LAPORAN TUGAS BESAR 1

IF2123 ALJABAR LINIER DAN GEOMETRI

SISTEM PERSAMAAN LINIER, DETERMINAN, DAN APLIKASINYA

oleh

Ahmad Romy Zahran	13520009
Firizky Ardiansyah	13520095
Muhammad Fahmi Irfan	13520152

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

BAB 1
DESKRIPSI MASALAH

Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. 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 peubah dan n persamaan). Solusi sebuah SPL mungkin tidak ada, banyak (tidak berhingga), atau hanya satu (unik/tunggal).

Di dalam Tugas Besar 1 ini, kami diminta membuat satu atau lebih pustaka aljabar linier dalam Bahasa Java. Pustaka tersebut berisi fungsi-fungsi seperti eliminasi Gauss, eliminasi Gauss-Jordan, menentukan balikan matriks, menghitung determinan, kaidah Cramer (kaidah Cramer khusus untuk SPL dengan n peubah dan n persamaan). Selanjutnya, pustaka ini akan digunakan untuk menyelesaikan berbagai persoalan yang dimodelkan dalam bentuk SPL, persoalan interpolasi, dan persoalan regresi.

BAB 2
TEORI SINGKAT

A. Metode Eliminasi Gauss

Metode eliminasi Gauss merupakan salah satu metode dalam menentukan solusi dari suatu SPL. Metode ini memanfaatkan matriks *augmented*, yaitu matriks yang disusun oleh koefisien-koefisien beserta konstanta dari SPL. Metode ini terdiri dari dua tahap, yaitu tahap eliminasi dan tahap penyulihan mundur. Pada tahap eliminasi, matriks *augmented* dari SPL tersebut akan dioperasikan dengan Operasi Baris Elementer (OBE), yaitu mengalikan suatu baris dengan konstanta taknol, menukar dua baris yang berbeda, atau menambahkan kelipatan suatu baris ke baris lain. Pengoperasian matriks dengan OBE bertujuan untuk memperoleh matriks eselon baris, yaitu matriks yang memenuhi ketentuan berikut.

1. Jika suatu baris mengandung bilangan taknol, bilangan taknol pertamanya ialah 1. Selanjutnya, bilangan 1 pertama ini akan disebut *leading 1*.
2. Jika ada beberapa baris yang tidak mengandung bilangan taknol, baris-baris tersebut diletakkan di baris-baris paling bawah.
3. Jika ada dua baris berurutan yang mengandung bilangan taknol, *leading 1* baris yang di atas lebih kiri daripada *leading 1* baris yang di bawah.

Pada tahap penyulihan mundur, matriks eselon baris yang diperoleh dapat digunakan untuk membuat suatu SPL berkorespondensi dengan matriks tersebut. Akar-akar dari SPL ini dapat ditentukan menggunakan teknik penyulihan mundur, yaitu menyelesaikan SPL dari persamaan paling bawah, lalu sulih solusi-solusi yang sudah diperoleh ke persamaan yang di atasnya.

B. Metode Eliminasi Gauss-Jordan

Metode ini tidak jauh berbeda dengan metode sebelumnya. Perbedaannya ada pada tahap keduanya. Jika sebelumnya matriks eselon baris yang diperoleh langsung diubah menjadi SPL, pada metode ini OBE tetap dilakukan untuk memperoleh matriks eselon baris tereduksi, yaitu matriks eselon baris dengan satu ketentuan tambahan, yaitu kolom yang memuat *leading 1* tak boleh memuat bilangan taknol selain pada baris tersebut. Solusi SPL dapat ditemukan setelah mengubah matriks eselon baris tereduksi ke SPL yang berkorespondensi.

C. Determinan

Determinan merupakan suatu fungsi dari matriks yang menghasilkan suatu bilangan riil. Determinan ini dapat digunakan untuk menentukan invers dari suatu matriks dan solusi dari suatu SPL. Terdapat beberapa cara untuk menentukan determinan dari suatu fungsi.

1. Ekspansi Kofaktor

Minor dari entri baris ke-i dan kolom ke-j dari suatu matriks persegi A (disimbolkan M_{ij}) didefinisikan sebagai determinan dari matriks A dengan baris ke-i dan kolom ke-j dihapus. Jika M_{ij} dikalikan dengan $(-1)^{i+j}$, hasil dari perkalian tersebut merupakan kofaktor dari entri baris ke-i dan kolom ke-j (disimbolkan C_{ij}). Untuk Matriks berukuran $n \times n$ (dengan $n \geq 2$), determinan dapat ditentukan dengan rumus berikut.

$$\det(A) = a_{1j}C_{1j} + a_{2j}C_{2j} + \cdots + a_{nj}C_{nj}$$

Pada metode ini, determinan didefinisikan dengan determinan dari matriks yang lebih kecil, sehingga membentuk pola rekursif. Basis dari pola rekursif ini ialah $n = 1$, yaitu $\det(A) = a_{11}$.

2. Operasi Baris Elementer

Determinan juga dapat ditentukan dengan Operasi Baris Elementer. Operasi Baris Elementer dilakukan sehingga matriks yang determinannya akan dicari menjadi matriks segitiga atas, matriks segitiga bawah, ataupun matriks diagonal. Matriks segitiga atas ialah matriks yang untuk setiap i dan j indeks baris dan kolom dari suatu entri di matriks tersebut, jika $i - j > 0$ maka $a_{ij} = 0$. Sebaliknya, matriks segitiga bawah sama seperti matriks segitiga atas, namun syaratnya jika $j - i > 0$ maka $a_{ij} = 0$. Selain itu, ada juga matriks diagonal, yaitu matriks yang memenuhi $a_{ij} = 0$ untuk setiap $i \neq j$. Dalam mencari determinan menggunakan OBE, ada beberapa persyaratan, yaitu

a. Jika suatu baris dikali k , kalikan determinannya dengan k

b. Jika suatu baris ditukar dengan baris lain, kalikan -1

c. Suatu baris bisa dijumlahkan dengan kelipatan baris lain, tanpa perlu mengubah determinan.

Setelah didapat matriks segitiga ataupun matriks diagonal, determinan dapat ditentukan dengan cara mengalikan setiap komponen diagonal dari matriks tersebut, lalu kalikan juga hasil dari syarat-syarat OBE di atas.

3. Matriks Balikan

Matriks balikan dari suatu matriks A merupakan matriks yang jika dikalikan dengan matriks A akan menghasilkan suatu matriks identitas, yaitu matriks diagonal yang semua komponen diagonal utamanya 1. Matriks balikan dapat ditentukan dengan beberapa cara

Matriks Kofaktor dan Matriks Adjoin

Matriks kofaktor merupakan matriks yang komponen baris ke-i dan kolom ke-j -nya ialah C_{ij}. Matriks adjoin ialah transpos dari matriks kofaktor. Dengan matriks adjoin, matriks balikan dapat ditentukan, yaitu matriks adjoin yang dibagi dengan determinan dari matriks yang akan dicari balkannya.

Operasi Baris Elementer

Matriks balikan dapat ditentukan dengan OBE, yaitu dengan menggabungkan matriks tersebut dengan matriks indentitas menjadi suatu matriks *augmented*, lalu OBE dilakukan sehingga matriks kiri menjadi matriks identitas.

4. Kaidah Cramer

Kaidah Cramer merupakan suatu metode untuk menentukan solusi SPL. Misalkan terdapat SPL dengan variabel-variabelnya $x_1, x_2, \ldots, x_n$. Misalkan juga A merupakan matriks koefisien SPL tersebut (dengan kata lain, $Ax = B$). Maka, nilai dari x_i dapat ditentukan dengan formula berikut

$$x_i = \frac{\det(A_i)}{\det(A)}$$

Dengan A_i merupakan matriks A yang kolom-kolom ke-i -nya diganti dengan matriks B , yaitu matriks konstanta.

D. Interpolasi Polinom

Pada kumpulan n+1 titik, kita dapat membuat suatu polinomial berderajat n $p_n(x)$ sehingga kurva $y = p_n(x)$ melalui titik-titik tersebut. Hal ini disebut dengan interpolasi polinom. Untuk menentukan polinomial tersebut, cukup dengan menyulihkan titik-titik tersebut ke dalam polinomial tersebut sehingga membentuk SPL dengan n+1 variabel. Solusi dari SPL tersebut ialah koefisien dari polinomial tersebut.

E. Regresi Linier Berganda

Regresi linier berganda merupakan metode untuk memprediksi suatu hubungan dari suatu variabel terikat dengan beberapa variabel bebas. Bentuk umum dari regresi linier berganda ialah

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \epsilon$$

Nilai dari $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$ dapat ditentukan dari SPL berikut

$$n\beta_0 + \beta_1 \sum_{i=1}^{n} x_{1i} + \beta_2 \sum_{i=1}^{n} x_{2i} + \ldots + \beta_k \sum_{i=1}^{n} x_{ki} = \sum_{i=1}^{n} y_i$$

$$\beta_0 \sum_{i=1}^{n} x_{1i} + \beta_1 \sum_{i=1}^{n} x_{1i}^2 + \beta_2 \sum_{i=1}^{n} x_{1i}\, x_{2i} + \cdots + \beta_k \sum_{i=1}^{n} x_{1i}\, x_{ki} = \sum_{i=1}^{n} x_{1i}\, y_i \vdots$$

$$\beta_0 \sum_{i=1}^{n} x_{ki} + \beta_1 \sum_{i=1}^{n} x_{1i}\, x_{ki} + \beta_2 \sum_{i=1}^{n} x_{2i}\, x_{ki} + \ldots + \beta_k \sum_{i=1}^{n} x_{ki}^2 = \sum_{i=1}^{n} x_{ki}\, y_i$$

BAB 3
IMPLEMENTASI PUSTAKA

A. Struktur Program

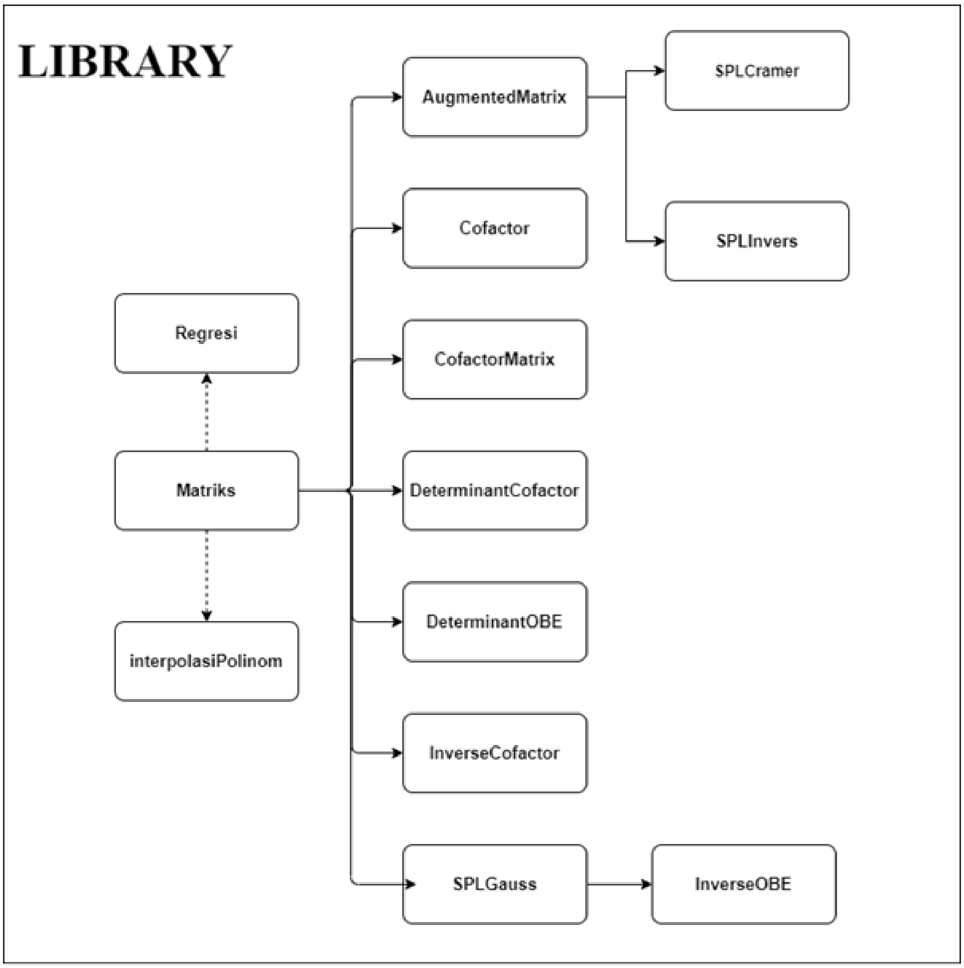
Implementasi pustaka aljabar linier untuk menyelesaikan permasalahan sistem persamaan linier dan matriks ini diselesaikan dengan paradigma pemrograman *Object Oriented Programming* (OOP) dalam bahasa Java. Program tersusun atas beberapa modul-modul berisi objek *class* dan/atau struktur data yang membangun fungsi utama program, yaitu menyelesaikan permasalahan aljabar linier. Modul-modul objek dikelompokkan dalam direktori yang disesuaikan berdasarkan fungsi umum modul tersebut.

Pustaka yang penulis buat, dibangun atas dua buah modul utama, yaitu *library* dan *driver*. *Library* berisi kelas-kelas berupa struktur data beserta atribut fungsi dan prosedur yang mendukung. *Driver* berisi program yang menghubungkan cetak biru kelas-kelas objek pada *library* sekaligus program untuk *Graphical User Interface* (GUI) pustaka.

Struktur program utama pustaka ini, disimpan dalam modul dasar bernama *src*, kemudian diisi oleh *package* berisi program utama pustaka. *library* dan GUI, disimpan dalam *src/algeo/lib* dan *src/algeo/IO*. Program ini dikompilasi menjadi *binary* pada folder *bin/algeo*, dengan *algeo* adalah nama *package* yang penulis gunakan. Data uji program akan disimpan dalam file *test* dan akan dikelompokkan menjadi file *input* dan file *output* berdasarkan fungsinya (menyimpan keluaran dan algoritma masukan dari file).

Cetak biru program utama modul *library* meliputi 13 *class* yaitu, Matriks, AugmentedMatrix, Cofactor, CofactorMatrix, DeterminantCofactor, DeterminantOBE, interpolasiPolinom, InverseCofactor, InversOBE, SPLCramer, SPLGauss, SPLInvers, dan RLB. Kelas ini disusun secara hierarki, dengan metode *inheritance*, sehingga terdapat *class* yang merupakan *superclass* dari *class* yang di-*inherited* dan terdapat *class* dari di-*inherit* disebut *subclass*.

Hierarki tersebut secara sederhana dapat digambarkan pada diagram di bawah ini.

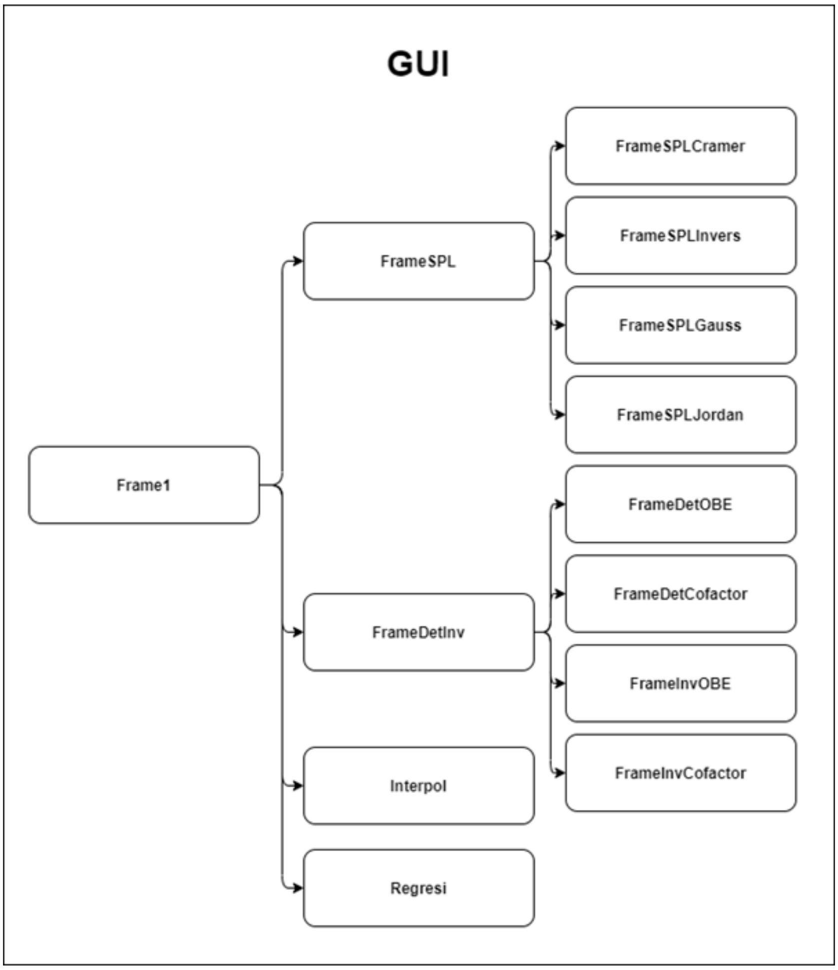


Bagan 1 hirarki pustaka program

Tanda panah menunjukkan bahwa objek kelas tersebut merupakan subkelas dari kelas yang menunjuk. Konsep subkelas dan superkelas pada dasarnya adalah subkelas akan mengambil atau mewarisi seluruh atribut superkelas untuk dapat digunakan dalam subkelas, sehingga superkelas yang mewariskan atribut dapat diperluas menjadi objek baru yang memiliki karakteristik sama dengan dirinya, tetapi ditambahkan dengan atribut unik miliknya sendiri (atribut yang tidak terdapat di superkelas tetapi tersedia di subkelas).

Pada bagian *Graphical User Interface* (GUI) modulnya tersusun atas *frame-frame* yang disesuaikan dengan fungsionalitasnya. Penulis menggunakan IntelliJ sebagai *builder interface* menggunakan *external library* yang disebut Java Swing yang implementasinya menggunakan JFrame. Kelas yang mengandung interface melakukan *inheritance* terhadap kelas yang disebut JFrame, sebuah *class* dari *external library* yang telah diimpor.

Adapun hierarki program GUI secara umum adalah sebagai berikut



Bagan 2 Hirarki GUI program

Tanda panah pada diagram menunjukkan hal yang sama dengan definisi sebelumnya. Di dalam algoritmanya cetak biru pada modul *library* akan digunakan pada implementasi lebih khusus pada bagian antarmuka.

B. Struktur *Class*

Kelas pada umumnya terdiri dari beberapa atribut (baik bersifat *public* maupun *private*) dan beberapa *method* atau primitif, berbentuk fungsi dan prosedur. Seluruh objek memiliki primitif yang disebut konstruktor. Objek juga biasanya memiliki primitif lain yang disebut sebagai *getter*, *setter*, predikat, dan primitif input/output.

Atribut dari sebuah objek dapat di-*set* hanya dapat diakses kelas tersebut (*private*/*protected*) atau bisa diakses seluruh kelas (*public*). Sifat atribut ini akan menentukan bagaimana cara pengolahan objek dalam kelas saat diimplementasikan di dalam driver atau kelas lainnya. Sifat atribut ini juga memungkinkan enkapsulasi informasi, sehingga suatu atribut dapat dibuat menjadi *read-only*, *write-only*, ataupun *write* and *read*.

Primitif dari sebuah kelas yang paling umum adalah konstruktor, yang sesuai namanya, fungsi dari primitif ini adalah membangun objek tersebut. Sebuah konstruktor dalam program Java didefinisikan dengan nama dari *object* yang dikonstruksi. Sebuah kelas dapat memiliki banyak konstruktor dengan parameter primitif yang bisa dibuat berbeda, disesuaikan fungsionalitasnya.

Dampak dari enkapsulasi adalah diperlukan algoritma untuk mengakses atribut yang hanya bisa diakses dalam kelas tersebut. *Setter* dan *getter* berperan dalam mengubah dan mengambil atribut yang terproteksi di dalam kelas tersebut. Adapun primitif lainnya, predikat adalah sebuah fungsi yang mengembalikan boolean, dan input/output menangani keluaran dan masukan objek.

Pada program pustaka ini, dalam implementasinya digunakan atribut dan primitif yang dijelaskan di atas. Pada bagian *library* atribut dan primitif disesuaikan dengan kebutuhan dan spesifikasi program. Adapun secara detail, struktur kelas dari program ini sebagai berikut.

Tabel 1 Struktur kelas

No.	**Nama *Class***	**Object (Attributes/Methods)**	**Keterangan**
1.	Matriks	**Attributes**	
		elmt	Bertipe array of array of double. Elemen matriks disimpan dalam elmt.
		nRow	Bertipe integer, menyimpan banyaknya baris pada matriks.
		nCol	Bertipa integer, menyimpan banyaknya kolom pada matriks.
		Methods	
		Matriks	Konstruktor ADT. Meliputi prosedur input dari file, dari keyboard, dan konstruksi matriks null
		tulisMatriks1	Output matriks ke file
		tulisMatriks2	Output matriks ke terminal (hanya untuk keperluan *debugging*)
		Eq	Mereturn true ketika nilai yang bertipe Double sama dengan presisi tertentu

		swaprow	Prosedur menukar baris untuk keperluan operasi baris elementer
2.	AugmentedMatrix	**Methods**	
		AugmentedMatrix	Konstruktor ADT. Memisahkan matriks augmented menjadi matriks koefisien dan array konstanta
		getCoefCol	Mengembalikan banyak kolom dari matriks koefisien
		getCoefRow	Mengembalikan banyak baris dari matriks koefisien
		constElmt	Mengembalikan elemen ke-i dari array konstan
3.	Cofactor	**Methods**	
		Cofactor	Konstruktor kelas, mencari minor matriks/submatriks
4.	CofactorMatrix	**Methods**	
		CofactorMatrix	Konstruktor kelas, menyusun minor submatriks menjadi matriks kofaktor utuh
		adjoin	Transpose objek menjadi adjoin matriks
5.	DeterminantCofactor	**Attributes**	
		M	Salinan matriks
		Methods	
		DeterminantCofactor	Konstruktor kelas, menyalin elemen matriks
		Determinant	Mengembalikan nilai determinan
		hasDeterminant	Mengembalikan true jika determinan ada dan valid
6.	DeterminantOBE	**Attributes**	
		M	Salinan matriks
		Methods	
		DeterminantOBE	Konstruktor kelas, menyalin elemen matriks
		firstNonZeroOccurance	Mencari pivot untuk operasi baris elementer
		Determinant	Mengembalikan nilai determinan
		hasDeterminant	Mengembalikan true jika determinan ada dan valid
7.	interpolasiPolinom	**Attributes**	
		points	Atribut input
		coefPolinom	Menyimpan hasil pencarian solusi
		Methods	
		interpolasiPolinom	Konstruktor, inisialisasi atribut dan pemodelan
		getHampiran	Penyulangan nilai yang ingin dicari hampirannya dari solusi

		getCoefPolinom	Mencari solusi hasil pemodelan
		getPolinomString	Mengubah tipe data solusi menjadi string (agar lebih mudah dibaca)
8.	InverseCofactor	**Attributes**	
		res	Menyimpan matriks invers
		Methods	
		InverseCofactor	Pemrosesan matriks menjadi matriks invers
		getInverse	Mengembalikan matriks invers
		hasInverse	Mengembalikan true jika matriks invers ada dan valid
9.	InverseOBE	**Attributes**	
		res	Menyimpan matriks invers
		Methods	
		InversOBE	Inisialisasi matriks, mengaugmentasi matriks dengan matriks identitas
		InversProcess	Pemrosesan matriks menjadi matriks invers
		getInverse	Mengembalikan matriks invers
		hasInverse	Mengembalikan true jika matriks invers ada dan valid
10.	SPLCramer	**Methods**	
		SPLCramer	Konstruktor matriks, inisialisasi matriks
		getSolutionVal	Mengembalikan sebuah array dari solusi yang didapatkan
		getSolutionString	Konversi solusi menjadi string
11.	SPLGauss	**Attributes**	
		M	Salinan matriks
		Methods	
		SPLGauss	Inisialisasi matriks
		firstNonZeroOccurence	Mencari pivot untuk dilakukan operasi baris elementer
		GaussProcess	Mencari solusi SPL dengan metode Gauss
		JordanProcess	Mencari solusi SPL dengan metode Gauss-Jordan
		getSolution	Membangun solusi melalui proses penyulangan balik dan memprosesnya menjadi sebuah array dari string solusi
12.	SPLInvers	**Methods**	
		SPLInvers	Inisialisasi matriks
		getSolutionVal	Mengembalikan solusi
		getSolutionString	Mengembalikan konversi solusi menjadi string

		Methods	
13.	RLB	RLB	Kontruktor, inisialisasi matriks
		getMatriksCoef	Pemodelan matriks menjadi matriks koefisien
		solveCoefReg	Pemrosesan matriks koefisien hasil pemodelan
		getRegEq	Mengembalikan solusi persamaan
		getEstimasi	Mencari nilai hampiran

Perhatikan terdapat **tiga**[bisa jadi berubah] buah ‘*parent’* atau bisa disebut tipe data utama yang digunakan dalam program utama, yaitu Matriks, AugmentedMatrix, dan interpolasiPolinomial [bisa berubah]. Kelas ini berperan sebagai *abstract data type* yang akan digunakan untuk keperluan kelas lainnya. Perhatikan juga bahwa konstruktor dari sebuah kelas dapat terdiri dari beberapa jenis bergantung pada kebutuhan objek tersebut.

Adapun struktur kelas pada *GUI*, pada umumnya hanya terdiri dari atribut panel, teks, dan tombol serta primitif berupa konstruktor dan primitif untuk melakukan aksi berdasarkan tombol dan teks yang dimasukan saat *runtime*. Bagian konstruktor GUI hampir seluruhnya berstruktur sama, yaitu inisialisasi tombol, panel, dan/atau teks, serta memberikan antarmuka pada setiap *frame*nya. sedangkan pada bagian primitif *action* terhadap masukan, berisi lojik yang dijalankan algoritma hasil dari masukan.

C. Garis Besar Program

Program pustaka ini dalam implementasinya hanya terdapat beberapa algoritma penting yang merupakan garis besar program dalam menjalankan fungsionalitasnya. Terdapat lima fungsionalitas yang utama dalam penyelesaian masalah aljabar linier ini, diantaranya yaitu, Algoritma pencarian determinan, pencarian invers, pencarian solusi SPL, interpolasi polinomial, dan regresi linier berganda. Masing-masing fungsionalitas ini diimplementasikan dengan beberapa metode. Berikut adalah penjelasan algoritma-algoritma tersebut.

1. Determinan

Algoritma determinan yang digunakan dalam program ini adalah menggunakan metode kofaktor dan reduksi baris menjadi matriks segitiga. Berikut adalah implementasi pencarian determinan menggunakan metode kofaktor.

```
public Double Determinant(){
     if(!this.hasDeterminant()){
         return Double.NaN;
```

```
        }
        Double det = 0.0;
        if (M.nRow() == 1) return M.elmt[0][0];
        else {
            for (int j = 0; j < M.nCol(); j++) {
                Cofactor cof = new Cofactor(M, 0, j);
                DeterminantCofactor Det = new DeterminantCofactor(cof);
                if (j % 2 == 0) det += M.elmt[0][j] * Det.Determinant();
                else det -= M.elmt[0][j] * Det.Determinant();
            }
            return det;
        }
}
```

Proses yang dijalankan adalah iterasi pada baris pertama kemudian cari kofaktor dari submatriks yang tidak mengandung baris dan kolom dari elemen yang bersangkutan. Perhatikan pada bagian awal, diperiksa apakah determinan valid (matriks berupa matriks persegi), jika tidak akan dikembalikan nilai *NaN*.

Adapun implementasi pencarian determinan menggunakan metode reduksi baris, sebagai berikut.

```
public Double Determinant() {
    if(!this.hasDeterminant()){
        return Double.NaN;
    }
    Double res = 1.0;
    int numOfSwaps = 0;
    int i, j, k, p;
    i = k = 0;
    while (i < M.nRow() && k < M.nCol()) {
        int i_max = firstNonZeroOccurence(i, k);
        if (Eq(M.elmt[i_max][k], 0.0)) {
            k++;
        } else {
            if (i_max != i) {
                M.swaprow(i_max, i);
                numOfSwaps++;
            }
            for (j = i + 1; j < M.nRow(); j++) {
                for (p = k + 1; p < M.nCol(); p++) {
                    if(!Eq(M.elmt[i][k], 0.0)){
                        M.elmt[j][p] = M.elmt[j][p] - M.elmt[i][p] *
M.elmt[j][k] / M.elmt[i][k];
                    }

                }
                M.elmt[j][k] = 0.0;
            }
            i++;
            k++;
        }
    }
    res *= Math.pow(-1, numOfSwaps);
    for(int l=0; l<M.nRow(); l++){
        res *= M.elmt[l][l];
```

```
    }
    return res;
}
```

Program ini melakukan proses yang mirip dengan metode Gauss, yang akan dijelaskan selanjutnya. Algoritma pada intinya adalah mengiterasi titik tumpu untuk melakukan reduksi baris menggunakan operasi baris elementer kemudian titik tumpu ini menjadi pembuat nol baris-baris yang ada di bawahnya. Perhatikan bahwa untuk memperoleh determinan, pada akhir iterasi, dilakukan dengan cara mengalikan seluruh elemen pada diagonal utama hasil matriks reduksi dan mengalikannya dengan , dengan *numofswap* adalah banyaknya penukaran baris yang sudah dilakukan saat melakukan proses baris elementer.

2. Invers

Sama dengan determinan, pencarian invers dilakukan dengan dua metode, yaitu menggunakan operasi baris elementer dan ekspansi kofaktor. Berikut adalah implementasi pencarian invers matriks menggunakan ekspansi kofaktor.

```
public InverseCofactor(Matriks m){
    super(m);
    DeterminantCofactor detcof = new DeterminantCofactor(m);
    double det = detcof.Determinant();
    if (Eq(det,0.00)) {
        System.out.println("Tidak memliki balikan");
        res = null;
    }
    else if(nRow!=nCol){
        System.out.println("Balikan Tidak Terdefinisi, Jumlah baris dan
kolom harus sama");
        res = null;
    }
    else {
        res = new Matriks(nRow, nCol);
        for (int i = 0; i < nRow; i++) {
            for (int j = 0; j < nCol; j++) {
                res.elmt[i][j] = m.elmt[i][j];
            }
        }
        CofactorMatrix cofmat = new CofactorMatrix(m);
        for (int i = 0; i < m.nRow(); i++) {
            for (int j = 0; j < m.nCol(); j++) {
                res.elmt[i][j] = cofmat.adjoin().elmt[i][j] / det;
            }
        }
    }
}
```

Algoritma ini pada dasarnya adalah mencari seluruh minor yaitu submatriks yang tidak mengandung sebuah elemen a_{ij} untuk setiap i, j indeks valid matriks. Minor ini disusun dalam kesatuan matriks dengan pola min, plus yang disebut sebagai kofaktor matriks. Hasil transpose dari matriks kofaktor disebut sebagai adjoin matriks, yang jika matriks adjoin ini dibagi skalar dengan determinan matriks tersebut, akan diperoleh invers matriks, sesuai yang diinginkan. Perhatikan bahwa, jika determinan matriks sama dengan nol atau batasan matriks tidak valid, akan diperoleh matriks yang *non-invertible* atau tidak memiliki balikan.

Adapun metode pencarian invers menggunakan metode operasi baris elementer adalah sebagai berikut.

```
public InverseOBE(Matriks m){
    super(m);
    res = new Matriks(nRow, nCol);
    if(nRow!=nCol){
        System.out.println("Balikan Tidak Terdefinisi, Jumlah baris dan
kolom harus sama");
        M = null;
    }
    else {
        M = new Matriks(nRow, 2*nCol);
        DeterminantOBE Det = new DeterminantOBE(m);
        if(Eq(Det.Determinant(), 0.0)){
            System.out.println("Tidak memiliki balikan");
            M = null;
        }
        else{
            for(int i=0; i<nRow; i++){
                for(int j=0; j<2*nCol; j++){
                    if(j<nCol) {
                        M.elmt[i][j] = m.elmt[i][j];
                        res.elmt[i][j] = m.elmt[i][j];
                    }
                    else{
                        if(j-(nCol)==i){
                            M.elmt[i][j] = 1.0;
                        }
                        else{
                            M.elmt[i][j] = 0.0;
                        }
                    }
                }
            }
        }
    }
}
private Matriks InversProcess(){
    SPLGauss inv = new SPLGauss(this.M);
    DeterminantOBE Det = new DeterminantOBE(res);

    inv.JordanProcess();
    return inv.M;

}
```

Pada metode ini, yang dilakukan adalah dua langkah besar yaitu mengaugmentasi matriks identitas dengan matriks yang dicari inversnya, kemudian hasil matriks teraugmentasi ini dilakukan proses Gauss-Jordan elimination, sehingga hasil operasi matriks akan mengasilkan matriks sebagai berikut

$$A|I \rightarrow I|A^{-1}$$

Perhatikan bahwa, matriks yang dicari dipastikan memiliki invers, sebab sudah diperiksa apakah determinan dan batasan matriks bernilai valid atau tidak.

3. Sistem Persamaan Linier

Penyelesaian sistem persamaan linier dilakukan dengan empat metode di antaranya adalah, eliminasi Gauss, eliminasi Gauss-Jordan, menggunakan matriks balikan, dan kaidah Cramer. Metode Gauss dan Gauss-Jordan implementasinya hampir sama persis dengan implementasi reduksi baris pada pencarian determinan, yang menjadi beda hanyalah bagian operasi pada Gauss, baris pada titik pivot dibagi dengan titik pivotnya, sedangkan pada Gauss-Jordan, mirip dengan Gauss, hanya iterasi "membuat nol" dilakukan dari mulai baris ke nol, hingga akhir baris.

Adapun implementasi pencarian sistem persamaan linier menggunakan balikan matriks, sebagai berikut.

```
public Double[] getSolutionVal(){
    Double[] solution = new Double[nCol-1];
    Matriks coef = new Matriks(coefRow,coefCol);     // buat matriks coef 
sendiri hehe
    coef.elmt = coefficient;                         // semoga bisa
    InverseCofactor invcoef = new InverseCofactor(coef);
    if(!invcoef.hasInverse()){
        Arrays.fill(solution, Double.NaN);
        return solution;
    }
    for(int i=0; i<nCol-1; i++){             // jika Ax = B, maka x = 
inverse(A) * B
        Double res = 0.0;
        for(int j=0; j<invcoef.nRow(); j++){
            res += invcoef.res.elmt[i][j] * constant[j];
        }
        solution[i] = res;
    }
    return solution;

}
```

Pada primitif ini, langkah yang dilakukan adalah mencari balikan dari matriks koefisien kemudian mengalikannya dengan konstanta persamaan, sesuai dengan persamaan

$$AX = B \implies X = A^{-1}B$$

Dengan A adalah matriks koefisien, X adalah matriks variabel yang dicari, dan B adalah matriks konstanta.

Terakhir, pencarian solusi SPL menggunakan kaidah Cramer, diimplementasikan sebagai berikut

```
public Double[] getSolutionVal() {
    int n = nCol-1;
    Double[] solution = new Double[n];
    Matriks matrixj = new Matriks(coefRow,coefCol);
    int i,j;
    for(i=0;i<coefRow;i++) {
        for(j=0;j<coefCol;j++) {
            matrixj.elmt[i][j] = coefficient[i][j];
        }
    }
    DeterminantOBE detClass = new DeterminantOBE(matrixj);
    if(!detClass.hasDeterminant()){
        Arrays.fill(solution, Double.NaN);
        return solution;
    }
    else{
        Double detCoef = detClass.Determinant();
        if(Eq(detCoef, 0.0)){
            Arrays.fill(solution, Double.NaN);
            return solution;
        }
        for(j=0;j<nCol-1;j++) {
            for(i=0;i<coefRow;i++) {
                matrixj.elmt[i][j] = constant[i];
            }
            Double detj = new DeterminantOBE(matrixj).Determinant();
            solution[j] = detj/detCoef;
            for(i=0;i<coefRow;i++) {
                matrixj.elmt[i][j] = coefficient[i][j];
            }
        }
    }
    return solution;
}
```

Langkah awal adalah pencarian determinan matriks koefisien sebagai pembagi dari solusi persamaan. Kolom matriks koefisien satu persatu di iterasi dengan iterasi ke-i dilakukan penukaran kolom dengan matriks konstanta. Matriks hasil pertukaran kemudian dicari determinannya, sehingga akan diperoleh

$$x_i = \frac{\Delta_i}{\Delta}$$

Dengan Δ_i adalah determinan matriks penggatian ke$-i$ dan Δ adalah determinan matriks koefisien. Sehingga akan didapatkan solusi utuh sesuai dengan kaidah Cramer.

4. Interpolasi Polinomial

Metode yang digunakan pada interpolasi tidak diberikan pada spesifikasi, sehingga penulis menggunakan metode yang tersedia pada saat membangun program ini (saat penulisan program hanya ada algoritma pencarian solusi SPL dengan metode Gauss). Implementasinya adalah sebagai berikut.

```
public interpolasiPolinom(Matriks points) {
    this.points = new Matriks(points);
    int n = points.nRow();
    Matriks matriksAugmented = new Matriks(n,n+1);
    int i,j;

    for(i=0;i<n;i++) {
        for(j=0;j<n+1;j++) {
            if(j==0) {
                matriksAugmented.elmt[i][j]=1.0;
            } else if(j==n) {
                matriksAugmented.elmt[i][j]=points.elmt[i][1];
            } else {
                matriksAugmented.elmt[i][j]=matriksAugmented.elmt[i][j-
1]*points.elmt[i][0];
            }
        }
    }
    // interpolasiPolinom dijamin solusinya unik jika koordinat x tiap
titik berbeda
    SPLGauss splAugmented = new SPLGauss(matriksAugmented);
    splAugmented.JordanProcess();
    String[] coefPolinomString = splAugmented.getSolution();
    coefPolinom = new Double[n];
    for(i=0;i<n;i++) {
        coefPolinom[i] = Double.valueOf(coefPolinomString[i]);
    }
}
```

Pencarian solusi diprekomputasi pada bagian konstruktor. Solusi dibangun dari masukan berupa koordinat-koordinat yang kemudian diolah menjadi matriks persamaan linier yang perlu dicari solusinya. Karena implementasi pencarian SPL sudah ada sebelumnya, sehingga algoritmanya hanya perlu memanggil objeknya saja. Hasil solusi SPL ini, diolah lebih lanjut agar berbentuk polinomial.

5. Regresi Linier Berganda

Berikut adalah implementasi regresi linear berganda dalam bahasa Java

```
public void getMatriksCoef(){
    for(int i=0; i<=k; i++){
        for(int j=0; j<=k; j++){
            Double xim, xjm;
            double coef = 0d;
            for(int l=0; l<n; l++){
                if(i==0) xim = 1d;
                else xim = x[i-1][l];
                if(j==0) xjm = 1d;
                else xjm = x[j-1][l];
                coef += xim*xjm;
            }
            m.elmt[i][j] = coef;
        }

        m.elmt[i][k+1] = 0d;
        for(int j=0; j<n; j++){
            Double xi;
            if(i==0) xi = 1d;
            else xi = x[i-1][j];
            m.elmt[i][k+1] += xi*y[j];
        }
    }
}

public void solveCoefReg(){
    SPLInvers splinv = new SPLInvers(this.m);
    b = splinv.getSolutionVal();
}
```

Pada primitif getMatriksCoef, permasalahan regresi terlebih dahulu dimodelkan menjadi sistem persamaan linier. Setelah matriks koefisien hasil pemodelan didapatkan, primitif selanjutnya digunakan untuk mencari solusi dari SPL tersebut. Lebih lanjut, solusi persamaan ini disubstitusi ulang ke model awal regresi untuk mencari nilai hampiran yang dihendaki.

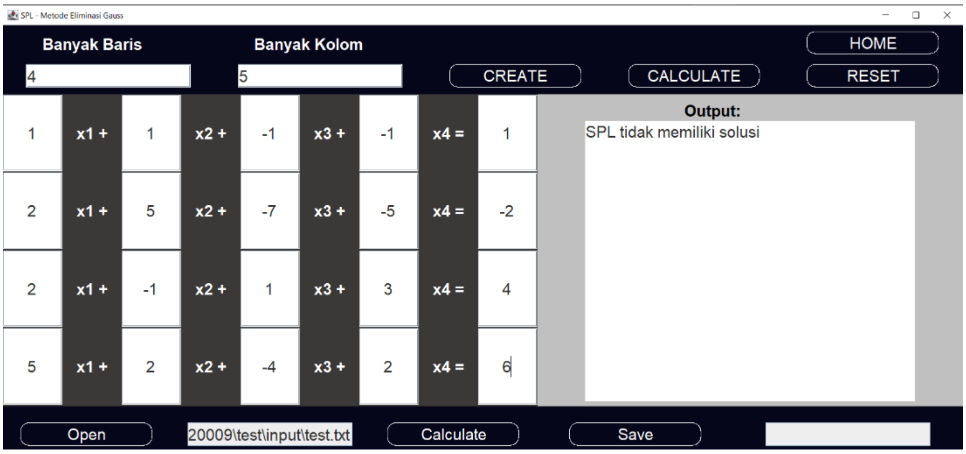
BAB 4
EKSPERIMEN

F. Menentukan Solusi Ax = b

Terdapat empat kasus uji yang akan diuji. Kasus pertama ialah

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

Menggunakan metode eliminasi Gauss, diperoleh bahwa Ax = b tidak punya solusi. Begitu juga dengan metode eliminasi Gauss-Jordan**. Untuk** metode matriks balikan, dan metode Cramer **sendiri tidak mengeluarkan output, yang artinya solusi SPL banyak atau tidak ada.**



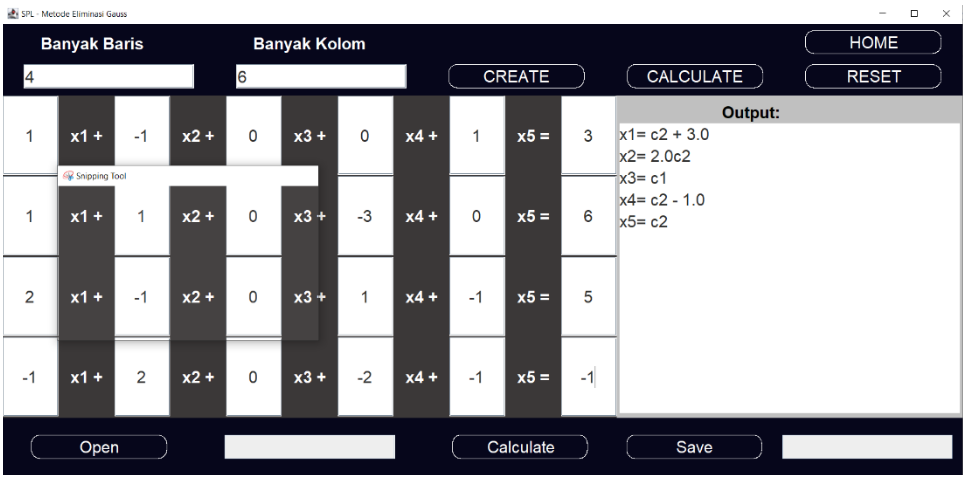
Gambar 1 Hasil masukan dan keluaran SPL

Untuk kasus kedua, matriks A dan b-nya ialah.

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

Dari kedua matriks di atas, diperoleh matriks x yang merupakan solusi dari permasalahan ini.

$$(x_1, x_2, x_3, x_4, x_5) = (c_1 + 3{,}2c_2, c_1, c_2 - 1, c_2)$$



Gambar 2 Hasil masukan dan keluaran SPL

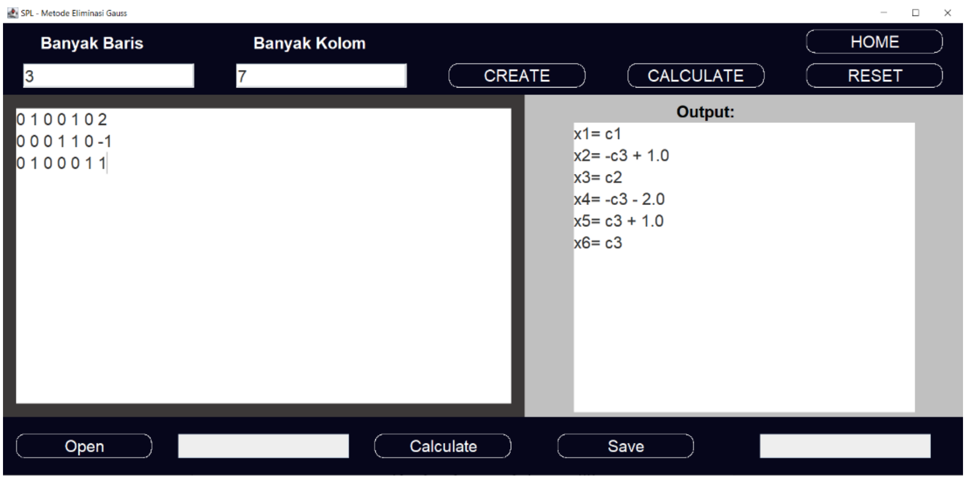
Pada kasus ini, sudah pasti bahwa solusi tidak mungkin tunggal, karena banyaknya baris lebih sedikit daripada banyaknya kolom pada matriks A.

Untuk kasus ketiga, matriks

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

Dari kedua matriks di atas, diperoleh matriks x yang merupakan solusi dari permasalahan ini.

$$(x_1, x_2, x_3, x_4, x_5, x_6) = (c_1, -c_3 + 1, c_2, -c_3 - 2, c_3 + 1, c_3)$$



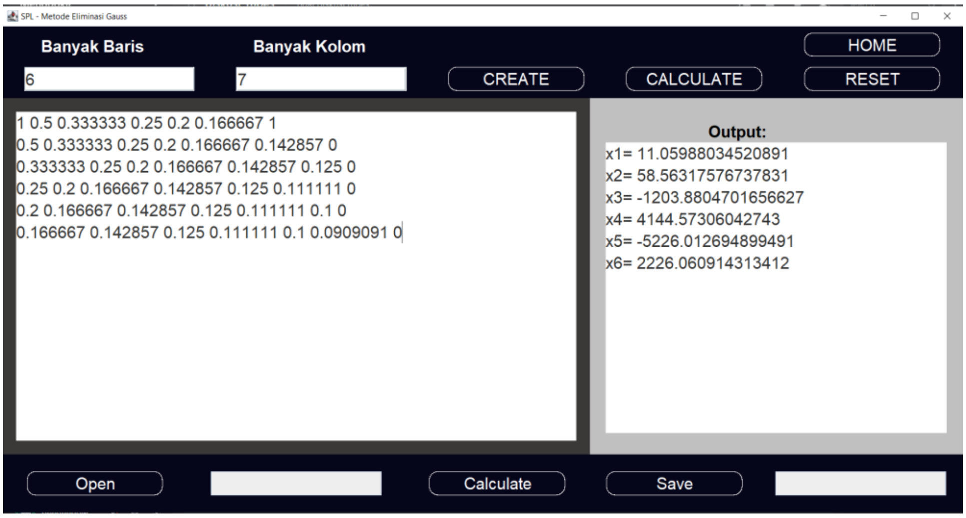
Gambar 3 Hasil masukan dan keluaran SPL

Dari matriks A, terlihat bahwa kolom 1 dan kolom 3 hanya mengandung 0, sehingga berapapun nilai dari variabel x1 dan x3 tidak akan memengaruhi variabel lain.

Untuk kasus uji keempat adalah matriks Hilbert dengan $n = 6$ dan $n = 10$.

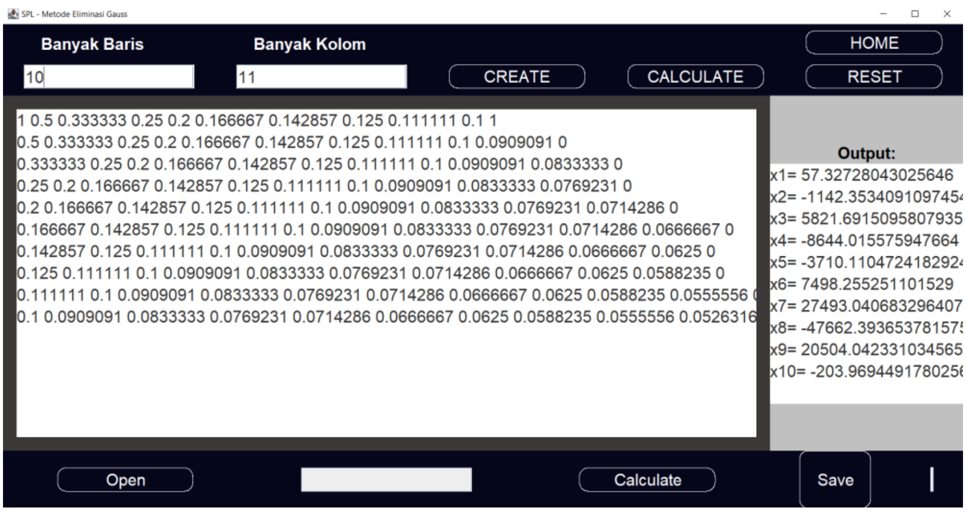
$$A = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2}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

Untuk n = 6, jika dimasukkan ke dalam program yang telah dibuat, diperoleh



Gambar 4 Hasil masukan dan keluaran SPL

Untuk n = 10, jika dimasukkan ke dalam program yang telah dibuat, diperoleh



Gambar 5 Hasil masukan dan keluaran SPL

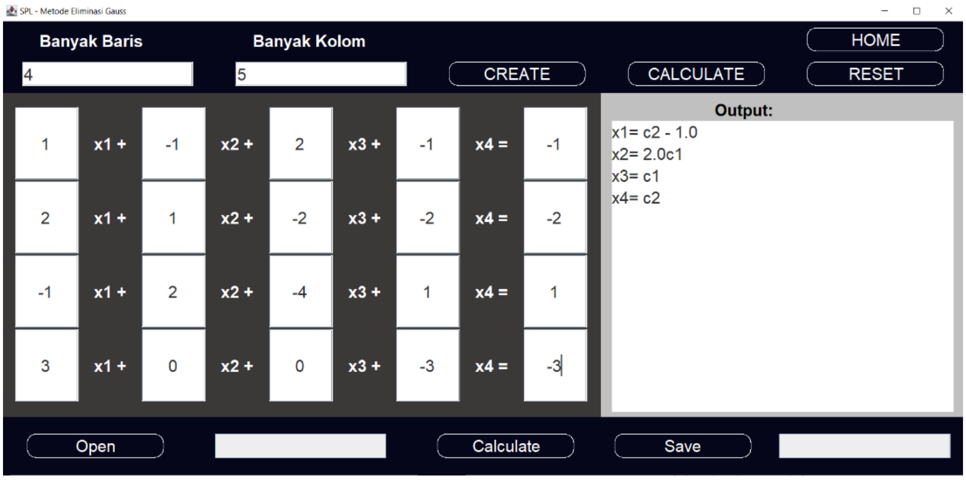
G. SPL Berbentuk matriks *augmented*

Terdapat dua kasus SPL yang berbentuk matriks *augmented*. Untuk kasus pertama, matriks *augmented*-nya ialah

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

Dengan memasukkan matriks tersebut ke program yang telah dibuat, diperoleh hasil sebagai berikut.

$$(x_1, x_2, x_3, x_4) = (c_2 - 1, 2c_1, c_1, c_2)$$



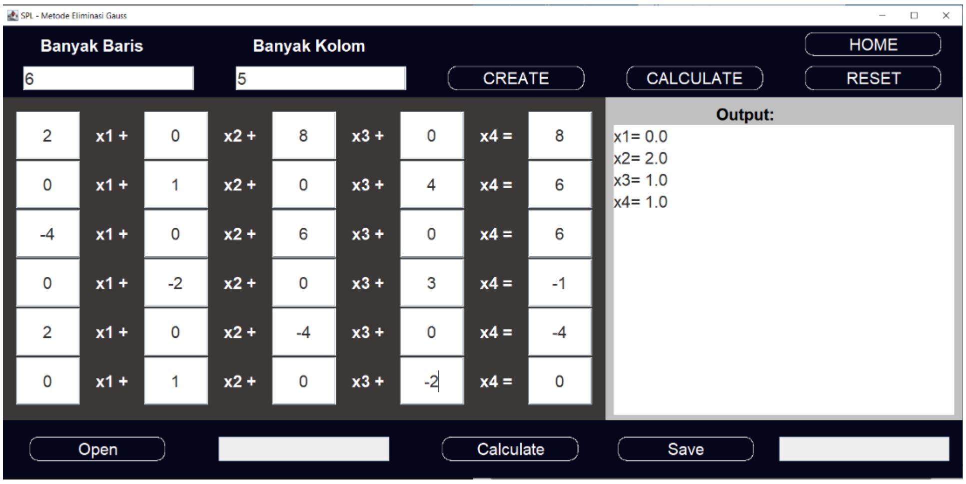
Gambar 6 Hasil masukan dan keluaran matriks augmented

Untuk kasus kedua, diberikan matriks *augmented* berikut.

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

Dengan memasukkan matriks tersebut ke program yang telah dibuat, diperoleh

$$(x_1, x_2, x_3, x_4) = (0,2,1,1)$$



Gambar 7 Hasil masukan dan keluaran matriks augmented

H. SPL Berbentuk Sekumpulan Persamaan

SPL pertama yang akan dicari solusinya ialah sebagai berikut

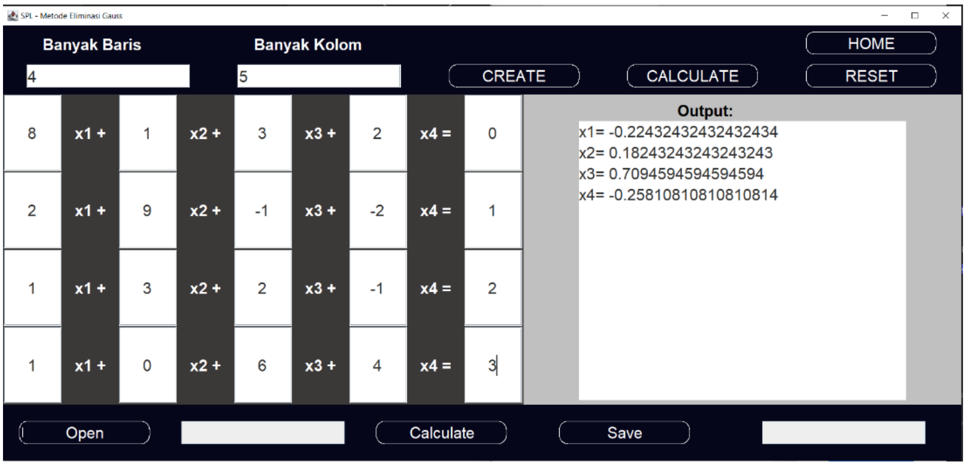
$$\begin{aligned}
8x_1 + x_2 + 3x_3 + 2x_4 &= 0\\
2x_1 + 9x_2 - x_3 - 2x_4 &= 1\\
x_1 + 3x_2 + 2x_3 - x_4 &= 2\\
x_1 + 6x_3 + 4x_4 &= 3
\end{aligned}$$

SPL di atas dapat diubah ke bentuk matriks *augmented*, diperoleh

$$\begin{bmatrix} 8 & 1 & 3 & 2 & 0\\ 2 & 9 & -1 & -2 & 1\\ 1 & 3 & 2 & -1 & 2\\ 1 & 0 & 6 & 4 & 3 \end{bmatrix}$$

Dengan memberi masukan ke program berupa matriks di atas, diperoleh hasil berikut.

$$(x_1, x_2, x_3, x_4) = (-0.22, 0.18, 0.71, -0.26)$$



Gambar 8 Hasil masukan dan keluaran SPL

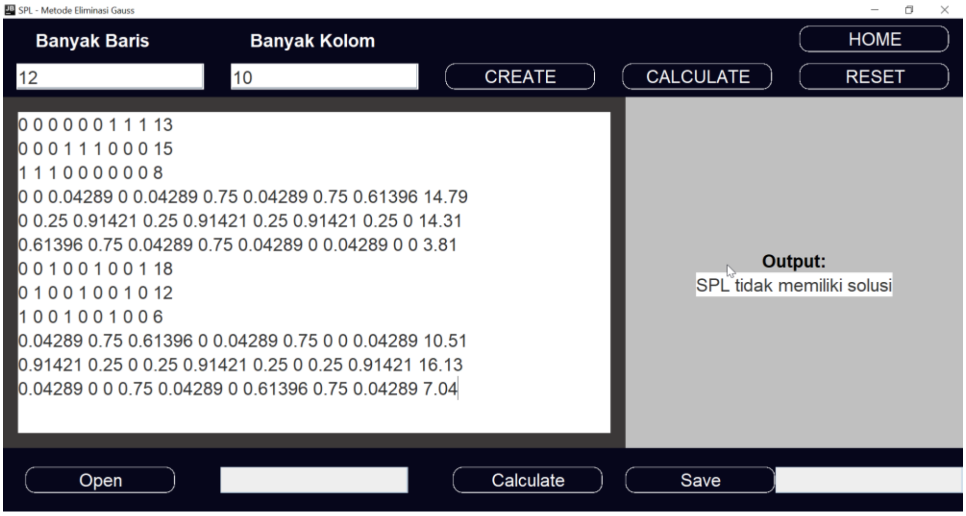
Selanjutnya, SPL yang solusinya akan dicari ialah SPL berikut.

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

SPL di atas dapat diubah ke bentuk matriks *augmented*, sehingga diperoleh

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 13 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 15 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \\
0 & 0 & 0.04289 & 0 & 0.04289 & 0.75 & 0.04289 & 0.75 & 0.61396 & 14.79 \\
0 & 0.25 & 0.91421 & 0.75 & 0.91421 & 0.25 & 0.91421 & 0.25 & 0 & 14.31 \\
0 & 0.75 & 0.04289 & 0.75 & 0.04289 & 0 & 0.04289 & 0 & 0.61396 & 3.81 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 18 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 12 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 6 \\
0.04289 & 0.75 & 0.61396 & 0 & 0.04289 & 0.75 & 0 & 0 & 0.04289 & 10.51 \\
0.91421 & 0.25 & 0 & 0.25 & 0.91421 & 0.25 & 0 & 0.25 & 0.91421 & 16.13 \\
0.04289 & 0 & 0 & 0.75 & 0.04289 & 0 & 0 & 0.75 & 0.04289 & 7.04
\end{bmatrix}$$

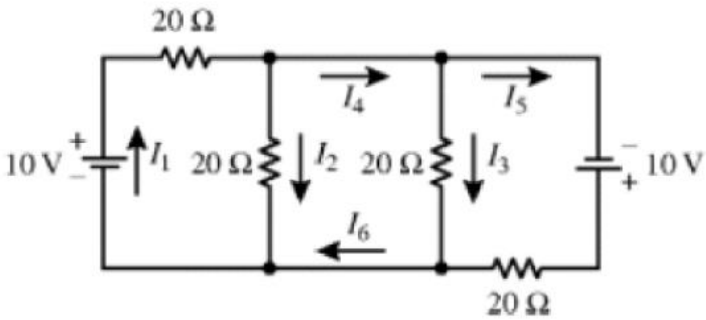
Dengan memasukkan matriks di atas ke program yang telah dibuat, diperoleh bahwa SPL tidak memiliki solusi.



Gambar 9 Hasil masukan dan keluaran SPL

I. Menentukan Arus pada Rangkaian Listrik

Pada kasus ini, diberikan suatu rangkaian listrik seperti berikut.



Gambar 10 Ilustrasi rangkaian listrik

Dari rangkaian di atas, diperoleh SPL dengan enam variabel.

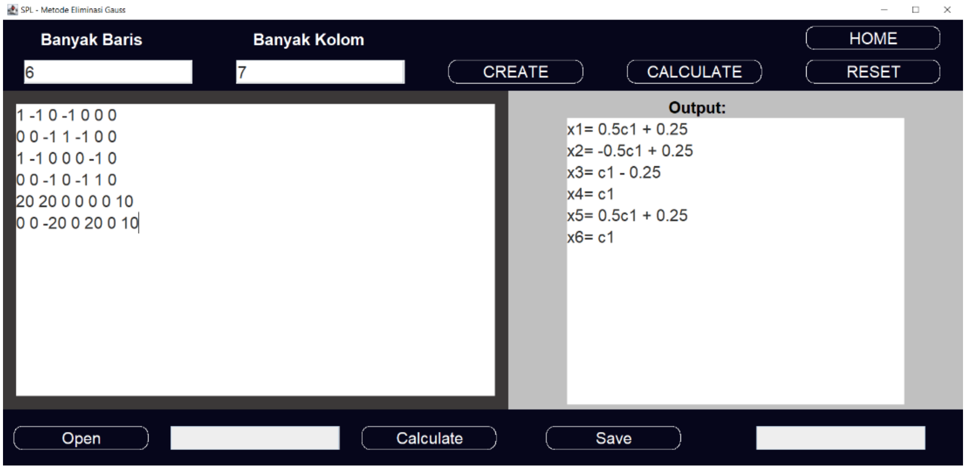
$$
\begin{array}{cccccccc}
i_1 & -i_2 & & -i_4 & & & = 0 \\
 & & -i_3 & +i_4 & -i_5 & & = 0 \\
i_1 & -i_2 & & & & -i_6 & = 0 \\
 & & -i_3 & & -i_5 & +i_6 & = 0 \\
20i_1 & +20i_2 & & & & & = 10 \\
 & & -20i_3 & & +20i_5 & & = 10
\end{array}
$$

Dari SPL di atas, diperoleh suatu matriks *augmented* ah weit gua sala

$$\begin{bmatrix} 1 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 \\ 20 & 20 & 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & -20 & 0 & 20 & 0 & 10 \end{bmatrix}$$

Jika matriks di atas dimasukkan ke dalam program yang telah dibuat, diperoleh

$$(I_1, I_2, I_3, I_4, I_5, i_6) = (0.5c_1 + 0.25, -0.5c_1 + 0.25, c_1 - 0.25, c_1, 0.5c_1 + 0.25, c_1)$$



Gambar 11 Hasil masukan dan keluaran SPL pemodelan rangkaian listrik

J. Permasalahan pada Sistem Reaktor

Permasalahan pada kasus ini sudah disederhanakan ke dalam SPL berikut,

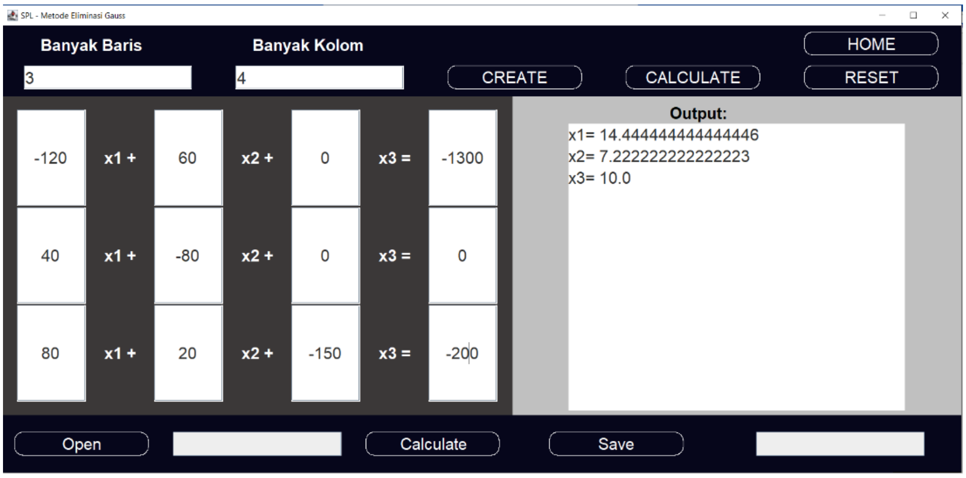
$$\begin{aligned} m_{A_{in}} + Q_{BA}x_B - Q_{AB}x_A - Q_{AC}x_A &= 0 \\ Q_{AB}x_A - Q_{BA}x_B - Q_{BC}x_B &= 0 \\ m_{C_{in}} + Q_{AC}x_A + Q_{BC}x_B - Q_{C_{out}}x_C &= 0 \end{aligned}$$

dengan $Q_{AB} = 40$, $Q_{AC} = 80$, $Q_{BA} = 60$, $Q_{BC} = 20$, $Q_{Cout} = 150$, $m_{Ain} = 1300$, dan $m_{Cin} = 200$. Dengan menyulihkan variabel-variabel tersebut ke SPL di atas dan menyusun ulang SPL, diperoleh suatu matriks *augmented* berikut.

$$\begin{bmatrix} -120 & 60 & 0 & -1300 \\ 40 & -80 & 0 & 0 \\ 80 & 20 & -150 & -200 \end{bmatrix}$$

Dengan memasukkan matriks tersebut ke program yang telah dibuat, diperoleh

$$(x_A, x_B, x_C) = (\mathbf{14.44}, \mathbf{7.22}, \mathbf{10})$$



Gambar 12 Hasil masukan dan keluaran SPL sistem reaktor

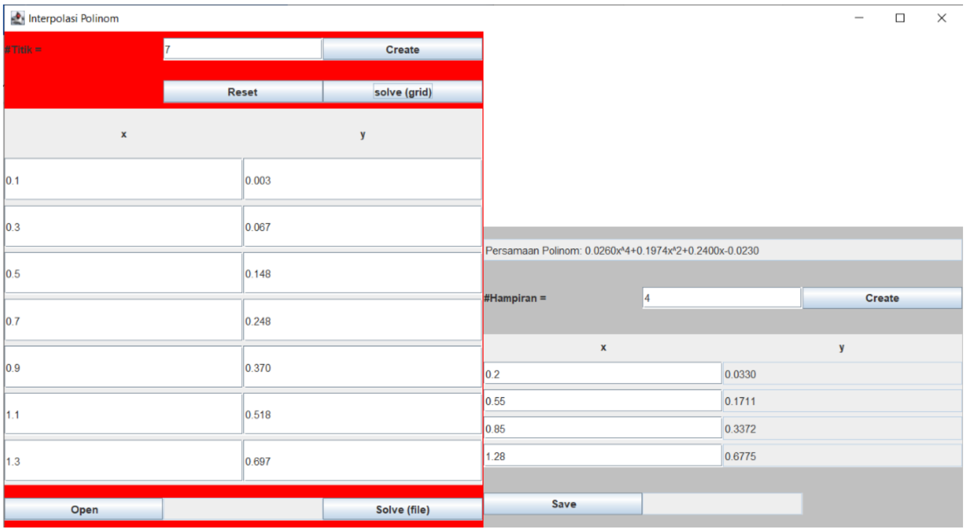
K. Studi Kasus Interpolasi

Pada kasus pertama interpolasi, diberikan tujuh titik yang akan dicari polinom interpolasinya. Dengan memasukkan titik-titik tersebut ke program yang telah dibuat, diperoleh polinomnya, yaitu $\boldsymbol{f(x)} = \mathbf{0.0260}\boldsymbol{x}^{\mathbf{4}} + \mathbf{0.1974}\,\boldsymbol{x}^{\mathbf{2}} + \mathbf{0.2400}\boldsymbol{x} - \mathbf{0.0230}$

Walaupun terdapat tujuh titik yang diberikan untuk mencari polinom interpolasinya, polinom yang diperoleh berderajat 4. Hal ini disebabkan karena koefisien dari x6 dan x5 sangat kecil sehingga terabaikan oleh program.

Selain itu, nilai-nilai hampiran dari beberapa x diperoleh

x	y
0.2	0.0330
0.55	0.1711
0.85	0.3372
1.28	0.6775



Gambar 13 Studi kasus interpolasi

Pada kasus kedua, diberikan data jumlah kasus baru Covid-19 di Indonesia yang berisi tanggal dan jumlah kasus baru. Tanggal dikonversi menjadi desimal dengan formula berikut.

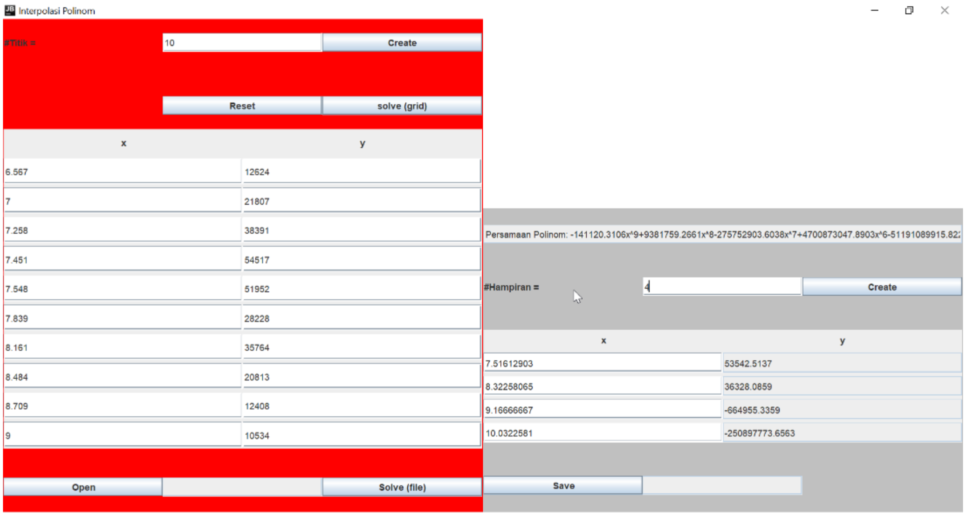
$$\text{tanggal(desimal)} = \text{bulan} + (\text{tanggal/jumlah hari pada bulan tersebut})$$

Dengan formula di atas, diperoleh data yang berisi tanggal dalam bentuk desimal dan jumlah kasus harian Covid-19 di Indonesia. Dengan memasukkan data ini ke program yang telah dibuat, diperoleh fungsi berikut

$$\begin{aligned} F(x) \;=\; & -141120.3106x_9 + 9381759.2661x_8 - 275752903.6038x_7 \\ & + 4700873047.8903x_6 - 51191089915.8224x_5 \\ & + 369011568500.2981x_4 - 1759197443156.9820x_3 \\ & + 5342144345319.0420x_2 - 9362383549278.9180x \\ & + 7200305831156.5590 \end{aligned}$$

Diperoleh juga hampiran dari beberapa nilai x.

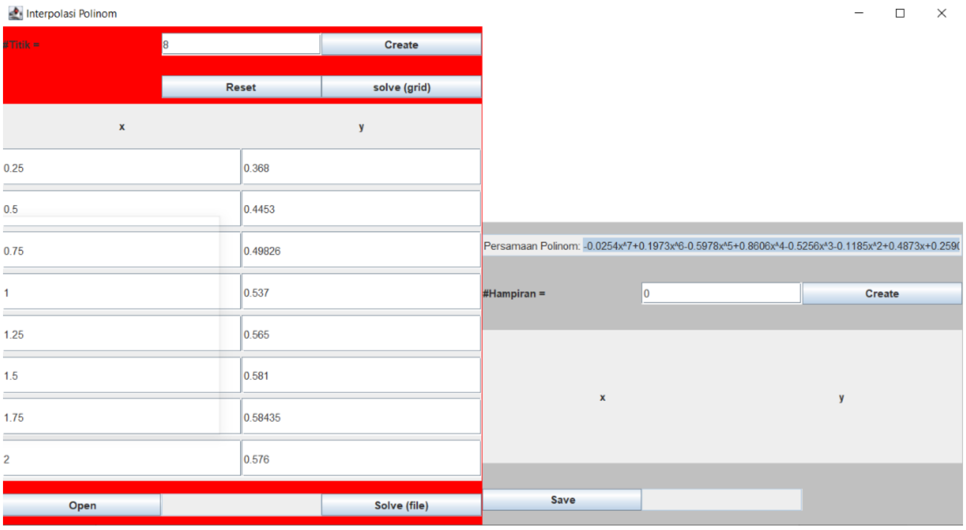
x	y
7.52	53543
8.32	36328
9.17	-664955
10.03	-250897774



Gambar 14 Studi kasus interpolasi

Pada kasus ketiga, diberikan sebuah fungsi yang akan dicari polinom interpolasinya. Dengan mengambil n=8, diperoleh

$$F(x) = -0.0254x^7 + 0.1973x^6 - 0.5978x^5 + 0.8606x^4 - 0.5256x^3 - 0.1185x^2 + 0.4873x + 0.2590$$

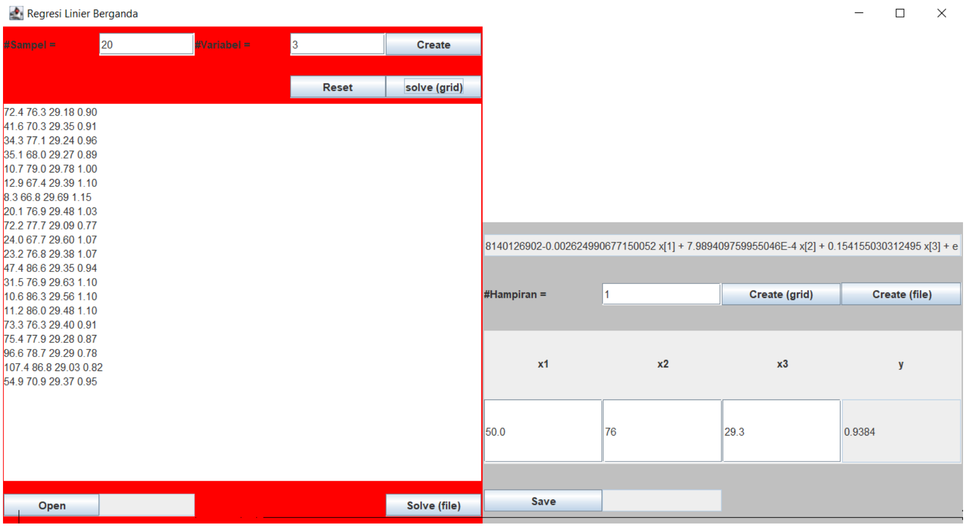


Gambar 15 Studi kasus interpolasi

L. Studi Kasus Regresi Linier Berganda

Pada kasus ini, diberikan 20 data tentang nilai Nitrous Oxide yang dipengaruhi oleh tiga hal, yaitu kelembapan, temperatur, dan tekanan. Data-data ini kemudian dimasukkan ke dalam program yang telah dibuat, lalu diperoleh persamaan regresinya, yaitu y = -3.5077781401269O2-0.002624990677150052 x[1] + 7.989409759955046E-4 x[2] + 0.154155030312495 x[3] + e

Selain itu, diperoleh juga nilai hampiran jika kelembapan 50%, temperatur 76°F, dan tekanan udara 29.30, yaitu 0.9384.



Gambar 16 Studi kasus regresi linier berganda

BAB 5
KESIMPULAN, SARAN, DAN REFLEKSI

M. Kesimpulan

Setelah membuat program dan mengetesnya dengan beberapa persoalan, dicapai kesimpulan sebagai berikut.

1. Telah dibuat 13 kelas sebagai pustaka aljabar linier. Setiap satu atau dua metode untuk menyelesaikan persoalan dijadikan suatu objek kelas. Namun, antar objek kelas juga memiliki hubungan pewarisan dan/atau komposisi.
2. Telah dibuat antarmuka pengguna grafis yang memanfaatkan *external library* Java Swing untuk memudahkan pengguna, contohnya ketika ingin mencari determinan matriks $6 * 6$ akan dibuat sel $6 * 6$ yang perlu diisi, hal ini juga membantu pembuat program.
3. Program memenuhi spesifikasi yang diberikan dan bekerja baik pada setiap kasus uji

N. Saran

Terdapat beberapa saran untuk pengembangan program ini, yaitu sebagai berikut.

1. Pustaka yang dibuat ini terdiri dari banyak file khusus metode untuk persoalan tertentu. Bila dirasa penting untuk membuat satu objek matriks yang bisa menyelesaikan semua persoalan dapat dibuat kelas berisi seluruh atribut dan metode yang ada.
2. Antarmuka pengguna grafis dapat dikembangkan lagi.
3. Pustaka masih dapat ditambah dengan properti matriks aljabar linier lainnya, seperti nilai eigen, rank, dan diagonalisasi.

O. Refleksi

Adapun refleksi atau *lesson learned* dari tugas ini adalah sebagai berikut.

1. Dapat belajar bahasa baru, yaitu Java yang sangat mendukung pemrograman berorientasi objek. Selain itu, Java juga dapat dijalankan pada berbagai sistem operasi.
2. Selama mengerjakan tugas ini juga digunakan Intellij yang mendukung Java dan github.

3. Dalam mengerjakan tugas besar, dapat didiskusikan garis besar program dan pengaturan teknologi yang digunakan di awal agar lebih terarah. Selain itu, pembagian pj di awal juga membantu.

Daftar Pustaka

Anton, H., & Rorres, C. (2005). *Elementary Linear Algebra, ninth edition: Applications version*. Wiley.