LAPORAN TUGAS BESAR I

IF2123 ALJABAR LINIER DAN GEOMETRI

SISTEM PERSAMAAN LINIER, DETERMINAN, DAN APLIKASINYA

SEMESTER I 2019/2020

Disusun oleh:

Rinda Nur Hafizha 13516151

Harry Prabowo 13517094

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

2019

BAB I

DESKRIPSI MASALAH

A. Spesifikasi Masalah

(i) Sistem persamaan linier (SPL) Ax=b dengan n peubah (variable) dan m persamaan adalah berbentuk

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n &= b_2 \\ \vdots \qquad\qquad & \quad \vdots \\ \vdots \qquad\qquad & \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n &= b_m \end{aligned}$$

yang dalam hal ini x_i adalah peubah, a_{ij} dan b_i adalah koefisien $\in$ R. Sembarang SPL dapat diselesaikan dengan beberapa metode, yaitu metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan ($x= A^{-1}b$), dan kaidah Cramer (khusus untuk SPL dengan n pebuah dan n persamaan). Solusi sebuah SPL mungkin tidak ada, banyak, atau hanya satu (unik/tunggal).

(ii) Sebuah matriks M berukuran *n* x *n*

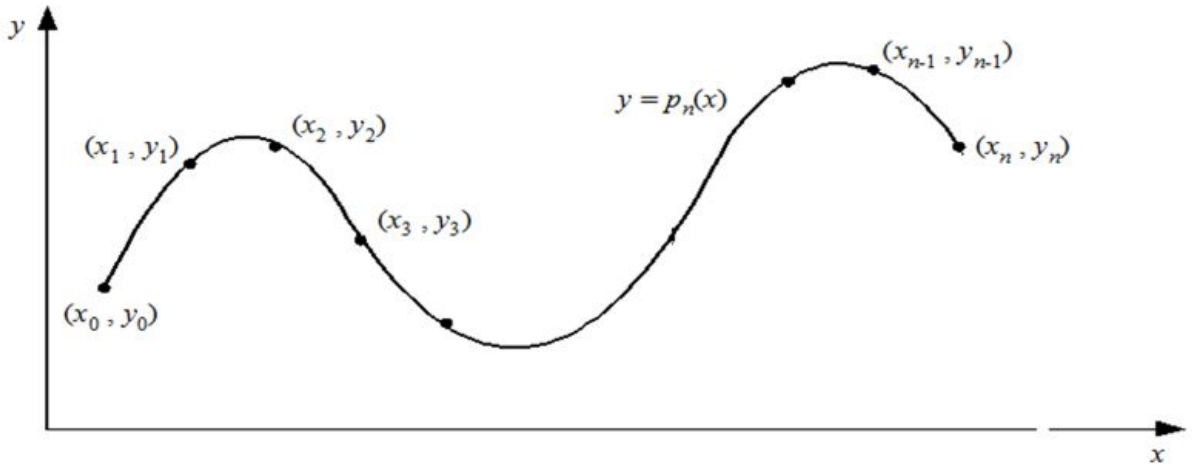
$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{bmatrix}$$

Determinannya adalah

$$\det(M) = \begin{vmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{vmatrix}$$

(iii) Balikan (inverse) matriks M berukuran n x n dapat dihitung dengan banyak cara: menggunakan metode eliminasi Gauss-Jordan dan menggunakan matriks adjoin.

Kembali ke sistem persamaan linier (SPL). SPL memiliki banyak aplikasi dalam bidang sains dan rekayasa, salah satunya adalah mengestimasi nilai fungsi dengan interpolasi polinom. Persoalan interpolasi polinom adalah sebagai berikut: Diberikan n+1 buah titik berbeda, (x_0, y_0), (x_1, y_1), ..., (x_n, y_n). Tentukan polinom $p_n(x)$ yang menginterpolasi (melewati) semua titik-titik tersebut sedemikian rupa sehingga $y_i = p_n(x_i)$ untuk $i = 0, 1, 2, ..., n$.



Setelah polinom interpolasi $p_n(x)$ ditemukan, $p_n(x)$ dapat digunakan untuk menghitung perkiraan nilai y di sembarang titik di dalam selang $[x_0, x_n]$.

Polinom interpolasi derajat n yang menginterpolasi titik-titik (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) adalah berbentuk $p_n(x) = a_0 + a_1x + a_2x_2 + ... + a_nx_n$. Jika hanya ada dua titik, (x_0, y_0) dan (x_1, y_1), maka polinom yang menginterpolasi kedua titik tersebut adalah $p_1(x) = a_0 + a_1x$ yaitu berupa persamaan garis lurus. Jika tersedia tiga titik, (x_0, y_0), (x_1, y_1), dan (x_2, y_2), maka polinom yang menginterpolasi ketiga titik tersebut adalah $p_2(x) = a_0 + a_1x + a_2x_2$ atau persaman kuadrat dan kurvanya berupa parabola. Jika tersedia empat titik, (x_0, y_0), (x_1, y_1), (x_2, y_2), dan (x_3, y_3), polinom yang menginterpolasi keempat titik tersebut adalah $p_3(x) = a_0 + a_1x + a_2x_2 + a_3x_3$, demikian seterusnya. Dengan cara yang sama kita dapat membuat polinom interpolasi berderajat n untuk n yang lebih tinggi asalkan tersedia (n+1) buah titik data. Dengan

menyulihkan (x_i, y_i) ke dalam persamaan polinom $p_n(x) = a_0 + a_1x + a_2x_2 + ... + a_nx_n$ untuk $i = 0, 1, 2, ..., n$, akan diperoleh n buah sistem persamaan lanjar dalam $a_0, a_1, a_2, ..., a_n$,

$$
\begin{aligned}
&a_0 + a_1x_0 + a_2{x_0}^2 + ... + a_n {x_0}^n = y_0 \\
&a_0 + a_1x_1 + a_2{x_1}^2 + ... + a_n {x_1}^n = y_1 \\
&... \qquad\qquad ... \\
&a_0 + a_1x_n + a_2{x_n}^2 + ... + a_n {x_n}^n = y_n
\end{aligned}
$$

Solusi sistem persamaan lanjar ini, yaitu nilai $a_0, a_1, ..., a_n$, diperoleh dengan menggunakan metode eliminasi Gauss yang sudah anda pelajari. Sebagai contoh, misalkan diberikan tiga buah titik yaitu (8.0, 2.0794), (9.0, 2.1972), dan (9.5, 2.2513). Tentukan polinom interpolasi kuadratik lalu estimasi nilai fungsi pada x= 9.2. Polinom kuadratik berbentuk $p_2(x) = a_0 + a_1x + a_2x_2$. Dengan menyulihkan ketiga buah titik data ke dalam polinom tersebut, diperoleh sistem persamaan lanjar yang terbentuk adalah

$$
\begin{aligned}
&a_0 + 8.0a_1 + 64.00a_2 = 2.0794 \\
&a_0 + 9.0a_1 + 81.00a_2 = 2.1972 \\
&a_0 + 9.5a_1 + 90.25a_2 = 2.2513
\end{aligned}
$$

Penyelesaian sistem persamaan dengan metode eliminasi Gauss menghasilkan a_0= 0.6762, a_1 = 0.2266, dan a_2 = -0.0064. Polinom interpolasi yang melalui ketiga buah titik tersebut adalah $p_2(x) = 0.6762 + 0.2266x - 0.0064x_2$. Dengan menggunakan polinom ini, maka nilai fungsi pada x = 9.2 dapat ditaksir sebagai berikut:

$$p_2(9.2) = 0.6762 + 0.2266(9.2) - 0.0064(9.2)^2 = 2.2192.$$

BAB II

TEORI SINGKAT

A. Metode Eliminasi Gauss

Metode Eliminasi Gauss adalah suatu metode untuk mengoperasikan nilai-nilai di dalam

matriks sehingga menjadi matriks yang lebih sederhana lagi. Operasi yang dilakukan adalah operasi baris elementer, yaitu suatu operasi aljabar yang dilakukan pada suatu sistem yang tidak akan mengubah set solusinya dan menghasilkan sistem yang jauh lebih sederhana. Metode ini dapat diterapkan kepada suatu sistem persamaan linear yang disajikan dalam bentuk matriks *augmented* untuk mencari solusi dari sistem tersebut. Biasanya, operasi aljabar yang dilakukan adalah:

1. Kalikan sebuah baris dengan suatu konstanta bukan nol
2. Tukar dua baris
3. Tambahkan sebuah baris yang sudah dikalikan dengan konstanta kepada baris lain.

Eliminasi Gauss akan menghasilkan bentuk *row echelon*, yaitu matriks yang bentuknya seperti berikut., di mana tanda bintang merupakan angka apapun.

$$\begin{bmatrix} 1 & * & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 1 & * \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & * & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & * & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & * & * & * & * & * & * & * \\ 0 & 0 & 0 & 1 & * & * & * & * & * \\ 0 & 0 & 0 & 0 & 1 & * & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 1 & * & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & * \end{bmatrix}$$

Gambar 1. Matriks yang memiliki bentuk *row echelon*

Sebuah matriks yang memiliki bentuk *row echelon* memiliki sifat seperti berikut.

1. Bila suatu baris tidak seluruhnya terdiri dari nol, maka angka bukan nol pertama pada baris tersebut harus bernilai 1, atau bisa disebut memiliki *leading* 1.
2. Bila ada suatu baris yang seluruhnya terdiri dari nol, maka baris tersebut dipindah ke baris paling bawah
3. Pada sembarang dua baris berurutan yang tidak seluruhnya terdiri dari nol, posisi *leading* 1 pada baris yang lebih rendah posisinya terletak pada sisi yang lebih kanan dibandingkan dengan posisi *leading* 1 yang berada di baris yang lebih tinggi posisinya.

Prosedur untuk mereduksi sebuah matriks menjadi bentuk *row echelon* menggunakan fase *forward*, yaitu mengubah bentuk matriks menjadi 0 untuk bagian yang berada dibawah *leading* 1.

B. Metode Eliminasi Gauss-Jordan

Metode Eliminasi Gauss-Jordan merupakan metode yang mirip dengan Metode Eliminasi Gauss dan memiliki tujuan yang sama namun metode ini akan menghasilkan bentuk *reduced row echelon*, yaitu bentuk yang lebih sederhana lagi daripada bentuk *row echelon*. Berikut merupakan bentuk dari *reduced row echelon*.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & * \\ 0 & 1 & 0 & * \\ 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & * & 0 & 0 & 0 & * & * & 0 & * \\ 0 & 0 & 0 & 1 & 0 & 0 & * & * & 0 & * \\ 0 & 0 & 0 & 0 & 1 & 0 & * & * & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 1 & * & * & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & * \end{bmatrix}$$

Gambar 2. Matriks yang memiliki bentuk *reduced row echelon*

Sebuah matriks yang memiliki bentuk *reduced row echelon* memiliki sifat seperti berikut.

1. Bila suatu baris tidak seluruhnya terdiri dari nol, maka angka bukan nol pertama pada baris tersebut harus bernilai 1, atau bisa disebut memiliki *leading* 1.
2. Bila ada suatu baris yang seluruhnya terdiri dari nol, maka baris tersebut dipindah ke baris paling bawah
3. Pada sembarang dua baris berurutan yang tidak seluruhnya terdiri dari nol, posisi *leading* 1 pada baris yang lebih rendah posisinya terletak pada sisi yang lebih kanan dibandingkan dengan posisi *leading* 1 yang berada di baris yang lebih tinggi posisinya.
4. Tiap kolom yang memiliki *leading* 1 memiliki nilai nol di tempat lain di kolom tersebut.

Prosedur untuk mereduksi sebuah matriks menjadi bentuk *reduced row echelon* menggunakan 2 fase, yaitu fase *forward* dan *backward*, dimana fase *backward* mengubah bentuk matriks menjadi 0 untuk bagian yang berada di atas *leading* 1.

C. Determinan

Determinan matriks adalah sebuah skalar yang diperoleh dari elemen-elemen matriks tersebut dengan operasi tertentu. Determinan hanya dapat dihitung pada matriks yang memiliki jumlah baris dan kolom yang sama atau disebut dengan matriks persegi.

Determinan ditulis persis seperti lambang absolut, yakni dua garis tegak lurus yang mengapit elemen-elemen matriks, sebagai berikut.

$$\det(A) = |A| = \begin{vmatrix} 1 & 2 \\ 4 & 3 \end{vmatrix}$$

Untuk matriks berukuran 2x2, determinan dihitung menurut cara berikut,

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

Untuk matriks berukuran 3x3, determinan dihitung dengan cara berikut.

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

Ada beberapa sifat determinan matriks, di antaranya:

1. Apabila semua elemen dari salah satu baris atau kolom sama dengan nol, maka determinan matriks tersebut adalah nol.

$$A = \begin{bmatrix} 0 & 0 \\ 2 & 3 \end{bmatrix} \rightarrow |A| = 0;\ B = \begin{bmatrix} 2 & 3 & 1 \\ 0 & 0 & 0 \\ 5 & 4 & 1 \end{bmatrix} \rightarrow |B| = 0$$

2. Apabila semua elemen dari salah satu baris atau kolom itu sama dengan elemen-elemen baris atau kolom lain, maka determinan matriks tersebut adalah nol.

$$B = \begin{bmatrix} 4 & 3 & 2 \\ 5 & 7 & 8 \\ 4 & 3 & 2 \end{bmatrix} \rightarrow |B| = 0$$

(Sebab elemen-elemen baris ke-1 dan ke-3 adalah sama).

3. Apabila elemen-elemen salah satu dari baris atau kolom adalah merupakan kelipatan dari elemen-elemen baris atau kolom lain maka determinan matriks tersebut adalah nol.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 7 & 0 \\ 2 & 4 & 6 \end{bmatrix} \rightarrow |A| = 0$$

(Sebab elemen-elemen baris ke-3 sama dengan kelipatan elemen-elemen baris ke-1).

4. $|AB| = |A| \times |B|$
5. $|A^T| = |A|$, untuk A^T ialah transpose dari matriks A.
6. $|A^{-1}| = \frac{1}{|A|}$, untuk A^{-1} ialah invers dari matriks A.

7. $|kA| = kn\,|A|$, untuk A ordo $n \times n$, dan k suatu konstanta.

D. Matriks Balikan

Secara umum, invers dari matriks persegi ditulis sebagai berikut.

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

Matriks balikan atau bisa disebut dengan matriks invers adalah kebalikan dari suatu matriks tertentu. Jika A dan B adalah matriks persegi, dan berlaku AB = BA = I. maka dikatakan matriks A dan B saling invers. Kemudian, jika suatu matriks bujur sangkar A dikalikan terhadap inversnya yaitu matriks bujur sangkar A^{-1}, akan menghasilkan matriks I (matriks identitas pada operasi perkalian matriks).

Berikut ciri-ciri lain dari matriks balikan:

1. $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$
2. $(A^{-1})^t = (A^t)^{-1}$

Matriks yang mempunyai invers disebut *invertible* atau matriks non-singular, sedangkan matriks yang tidak mempunyai invers disebut matriks singular.

Untuk mencari invers matriks persegi berordo 2×2 sebagai berikut.

$$\mathrm{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \Rightarrow A^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Jika ad - bc = 0 maka matriks tersebut tidak mempunyai invers, atau disebut matriks singular.

E. Matriks Kofaktor

Kofaktor adalah 1 dipangkatkan dengan jumlah baris ke-a dan kolom ke -b dari suatu matriks, kemudian dikalikan dengan minornya. Jika A adalah matriks kuadrat, maka minor a_{ij} dinyatakan oleh M_{ij} adalah submatriks A yang didapat dengan jalan menghilangkan baris ke-i dan kolom ke - j.

Kofaktor a_{ij} dinyatakan oleh C_{ij} didefinisikan sebagai: $C_{ij} = (-1)^{I+j} .M_{ij.}$

$$\begin{vmatrix} + & - & + & \dots \\ - & + & - & \dots \\ + & - & + & \dots \\ \dots & \dots & \dots & \dots \end{vmatrix}$$

Sebuah matriks kofaktor adalah hasil konstruksi kofaktor masing-masing elemen matriks sesuai alamat elemen awalnya.

F. Matriks Adjoin

Adjoin suatu matriks ialah nilai hasil transpose dari kofaktor matriks tersebut. Artinya, untuk membentuk matriks adjoin dari suatu matriks, dilakukan:

- Pembentukan matrik kofaktor C
- Transpose dari matrik C yaitu C^T

G. Kaidah Crammer

Kaidah Cramer adalah rumus yang dapat digunakan untuk menyelesaikan sistem persamaan linear. Metode ini menggunakan determinan suatu matriks dan matriks lain yang diperoleh dengan mengganti salah satu kolom dengan vektor yang terdiri dari angka di sebelah kanan persamaannya.

Misalkan sebuah matriks 3 x 3 sebagai berikut.

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

Persamaan ini diubah menjadi *augmented matrix*.

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

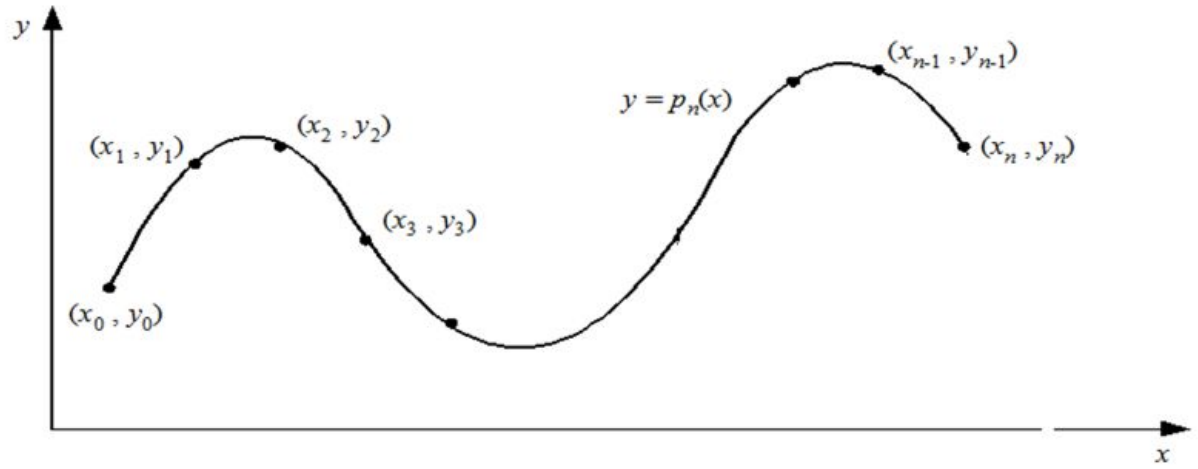
Lalu nilai *x*, *y*, dan *z* dicari sesuai cara berikut.

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

Kaidah Cramer tidak efisien untuk sistem dengan lebih dari dua atau tiga persamaan. Kaidah Cramer juga tidak stabil secara numerik, termasuk untuk sistem 2×2.

H. Interpolasi Polinom

Diberikan n+1 buah titik berbeda, (x_0, y_0), (x_1, y_1), ..., (x_n, y_n). Tentukan polinom $p_n(x)$ yang menginterpolasi (melewati) semua titik-titik tersebut sedemikian rupa sehingga $y_i = p_n(x_i)$ untuk $i = 0, 1, 2, ..., n$.



Setelah polinom interpolasi $p_n(x)$ ditemukan, $p_n(x)$ dapat digunakan untuk menghitung perkiraan nilai y di sembarang titik di dalam selang $[x_0, x_n]$.

Polinom interpolasi derajat n yang menginterpolasi titik-titik (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) adalah berbentuk $p_n(x) = a_0 + a_1x + a_2x_2 + ... + a_nx_n$. Jika hanya ada dua titik, (x_0, y_0) dan (x_1, y_1), maka polinom yang menginterpolasi kedua titik tersebut adalah $p_1(x) = a_0 + a_1x$ yaitu berupa persamaan garis lurus. Jika tersedia tiga titik, (x_0, y_0), (x_1, y_1), dan (x_2, y_2), maka polinom yang menginterpolasi ketiga titik tersebut adalah $p_2(x) = a_0 + a_1x + a_2x_2$ atau persaman kuadrat dan kurvanya berupa parabola. Jika tersedia empat titik, (x_0, y_0), (x_1, y_1), (x_2, y_2), dan (x_3, y_3), polinom yang menginterpolasi keempat titik tersebut adalah $p_3(x) = a_0 + a_1x + a_2x_2 + a_3x_3$, demikian seterusnya. Dengan cara yang sama kita

dapat membuat polinom interpolasi berderajat n untuk n yang lebih tinggi asalkan tersedia (n+1) buah titik data. Dengan menyulihkan (x_i, y_i) ke dalam persamaan polinom $p_n(x) = a_0 + a_1x + a_2x_2 + ... + a_nx_n$ untuk i = 0, 1, 2, ..., n, akan diperoleh n buah sistem persamaan lanjar dalam a_0, a_1, a_2, ..., a_n,

$$
\begin{aligned}
&a_0 + a_1x_0 + a_2{x_0}^2 + ... + a_n\,{x_0}^n = y_0\\
&a_0 + a_1x_1 + a_2{x_1}^2 + ... + a_n\,{x_1}^n = y_1\\
&\quad ... \qquad\qquad\quad ...\\
&a_0 + a_1x_n + a_2{x_n}^2 + ... + a_n\,{x_n}^n = y_n
\end{aligned}
$$

Solusi sistem persamaan lanjar ini, yaitu nilai a_0, a_1, ..., a_n, diperoleh dengan menggunakan metode eliminasi Gauss.

BAB III

IMPLEMENTASI PROGRAM

A. Kelas Matriks

Merupakan kelas yang mendeklarasikan atribut yang dimiliki matriks yang akan digunakan untuk menyelesaikan persoalan yang dispesifikasikan serta terdapat *method* yang berkaitan dengan Matriks. Di kelas ini merupakan tempat mendeklarasikan operasi-operasi yang dispesifikasikan oleh tugas ini.

```
import java.util.Scanner;
import java.io.*;

public class Matriks {
    private static Scanner scan = new Scanner(System.in);
    private static int ADD_ROW = 1;
    private static int ADD_COL = 2;
    private static double [][] matriks;
    private boolean is_augmented;
    private int row;    // m
    private int col;    // n

    // ctor
    public Matriks() {
          this.row = 100;
          this.col = 100;
          this.is_augmented = true;
          matriks = new double[this.row + ADD_ROW][this.col + ADD_COL];
    }

    public Matriks(int row, int col, boolean is_augmented){
          this.row = row;
          this.col = col;
          this.is_augmented = is_augmented;

          if (is_augmented) {
                   matriks = new double[this.row+ADD_ROW][this.col+ADD_COL];
          } else {
                   matriks = new double[this.row+ADD_ROW][this.col+ADD_ROW];
          }
    }

    public void readMatriksFromKeyboard() {
          String r;
          int row, column;
          row = this.row + ADD_ROW;
          if (this.is_augmented) {
                   column = this.col + ADD_COL;
          } else {
                   column = this.col + ADD_ROW;
          }
          String [][] temp = new String[row][column];

          for (int i=1; i< row; i++) {
                   r = scan.nextLine();
                   temp[i] = r.trim().split("\\s+");
                   for (int j=1; j<column; j++) {
                            this.matriks[i][j] = Double.parseDouble(temp[i][j-1]);
                   }
          }
    }

    public void readMatriksFromFile() throws IOException {
          System.out.println("Masukkan nama input file (xxx.txt):");
          System.out.print(">> ");
          String infile = scan.nextLine();

          BufferedReader in = new BufferedReader (new FileReader(infile));

          String line = in.readLine();
```

```
	int m=0;
	int n=0;

	while (line != null) {
		m++;

		String[] temp = line.trim().split("\\s+");

		for (n=1; n <= temp.length; n++) {
			this.matriks[m][n] = Double.parseDouble(temp[n-1]);
		}

		line = in.readLine();
	}
	this.row = m;
	this.col = n - ADD_COL;

	in.close();
}

private void swap2Rows(double[][] matrix, int i, int k, int j) {
	double temp;
	int column = matrix[0].length;
	for (int q = j; q < column; q++) {
		temp = matrix[i][q];
		matrix[i][q] = matrix[k][q];
		matrix[k][q] = temp;
		}
}

private void rowDivider(double[][] matrix, int i, int j, double divider) {
	int column = matrix[0].length;
	for (int q = j; q < column; q++) {
		matrix[i][q] /= divider;
	}
}

private void rowReducer(double[][] matrix, int i, int j, boolean is_gaussjordan) {
	int start;
	if (is_gaussjordan) {
		start = 1;
	} else {
		start = i;
	}
	int r = matrix.length;
	int column = matrix[0].length;

	for (int p=start; p < r; p++) {
		double temp = matrix[p][j];
		if (p != i && temp != 0) {
			for (int q=j; q < column; q++) {
				matrix[p][q] -= temp * matrix[i][q];
			}
		}
	}
}

private Boolean rowReducerForInverse(double[][] matrix, double[][] identity, int i, int j) {
	int start=1;
	int r = matrix.length;
	int column = matrix[0].length;

	for (int p=start; p < r; p++) {
		double temp = matrix[p][j];
		if (p != i && temp != 0) {
			for (int q = j; q < column; q++) {
				matrix[p][q] -= temp * matrix[i][q];
			}
			for (int q = 1; q < column; q++) {
				identity[p][q] -= temp * identity[i][q];
			}
		}
	}
	if (isMatrixNotInvertible(matrix)) {
		return false;
	}
```

```
        return true;

    }

    private void Gauss(double[][] matrix, boolean is_gaussjordan) {
        int i=1;
        int j=1;
        int k;
        while (i<=this.row && j<=this.col) {

                // look for a non-zero entry in col j or below row i
                k = i;
                while (k <= this.row && matrix[k][j]==0) {
                        k++;
                }

                // if non-zero entry is found
                if (k <= this.row) {
                        if (k != i) {
                                swap2Rows(matrix, i, k, j);
                        }

                        if (this.matriks[i][j] != 1) {
                                rowDivider(matrix, i,j, matrix[i][j]);
                        }

                        // pengurang baris gauss
                        rowReducer(matrix, i,j,is_gaussjordan);
                        i++;
                }
                j++;
        }
    }

    public void GaussSolution() throws IOException {
        Gauss(this.matriks, false);
        System.out.println("\nHasil operasi Gauss dalam bentuk matriks");
        printMatriks();
        Gauss(this.matriks, true);
        System.out.println();
        SPLSolution();
    }

    public void GaussJordanSolution() throws IOException {
        Gauss(this.matriks, true);
        System.out.println("\nHasil operasi Gauss-Jordan dalam bentuk matriks");
        printMatriks();
        System.out.println();
        SPLSolution();
    }

    private void SPLSolution() throws IOException {
        String solution = "";
        if (isNoSolution()) {
                solution = "Solusi SPL tidak ada";
                System.out.println("Solusi SPL tidak ada");
        } else
        if (isSolutionUnique()) {
                solution = "Solusi SPL Unik\n" + calculateUniqueSolution();
                System.out.println(solution);

        } else
        if (isSolutionInfinite()) {
                solution = "Solusi SPL Infinite";
                System.out.println("Solusi SPL Infinite");
                // calculateInfiniteSolution();
        }
        saveToFile(solution);
    }

    private String calculateUniqueSolution() {
        String solution = "";
        for (int i=1; i<= this.row; i++) {
                solution += "x";
                solution += Integer.toString(i);
                solution += " = ";
                solution += Double.toString(this.matriks[i][this.col + 1]);
```

```
                solution += "\n";
        }
        return solution;
    }

    private void calculateInfiniteSolution() {
        for (int i=1; i<=this.row; i++) {
                for (int j=1; j<=this.col; j++) {
                        if (this.matriks[i][j] == 1) {
                                System.out.print("x");
                                System.out.print(j);
                                System.out.print("= ");
                                System.out.print(matriks[i][this.col + 1]);

                                for (int p=j+1; p<=this.col; p++) {
                                        if (this.matriks[i][p] != 0) {
                                                double coeff = this.matriks[i][p] * -1;
                                                if (coeff > 0) {
                                                        System.out.print('+');
                                                        System.out.print(coeff);
                                                } else
                                                if (this.matriks[i][p] < 0) {
                                                        System.out.print(coeff);
                                                }
                                                System.out.print("t");
                                                System.out.print(p);
                                                // p++;
                                        }
                                }
                        } else
                        if (this.matriks[i][j]!=0 && this.matriks[i][j]!=1) {
                                System.out.println();
                                System.out.print("x");
                                System.out.print(j);
                                System.out.print("= ");
                                System.out.print("t");
                                System.out.println(j);
                        }
                }
        }
    }

    private void calculateInfiniteSolution1() {
                                char[]                                  list_of_char                                    =
{'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','y','z'};
        int idxchar = 0;
        String solution;
        String[] list_of_solution = new String[this.col + 1];
        Boolean[] list_sol = new Boolean[this.col + 1];

        for (int j = 1; j<=this.col; j++) {
                list_sol[j] = false;
        }

        for (int i = 1; i<=this.row; i++) {
                for (int j=1; j<=this.col; j++) {
                        if (this.matriks[i][j] == 1) {
                                list_sol[j] = true;
                                j = this.col;
                        }
                }
        }

        for (int j = 1; j<=this.col; j++) {
                if (!list_sol[j]) {
                        list_of_solution[j] = Character.toString(list_of_char[idxchar]);
                        idxchar++;
                }
        }

        // for (int kol=1; kol<=this.col; kol++) {
        //     if (list_sol[kol]) {
        //       solution =
        //     }
        // }
```

```
	for (int j = 1; j<=this.col; j++) {
		System.out.println(list_of_solution[j]);
	}


}

private Boolean isRowAllZero(double[] rowMatrix) {
	for (int i=1; i<=this.col+1; i++) {
		if (rowMatrix[i]!=0) {
			return false;
		}
	}
	return true;
}

private Boolean GaussJordanForInverse(double[][] matrix, double[][] identity) {
	int i=1;
	int j=1;
	int k;
	while (i<=this.row && j<=this.col) {

		// look for a non-zero entry in col j or below row i
		k = i;
		while (k <= this.row && matrix[k][j]==0) {
			k++;
		}

		// if non-zero entry is found
		if (k <= this.row) {
			if (k != i) {
				swap2Rows(matrix, i, k, j);
				swap2Rows(identity, i, k, j);
			}

			if (this.matriks[i][j] != 1) {
				double divider = matrix[i][j];
				rowDivider(matrix, i,j, divider);
				rowDivider(identity, i, 1, divider);
			}

			// pengurang baris gauss
			boolean is_invertible = rowReducerForInverse(matrix, identity, i,j);
			if (!is_invertible) {
				return false;
			}
			i++;
		}
		j++;
	}

	return true;
}

public void findInverseUsingOBE() {
	double[][] identity = new double[this.row + ADD_ROW][this.col + ADD_ROW];
	for (int i=1; i<=this.row; i++) {
		for (int j=1; j<=this.col; j++) {
			if (i==j) {
				identity[i][j] = 1;
			}

		}
	}

	double[][] inverse = identity;

	boolean is_invertible = GaussJordanForInverse(this.matriks, inverse);

	if (is_invertible) {
		System.out.println("Matrix is invertible");
		System.out.println("Hasil inverse");
		for (int i=1; i<= this.row; i++) {
			for (int j=1; j<= this.col; j++) {
				System.out.print(inverse[i][j] + " ");
			}
			System.out.println();
```

```
                }
                } else {
                        System.out.println("Matrix is not invertible");
                }

    }

    public void printMatriks() {
          int row, column;
          row = this.row + ADD_ROW;
          if (this.is_augmented) {
                  column = this.col + ADD_COL;
          } else {
                  column = this.col + ADD_ROW;
          }
          System.out.println();
          for (int i=1; i< row; i++) {
                  for (int j=1; j< column; j++) {
                          System.out.print(this.matriks[i][j] + " ");
                  }
                  System.out.println();
          }
    }

    private Boolean isMatrixEqual(double[][] m1, double[][] m2) {
          for (int i=1; i<m1.length; i++) {
                  for (int j=1; j<m1[0].length; j++) {
                          if (m1[i][j] != m2[i][j]) {
                                  return false;
                          }
                  }
          }
          return true;
    }

    private Boolean isMatrixNotInvertible(double[][] m) {
          for (int i=1; i<m[1].length; i++) {
                  if (m[m.length-1][i] != 0) {
                          return false;
                  }
          }
          return true;
    }

    private Boolean isSolutionUnique() {
          if (this.matriks[this.row][this.col] == 1 && this.matriks[this.row][this.col + 1] != 0) {
                  return true;
          }
          return false;
    }

    private Boolean isSolutionInfinite() {
          if (this.matriks[this.row][this.col] == 0 && this.matriks[this.row][this.col + 1] == 0) {
                  return true;
          }
          return false;
    }

    private Boolean isNoSolution() {
          if (this.matriks[this.row][this.col] == 0 && this.matriks[this.row][this.col + 1] != 0) {
                  return true;
          }
          return false;
    }

    private void saveToFile(String solution) throws IOException {
          System.out.println("Masukkan nama output file (xxx.txt):");
          System.out.print(">> ");

          String outfile = scan.nextLine();
          PrintWriter out = new PrintWriter(new FileWriter(outfile));
          out.print(solution);
          out.close();
    }

    public static void main(String[] args) throws IOException {
          // Input dari keyboard
```

```
        // System.out.print("Masukkan nilai m (jumlah baris) >> ");
        // int row = scan.nextInt();
        // System.out.print("Masukkan nilai n (jumlah kolom) >> ");
        // int col = scan.nextInt();

        // Matriks matrix = new Matriks(row, col);
        // matrix.readMatriksFromKeyboard();

        // input dari file
        // Matriks matrix = new Matriks();
        // matrix.readMatriksFromFile();
        // matrix.printMatriks();
    }
}
```

B. Kelas Main

Merupakan kelas yang berfungsi untuk menangani interaksi dengan pengguna serta pemanfaatan operasi-operasi yang sudah dideklarasikan di kelas Matriks.

```
import java.util.Scanner;
import java.io.*;

public class Main{
    private static Scanner scan = new Scanner(System.in);
    private static Matriks matrix;
    public static void main(String[] args) throws IOException {
        Main main = new Main();
        main.start();
    }

    private void start() throws IOException {
        System.out.println("========================================================================");
        System.out.println("==                                                                    WELCOME
                                                    ==");
        System.out.println("========================================================================");

        do {

        } while (!mainMenu());
    }

    private Boolean mainMenu() throws IOException {
        System.out.println("\nMENU");
        System.out.println("1. Sistem Persamaan Linier");
        System.out.println("2. Determinan");
        System.out.println("3. Matriks Balikan");
        System.out.println("4. Matriks Kofaktor");
        System.out.println("5. Adjoin");
        System.out.println("6. Interpolasi Polinom");
        System.out.println("7. Keluar");
        System.out.print(">> ");
        int option = scan.nextInt();

        if (option == 7) {
                return true;
        } else
        if (option == 1) {
                return subMenuSPL();
        } else
        if (option == 2) {
                return subMenuDet();
        } else
        if (option == 3) {
                return subMenuInvers();
        } else {
                return false;
        }
    }
```

```
    private Boolean subMenuSPL() throws IOException {
          System.out.println("\nPilih Metode");
          System.out.println("1. Metode Eliminasi Gauss");
          System.out.println("2. Metode Eliminasi Gauss-Jordan");
          System.out.println("3. Metode Matriks Balikan");
          System.out.println("4. Kaidah Cramer");
          System.out.print(">> ");
          int option = scan.nextInt();
          selectInputType(true);

          if (option == 1) {
                  matrix.GaussSolution();
          } else if (option == 2) {
                  matrix.GaussJordanSolution();
          }
          return false;
    }

    private void selectInputType(boolean is_augmented) throws IOException {
          System.out.println("\nPilih media input matriks");
          System.out.println("1. Keyboard");
          System.out.println("2. File eksternal");
          System.out.print(">> ");
          int option = scan.nextInt();
          if (option == 1) {
                  inputMatrixFromKeyboard(is_augmented);
          } else
          if (option==2) {
                  inputMatrixFromFile(is_augmented);
          }
    }

    private void inputMatrixFromFile(boolean is_augmented) throws IOException {
          if (!is_augmented) {
                  System.out.println("Maaf belum bisa");
          } else {
                  matrix = new Matriks();
                  matrix.readMatriksFromFile();
                  System.out.println("Hasil baca matriks dari file");
                  matrix.printMatriks();
                  System.out.println();
          }

    }

    private void inputMatrixFromKeyboard(boolean is_augmented) {
          int col, row;
          if (is_augmented) {
                  System.out.print("Masukkan nilai m (jumlah baris) >> ");
                  row = scan.nextInt();
                  System.out.print("Masukkan nilai n (jumlah kolom) >> ");
                  col = scan.nextInt();
          } else {
                  System.out.print("Masukkan nilai n (jumlah baris dan kolom) >> ");
                  col = scan.nextInt();
                  row = col;
          }


          matrix = new Matriks(row, col, is_augmented);
          matrix.readMatriksFromKeyboard();
    }

    private Boolean subMenuDet() {
          // System.out.println("\nPilih Metode");
          // System.out.println("1. Metode Eliminasi Gauss");
          // System.out.println("2. Metode Eliminasi Gauss-Jordan");
          // System.out.println("3. Metode Matriks Balikan");
          // System.out.println("4. Kaidah Cramer");
          // System.out.print(">> ");
          // int option = scan.nextInt();
          System.out.println("Maaf belum bisa");
          return false;
    }

    private Boolean subMenuInvers() throws IOException {
          System.out.println("\nPilih Metode");
```

```
            System.out.println("1. Metode Eliminasi Gauss-Jordan");
            System.out.println("2. Menggunakan Matriks Adjoin (Belum bisa)");
            System.out.print(">> ");
            int option = scan.nextInt();
            selectInputType(false);
            if (option==1) {
                    inverseGaussJordan();
            }
            return false;
    }

    private void inverseGaussJordan() {
            matrix.findInverseUsingOBE();
    }
}
```

BAB IV

EKSPERIMEN

Berikut merupakan hasil eksekusi program yang telah kami implementasikan terhadap contoh-contoh kasus yang diberikan serta analisis hasil eksekusinya.

1. Solusi SPL

No.	Persoalan	Hasil Implementasi
1.	$A = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 5 & -7 & -5 \\ 2 & -1 & 1 & 3 \\ 5 & 2 & -4 & 2 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ -2 \\ 4 \\ 6 \end{bmatrix}$	MENU 1. Sistem Persamaan Linier 2. Determinan 3. Matriks Balikan 4. Matriks Kofaktor 5. Adjoin 6. Interpolasi Polinom 7. Keluar >> 1 Pilih Metode 1. Metode Eliminasi Gauss 2. Metode Eliminasi Gauss-Jordan 3. Metode Matriks Balikan 4. Kaidah Cramer >> 1 Pilih media input matriks 1. Keyboard 2. File eksternal >> 1 Masukkan nilai m (jumlah baris) >> 4 Masukkan nilai n (jumlah kolom) >> 4 1 1 -1 -1 1 2 5 -7 -5 -2 2 -1 1 3 4 5 2 -4 2 6 Hasil operasi Gauss dalam bentuk matriks 1.0 1.0 -1.0 -1.0 1.0 0.0 1.0 -1.6666666666666667 -1.0 -1.3333333333333333 0.0 0.0 1.0 -1.0 1.0 0.0 0.0 0.0 0.0 1.0 Solusi SPL tidak ada Masukkan nama output file (xxx.txt): >> soal1.txt

2.

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

```
MENU
1. Sistem Persamaan Linier
2. Determinan
3. Matriks Balikan
4. Matriks Kofaktor
5. Adjoin
6. Interpolasi Polinom
7. Keluar
>> 1

Pilih Metode
1. Metode Eliminasi Gauss
2. Metode Eliminasi Gauss-Jordan
3. Metode Matriks Balikan
4. Kaidah Cramer
>> 2

Pilih media input matriks
1. Keyboard
2. File eksternal
>> 1
Masukkan nilai m (jumlah baris) >> 4
Masukkan nilai n (jumlah kolom) >> 5
1 -1 0 0 1 3
1 1 0 -3 0 6
2 -1 0 1 -1 5
-1 2 0 -2 -1 -1

Hasil operasi Gauss Jordan dalam bentuk matriks

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 SPL Infinite
Masukkan nama output file (xxx.txt):
>> soal1b.txt
```

3.

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

```
MENU
1. Sistem Persamaan Linier
2. Determinan
3. Matriks Balikan
4. Matriks Kofaktor
5. Adjoin
6. Interpolasi Polinom
7. Keluar
>> 1

Pilih Metode
1. Metode Eliminasi Gauss
2. Metode Eliminasi Gauss Jordan
3. Metode Matriks Balikan
4. Kaidah Cramer
>> 2

Pilih media input matriks
1. Keyboard
2. File eksternal
>> 1
Masukkan nilai m (jumlah baris) >> 3
Masukkan nilai n (jumlah kolom) >> 6
0 1 0 0 1 0 2
0 0 0 1 1 0 -1
0 1 0 0 0 1 1

Hasil operasi Gauss-Jordan dalam bentuk matriks

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

Masukkan nama output file (xxx.txt):
>> soal1c.txt
```

2. SPL berbentuk matriks *augmented*

No.	Persoalan	Hasil Implementasi
1.	$\begin{bmatrix} 1 & -1 & 2 & -1 & -1 \\ 2 & 1 & -2 & -2 & -2 \\ -1 & 2 & -4 & 1 & 1 \\ 3 & 0 & 0 & -3 & -3 \end{bmatrix}$.	MENU 1. Sistem Persamaan Linier 2. Determinan 3. Matriks Balikan 4. Matriks Kofaktor 5. Adjoin 6. Interpolasi Polinom 7. Keluar >> 1 Pilih Metode 1. Metode Eliminasi Gauss 2. Metode Eliminasi Gauss-Jordan 3. Metode Matriks Balikan 4. Kaidah Cramer >> 2 Pilih media input matriks 1. Keyboard 2. File eksternal >> 2 Masukkan nama input file (xxx.txt): >> test/2a.txt Hasil baca matriks dari file 1.0 1.0 2.0 1.0 1.0 2.0 1.0 -2.0 -2.0 -2.0 -1.0 2.0 -4.0 1.0 1.0 3.0 0.0 0.0 -3.0 -3.0 Hasil operasi Gauss Jordan dalam bentuk matriks 1.0 0.0 0.0 -1.0 -1.0 0.0 1.0 -2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 Solusi SPL Infinite Masukkan nama output file (xxx.txt): >> hasil2a.txt

2.

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

```
>> 1

Pilih Metode
1. Metode Eliminasi Gauss
2. Metode Eliminasi Gauss-Jordan
3. Metode Matriks Balikan
4. Kaidah Cramer
>> 2

Pilih media input matriks
1. Keyboard
2. File eksternal
>> 2
Masukkan nama input file (xxx.txt):
>> test/3b.txt
Hasil baca matriks dari file

2.0 0.0 8.0 0.0 8.0
0.0 1.0 0.0 4.0 6.0
-4.0 0.0 6.0 0.0 6.0
0.0 -2.0 0.0 3.0 -1.0
2.0 0.0  4.0 0.0  4.0
0.0 1.0 0.0 -2.0 0.0


Hasil operasi Gauss-Jordan dalam bentuk matriks

1.0 0.0 0.0 0.0 0.0
0.0 1.0 0.0 0.0 2.0
0.0 0.0 1.0 0.0 1.0
0.0 0.0 0.0 1.0 1.0
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0

Solusi SPL infinite
Masukkan nama output file (xxx.txt):
>> hasil3b.txt
```

3. SPL berbentuk persamaan linear

No.	Persoalan	Hasil Implementasi
1.	(i) $8x_1 + x_2 + 3x_3 + 2x_4 = 0$ $2x_1 + 9x_2 - x_3 - 2x_4 = 1$ $x_1 + 3x_2 + 2x_3 - x_4 = 2$ $x_1 + 6x_3 + 4x_4 = 3$	

4. Matriks Balikan (5 x 5)

```
MENU
1. Sistem Persamaan Linier
2. Determinan
3. Matriks Balikan
4. Matriks Kofaktor
5. Adjoin
6. Interpolasi Polinom
7. Keluar
>> 3

Pilih Metode
1. Metode Eliminasi Gauss-Jordan
2. Menggunakan Matriks Adjoin (Belum bisa)
>> 1

Pilih media input matriks
1. Keyboard
2. File eksternal
>> 1
Masukkan nilai n (jumlah baris dan kolom) >> 5
6 10 3 11 4
13 5 12 7 14
15 8 16 2 9
17 18 19 20 21
22 23 24 25 26
Matrix is invertible
Hasil inverse
0.12903225806451719 0.16129032258064535 1.1796119636042288E-15 -0.6193548387096602 0.3935483870967595
-1.7096774193548288 -0.38709677419354977 -0.9999999999999869 -15.993548387096595 13.735483870967592
1.4193548387096662 0.10752688172043137 0.0000000000000887 14.587096774193305 -12.464301075268688
1.7741935483870879 0.3010752688172056 0.999999999999987 16.083870967741776 -13.772043010752538
-1.6129032258064360 -0.18279569892473257 -0.0000000000000891 -14.258064516128886 12.247311827956866
```

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Program ini dapat menyelesaikan SPL dengan berbagai metode, yaitu Gauss, Gauss-Jordan, metode matriks balikan, serta dengan kaidah cramer. Selain itu, program ini dapat melakukan berbagai operasi yang dapat diterapkan kepada matriks, yaitu mencari determinan, matriks kofaktor, adjoin, serta inverse dengan berbagai metode. Selain itu, program ini juga dapat menyelesaikan persoalan interpolasi polinom. Solusi-solusi yang ditawarkan program ini sangat berguna untuk menyelesaikan persoalan yang ada di kehidupan, yaitu persoalan yang bisa dinotasikan dengan sistem persamaan linier dan yang bisa disajikan dengan bentuk matriks, terutama persoalan yang kompleks dan sulit untuk dikerjakan secara manual, bila menggunakan program ini maka pencarian solusi menjadi lebih mudah.

B. Saran

Beberapa kegiatan seperti menentukan tegangan listrik dan arus listrik dapat diaplikasikan dengan aljabar lanjar yang dihasilkan oleh interpolasi titik. Saran kami agar metode tersebut dapat lebih diaplikasikan lagi pada hal lain, sehingga program ini dapat dipakai pada kehidupan sehari – hari.

Selain itu, spesifikasi dari tugas besar ini menurut saya terlalu banyak, apalagi dibandingkan dengan tugas besar yang sama yang diberikan satu tahun hingga dua tahun yang lalu. Sarannya semoga lingkup tugas yang diberikan bisa dipertimbangkan lagi kedepannya.

DAFTAR REFERENSI

[1] Anton, Howard dan Chris Rorres. 2014. Elementary Linear Algebra : applications version -- 11th edition. United States of America: Wiley.

[2]http://contohdanpenyelesaianmatrix.blogspot.com/2014/06/invers-matriks_5.html?m=1