LAPORAN TUGAS BESAR
SISTEM PERSAMAAN LINEAR, DETERMINAN, DAN APLIKASINYA
IF2123 ALJABAR LINEAR DAN GEOMETRI

Disusun oleh:
Yahya (13518029)
Muhammad Cisco Zulfikar (13518073)
Naufal Prima Yoriko (13518146)

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

2019

DAFTAR ISI

BAB I
Deskripsi Masalah

Dengan menggunakan Bahasa Java, kami diminta membuat program untuk

1. Menghitung solusi SPL dengan metode eliminasi metode eliminasi Gauss, metode Eliminasi Gauss-Jordan, metode matriks balikan, dan kaidah Cramer (kaidah Cramer khusus untuk SPL dengan *n* pebuah dan *n* persamaan);
2. Menyelesaikan persoalan interpolasi;
3. Menghitung determinan matriks dengan berbagai cara yang disebutkan di atas, matriks kofaktor, dan matriks adjoin dari sebuah matriks $n \times n$.

Program yang dibuat harus memenuhi spesifikasi

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 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 $x_1 = t$.)
5. Untuk persoalan determinan, matriks balikan, matriks kofaktor, dan adjoin, maka luarannya sesuai dengan persoalan masing-masing
6. Untuk persoalan polinom interpolasi, luarannya adalah persamaan polinom dan taksiran nilai fungsi pada x yang diberikan.
7. Luaran program harus dapat ditampilkan pada layar komputer dan dapat disimpan ke dalam file.
8. Bahasa program yang digunakan adalah Java
9. Program tidak harus berbasis GUI, cukup text-based saja, namun boleh menggunakan GUI (memakai kakas Eclipse misalnya).
10. Program dapat dibuat dengan pilihan menu. Urutan menu dan isinya dipersilakan ditrancang masing-masing. Misalnya, menu:

```
MENU
1. Sistem Persamaaan Linier
2. Determinan
3. Matriks balikan
4. Matriks kofaktor
5. Adjoin
6. Interpolasi Polinom
7. 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 2 dan 3.

11. Sebagai pembanding, bandingkan solusi program anda dengan hasil dari Wolfram Alpha.

BAB II
Teori Dasar

1. Metode Eliminasi Gauss

Metode Eliminasi Gauss, juga dikenal sebagai reduksi baris, adalah algoritma dalam aljabar linier untuk memecahkan sistem persamaan linear. Biasanya dipahami sebagai urutan operasi yang dilakukan pada matriks koefisien yang sesuai. Metode ini juga dapat digunakan untuk menghitung determinan matriks, dan menghitung kebalikan dari matriks kuadrat terbalik. Metode ini dinamai dari nama penemunya Carl Friedrich Gauss (1777-1855).

Untuk melakukan metode ini, dapat menggunakan operasi baris elementer untuk memodifikasi matriks sampai sudut kiri bawah dari matriks diisi dengan nol, sebanyak mungkin. Ada tiga jenis operasi baris dasar:

a. Mengganti urutan dua baris
b. Mengalikan baris dengan angka yang bukan nol
c. Menambah suatu baris dengan baris yang lainnya

Sebagai contoh:

$\begin{aligned} 2x+y-z&=8\\ -3x-y+2z&=-11\\ -2x+y+2z&=-3 \end{aligned}$		$\left[\begin{array}{rrr\|r} 2 & 1 & -1 & 8\\ -3 & -1 & 2 & -11\\ -2 & 1 & 2 & -3 \end{array}\right]$
$\begin{aligned} 2x+y-z&=8\\ \tfrac{1}{2}y+\tfrac{1}{2}z&=1\\ 2y+z&=5 \end{aligned}$	$\begin{aligned} L_2+\tfrac{3}{2}L_1&\to L_2\\ L_3+L_1&\to L_3 \end{aligned}$	$\left[\begin{array}{rrr\|r} 2 & 1 & -1 & 8\\ 0 & \frac{1}{2} & \frac{1}{2} & 1\\ 0 & 2 & 1 & 5 \end{array}\right]$
$\begin{aligned} 2x+y-z&=8\\ \tfrac{1}{2}y+\tfrac{1}{2}z&=1\\ -z&=1 \end{aligned}$	$L_3+-4L_2\to L_3$	$\left[\begin{array}{rrr\|r} 2 & 1 & -1 & 8\\ 0 & \frac{1}{2} & \frac{1}{2} & 1\\ 0 & 0 & -1 & 1 \end{array}\right]$

Gambar 2.1. Contoh penggunaan Metode Eliminasi Gauss dengan menggunakan operasi baris elementer.

Matriks di atas diselesaikan dengan menggunakan operasi baris elementer. Matriks tersebut diselesaikan hingga matriks tersebut menjadi matriks dalam bentuk *echelon*.

2. Metode Eliminasi Gauss-Jordan

Metode Eliminasi Gauss-Jordan adalah algoritma dalam aljabar linear untuk memecahkan sistem persamaan linear dan merupakan lanjutan dari Metode Eliminasi Gauss. Alih-alih berhenti hingga matriks menjadi dalam bentuk *echelon*, dengan Metode Eliminasi Gauss-Jordan matriks diubah hingga menjadi matriks dalam bentuk *reduced echelon*. Matriks dengan bentuk *reduced echelon* adalah sebuah matriks dengan sudut kiri bawah dan sudut kanan atas dari matriks diisi dengan nol sebanyak mungkin, dan menyisakan diagonal utama dari matriks tersebut dengan satu.

Sebagai contoh (melanjutkan Gambar 2.1.):

$\begin{aligned} 2x+y&=7\\ \tfrac{1}{2}y&=\tfrac{3}{2}\\ -z&=1 \end{aligned}$	$\begin{aligned} L_2+\tfrac{1}{2}L_3&\to L_2\\ L_1-L_3&\to L_1 \end{aligned}$	$\left[\begin{array}{rrr\|r} 2 & 1 & 0 & 7\\ 0 & \frac{1}{2} & 0 & \frac{3}{2}\\ 0 & 0 & -1 & 1 \end{array}\right]$
$\begin{aligned} 2x+y&=7\\ y&=3\\ z&=-1 \end{aligned}$	$\begin{aligned} 2L_2&\to L_2\\ -L_3&\to L_3 \end{aligned}$	$\left[\begin{array}{rrr\|r} 2 & 1 & 0 & 7\\ 0 & 1 & 0 & 3\\ 0 & 0 & 1 & -1 \end{array}\right]$
$\begin{aligned} x&=2\\ y&=3\\ z&=-1 \end{aligned}$	$\begin{aligned} L_1-L_2&\to L_1\\ \tfrac{1}{2}L_1&\to L_1 \end{aligned}$	$\left[\begin{array}{rrr\|r} 1 & 0 & 0 & 2\\ 0 & 1 & 0 & 3\\ 0 & 0 & 1 & -1 \end{array}\right]$

Gambar 2.2. Contoh penggunaan Metode Eliminasi Gauss-Jordan.

3. Determinan

Dalam aljabar linear, determinan adalah nilai skalar yang dapat dihitung dari unsur-unsur matriks kuadrat dan mengkodekan sifat-sifat tertentu dari transformasi linear yang dijelaskan oleh matriks. Determinan matriks A dinotasikan dengan det (A), det A, atau | A |. Secara geometris, ini dapat dilihat sebagai faktor penskalaan volume dari transformasi linear yang dijelaskan oleh matriks.

Dalam kasus matriks 2 × 2, determinan matriks dapat didefinisikan sebagai:

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Gambar 2.3. Determinan matriks 2 × 2.

Dalam kasus matriks 3 × 3, determinan matriks dapat didefinisikan sebagai:

$$\begin{aligned} |A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} &= a \begin{vmatrix} \square & \square & \square \\ \square & e & f \\ \square & h & i \end{vmatrix} - b \begin{vmatrix} \square & \square & \square \\ d & \square & f \\ g & \square & i \end{vmatrix} + c \begin{vmatrix} \square & \square & \square \\ d & e & \square \\ g & h & \square \end{vmatrix} \\ &= a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix} \\ &= aei + bfg + cdh - ceg - bdi - afh. \end{aligned}$$

Gambar 2.4. Determinan matriks 3 × 3.

Untuk matriks n × n, determinan matriks bisa didapatkan dengan menggunakan persamaan Leibniz:

$$\det(A) = \sum_{\sigma \in S_n} \left(\operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma_i} \right)$$

Gambar 2.5. Persamaan Leibniz

Selain dengan cara di atas, determinan matriks n × n dapat dicari dengan menggunakan Metode Eliminasi Gauss dan mengalikan anggota matriks yang berada di diagonal utama.

4. Matriks Balikan

Matriks balikan atau matriks *inverse* adalah sebuah matriks yang dapat dikatakan sebagai kebalikan dari suatu matriks tertentu. Jika suatu matriks persegi A dikalikan dengan matriks balikannya (A^{-1}), maka akan menghasilkan matriks identitas.

Pada matriks 2 × 2 dan 3 × 3, matriks balikan bisa didapatkan dengan menggunakan persamaan

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

Gambar 2.6. Persamaan matriks balikan untuk matriks 2 × 2 dan 3 × 3

Selain cara diatas, matriks balikan dapat dicari dengan menggunakan operasi baris elementer pada matriks (A) bersamaan dengan matriks identitas (*I*) yang digabungkan dengan matriks (A) di sebelah kanan sehingga matriks yang awalnya berupa [A | *I*] berubah menjadi [*I* | A^{-1}].

5. Matriks Kofaktor

Sebelum mengetahui apa itu matriks kofaktor, perlu diketahui apa yang disebut dengan minor. Minor adalah determinan dari sebuah submatriks dari matriks yang lebih besar. Minor biasanya ditulis sebagai $M_{i, j}$.

Untuk mendapatkan minor dari baris ke-*i* dan kolom ke-*j* artinya semua elemen di baris ke-*i* dan kolom ke-*j* tersebut dihapus dan yang tersisa hanyalah submatriks dari matriks tersebut.

Sebagai contoh, misalkan terdapat matriks A dengan ukuran 3 × 3,

$$A = \begin{bmatrix} 3 & 1 & -4 \\ 2 & 5 & 6 \\ 1 & 4 & 8 \end{bmatrix},$$

Gambar 2.7. Contoh matriks A

maka, $M_{1,1}$ adalah

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

Gambar 2.8 Minor baris ke-1 dan kolom ke-1

Setelah diketahui minornya, maka dapat dicari kofaktor dari elemen (1, 1) dengan persamaan
$C_{i,j} = (-1)^{(i+j)} M_{i,j}$

Jadi, kofaktor dari elemen (1, 1) adalah

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

Gambar 2.9. Kofaktor elemen (1, 1)

Setelah mengetahui apa yang disebut dengan kofaktor, maka matriks kofaktor adalah matriks yang berisikan kofaktor elemen-elemen yang ada dalam matriks tersebut. Matriks kofaktor dengan ukuran n × n dapat digambarkan sebagai

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

Gambar 2.10. Matriks kofaktor n × n

6. Matriks Adjoin

Matriks adjoin adalah matriks kofaktor yang telah di*transpose*. Biasanya matriks adjoin dari sebuah matriks, misalkan matriks A, akan ditulis sebagai *adj(A)*.

7. Kaidah Cramer

Dalam aljabar linear, Kaidah Cramer adalah sebuah persamaan untuk menyelesaikan permasalahan sistem persamaan linear. Walaupun dalam sistem persamaan linear tersebut terdapat banyak variabel tanpa konstanta, Kaidah Cramer akan tetap berlaku selama sistem persamaan linear tersebut memiliki satu jawaban untuk setiap variabel. Kaidah Cramer dinamai setelah penemunya Gabriel Cramer (1704-1752).

Sebagai contoh,

$$\begin{cases} a_1 x + b_1 y + c_1 z = d_1 \\ a_2 x + b_2 y + c_2 z = d_2 \\ a_3 x + b_3 y + c_3 z = d_3 \end{cases}$$

Gambar 2.11. Contoh sistem persamaan linear dengan 3 variabel

diubah kedalam bentuk matriks sehingga menjadi

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

Gambar 2.12. Sistem persamaan linear yang diubah kedalam bentuk matriks

Lalu, untuk mendapatkan nilai *x*, *y*, dan *z* dapat dicari dengan cara

$$x = \frac{\begin{vmatrix} d_1 & b_1 & c_1 \\ d_2 & b_2 & c_2 \\ d_3 & b_3 & c_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}, \quad y = \frac{\begin{vmatrix} a_1 & d_1 & c_1 \\ a_2 & d_2 & c_2 \\ a_3 & d_3 & c_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}, \text{ and } z = \frac{\begin{vmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \\ a_3 & b_3 & d_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}$$

Gambar 2.13. Mencari *x, y* dan *z* menggunakan Kaidah Cramer

8. Interpolasi Polinom

Interpolasi polinom adalah sebuah cara untuk mencari sebuah persamaan polinomial yang jika digambar, maka garis tersebut akan melalui titik-titik yang ada. Menurut teorema interpolasi polinom, jika diberikan sebanyak *n* titik dalam dimensi-*xy* yang memiliki koordinat *x* yang berbeda, maka akan terdapat persamaan polinomial yang unik dengan derajat *n - 1* atau kurang yang dapat melalui titik-titik tersebut.

Secara general, misalkan terdapat titik-titik

$$(x_1, y_1), \quad (x_2, y_2), \quad (x_3, y_3), \ldots, \quad (x_n, y_n)$$

Gambar 2.14. Titik-titik sebanyak *n*

Karena terdapat *n* titik yang harus dipenuhi, maka harus dicari persamaan polinomial dengan bentuk

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

Gambar 2.15. Bentuk persamaan polinomial jika terdapat titik sebanyak *n*

Persamaan di atas menunjukkan bahwa koordinat-koordinat dari titik-titik harus memenuhi

$$\begin{aligned} a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_{n-1} x_1^{n-1} &= y_1 \\ a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_{n-1} x_2^{n-1} &= y_2 \\ \vdots \quad \vdots \quad \vdots \qquad\qquad \vdots \quad \vdots & \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_{n-1} x_n^{n-1} &= y_n \end{aligned}$$

Gambar 2.16.

Dalam persamaan-persamaan di atas, nilai dari seluruh *x* dan *y* diasumsikan diketahui sehingga dapat dituliskan sebagai sistem linear $a_0, a_1, a_2, \ldots, a_{n-1}$ yang tidak diketahui. Dari sini dapat dibuat matriks *augmented* untuk sistemnya menjadi

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} & y_1 \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} & y_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} & y_n \end{bmatrix}$$

Gambar 2.17.

sehingga demikian, interpolasi polinomial dapat dicari dengan menggunakan eliminasi Gauss-Jordan.

BAB III
Implementasi Program dengan Menggunakan Bahasa Java

class Matriks

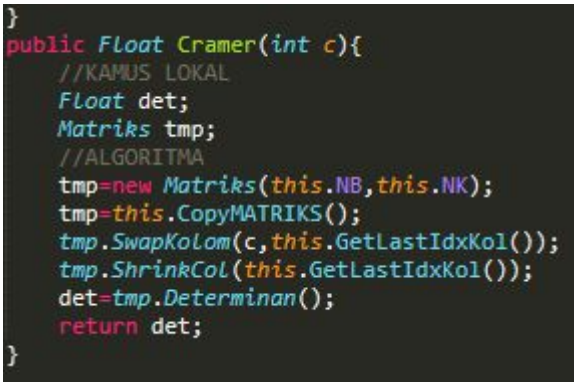
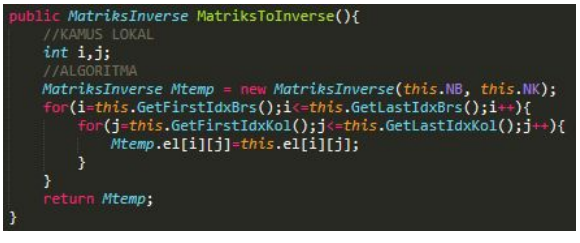
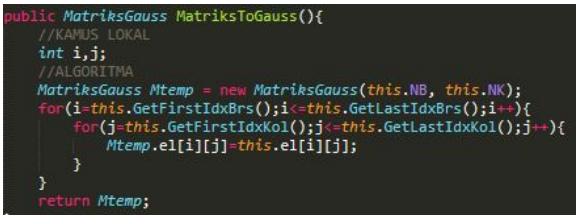
<table>
<tr>
<td>

```
public class Matriks{
    //Atribut
    int NB;
    int NK;
    float[][] el = new float[1001][1001];

    //METHOD:
    //Konstruktor
    Matriks(int nb, int nk){
    //KAMUS LOKAL
        int i,j;
    //ALGORITMA
        this.NB=nb;
        this.NK=nk;
        for(i=1;i<=1000;i++){
            for(j=1;j<=1000;j++){
                this.el[i][j]=0;
            }
        }
    }
```

</td>
<td>Type Matriks()</td>
</tr>
<tr>
<td>

```
//Input
public void BacaMATRIKS(){
    //KAMUS LOKAL
    Scanner in = new Scanner(System.in);
    int i,j;
    int m,n;
    //ALGORITMA
    m=in.nextInt();
    n=in.nextInt();
    this.NB=m;
    this.NK=n;
    for(i=1;i<=this.NB;i++){
        for(j=1;j<=this.NK;j++){
            this.el[i][j]=in.nextFloat();
        }
    }
}
//Output
public void TulisMATRIKS(){
    //KAMUS LOKAL
    int i,j;
    //ALGORITMA
    for(i=1;i<=this.NB;i++){
        for(j=1;j<=this.NK;j++){
            if(j!=this.NK){
                System.out.print(this.el[i][j]+" ");
            }else{
                System.out.printf("%.4f%n", this.el[i][j]);
            }
        }
    }
}
```

</td>
<td>input/output Matriks</td>
</tr>
</table>

	Determinan matriks
	Minor dari sebuah matriks
	Kaidah Cramer
	Matriks Inverse
	Metode Eliminasi Gauss

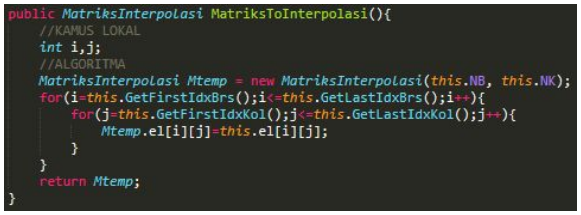
```
}

public Float Determinan(){
    float res;
    int i,j,k,cnt;
    if (this.NK==1){
        res = this.el[this.GetFirstIdxBrs()][this.GetFirstIdxBrs()];
    }
    else{
        res = 0;
        Matriks Ma;
        Ma=new Matriks(this.NB-1,this.NK-1);
        for(i=this.GetFirstIdxBrs();i<=this.GetLastIdxBrs();i++){
            for(j=this.GetFirstIdxBrs()+1;j<=this.GetLastIdxBrs();j++){
                cnt=0;
                for(k=this.GetFirstIdxBrs();k<=this.GetLastIdxBrs();k++){
                    if(k==i) cnt=-1;
                    else Ma.el[j-1][k+cnt] = this.el[j][k];
                }
            }
            if (this.el[this.GetFirstIdxBrs()][i]!=0){
                if(i%2==1)  res += this.el[this.GetFirstIdxBrs()][i]* Ma.Determinan();
                    else res -= this.el[this.GetFirstIdxBrs()][i]*Ma.Determinan();
            }
        }
    }
    return res;
}
```

```
}

public Float Minor(int r, int c){
    //KAMUS LOKAL
    Float res;
    int i,j;
    Matriks tmp;
    //ALGORITMA
    tmp=new Matriks(this.NB,this.NK);
    tmp=this.CopyMATRIKS();
    tmp.ShrinkCol(c);
    tmp.ShrinkRow(r);
    res=tmp.Determinan();
    return res;
}
```

```
}
public Float Cramer(int c){
    //KAMUS LOKAL
    Float det;
    Matriks tmp;
    //ALGORITMA
    tmp=new Matriks(this.NB,this.NK);
    tmp=this.CopyMATRIKS();
    tmp.SwapKolom(c,this.GetLastIdxKol());
    tmp.ShrinkCol(this.GetLastIdxKol());
    det=tmp.Determinan();
    return det;
}
```

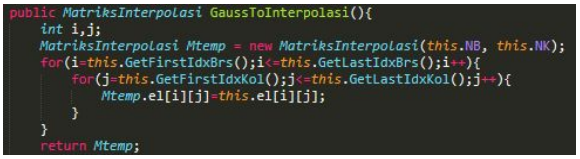
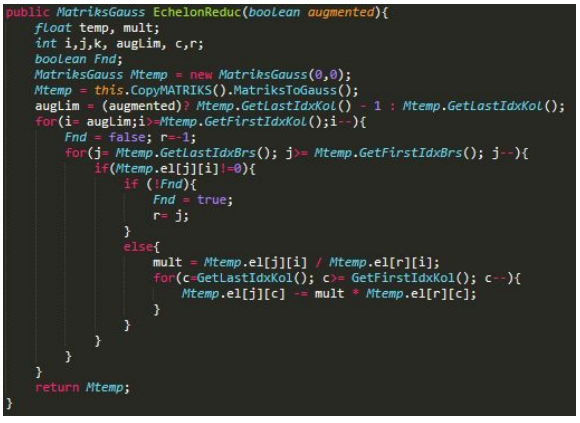
```
public MatriksInverse MatriksToInverse(){
    //KAMUS LOKAL
    int i,j;
    //ALGORITMA
    MatriksInverse Mtemp = new MatriksInverse(this.NB, this.NK);
    for(i=this.GetFirstIdxBrs();i<=this.GetLastIdxBrs();i++){
        for(j=this.GetFirstIdxKol();j<=this.GetLastIdxKol();j++){
            Mtemp.el[i][j]=this.el[i][j];
        }
    }
    return Mtemp;
}
```

```
public MatriksGauss MatriksToGauss(){
    //KAMUS LOKAL
    int i,j;
    //ALGORITMA
    MatriksGauss Mtemp = new MatriksGauss(this.NB, this.NK);
    for(i=this.GetFirstIdxBrs();i<=this.GetLastIdxBrs();i++){
        for(j=this.GetFirstIdxKol();j<=this.GetLastIdxKol();j++){
            Mtemp.el[i][j]=this.el[i][j];
        }
    }
    return Mtemp;
```

	Metode Interpolasi Polinomial
	Input dari file
	Save ke file

class MatriksGauss

	Gauss ke bentuk Matriks

	Metode Gauss ke Metode Interpolasi
	Matriks *echelon*
	Matriks *reduced echelon*

```
public MatriksInterpolasi GaussToInterpolasi(){
    int i,j;
    MatriksInterpolasi Mtemp = new MatriksInterpolasi(this.NB, this.NK);
    for(i=this.GetFirstIdxBrs();i<=this.GetLastIdxBrs();i++){
        for(j=this.GetFirstIdxKol();j<=this.GetLastIdxKol();j++){
            Mtemp.el[i][j]=this.el[i][j];
        }
    }
    return Mtemp;
```

```
public MatriksGauss Echelon(boolean augmented){
    float temp, mult;
    int i,j,k, auglim, p,q;
    boolean Fnd;
    MatriksGauss Mtemp = new MatriksGauss(0,0);
    Mtemp = this.CopyMATRIKS().MatriksToGauss();
    auglim = (augmented)? Mtemp.GetLastIdxKol() - 1 : Mtemp.GetLastIdxKol();
    j= Mtemp.GetFirstIdxBrs();
    for(i=Mtemp.GetFirstIdxKol(); i<=auglim && j<= Mtemp.GetLastIdxBrs(); i++){
        q = j-1; Fnd = false;
        do {
            q++;
            if (Mtemp.el[q][i]!=0){
                Fnd = true;
            }
        } while (!Fnd && q<Mtemp.GetLastIdxBrs());

        if(Fnd){
            if (q!=j){
                for(k = i; k<=Mtemp.GetLastIdxKol(); k++){
                    temp = Mtemp.el[q][k];
                    Mtemp.el[q][k] = Mtemp.el[j][k];
                    Mtemp.el[j][k] = temp;
                }
            }
            temp = Mtemp.el[j][i];
            for(k=i;k<=Mtemp.GetLastIdxKol();k++){
                Mtemp.el[j][k] /= temp;
            }

            for(k = j+1; k<= Mtemp.GetLastIdxBrs(); k++){
                if (Mtemp.el[k][i]!=0){
                    mult = Mtemp.el[k][i]/Mtemp.el[j][i];
                    for (p=i; p<=Mtemp.GetLastIdxKol(); p++){
                        Mtemp.el[k][p] -= mult * Mtemp.el[j][p];
                    }
                }
            }
            j++;
        }
    }
    return Mtemp;
```

```
public MatriksGauss EchelonReduc(boolean augmented){
    float temp, mult;
    int i,j,k, auglim, c,r;
    boolean Fnd;
    MatriksGauss Mtemp = new MatriksGauss(0,0);
    Mtemp = this.CopyMATRIKS().MatriksToGauss();
    auglim = (augmented)? Mtemp.GetLastIdxKol() - 1 : Mtemp.GetLastIdxKol();
    for(i= auglim;i>=Mtemp.GetFirstIdxKol();i--){
        Fnd = false; r=-1;
        for(j= Mtemp.GetLastIdxBrs(); j>= Mtemp.GetFirstIdxBrs(); j--){
            if(Mtemp.el[j][i]!=0){
                if (!Fnd){
                    Fnd = true;
                    r= j;
                }
                else{
                    mult = Mtemp.el[j][i] / Mtemp.el[r][i];
                    for(c=GetLastIdxKol(); c>= GetFirstIdxKol(); c--){
                        Mtemp.el[j][c] -= mult * Mtemp.el[r][c];
                    }
                }
            }
        }
    }
    return Mtemp;
}
```

<table>
<tr>
<td>

```
public void Solver(){
    int i,j, k,cnt;
    int[] nNull = new int[this.GetLastIdxBrs()+1];
    int[] fIdx = new int[this.GetLastIdxBrs()+1];
    int[] lIdx = new int[this.GetLastIdxBrs()+1];
    boolean loop = true;
    for(i=this.GetLastIdxBrs(); i>=this.GetFirstIdxBrs() && loop; i--){
        cnt=0;
        fIdx[i]=-1;
        for(j=this.GetLastIdxKol() - 1; j>=this.GetFirstIdxKol(); j--){
            if(this.el[i][j]!=0){
                if(cnt==0)fIdx[i] = j;
                lIdx[i]= j;
                cnt++;
            }
        }
        nNull[i]=cnt;
        if(cnt==0 && this.el[i][this.GetLastIdxKol()]!=0){
            System.out.printf("Tidak Ada Solusi Valid%n");
            loop = false;
        }
        else if(cnt==1){
            for(j=i-1; j>=this.GetFirstIdxBrs(); j--){
                this.el[j][this.GetLastIdxKol()] -=this.el[j][fIdx[i]] * this.el[i][this.GetLastIdxKol()];
                this.el[j][fIdx[i]] = 0;
            }
        }
    }
    this.writeGauss(nNull,fIdx,lIdx,loop);
    if (loop){
        System.out.println("Solusi dari SPL tersebut adalah");
        for(i=this.GetFirstIdxBrs(); i<=this.GetLastIdxBrs(); i++){
            if(nNull[i]==1){
                if (this.el[i][fIdx[i]]!=1) System.out.printf("%.2f", this.el[i][fIdx[i]]);
                System.out.printf("X%d ", fIdx[i]);
                System.out.printf("= %.3f", this.el[i][this.GetLastIdxKol()]);
                System.out.printf("%n");
            }
            else if(nNull[i]>1){
                cnt = 0;
                for(j=lIdx[i]; cnt < nNull[i]-1; j++){
                    if(this.el[i][j]!=0){
                        System.out.printf("X%d ", j);
                        System.out.printf("- t%d", j);
                        System.out.printf("%n");
                        cnt++;
                    }
                }
```

```
                System.out.printf("X%d =", fIdx[i]);
                if (this.el[i][this.GetLastIdxKol()]!=0) System.out.printf("%.3f", this.el[i][this.GetLastIdxKol()]);
                cnt = 0;
                for(j=lIdx[i]; cnt < nNull[i]-1; j++){
                    if(this.el[i][j]!=0){
                        System.out.printf(" - ");
                        if (this.el[i][j]!=1) System.out.printf("%.2f", this.el[i][j]);
                        System.out.printf("t%d", j);
                        cnt++;
                    }
                }
                System.out.printf("%n");
            }
        }
    }
}
```

</td>
<td>Solver</td>
</tr>
<tr>
<td>

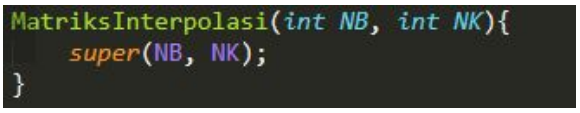
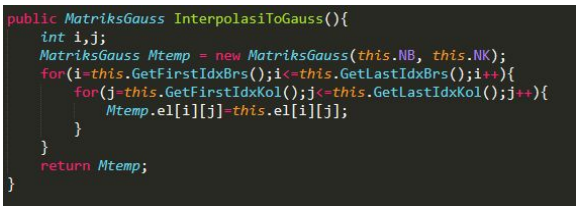
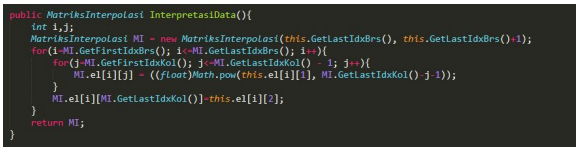
```
public void writeGauss(int[] nNull, int[] fIdx, int[] LIdx, boolean loop){
    //KAMUS LOKAL
    int i,j, k, cnt;
    StringBuffer strBuff;
    String s;
    //ALGORITMA
    strBuff=new StringBuffer();

    try (FileWriter writer = new FileWriter("outputGauss.txt");
        BufferedWriter bw = new BufferedWriter(writer)) {
    if(loop){
        strBuff.append("Solusi dari SPL tersebut adalah");
        s=strBuff.toString();
        bw.write(s);
        bw.write(System.lineSeparator());
        strBuff.delete(0,strBuff.length());
        for(i=this.GetFirstIdxBrs(); i<=this.GetLastIdxBrs(); i++){
            if(nNull[i]==1){
                if (this.el[i][fIdx[i]]!=1){
                    strBuff.append(this.el[i][fIdx[i]]);
                    System.out.printf("%.2f", this.el[i][fIdx[i]]);
                }
                strBuff.append("X");
                strBuff.append(fIdx[i]);
                System.out.printf("X%d ", fIdx[i]);
                strBuff.append(" = ");
                strBuff.append(this.el[i][this.GetLastIdxKol()]);
                System.out.printf("= %.3f", this.el[i][this.GetLastIdxKol()]);
                s=strBuff.toString();
                bw.write(s);
                bw.write(System.lineSeparator());
                System.out.printf("%n");
                strBuff.delete(0,strBuff.length());
            }
```

</td>
<td>Menuliskan matriks yang sudah diubah dengan metode eliminasi Gauss</td>
</tr>
</table>

```
        else if(nNull[i]>1){
            cnt = 0;
            for(j=lIdx[i]; cnt < nNull[i]-1; j++){
                if(this.el[i][j]!=0){
                    strBuff.append("X");
                    strBuff.append(j);
                    System.out.printf("X%d ", j);
                    strBuff.append(" = ");
                    strBuff.append(j);
                    System.out.printf("= t%d", j);
                    System.out.printf("%n");
                    s=strBuff.toString();
                    bw.write(s);
                    bw.write(System.lineSeparator());
                    strBuff.delete(0,strBuff.length());
                    cnt++;
                }
            }
            strBuff.append("X");
            strBuff.append(fIdx[i]);
            strBuff.append(" =");
            System.out.printf("X%d =", fIdx[i]);
            if (this.el[i][this.GetLastIdxKol()]!=0){
                strBuff.append(this.el[i][this.GetLastIdxKol()]);
                System.out.printf("%.3f", this.el[i][this.GetLastIdxKol()]);
            }
            cnt = 0;
            for(j=lIdx[i]; cnt < nNull[i]-1; j++){
                if(this.el[i][j]!=0){
                    strBuff.append(" - ");
                    System.out.printf(" - ");
                    if (this.el[i][j]!=1) {
                        strBuff.append(this.el[i][j]);
                        System.out.printf("%.2f", this.el[i][j]);
                    }
                        strBuff.append("t");
                        strBuff.append(j);
                        System.out.printf("t%d", j);
                        cnt++;
                }
            }
```

```
                }
            }
        }
        else{
            strBuff.append("Tidak ada solusi valid");
            s=strBuff.toString();
            bw.write(s);
        }
        }
        catch (IOException e) {
            System.err.format("IOException: %s%n", e);
        }
}
```

Class MatriksInterpolasi

	Konstruktor
	Matriks interpolasi ke Matriks Gauss
	Bentuk Pair ke Matriks
	Save ke file

	Lanjutan save ke file

```
        bw.write(System.lineSeparator());
        strBuff.delete(0,strBuff.length());
            System.out.println("Koefisien X tidak unik");
            loop = false;
        }
        else if(cnt==-1){
            for(j=i-1; j>=this.GetFirstIdxBrs(); j--){
                this.el[j][this.GetLastIdxKol()]-=this.el[j][fIdx] * this.el[i][this.GetLastIdxKol()];
                this.el[j][fIdx] = 0;
            }
        }
    }
    if (loop){
    strBuff.append("Tidak Ada Solusi Valid");
        s=strBuff.toString();
        bw.write(s);
        bw.write(System.lineSeparator());
        strBuff.delete(0,strBuff.length());
        System.out.println("Solusi dari Interpolasi titik-titik tersebut adalah");
        strBuff.append("f(X) = ");

        System.out.print("f(X) = ");
        cnt = 0;
        for(i=this.GetFirstIdxBrs();i<=this.GetLastIdxBrs();i++){
            if(this.el[i][this.GetLastIdxKol()]!= 0){
                loop = true;
                for(j=this.GetFirstIdxKol();j<=this.GetLastIdxKol()-1 && loop;j++){
                    if(this.el[i][j]!=0){
                        if(cnt!=0 && this.el[i][this.GetLastIdxKol()]>0) {
```

```
                            strBuff.append(" + ");
                            System.out.printf(" + ");
                            }
                            if(this.el[i][this.GetLastIdxKol()]<0){
                            strBuff.append(" - ");
                            System.out.printf(" - ");
                            }
                            strBuff.append(Math.abs(this.el[i][this.GetLastIdxKol()]));
                            System.out.printf("%.2f", Math.abs(this.el[i][this.GetLastIdxKol()]));
                            if(this.GetLastIdxKol()-j-1 != 0) {
                            strBuff.append("X");
                            strBuff.append(this.GetLastIdxKol()-j-1);
                            System.out.printf("X^%d", this.GetLastIdxKol()-j-1);
                            }
                            loop = false;
                            cnt++;
                        }
                    }
                }
            }
            s=strBuff.toString();
            bw.write(s);
            bw.write(System.lineSeparator());
            System.out.printf("%n");
        }

    } catch (IOException e) {
        System.err.format("IOException: %s%n", e);
    }
}
```

BAB IV
Eksperimen

1. **Menemukan solusi SPL $Ax = b$**

a. $$A = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 5 & -7 & -5 \\ 2 & -1 & 1 & 3 \\ 5 & 2 & -4 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -2 \\ 4 \\ 6 \end{bmatrix}$$

Solusi yang diberikan oleh Wolfram Alpha:
Tidak valid karena det(A) = 0
Solusi program yang dibuat:

```
1.0 0.0 0.0 0.6666665 1.666667
0.0 1.0 0.0 -2.6666665 0.33333302
0.0 0.0 1.0 -1.0 1.0
0.0 0.0 0.0 0.0 1.0
Tidak Ada Solusi Valid
```

b. $$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & -3 & 0 \\ 2 & -1 & 0 & 1 & -1 \\ -1 & 2 & 0 & -2 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 6 \\ 5 \\ -1 \end{bmatrix}$$

Solusi yang diberikan oleh Wolfram Alpha:
Gagal output.
Solusi program yang dibuat:

```
1.0 0.0 0.0 0.0 -1.0 3.0
0.0 1.0 0.0 0.0 -2.0 0.0
0.0 0.0 0.0 1.0 -1.0 -1.0
0.0 0.0 0.0 0.0 0.0 0.0
Solusi dari SPL tersebut adalah
X1 = t1
X5 =3.000 - t1
X2 = t2
X5 = - t2
X4 = t4
X5 =-1.000 - t4
```

c. $$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$$

Solusi yang diberikan oleh Wolfram Alpha:
Gagal output
Solusi program yang dibuat:

```
0.0 1.0 0.0 0.0 0.0 1.0 1.0
0.0 0.0 0.0 1.0 0.0 1.0 -2.0
0.0 0.0 0.0 0.0 1.0 -1.0 1.0
Solusi dari SPL tersebut adalah
X2 = t2
X6 =1.000 - t2
X4 = t4
X6 =-2.000 - t4
X5 = t5
X6 =1.000 - t5
```

d.

$$H=\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n+1} \end{bmatrix}, \quad b=\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Matriks H adalah matriks Hilbert dengan $n = 6$ atau $n = 10$.

Solusi yang diberikan oleh Wolfram Alpha:

$n = 6$

$$6\begin{pmatrix} 6 \\ -105 \\ 560 \\ -1260 \\ 1260 \\ -462 \end{pmatrix}$$

$n = 10$

$$10\begin{pmatrix} 10 \\ -495 \\ 7920 \\ -60\,060 \\ 252\,252 \\ -630\,630 \\ 960\,960 \\ -875\,160 \\ 437\,580 \\ -92\,378 \end{pmatrix}$$

Solusi program yang dibuat:

$n = 6$

```
1.0 0.0 0.0 0.0 0.0 0.0 11.2846
0.0 1.0 0.0 0.0 0.0 0.0 53.0499
0.0 0.0 1.0 0.0 0.0 0.0 -1170.3965
0.0 0.0 0.0 1.0 0.0 0.0 4064.3052
0.0 0.0 0.0 0.0 1.0 0.0 -5142.9233
0.0 0.0 0.0 0.0 0.0 1.0 2194.9790
Solusi dari SPL tersebut adalah
X1 = 11.285
X2 = 53.050
X3 = -1170.396
X4 = 4064.305
X5 = -5142.923
X6 = 2194.979
```

$n = 10$

```
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 61.3433
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -1254.3906
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 6489.4556
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 -9697.0977
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 -4575.5996
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 10235.9766
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 27045.0234
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 -48029.9570
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 18291.4629
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 1448.8837
Solusi dari SPL tersebut adalah
X1 = 61.343
X2 = -1254.391
X3 = 6489.456
X4 = -9697.098
X5 = -4575.600
X6 = 10235.977
X7 = 27045.023
X8 = -48029.957
X9 = 18291.463
X10 = 1448.884
```

2. SPL berbentuk matriks *augmented*

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

$$\begin{bmatrix} 2 & 0 & 8 & 0 & 8 \\ 0 & 1 & 0 & 4 & 6 \\ -4 & 0 & 6 & 0 & 6 \\ 0 & -2 & 0 & 3 & -1 \\ 2 & 0 & -4 & 0 & -4 \\ 0 & 1 & 0 & -2 & 0 \end{bmatrix}$$

SPL berbentuk …

a.

$$\begin{aligned} 8x_1 + x_2 + 3x_3 + 2x_4 &= 0 \\ 2x_1 + 9x_2 - x_3 - 2x_4 &= 1 \\ x_1 + 3x_2 + 2x_3 - x_4 &= 2 \\ x_1 + 6x_3 + 4x_4 &= 3 \end{aligned}$$

Solusi yang diberikan oleh Wolfram Alpha:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & -0.224324 \\ 0 & 1 & 0 & 0 & 0.182432 \\ 0 & 0 & 1 & 0 & 0.709459 \\ 0 & 0 & 0 & 1 & -0.258108 \end{pmatrix}$$

Solusi program yang dibuat:

```
1.0 0.0 0.0 0.0 -0.2243
0.0 1.0 0.0 0.0 0.1824
0.0 0.0 1.0 0.0 0.7095
0.0 0.0 0.0 1.0 -0.2581
Solusi dari SPL tersebut adalah
X1 = -0.224
X2 = 0.182
X3 = 0.709
X4 = -0.258
```

b.

$$\begin{aligned} x_7 + x_8 + x_9 &= 13.00 \\ x_4 + x_5 + x_6 &= 15.00 \\ x_1 + x_2 + x_3 &= 8.00 \\ 0.04289(x_3 + x_5 + x_7) + 0.75(x_6 + x_8) + 0.61396x_9 &= 14.79 \\ 0.91421(x_3 + x_5 + x_7) + 0.25(x_2 + x_4 + x_6 + x_8) &= 14.31 \\ 0.04289(x_3 + x_5 + x_7) + 0.75(x_2 + x_4) + 0.61396x_1 &= 3.81 \\ x_3 + x_6 + x_9 &= 18.00 \\ x_2 + x_5 + x_8 &= 12.00 \\ x_1 + x_4 + x_7 &= 6.00 \\ 0.04289(x_1 + x_5 + x_9) + 0.75(x_2 + x_6) + 0.61396x_3 &= 10.51 \\ 0.91421(x_1 + x_5 + x_9) + 0.25(x_2 + x_4 + x_6 + x_8) &= 16.13 \\ 0.04289(x_1 + x_5 + x_9) + 0.75(x_4 + x_8) + 0.61396x_7 &= 7.04 \end{aligned}$$

Solusi yang diberikan oleh Wolfram Alpha:

Gagal output

Solusi program yang dibuat:

```
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 3.0031
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.9945
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 4.0024
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.9977
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 5.0055
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 8.9968
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 1.9991
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 6.0000
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 5.0009
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -0.0107
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0000
Tidak Ada Solusi Valid
```

3. **Untuk persoalan determinan, matriks balikan, dan matriks kofaktor, cari masing-masing dua buah matriks yang berukuran 5 × 5 dan 10 × 10.**

Matriks 5 × 5

$$\begin{pmatrix} 7 & 12 & 15 & -6 & -5 \\ 9 & -10 & -11 & 7 & -5 \\ 14 & -5 & 8 & -4 & 9 \\ 14 & 15 & -2 & 7 & 14 \\ -11 & 1 & 1 & 0 & -15 \end{pmatrix}$$

Solusi yang diberikan oleh Wolfram Alpha:

Determinan 5 × 5

$$\begin{vmatrix} 7 & 12 & 15 & -6 & -5 \\ 9 & -10 & -11 & 7 & -5 \\ 14 & -5 & 8 & -4 & 9 \\ 14 & 15 & -2 & 7 & 14 \\ -11 & 1 & 1 & 0 & -15 \end{vmatrix} = 263036$$

Matriks inverse 5 × 5

$$\frac{1}{263036}\begin{pmatrix} 16\,966 & 14\,453 & -10\,630 & -5985 & -22\,437 \\ 17\,772 & 2132 & -27\,848 & -2812 & -25\,968 \\ -30\,706 & -18\,499 & 61\,998 & 27\,607 & 79\,367 \\ -54\,180 & -15\,380 & 78\,504 & 51\,376 & 118\,240 \\ -13\,304 & -11\,690 & 10\,072 & 6042 & 2478 \end{pmatrix}$$

Matriks kofaktor 5 × 5

$$\begin{pmatrix} 16\,966 & 17\,772 & -30\,706 & -54\,180 & -13\,304 \\ 14\,453 & 2132 & -18\,499 & -15\,380 & -11\,690 \\ -10\,630 & -27\,848 & 61\,998 & 78\,504 & 10\,072 \\ -5985 & -2812 & 27\,607 & 51\,376 & 6042 \\ -22\,437 & -25\,968 & 79\,367 & 118\,240 & 2478 \end{pmatrix}$$

Solusi program yang dibuat:

Determinan 5 × 5

Matriks inverse 5 × 5

Matriks kofaktor 5 × 5

Matriks 10 × 10

$$\begin{bmatrix} 64 & 109 & 0 & 93 & 52 & 91 & -16 & 47 & -11 & 12 \\ 93 & 49 & -26 & 16 & 56 & 30 & 118 & 30 & -22 & 86 \\ 4 & -1 & 86 & 69 & -2 & 31 & 110 & 78 & 46 & 45 \\ 30 & 7 & -23 & 106 & -34 & 45 & 49 & 106 & 74 & 86 \\ 17 & 54 & 25 & -8 & 77 & 71 & 85 & 110 & 47 & -12 \\ 99 & 69 & -4 & 38 & 119 & 51 & 106 & 77 & 55 & 7 \\ 98 & 71 & 105 & 55 & 62 & 115 & 119 & 89 & 67 & 70 \\ 50 & 41 & 95 & 79 & 3 & 43 & 111 & 37 & 57 & 85 \\ 85 & 89 & -33 & 48 & 112 & 15 & 7 & 1 & 49 & 74 \\ -26 & 123 & 39 & -20 & -15 & -33 & 36 & 6 & -12 & 51 \end{bmatrix}$$

Solusi yang diberikan oleh Wolfram Alpha:

Determinan 10 × 10

$$\begin{vmatrix} 64 & 109 & 0 & 93 & 52 & 91 & -16 & 47 & -11 & 12 \\ 93 & 49 & -26 & 16 & 56 & 30 & 118 & 30 & -22 & 86 \\ 4 & -1 & 86 & 69 & -2 & 31 & 110 & 78 & 46 & 45 \\ 30 & 7 & -23 & 106 & -34 & 45 & 49 & 106 & 74 & 86 \\ 17 & 54 & 25 & -8 & 77 & 71 & 85 & 110 & 47 & -12 \\ 99 & 69 & -4 & 38 & 119 & 51 & 106 & 77 & 55 & 7 \\ 98 & 71 & 105 & 55 & 62 & 115 & 119 & 89 & 67 & 70 \\ 50 & 41 & 95 & 79 & 3 & 43 & 111 & 37 & 57 & 85 \\ 85 & 89 & -33 & 48 & 112 & 15 & 7 & 1 & 49 & 74 \\ -26 & 123 & 39 & -20 & -15 & -33 & 36 & 6 & -12 & 51 \end{vmatrix}$$ = -64 387 933 250 145 628 896

Matriks inverse 10 × 10

Gagal output.

Matriks kofaktor 10 × 10

Gagal output.

Solusi program yang dibuat:

Determinan 10 × 10

-

Matriks inverse 10 × 10

-

Matriks kofaktor 10 × 10

-

4. **Interpolasi**

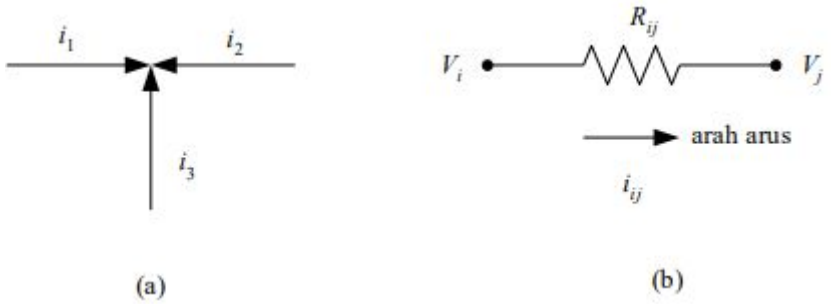
Dalam sebuah rangkaian listrik berlaku hukum-hukum arus Kirchoff menyatakan bahwa jumlah aljabar dari semua arus yang memasuki suatu simpul (Gambar 4.4a) haruslah nol:

$$\Sigma i = 0$$

Dalam hal ini, semua arus i yang memasuki simpul dianggap bertanda positif. Sedangkan hukum Ohm (Gambar 1) menyatakan bahwa arus i yang melalui suatu tahanan:

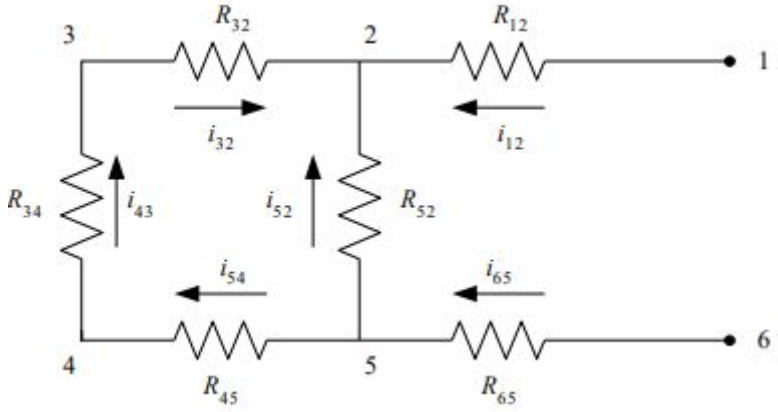
$$i_{ij} = \frac{V_i - V_j}{R_{ij}}$$

yang dalam hal ini V adalah tegangan dan R adalah tahanan.



Gambar 4.1. (a) Hukum Kirchoff, (b) Hukum Ohm

Diberikan sebuah rangkaian listrik dengan 6 buah tahanan seperti pada Gambar 2 Anda diminta menghitung arus pada masing-masing rangkaian.



Gambar 4.2. Rangkaian listrik dengan 6 buah tahanan

Arah arus dimisalkan seperti diatas. Dengan hukum Kirchoff diperoleh persamaan-persamaan berikut.

$$
\begin{aligned}
i_{12} &+ i_{52} &+ i_{32} &= 0\\
i_{65} &- i_{52} &- i_{54} &= 0\\
i_{43} &- i_{32} & &= 0\\
i_{54} &- i_{43} & &= 0
\end{aligned}
$$

Dari hukum Ohm didapat:

$$
\begin{aligned}
i_{32}\,R_{32} &- V_3 &+ V_2 &= 0\\
i_{43}\,R_{43} &- V_4 &+ V_3 &= 0\\
i_{65}\,R_{65} & &+ V_5 &= 0\\
i_{12}\,R_{12} & &+ V_2 &= 0\\
i_{54}\,R_{54} &- V_5 &+ V_4 &= 0\\
i_{52}\,R_{52} &- V_5 &+ V_2 &= 0
\end{aligned}
$$

Tentukan

i_{12} , i_{52} , i_{32} , i_{65} , i_{54} , i_{13} , V_2 , V_3 , V_4 , V_5

bila diketahui

R_{12} = 5 ohm , R_{52} = 10 ohm , R_{32} = 10 ohm
R_{65} = 20 ohm , R_{54} = 15 ohm , R_{14} = 5 ohm.
V_1 = 200 volt , V_6 = 0 volt.

Solusi yang diberikan oleh Wolfram Alpha:

Gagal output

Solusi program yang dibuat:

```
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0000
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0000
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0000
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0000
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0000
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0000
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0000
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0000
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0000
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -0.0 -0.0000
Solusi dari SPL tersebut adalah
X1 = 0.000
X2 = 0.000
X3 = 0.000
X4 = 0.000
X5 = 0.000
X6 = 0.000
X8 = 0.000
X9 = 0.000
X10 = 0.000
X11 = -0.000
```

i12	i52	i32	i65	i54	i43	v1	v2	v3	v4	v5	v6	kons
1	1	1	0	0	0	0	0	0	0	0	0	0
0	-1	0	1	-1	0	0	0	0	0	0	0	0
0	0	-1		0	1	0	0	0	0	0	0	0
0	0	0	0	1	-1	0	0	0	0	0	0	0
0	0	10	0	0	0	0	1	-1	0	0	0	0
0	0	0	0	0	5	0	0	1	-1	0	0	0
0	0	0	20	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	15	0	0	0	0	1	-1	0	0
0	10	0	0	0	0	0	1	0	0	-1	0	0
1	2	3	4	5	6	7	8	9	10	11	12	

5. Interpolasi

Jumlah penduduk Jawa Barat dari tahun 1971 hingga 2019 (dibulatkan ke juta) adalah sebagai berikut:

Tahun	Jumlah x 10^6)
1971	21,6
1980	27,4
1990	35,4
1995	39,2
2000	35,7
2010	43,2
2015	46,7
2019	49.1

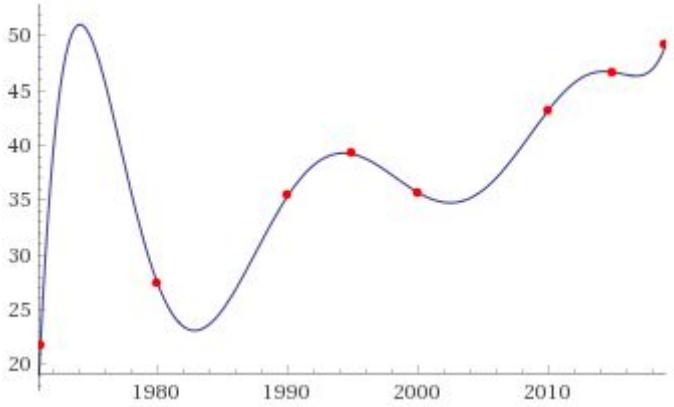
Berdasarkan data tersebut prediksilah jumlah penduduk Jawa Barat pada tahun 1975, 1983, 1992, 2005, 2012 (atau nilai lain sesuai masukan user) dengan menggunakan polinom interpolasi.

Solusi yang diberikan oleh Wolfram Alpha:

Persamaan interpolasi polinomial

$$6.5022\times10^{-8}\,x^7 - 0.000909145\,x^6 + 5.44783\,x^5 - 18\,135.8\,x^4 + 3.62239\times10^{7}\,x^3 - 4.3411\times10^{10}\,x^2 + 2.8902\times10^{13}\,x - 8.24655\times10^{15}$$

Grafik persamaan interpolasi polinomial



Solusi program yang dibuat:

```
Solusi dari Interpolasi titik-titik tersebut adalah
f(X) =  - 0.0000000002X^7 + 0.0000003284X^6 + 0.0033487007X^5 - 9.7198257446X^4 + 1379.2031250000X^3 + 10951647.0000000000X^2 +
 8164233216.0000000000X^1 - 18560443219968.0000000000
```

6. Menyederhanakan fungsi

$$f(x) = \frac{x^2 + \sqrt{x}}{e^x + x}$$

dengan polinom interpolasi derajat *n* di dalam selang [0, 2]. Sebagai contoh, jika *n* = 5, maka titik-titik *x* yang diambil di dalam selang [0, 2] berjarak *h* = (2 – 0)/5 = 0.4

BAB V
Kesimpulan, Saran, dan Refleksi

Kesimpulan

Dengan menggunakan Java, kita dapat membuat program matematis yang dapat membantu dalam topik aljabar linear.

Saran

Floating point jika dipangkatkan di fungsi interpolasi masih terdapat kekacauan sehingga masih harus dibenarkan. Algoritma mencari determinan belum optimal sehingga diperlukan riset algoritma yang lebih optimal. Parametrik belum sempurna disebabkan variabel parametrik belum minimum.

Refleksi

Mengoding dengan menggunakan Java lumayan ribet walaupun mirip dengan bahasa C.

DAFTAR REFERENSI

Anton, Howard and Chris Rorres. 2013. *Elementary Linear Algebra: Applications Version, 11th Edition.* New York: Wiley. ISBN: 978-1-118-43441-3.

Gaussian Elimination. Wikipedia. (tersedia secara online: *https://en.wikipedia.org/wiki/Gaussian_elimination*).

Cramer's Rule. Wikipedia. (tersedia secara online: *https://en.wikipedia.org/wiki/Cramer%27s_rule*).

WolframAlpha. (tersedia secara online: *https://www.wolframalpha.com*)