisi serta masukkan
// nilai variabel peubah yang diperlukan untuk menaksir suatu
// variabel respon terdefinisi
// F.S. Menampilkan persamaan yang dihasilkan dari regresi
// linier berganda serta hasil taksiran suatu variabel respon,
// atau hasil persamaan regresi dan hasil taksiran disimpan ke
// suatu file dengan format .txt
// Proses: Membuat matriks augmented yang berisikan SPL dari
// data yang diinput dengan rumus Normal Estimation Equation for
// Multiple Linear Regression, kemudian mencari solusi SPL
// tersebut untuk mendapatkan koefisien-koefisien variabel
// peubah
// dalam persamaan regresi, dan menghitung taksiran variabel
// respon sesuai dengan persamaan regresi

public double SumMul2BrsMLR (int IdxKol1, int IdxKol2)
// I.S. Sebuah matriks augmented (M) terdefinisi, dan IdxKol1
// serta IdxKol2 yang ingin dihitung jumlah dari hasil perkalian
// tiap elemennya terdefinisi.
// F.S. Mengembalikan hasil perhitungan jumlah dari hasil
// perkalian tiap elemen dari IdxKol1 dan IdxKol2 dari matriks M

public double[] SPLCramer (boolean isMLR)
// I.S. SPL dalam bentuk matriks augmented (M)
// F.S. Jika isMLR = True maka fungsi hanya mengembalikan array
// of double yang berisi solusi SPL, jika isMLR = false fungsi
// dapat menampilkan solusi SPL ke layer atau menyimpan ke dalam
// file serta tetap mengembalikan array of double yang berisi
// solusi SPL tersebut
```

```

// Multiple Linear Regression
public void MultipleLinearRegression() throws IOException{
    //Preparation: Membuat SPL dari Tabel Data dengan Normal Estimation Equation for Multiple Linear Regression
    int x = this.NKolEff+1;
    int y = this.NKolEff;
    Matriks spl = new Matriks(y, x);
    for(int i=0; i<y; i++){
        for(int j=0; j<x; j++){
            spl.Elm[i][j] = this.SumMul2BrsMLR(i, j);
        }
    }

    //Perhitungan bo, b1, b2, ..., bk
    double[] koef = new double[y];
    koef = spl.SPLCramer(true);
    int n = koef.length;

    // Menerima masukan user untuk nilai x1..xn
    System.out.println("\nUntuk mengetahui estimasi nilai y, masukkan nilai variabel peubah");
    Scanner scan = new Scanner(System.in);
    double[] variabel = new double[n];
    variabel[0] = 1;
    for(int i=1;i<n;i++){
        System.out.print("Nilai x"+i+": ");
        variabel[i] = scan.nextDouble();
    }

    System.out.println ("#=====");
    System.out.println ("# PENULISAN PERSAMAAN REGRESI DAN TAKSIRAN NILAI FUNGSI TERHADAP X YANG DIBERIKAN      #");
    System.out.println ("#-----#");
    System.out.println ("# Silakan pilih salah satu dari pilihan berikut!          #");
    System.out.println ("#=====");
    System.out.println ("# 1. Tampilan pada layar           #");
    System.out.println ("# 2. Simpan dalam file           #");
    System.out.println ("#=====");
    System.out.println ("# Ketik '1' atau '2' pada keyboard          #");

    // Menerima masukkan user untuk pilihan penulisan solusi regresi linier berganda
    int pilihan = scan.nextInt();
    while (pilihan<1 || pilihan>2){
        System.out.println("Masukan Anda salah. Silakan ulangi kembali.");
        pilihan = scan.nextInt();
    }

    if(pilihan ==1 ){
        //Print persamaan
        DecimalFormat eq = new DecimalFormat("#.####");
        System.out.println("\nBerikut adalah persamaan yang didapatkan dengan Regresi Linear Berganda:");
        String equation = "y = ";

        for(int i=0; i<n; i++){
            if(i==0){
                equation += eq.format(koef[i]);
            } else{
                if(koef[i]>0){
                    equation += " + "+eq.format(koef[i])+" x"+i;
                } else if (koef[i]<0){
                    equation += " - "+eq.format(Math.abs(koef[i]))+" x"+i;
                } else{
                    continue;
                }
            }
        }
    }
}

```

```

        }
    }
    System.out.println(equation);

    //Perhitungan nilai y
    DecimalFormat dfY = new DecimalFormat("#.##");
    double y1=0;
    for(int i=0; i<n; i++){
        y1 += koef[i]*variabel[i];
    }
    System.out.println("Berdasarkan nilai variabel peubah, didapatkan nilai y sebesar " + dfY.format(y1));

}else {

    //Print persamaan
    DecimalFormat eq = new DecimalFormat("#####");
    String final1 = "Berikut adalah persamaan yang didapatkan dengan Multiple Linear Regression:\n";
    String equation = "y = ";

    for(int i=0; i<n; i++){
        if(i==0){
            equation += eq.format(koef[i]);
        } else{
            if(koef[i]>0){
                equation += " + "+eq.format(koef[i])+" x"+i;
            } else if (koef[i]<0){
                equation += " - "+eq.format(Math.abs(koef[i]))+" x"+i;
            } else{
                continue;
            }
        }
    }
    equation += "\n";

    //Perhitungan nilai y
    DecimalFormat dfY = new DecimalFormat("#.##");
    double y1=0;
    for(int i=0; i<n; i++){
        y1 += koef[i]*variabel[i];
    }
    String final2 = "Berdasarkan nilai variabel peubah, didapatkan nilai y sebesar " + dfY.format(y1)+"\n";

    // Simpan dalam file
    Scanner sc = new Scanner(System.in);
    System.out.println("Masukkan nama File solusi beserta direktori dengan format nama_folder/nama_file.txt: ");
    System.out.println("Contoh: solutions/SolusiSPL.txt");
    String namafile = "../"+sc.nextLine();

    try (FileOutputStream file = new FileOutputStream(namafile)) {
        byte[] b;
        String final3 =final1+equation+final2;
        System.out.print(final3);
        b = final3.getBytes();
        file.write(b);
        System.out.println("Persamaan Regresi dan Taksiran Nilai Fungsi berhasil disimpan kedalam file");
    }
}
}

```

```

//Untuk Multiple Linear Regression
//Jumlah Perkalian 2 Kolom
public double SumMul2BrsMLR(int IdxKol1, int IdxKol2){
    // fungsi ini digunakan untuk menghasilkan jumlah dari perkalian setiap elemen pada kolom ke IdxKol1 dengan kolom ke IdxKol2
    // Untuk merealisasikan Normal Estimation Equation for Multiple Linear Regression
    double sum = 0;
    int n = this.NBrsEff;
    int k = this.NKolEff;
    if(IdxKol1==0 && IdxKol2==0){
        // untuk Elmt[0][0]
        sum = n;
    } else if(IdxKol2 == k){
        if(IdxKol1==0){
            // Kasus untuk Elmt[0][k] pada SPL
            for(int i=0; i<n ;i++){
                sum += this.Elmt[i][0];
            }
        } else{
            // Kasus Untuk kolom ke K selain Elmt[0][k]
            for(int i=0; i<n ;i++){
                sum += this.Elmt[i][IdxKol1]*this.Elmt[i][0];
            }
        }
    } else if(IdxKol1==0 || IdxKol2==0){
        if(IdxKol1==0){
            // Kasus Untuk kolom pertama
            for(int i=0; i<n; i++){
                sum += this.Elmt[i][IdxKol2];
            }
        } else{
            // Kasus untuk baris pertama selain selain Elmt[0][0] dan Elmt[0][k]
            for(int i=0; i<n; i++){
                sum+= this.Elmt[i][IdxKol1];
            }
        }
    } else{
        //Kasus Elmt ditengah
        for(int i=0; i<n; i++){
            sum+= this.Elmt[i][IdxKol1] * this.Elmt[i][IdxKol2];
        }
    }
    return sum;
}

```

3.3 Class InversMatriks

```
public InversMatriks (Matriks M)
// Konstruktor untuk membentuk matriks InversMatriks

public void OBEMatriksInvers ()
// I.S. Matriks terdefinisi dengan ukuran dan elemen dari pengguna
// F.S. Menghasilkan matriks eselon baris tereduksi

public void hasilInversOBE ()
// I.S. Matriks berbentuk eselon baris tereduksi dan mengandung
// matriks identitas
// F.S. Menghasilkan matriks invers sesuai dengan masukan pengguna

public boolean IsInversible ()
/* Mengembalikan nilai true jika matriks memiliki balikan, dan
false jika sebaliknya. Matriks memiliki balikan jika merupakan
matriks persegi dan determinannya tidak nol */

public void hitungKofaktor (Matriks temp, int p, int q)
/* I.S. Matriks temp terdefinisi, p dan q berturut-turut adalah
indeks baris dan kolom yang ingin dicari kofaktornya
F.S. Menghasilkan nilai Kofaktor dari Matriks temp pada baris p
dan kolom q */

public void hasilInversKofaktor ()
/* I.S. Matriks terdefinisi dengan ukuran dan elemen dari pengguna
dan invertible
F.S. Menghasilkan matriks invers yang sesuai */

public void TulisMatriksInvers ()
/* I.S. Matriks invers terdefinisi
F.S. Menulis hasil matriks invers yang dapat ditampilkan di layar
atau disimpan dalam sebuah file .txt */

public void SPLInvers ()
/* I.S. Matriks terdefinisi dengan ukuran dan elemen dari pengguna
F.S. Jika matriks SPL memiliki balikan dan solusinya unik, solusi
akan ditampilkan ke layar atau disimpan ke dalam file .txt. Jika
matriks SPL tidak memiliki balikan, akan ditampilkan "Matriks
tidak memiliki balikan". */

public void TulisSolusiSPL ()
/* I.S. Matriks terdefinisi dan berbentuk eselon baris tereduksi
F.S. Menulis solusi SPL yang dapat ditampilkan ke layar atau
disimpan ke dalam file dengan format .txt */

public void SolusiSPL ()
/* I.S. Matriks terdefinisi dan berbentuk eselon baris tereduksi
F.S. Menulis solusi SPL yang dapat ditampilkan ke layar */

public void SolusiSPLKeFile ()
/* I.S. Matriks terdefinisi dan berbentuk eselon baris tereduksi
F.S. Menulis solusi SPL disimpan dalam file dengan format .txt */
```

```

import java.io.FileOutputStream;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.Scanner;

public class InversMatriks {
    // ATTRIBUTES
    public Matriks matriks;

    // CONSTRUCTOR
    public InversMatriks(Matriks M) {
        this.matriks = M;
    }

    // METHODS
    // Menggunakan Operasi Baris Elementer
    public void OBEMatriksInvers() {
        /* I.S. Matriks terdefinisi dengan ukuran dan elemen dari pengguna
           F.S. Menghasilkan matriks eselon baris tereduksi */

        // Memperlebar jumlah kolom efektif matriks untuk penyisipan matriks identitas
        int KolAwal = this.matriks.NKolEff;
        this.matriks.NKolEff += this.matriks.NBrsEff;

        // Menambahkan matriks singular ke sisi kanan matriks
        for (int i = 0; i <= this.matriks.getLastIdxBrs(); i++) {
            for (int j = KolAwal; j <= this.matriks.getLastIdxKol(); j++) {
                if (j == KolAwal) {
                    this.matriks.Elmt[i][j] = 1;
                } else {
                    this.matriks.Elmt[i][j] = 0;
                }
            }
            KolAwal++;
        }
        this.matriks.EliminasiGaussJordan();
    }

    public void hasilInversOBE() {
        /* I.S. Matriks berbentuk eselon baris tereduksi dan mengandung matriks identitas
           F.S. Menghasilkan matriks invers sesuai dengan masukan pengguna */

        // Inisialisasi matriks temporary
        Matriks tmp = this.matriks;

        // Memindahkan matriks invers ke sebelah kiri matriks
        for (int i = 0; i <= this.matriks.getLastIdxBrs(); i++) {
            int x = 0;
            for (int j = (this.matriks.NBrsEff); j <= this.matriks.getLastIdxKol(); j++) {
                this.matriks.Elmt[i][x] = tmp.Elmt[i][j];
                x++;
            }
        }

        // Mengurangi jumlah kolom efektif matriks sehingga menyisakan matriks inversnya saja
        this.matriks.NKolEff -= this.matriks.NBrsEff;
    }

    public boolean IsInversible() {
        /* Mengembalikan nilai true jika matriks memiliki balikan, dan false jika sebaliknya.
           Matriks memiliki balikan jika merupakan matriks persegi dan determinannya tidak nol */

        boolean inversible = false;
        if ((this.matriks.NKolEff == this.matriks.NBrsEff) &&
        (this.matriks.DeterminanDenganKofaktor() != 0)) {
            inversible = true;
        }
    }
}

```

```

        return inversible;
    }

    // Menggunakan Determinan dan Kofaktor
    public void hitungKofaktor(Matriks temp, int p, int q){
        /* I.S. Matriks temp adalah matriks kosong yang terdefinisi, p dan q berturut-turut
        adalah indeks
            baris dan kolom yang ingin dicari kofaktornya
        F.S. Menghasilkan minor dari matriks atribut pada baris p dan kolom q dan menyimpannya
            ke dalam matriks temp */
        int brs = 0, kol = 0;

        for (int i = 0; i <= this.matriks.getLastIdxBrs(); i++){
            for (int j = 0; j <= this.matriks.getLastIdxKol(); j++){
                // Menyimpan elemen minor ke dalam temp, yaitu elemen saat indeks i != p dan i != q
                if (i != p && j != q){
                    temp.Elmt[brs][kol++] = this.matriks.Elmt[i][j];

                    if (kol == this.matriks.getLastIdxKol()){
                        kol = 0;
                        brs++;
                    }
                }
            }
        }
    }

    public void hasilInversKofaktor() {
        /* I.S. Matriks terdefinisi dengan ukuran dan elemen dari pengguna dan invertible
        F.S. Menghasilkan matriks invers yang sesuai */

        // Inisialisasi dan assignment sebuah matriks adjoin dan sebuah float determinan
        Matriks adj = new Matriks(this.matriks.NBrssEff, this.matriks.NKolsEff);
        double det = this.matriks.DeterminanDenganKofaktor();

        // Menghitung matriks adjoin
        if (this.matriks.NKolsEff == 1){
            adj.Elmt[0][0] = 1;
        } else {
            int sign = 1;

            for (int i = 0; i <= this.matriks.getLastIdxBrs(); i++) {
                for (int j = 0; j <= this.matriks.getLastIdxKol(); j++){
                    Matriks temp = new Matriks(this.matriks.NBrssEff, this.matriks.NKolsEff);
                    this.hitungKofaktor(temp, i, j);

                    if ((i+j) % 2 == 0){
                        sign = 1;
                    } else {
                        sign = -1;
                    }

                    temp.NBrssEff--;
                    temp.NKolsEff--;

                    adj.Elmt[j][i] = sign * temp.DeterminanDenganKofaktor();
                }
            }
        }
        this.matriks = adj;

        // Membagi seluruh elemen adjoin dengan determinan matriks
        for (int i = 0; i <= this.matriks.getLastIdxBrs(); i++){
            for (int j = 0; j <= this.matriks.getLastIdxKol(); j++) {
                this.matriks.Elmt[i][j] /= det;
            }
        }
    }
}

```

```

public void TulisMatriksInvers() throws IOException {
    /* I.S. Matriks invers terdefinisi
       F.S. Menulis hasil matriks invers yang dapat ditampilkan di layar atau disimpan dalam
sebuah file .txt */

    Scanner scan = new Scanner(System.in);
    System.out.println("=====#
");
    System.out.println ("# PENULISAN MATRIKS BALIKAN
#");
    System.out.println ("#-----#
");
    System.out.println ("# Silakan pilih salah satu dari pilihan berikut!
#");
    System.out.println ("#-----#
");
    System.out.println ("# 1. Tampilkan pada layar
#");
    System.out.println ("# 2. Simpan dalam file
#");
    System.out.println ("#-----#
");
    System.out.println ("# Ketik '1' atau '2' pada keyboard:
#");

    String pilihan = scan.nextLine();
    while (!pilihan.equals("1") && !pilihan.equals("2")){
        System.out.println("Masukan Anda salah. Silakan ulangi kembali.");
        pilihan = scan.nextLine();
    }

    if (pilihan.equals("1")) {
        System.out.println("Invers dari matriks di atas adalah ");
        this.matriks.TulisMatriks();
    } else {
        System.out.println("Invers dari matriks di atas adalah ");
        this.matriks.TulisMatriksKeFile();
    }
}

// Menyelesaikan SPL menggunakan Matriks Invers
public void SPLInvers() throws IOException{
    /* I.S. Matriks terdefinisi dengan ukuran dan elemen dari pengguna
       F.S. Jika matriks SPL memiliki balikan dan solusinya unik, solusi akan ditampilkan ke
layar atau
       disimpan ke dalam file .txt. Jika matriks SPL tidak memiliki balikan, akan
ditampilkan
       "Matriks tidak memiliki balikan". */

    // Memindahkan elemen-elemen kolom terakhir ke dalam matriks yang berbeda yaitu B
    Matriks B = new Matriks(this.matriks.NBrsEff, 1);
    for (int i = 0; i <= B.getLastIdxBrs(); i++) {
        B.Elmt[i][0] = this.matriks.Elmt[i][this.matriks.getLastIdxKol()];
        this.matriks.Elmt[i][this.matriks.getLastIdxKol()] = 0;
    }

    // Jumlah kolom efektif dikurangi 1
    this.matriks.NKolEff -= 1;

    if (this.IsInversible()){
        // Jika matriks invertible, maka proses penghitungan invers matriks dilakukan
        this.OBEMatriksInvers();
        this.hasilInversOBE();
        this.matriks = this.matriks.KalidenganMatriks(B);
        this.TulisSolusiSPL();
    } else {
        System.out.println("Matriks tidak memiliki balikan.");
    }
}

```

```

        }

    }

    public void TulisSolusiSPL() throws IOException {
        /* I.S. Matriks terdefinisi dan berbentuk eselon baris tereduksi
           F.S. Menulis solusi SPL yang dapat ditampilkan ke layar atau disimpan ke dalam file
           dengan format .txt */

        Scanner scan = new Scanner(System.in);
        System.out.println
        ("#=====");
        System.out.println ("# PENULISAN SOLUSI SPL
#");
        System.out.println ("#-----");
        System.out.println ("# Silakan pilih salah satu dari pilihan berikut!
#");
        System.out.println
        ("#=====");
        System.out.println ("# 1. Tampilkan pada layar
#");
        System.out.println ("# 2. Simpan dalam file
#");
        System.out.println
        ("#=====");
        System.out.println ("# Ketik '1' atau '2' pada keyboard:
#");

        // Menerima masukan pilihan dari pengguna
        String pilihan = scan.nextLine();
        while (!pilihan.equals("1") && !pilihan.equals("2")){
            System.out.println("Masukan Anda salah. Silakan ulangi kembali.");
            pilihan = scan.nextLine();
        }

        if (pilihan.equals("1")) {
            this.SolusiSPL();
        } else {
            this.SolusiSPLKeFile();
        }
    }

    public void SolusiSPL(){
        /* I.S. Matriks terdefinisi dan berbentuk eselon baris tereduksi
           F.S. Menulis solusi SPL yang dapat ditampilkan ke layar */

        for (int i = 0; i <= this.matriks.getLastIdxBrs(); i++){
            DecimalFormat df = new DecimalFormat("#.##");

            System.out.print("x" + (i+1) + " = " + df.format(this.matriks.Elmt[i][0]) + "\n");
        }
    }

    public void SolusiSPLKeFile() throws IOException {
        /* I.S. Matriks terdefinisi dan berbentuk eselon baris tereduksi
           F.S. Menulis solusi SPL disimpan dalam file dengan format .txt */

        // Menerima masukan nama file dari panggilan
        Scanner scan = new Scanner(System.in);
        System.out.println("Masukkan nama File solusi beserta direktori dengan format
nama_folder/nama_file.txt: ");
        System.out.println("Contoh: solutions/SolusiSPL.txt");
        String namafile = "../"+scan.nextLine();

        // Menampilkan solusi ke layar dan menyimpannya ke file
        try (FileOutputStream file = new FileOutputStream(namafile)) {

```

```

        byte[] b;
        for (int i = 0; i <= this.matriks.getLastIdxBrs(); i++) {
            DecimalFormat df = new DecimalFormat("#.###");
            String s = ("x" + (i + 1) + " = " + df.format(this.matriks.Elmt[i][0])) + "\n";
            System.out.print(s);
            b = s.getBytes();
            file.write(b);
        }
    }
}
}

```

3.4 Class TubesAlgeo

<pre> public static void TulisMenuWelcome () // I.S. Program TubesAlgeo berjalan // F.S. Menampilkan menu selamat datang ke layar user, atau // Status = N sehingga program berhenti. </pre>
<pre> public static void TulisMenuUtama () // I.S. Status = Y // F.S. Menampilkan Menu Utama ke layar user </pre>
<pre> public static void MasukkanMenuUtamaSalah () // I.S. masukkan user tidak valid, operasi<1 atau operasi>6 // F.S. Menampilkan pesan kesalahan masukkan pada menu utama ke // layar user </pre>
<pre> public static void TulisMenuPenyelesaian () // I.S. operasi = 1 // F.S. Menampilkan Menu Metode Penyelesaian SPL ke layar user </pre>
<pre> public static void MasukkanMetodeSalah () // I.S. masukkan user tidak valid, metode<1 atau metode>5 // F.S. Menampilkan pesan kesalahan masukkan pada Menu Metode // Penyelesaian SPL ke layar user </pre>
<pre> public static void TulisMenuDeterminan () // I.S. Operasi = 2 // F.S. Menampilkan Menu Metode Perhitungan Determinan ke layar // user </pre>
<pre> public static void MasukkanDeterminanSalah () // I.S. masukkan user tidak valid, metode<1 atau metode>3 // F.S. Menampilkan pesan kesalahan masukkan pada Menu Metode // Perhitungan Determinan ke layar user </pre>
<pre> public static void TulisMenuInvers () // I.S. Operasi = 3 // F.S. Menampilkan Menu Metode Perhitungan Matriks Balikan ke // layar user. </pre>
<pre> public static void MasukkanMetodeInversSalah () // I.S. masukkan user tidak valid, metode<1 atau metode>3 </pre>

```
// F.S. Menampilkan pesan kesalahan masukkan pada Menu Metode
// Perhitungan Matriks Balikan ke layar user

public static void TulisSolusiDet (double det)
// I.S. Determinan Matriks M terdefinisi
// F.S. Menulis determinan matriks M yang dapat ditampilkan
// ke layar atau disimpan ke dalam file dengan format .txt

public static void main (String [] args)
// I.S. TubesAlgeo.class (program) terdefinisi dan siap dijalankan
// F.S. User telah memasukkan '6' pada Menu Utama sehingga membuat
// status = 'N' dan mengakhiri program
```

```

Run | Debug
public static void main(String [] args) throws IOException {
    //START
    TulisMenuWelcome();
    Scanner scan = new Scanner(System.in);
    char status = scan.next().charAt(0);
    while(status == 'Y'){
        // Selama status = Y maka program akan berjalan
        TulisMenuUtama();
        // Menerima masukkan user untuk pilihan operasi yang ingin dilakukan
        scan = new Scanner(System.in);
        int operasi = scan.nextInt();
        while(operasi<1 || operasi>6){
            MasukanMenuUtamaSalah();
            operasi = scan.nextInt();
        }
        if(operasi==1){
            // SPL
            TulisMenuPenyelesaian();
            // Menerima masukkan user untuk pilihan metode untuk menyelesaikan SPL
            int metode = scan.nextInt();
            while(metode<1 || metode>5){
                MasukanMetodeSalah();
                metode = scan.nextInt();
            }
            if(metode == 1){
                //Eliminasi Gauss
                MatriksInit Mtemp = new MatriksInit(1);
                Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);

                Mtemp.toMatriks(A);
                A.SPLGauss();
            } else if (metode == 2){
                //Eliminasi Gauss-Jordan
                MatriksInit Mtemp = new MatriksInit(1);
                Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);

                Mtemp.toMatriks(A);
                A.SPLGaussJordan();
            } else if (metode == 3){
                //Matriks Balikan
                MatriksInit Mtemp = new MatriksInit(1);
                Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);

                Mtemp.toMatriks(A);
                InversMatriks InversA = new InversMatriks(A);
                InversA.SPLInvers();

            } else if(metode==4) {
                //Kaidah Cramer
                MatriksInit Mtemp = new MatriksInit(1);
                Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);

                Mtemp.toMatriks(A);
                double[] determinan = new double[A.NBrsEff];
                determinan = A.SPLCramer(false);
            }
            // else, metode==5 membuat program kembali ke menu utama
        } else if(operasi==2){
            // Determinan
            TulisMenuDeterminan();
            // Menerima masukkan user untuk pilihan metode untuk menghitung determinan sebuah matriks
            int metode = scan.nextInt();
            while(metode<1 || metode>3){
                MasukanMetodeDeterminanSalah();
                metode = scan.nextInt();
            }
        }
    }
}

```

```

if(metode == 1){
    // Metode Reduksi Baris
    MatriksInit Mtemp = new MatriksInit(2);
    Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);

    Mtemp.toMatriks(A);
    double determinan = A.DeterminanOBE();
    TulisSolusiDet(determinan);
} else if (metode == 2){
    // Metode Ekspansi Kofaktor
    MatriksInit Mtemp = new MatriksInit(2);
    Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);

    Mtemp.toMatriks(A);
    double determinan = A.DeterminanDenganKofaktor();
    TulisSolusiDet(determinan);
} // else, metode==3 membuat program kembali ke menu utama

} else if(operasi==3){
    //Matriks Balikan
    TulisMenuInvers();
    // Menerima masukkan user untuk pilihan metode untuk membuat matriks balikan dari sebuah matriks
    int metode = scan.nextInt();
    while(metode<1 || metode>3){
        MasukanMetodeInversSalah();
        metode = scan.nextInt();
    }
    if(metode == 1){
        // Metode Reduksi Baris
        MatriksInit Mtemp = new MatriksInit(3);
        Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);

        Mtemp.toMatriks(A);
    }
    if(metode == 1){
        // Metode Reduksi Baris
        MatriksInit Mtemp = new MatriksInit(3);
        Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);

        Mtemp.toMatriks(A);
        InversMatriks InversA = new InversMatriks(A);
        if (InversA.IsInversible()){
            InversA.OBEMatriksInvers();
            InversA.hasilInversOBE();
            InversA.TulisMatriksInvers();
        } else {
            System.out.println("Matriks tidak memiliki balikan.");
        }
    } else if (metode == 2){
        // Metode Ekspansi Kofaktor
        MatriksInit Mtemp = new MatriksInit(3);
        Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);

        Mtemp.toMatriks(A);
        InversMatriks InversA = new InversMatriks(A);
        if (!InversA.IsInversible()){
            System.out.println("Matriks tidak memiliki balikan.");
        } else {
            InversA.hasilInversKofaktor();
            InversA.TulisMatriksInvers();
        }
    }
}

```

```
    } else if (operasi == 4) {
        // Interpolasi Polinom
        MatriksInit Mtemp = new MatriksInit(4);
        Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);

        Mtemp.toMatriks(A);
        A.MakeMatriksInterpolasi(A.NBrsEff);
        A.SolusiInterpolasi();

    } else if(operasi==5){
        //Regresi Linear Berganda
        MatriksInit Mtemp = new MatriksInit(5);
        Matriks A = new Matriks(Mtemp.NBrsEff, Mtemp.NKolEff);
        Mtemp.toMatriks(A);
        A.MultipleLinearRegression();
    }else{
        // Keluar Program
        status = 'N';
    }
}
}
```

BAB IV

EKSPERIMENT

4.1 SPL

1. Test Case 1

Input	Output
$A = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 5 & -7 & -5 \\ 2 & -1 & 1 & 3 \\ 5 & 2 & -4 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -2 \\ 4 \\ 6 \end{bmatrix}$	SPL ini tidak memiliki solusi.
<p>Matriks Augmented:</p> $\begin{array}{ccccc c} 1 & 1 & -1 & -1 & 1 \\ 2 & 5 & -7 & -5 & -2 \\ 2 & -1 & 1 & 3 & 4 \\ 5 & 2 & -4 & 2 & 6 \end{array}$	

Catatan:
Hasil didapatkan dengan metode eliminasi Gauss, metode eliminasi Gauss-Jordan, dan metode matriks balikan.

2. Test Case 2

Input	Output
$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & -3 & 0 \\ 2 & -1 & 0 & 1 & -1 \\ -1 & 2 & 0 & -2 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 6 \\ 5 \\ -1 \end{bmatrix}$	SPL ini memiliki solusi banyak, yang mengikuti: $x_1 = 3+s$ $x_2 = 2s$ $x_3 = r$ $x_4 = -1+s$ $x_5 = s$
<p>Matriks Augmented:</p> $\begin{array}{ccccc c} 1 & -1 & 0 & 0 & 1 & 3 \\ 1 & 1 & 0 & -3 & 0 & 6 \\ 2 & -1 & 0 & 1 & -1 & 5 \\ -1 & 2 & 0 & -2 & -1 & -1 \end{array}$	

Catatan:
Hasil didapatkan dengan metode eliminasi Gauss dan metode eliminasi Gauss-Jordan.

3. Test Case 3

Input	Output
$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$ <p>Matriks Augmented:</p> $\begin{array}{ccccccc} 0 & 1 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}$	<pre>SPL ini memiliki solusi banyak, yang mengikuti: x1 = r x2 = 1-t x3 = s x4 = -2-t x5 = 1+t x6 = t</pre>
<p>Catatan:</p> <p>Hasil didapatkan dengan metode eliminasi Gauss dan metode eliminasi Gauss-Jordan.</p>	

4. Test Case 4

Input	Output
<p>Matriks Hilbert dengan n = 6 (augmented):</p> $\begin{array}{ccccccc} 1 & 0.5 & 0.33 & 0.25 & 0.2 & 0.167 & 1 \\ 0.5 & 0.33 & 0.25 & 0.2 & 0.167 & 0.143 & 0 \\ 0.33 & 0.25 & 0.2 & 0.167 & 0.143 & 0.125 & 0 \\ 0.25 & 0.2 & 0.167 & 0.143 & 0.125 & 0.111 & 0 \\ 0.2 & 0.167 & 0.143 & 0.125 & 0.111 & 0.1 & 0 \\ 0.167 & 0.143 & 0.125 & 0.111 & 0.1 & 0.0909 & 0 \end{array}$	<pre>SPL ini memiliki solusi unik, yaitu: x1 = 6.34 x2 = -9.1 x3 = -76.87 x4 = 148.1 x5 = -21.32 x6 = -49.02</pre>
<p>Catatan:</p> <p>Hasil didapatkan dengan metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan, dan metode Cramer.</p>	

5. Test Case 5

Input	Output
Matriks Hilbert dengan n = 10 (augmented): <pre>1 0.5 0.33 0.25 0.2 0.167 0.143 0.125 0.111 0.1 1 0.5 0.33 0.25 0.2 0.167 0.143 0.125 0.111 0.1 0.0909 0 0.33 0.25 0.2 0.167 0.143 0.125 0.111 0.1 0.0909 0.0833 0 0.25 0.2 0.167 0.143 0.125 0.111 0.1 0.0909 0.0833 0.0769 0 0.2 0.167 0.143 0.125 0.111 0.1 0.0909 0.0833 0.0769 0.0714 0 0.167 0.143 0.125 0.111 0.1 0.0909 0.0833 0.0769 0.0714 0.0667 0 0.143 0.125 0.111 0.1 0.0909 0.0833 0.0769 0.0714 0.0667 0.0625 0 0.125 0.111 0.1 0.0909 0.0833 0.0769 0.0714 0.0667 0.0625 0.0588 0 0.111 0.1 0.0909 0.0833 0.0769 0.0714 0.0667 0.0625 0.0588 0.0556 0 0.1 0.0909 0.0833 0.0769 0.0714 0.0667 0.0625 0.0588 0.0556 0.0526 0</pre>	SPL ini memiliki solusi unik, yaitu: x1 = -33.6 x2 = 241.28 x3 = -20.78 x4 = -825.4 x5 = 552.38 x6 = -946.66 x7 = 1258.16 x8 = 571.35 x9 = 584.72 x10 = -1414.58
Catatan: Hasil didapatkan dengan metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan, dan metode Cramer.	

6. Test Case 6

Input	Output
Matriks Augmented: $\left[\begin{array}{ccccc} 1 & -1 & 2 & -1 & -1 \\ 2 & 1 & -2 & -2 & -2 \\ -1 & 2 & -4 & 1 & 1 \\ 3 & 0 & 0 & -3 & -3 \end{array} \right]$	SPL ini memiliki solusi banyak, yang mengikuti: x1 = -1+s x2 = 2r x3 = r x4 = s
Catatan: Hasil didapatkan dengan metode eliminasi Gauss dan metode eliminasi Gauss-Jordan.	

7. Test Case 7

Input	Output
<p>Matriks Augmented:</p> $\left[\begin{array}{ccccc} 2 & 0 & 8 & 0 & 8 \\ 0 & 1 & 0 & 4 & 6 \\ -4 & 0 & 6 & 0 & 6 \\ 0 & -2 & 0 & 3 & -1 \\ 2 & 0 & -4 & 0 & -4 \\ 0 & 1 & 0 & -2 & 0 \end{array} \right]$	<p>SPL ini memiliki solusi unik, yaitu: $x_1 = 0$ $x_2 = 2$ $x_3 = 1$ $x_4 = 1$</p>
<p>Catatan:</p> <p>Hasil didapatkan dengan metode eliminasi Gauss dan metode eliminasi Gauss-Jordan.</p>	

8. Test Case 8

Input	Output
$\begin{aligned} 8x_1 + x_2 + 3x_3 + 2x_4 &= 0 \\ 2x_1 + 9x_2 - x_3 - 2x_4 &= 1 \\ x_1 + 3x_2 + 2x_3 - x_4 &= 2 \\ x_1 + 6x_3 + 4x_4 &= 3 \end{aligned}$	<p>SPL ini memiliki solusi unik, yaitu: $x_1 = -0.22$ $x_2 = 0.18$ $x_3 = 0.71$ $x_4 = -0.26$</p>
<p>Matriks Augmented:</p> $\left[\begin{array}{ccccc} 8 & 1 & 3 & 2 & 0 \\ 2 & 9 & -1 & -2 & 1 \\ 1 & 3 & 2 & -1 & 2 \\ 1 & 0 & 6 & 4 & 3 \end{array} \right]$	

Catatan:

Hasil didapatkan dengan metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan, dan metode Cramer.

9. Test Case 9

Input	Output
$x_7 + x_8 + x_9 = 13.00$ $x_4 + x_5 + x_6 = 15.00$ $x_1 + x_2 + x_3 = 8.00$ $0.04289(x_3 + x_5 + x_7) + 0.75(x_6 + x_8) + 0.61396x_9 = 14.79$ $0.91421(x_3 + x_5 + x_7) + 0.25(x_2 + x_4 + x_6 + x_8) = 14.31$ $0.04289(x_3 + x_5 + x_7) + 0.75(x_2 + x_4) + 0.61396x_1 = 3.81$ $x_3 + x_6 + x_9 = 18.00$ $x_2 + x_5 + x_8 = 12.00$ $x_1 + x_4 + x_7 = 6.00$ $0.04289(x_1 + x_5 + x_9) + 0.75(x_2 + x_6) + 0.61396x_3 = 10.51$ $0.91421(x_1 + x_5 + x_9) + 0.25(x_2 + x_4 + x_6 + x_8) = 16.13$ $0.04289(x_1 + x_5 + x_9) + 0.75(x_4 + x_8) + 0.61396x_7 = 7.04$	SPL ini tidak memiliki solusi.
Matriks Augmented:	
$ \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 13.00 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 15.00 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 8.00 \\ 0 & 0 & 0.04289 & 0 & 0.04289 & 0.75 & 0.04289 & 0.75 & 0.61396 & 14.79 \\ 0 & 0.25 & 0.91421 & 0.25 & 0.91421 & 0.25 & 0.91421 & 0.25 & 0 & 14.31 \\ 0.61396 & 0.75 & 0.04289 & 0.75 & 0.04289 & 0 & 0.04289 & 0 & 0 & 3.81 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 18.00 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 12.00 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 6.00 \\ 0.04289 & 0.75 & 0.61396 & 0 & 0.04289 & 0.75 & 0 & 0 & 0.04289 & 10.51 \\ 0.91421 & 0.25 & 0 & 0.25 & 0.91421 & 0.25 & 0 & 0.25 & 0.91421 & 16.13 \\ 0.04289 & 0 & 0 & 0.75 & 0.04289 & 0 & 0.61396 & 0.75 & 0.04289 & 7.04 \end{matrix} $	

Catatan:

Hasil didapatkan dengan metode eliminasi Gauss dan metode eliminasi Gauss-Jordan.

4.2 Determinan

1. Test Case 1

Input	Output
$ \begin{matrix} 2 & 3 & 4 \\ 5 & 4 & 3 \\ 7 & 0 & 1 \end{matrix} $	Determinan matriks A adalah -56
Catatan:	
Hasil didapatkan dengan metode operasi baris elementer dan metode ekspansi kofaktor.	

2. Test Case 2

Input	Output
<pre>2 3 5 4 3 3 4 4 3 2 2 3 6 3 5 3 4 3 3 1 4 5 5 1 3</pre>	<pre>Determinan matriks A adalah -1</pre>
Catatan: Hasil didapatkan dengan metode operasi baris elementer dan metode ekspansi kofaktor.	

4.3 Matriks Balikan

1. Test Case 1

Input	Output
<pre># Masukkan n: 3 ===== # Masukkan elemen-elemen matriks A, dengan urutan: # a11 a12 a13 ... a1m # a21 a22 a23 ... a2m # ... # an1 an2 an3 ... anm 5 15 56 -4 -11 -41 -1 -3 -11</pre>	<pre>Invers dari matriks di atas adalah -2 -3 1 -3 1 -19 1 0 5 "</pre>
Catatan: Baik metode reduksi baris maupun metode ekspansi kofaktor menghasilkan matriks balikan yang sama	

2. Test Case 2

Input	Output
<pre># Masukkan n: 5 ===== # Masukkan elemen-elemen matriks A, dengan urutan: # a11 a12 a13 ... a1m # a21 a22 a23 ... a2m # ... # an1 an2 an3 ... anm 1 -3 0 -1 0 0 0 -2 0 3 2 0 0 0 0 0 4 0 -4 0 5 0 -5 0 6</pre>	<pre>Invers dari matriks di atas adalah 0 0 0.5 0 0 -0.25 0 0.125 0.0625 0 0 2 2.5 0 -1 -0.25 0 0.125 -0.1875 0 0 1.66667 1.66667 0 -0.66667</pre>
Catatan: Baik metode reduksi baris maupun metode ekspansi kofaktor menghasilkan matriks balikan yang sama	

3. Test Case 3

Input	Output
<pre># Masukkan n: 3 ===== # Masukkan elemen-elemen matriks A, dengan urutan: # a11 a12 a13 ... a1m # a21 a22 a23 ... a2m # ... # an1 an2 an3 ... anm 1 6 4 2 4 -1 -1 2 5</pre>	<pre>Matriks tidak memiliki balikan.</pre>
Catatan: Baik metode reduksi baris maupun metode ekspansi kofaktor menghasilkan matriks balikan yang sama	

4.4 Interpolasi Polinom

1. Test Case 1

Input	<pre># Masukkan n: 7 #===== # Masukkan pasangan titik x dan y sebanyak n, dengan format: # x1 y1 # x2 y2 # ... # xn yn 0.1 0.003 0.3 0.067 0.5 0.148 0.7 0.248 0.9 0.370 1.1 0.518 1.3 0.697 #</pre>
Output	<pre>^C Hasil polinom interpolasinya adalah $P(x) = -0.022977 + 0.24x + 0.197396x^2 + 0.026042x^4$ Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 0.2 $P(0.2) \approx 0.032960937500000016$ Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 0.55 $P(0.55) \approx 0.17111865234375$ Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 0.85 $P(0.85) \approx 0.33723583984375$ Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 1.28 $P(1.28) \approx 0.6775418375$</pre>

2. Test Case 2

Input	<pre># Masukkan n: 10 #===== # Masukkan pasangan titik x dan y sebanyak n, dengan format: # x1 y1 # x2 y2 # ... # xn yn 4.800 8211 5 10118 5.516 17025 5.710 20796 6.5 39294 7.194 64958 8.097 113134 8.258 123503 9.033 177571 9.333 145510</pre>
-------	--

Output	<pre> Hasil polinom interpolasinya adalah P(x) = 227096350.507999 - 415842607.965956x + 318150046.631761x^2 - 136003278.463966x^3 + 36176037.884002x^4 - 6249554.331817x^5 + 704212.266415x^6 - 50061.953959x^7 + 2041.919709x^8 - 36.470957x^9 Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 5.833 P(5.833) ≈ 23368.095750629902 Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 9 P(9.0) ≈ 176872.85820007324 Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 9.5 P(9.5) ≈ 68216.42609405518 Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 10 P(10.0) ≈ -916715.5688858032 Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 10.333 P(10.333) ≈ -3111094.9518814087 Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 10.5 P(10.5) ≈ -5161349.341300964 </pre>
--------	--

3. Test Case 3

Input	<pre> # Masukkan n: 5 ===== # Masukkan pasangan titik x dan y sebanyak n, dengan format: # x1 y1 # x2 y2 # ... # xn yn 0.4 0.4189 0.8 0.5072 1.2 0.5609 1.6 0.5837 2 0.5766 </pre>
Output	<pre> Hasil polinom interpolasinya adalah P(x) = 0.2896 + 0.381521x - 0.155859x^2 + 0.027214x^3 - 0.004395x^4 Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 1 P(1.0) ≈ 0.5380804687500002 Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 3 P(3.0) ≈ 0.41023671874999906 Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 2.5 P(2.5) ≈ 0.522831201171875 Masukkan nilai x yang ingin ditaksir (Jika ingin kembali ke menu utama, masukkan -999): 1.5 P(1.5) ≈ 0.580796044921875 </pre>

4.5 Regresi Linier Berganda

1. Test Case 1

Input	<pre># Masukkan jumlah peubah (n): 3 # Masukkan banyaknya data (i): 20 ===== # Masukkan persamaan-persamaannya dalam bentuk: # y1 x11 x12 ... x1n # y2 x21 x22 ... x2n # ... # yn xn1 xn2 ... xnn 0.90 72.4 76.3 29.18 0.91 41.6 70.3 29.35 0.96 34.3 77.1 29.24 0.89 35.1 68.0 29.27 1.00 10.7 79.0 29.78 1.10 12.9 67.4 29.39 1.15 8.3 66.8 29.69 1.03 20.1 76.9 29.48 0.77 72.2 77.7 29.09 1.07 24.0 67.7 29.60 1.07 23.2 76.8 29.38 0.94 47.4 86.6 29.35 1.10 31.5 76.9 29.63 1.10 10.6 86.3 29.56 1.10 11.2 86.0 29.48 0.91 73.3 76.3 29.40 0.87 75.4 77.9 29.28 0.78 96.6 78.7 29.29 0.82 107.4 86.8 29.03 0.95 54.9 70.9 29.37 Untuk mengetahui estimasi nilai y, masukkan nilai variabel peubah Nilai x1: 50 Nilai x2: 76 Nilai x3: 29.30</pre>
Output	Berikut adalah persamaan yang didapatkan dengan Regresi Linear Berganda: $y = -3.50778 - 0.00262 x_1 + 0.0008 x_2 + 0.15416 x_3$ Berdasarkan nilai variabel peubah, didapatkan nilai y sebesar 0.94

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

Dari proses pembuatan program dan eksperimen yang dilakukan, penulis mendapat beberapa kesimpulan, yaitu:

1. Java merupakan bahasa pemrograman yang sangat modular, mudah dibaca, serta dapat dijalankan pada berbagai sistem operasi, seperti Windows, MacOS, dan Linux.
2. Tidak semua sistem persamaan linier memiliki solusi unik, ada pula yang memiliki solusi banyak berbentuk parametrik, bahkan ada juga yang tidak memiliki solusi sama sekali.
3. Penggunaan sistem persamaan linier, determinan, dan aplikasinya dapat diterapkan dalam berbagai bidang keilmuan baik ilmu murni maupun terapan.

5.2 Saran

Tim penulis memiliki beberapa saran mengenai pengembangan program ini, yaitu

1. Algoritma yang dijelaskan dalam laporan tugas akhir ini masih memiliki banyak kemungkinan untuk dilakukan efisiensi. Oleh karena itu, dalam pengembangan program ini, masih bisa dilakukan efisiensi kinerja.
2. Program ini bisa dikembangkan lebih lanjut dengan menambahkan antarmuka pengguna grafis untuk memudahkan pengguna berinteraksi dengan program ini, jadi pengguna tidak perlu mengetik teks perintah yang ingin dijalankan dan lebih *user friendly*.
3. Menimbang diperlukannya program ini dalam banyak mata kuliah di ITB, sebaiknya program ini juga dipublikasikan, setidaknya kepada khalayak ITB supaya program ini memiliki nilai kebermanfaatan yang lebih besar.

5.3 Refleksi

Dari proses pembuatan program dan eksperimen yang dilakukan, penulis merasa bahwa:

1. Pembagian tugas di awal dan linimasa penggerjaan sudah ditetapkan secara rinci dan jelas dan pada pelaksanaannya, linimasa yang telah ditetapkan dapat diikuti sebagaimana yang telah direncanakan.
2. Komunikasi antar anggota kelompok berjalan dengan baik, sehingga tidak terjadi miskomunikasi atau kesalahpahaman selama penggerjaan.
3. Selama proses pembuatan program, ketika ditemukan ketidaksesuaian saat proses eksperimen, temuan langsung dikomunikasikan kepada anggota kelompok lainnya dan bersama-sama mencari solusi.
4. Program telah berhasil dibuat sesuai dengan spesifikasi dan telah berhasil dilakukan uji coba dengan beberapa *testcase*.

DAFTAR REFERENSI

- Anton, Howard dan Chris Rorres. 2010. *Elementary Linear Algebra: Application Version*. USA: John Wiley & Sons
- Aprianti, N. 2020. Determinan Matriks [Flscirc] _R Bentuk Khusus $N \times N$, ($N \geq 3$) Menggunakan Ekspansi Kofaktor (Doctoral dissertation, Universitas Islam Negeri Sultan Syarif Kasim Riau).
- Bremer, M. 2012. Math 261A -Spring 2012 Multiple Linear Regression.
- Lyawati. 2020. Aplikasi Aljabar Linear pada Kriptografi [Skripsi]. Yogyakarta (ID): Universitas Sanata Dharma.
- Marsudi, Marjono. 2012. *Aljabar Linear*. Malang: Universitas Brawijaya Press.
- Sibaroni, Yuliant. 2002. *Buku Ajar Aljabar Linear*. Bandung: Sekolah Tinggi Teknologi Telkom.