LAPORAN TUGAS BESAR 1
ALJABAR LINIER DAN GEOMETRI
IF2123

Rexy Gamaliel Rumahorbo 13519010

Mohammad Afif Akromi 13519110

Muhammad Jafar Gundari 13519197

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2020

BAB I

DESKRIPSI MASALAH

Membuat program dalam Bahasa Java untuk

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

Spesifikasi program

1. Program dapat menerima masukan (input) baik dari keyboard maupun membaca masukan dari file text.Untuk SPL, masukan dari keyboard adalah m, n, koefisien aij, dan bi. 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 aij. 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, (x0, y0),(x1, y1),..., (xn, yn), dan nilai x yang akan ditaksir nilai fungsinya. Jika masukannya dari file, maka titik-titik dinyatakan pada setiap baris tanpa koma dan tanda kurung. Misalnya jika titik-titik datanya adalah (8.0, 2.0794), (9.0, 2.1972), dan (9.5, 2.2513), maka di dalam file text ditulis sebagai berikut:

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

4. Untuk persoalan regresi, masukannya jika dari keyboard adalah n (jumlah peubah x), semua nilai-nilai x_{1i}, x_{2i}, ...,x_{ni}, nilai y_i, dan nilai-nilai x_k yang akan ditaksir nilai fungsinya. Jika masukannya dari file, maka titik-titik dinyatakan pada setiap baris tanpa koma dan tanda kurung.
5. Untup persoalan SPL, luaran (output) program adalah solusi SPL. Jika solusinya tunggal, tuliskan nilainya. Jika solusinya tidak ada, tuliskan solusi tidak ada, jika solusinya banyak, maka tuliskan solusinya dalam bentuk parametrik (misalnya $x_4 = -2$, $x_3 = 2s–t$, $x_2 = s$, dan $x_1 = t$.)
6. Untuk persoalan determinan dan matriks balikan,maka luarannya sesuai dengan persoalan masing-masing
7. Untuk persoalan polinom interpolasi dan regresi, luarannya adalah persamaan polinom/regresi dan taksiran nilai fungsi pada x yang diberikan.
8. Luaran program harus dapat ditampilkan pada layar komputer dan dapat disimpan ke dalam file
9. Bahasa program yang digunakan adalah Java.
10. Program tidak harus berbasis GUI, cukup text-based saja, namun boleh menggunakan GUI (memakai kakas Eclipse misalnya).
11. Program dapat dibuat dengan pilihan menu. Urutan menu dan isinya dipersilakan dirancang masing-masing. Misalnya, menu:

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

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

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

BAB II

TEORI SINGKAT

I. Sistem Persamaan Linear

Sistem persamaan linier (SPL) dengan n peubah (variable) dan m persamaan adalah berbentuk

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

Sistem persamaan linear seperti ini dapat dinyatakan dalam bentuk perkalian matriks **Ax= b,** yang dalam hal ini x_i adalah peubah, a_{ij} dan b_i adalah koefisien ∈ 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 peubah dan n persamaan). Solusi sebuah SPL mungkin tidak ada, banyak, atau hanya satu (unik/tunggal).

Untuk memahami metode reduksi Gauss dan Gauss-Jordan, terlebih dahulu perlu diketahui istilah matriks eselon baris dan matriks eselon baris tereduksi, serta Operasi Baris Elementer (OBE). OBE merupakan operasi pada matriks berupa: pertukaran 2 buah baris dan penjumlahan suatu baris dengan k kali baris lainnya. Sementara itu, sebuah matriks M disebut matriks eselon baris jika memenuhi:

a. Jika suatu baris tidak terdiri dari seluruhnya angka 0, maka elemen tidak nol pertama yang muncul pada baris tersebut adalah 1, yang disebut sebagai leading one.
b. Untuk 2 buah baris yang masing-masing memiliki *leading one*, *leading one* baris yang terletak di bawah posisinya harus menjorok ke kanan dari *leading one* dari baris yang terletak di atas.
c. Setiap baris yang seluruh elemennya terdiri dari angka 0 terletak di bagian bawah matriks.

Metode Gauss menyederhanakan suatu matriks M menggunakan OBE sedemikian sehingga M menjadi matriks eselon baris. Dalam bentuk matriks eselon baris, dapat dicari

solusi dari SPL dengan membentuk persamaan yang direpresentasikan oleh matriks M, kemudian mencari solusinya dengan aljabar.

Gaussian elimination

$$\left[\begin{array}{ccc|c} 1 & * & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 1 & * \end{array}\right]$$

row echelon form

Gauss-Jordan elimination

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & * \\ 0 & 1 & 0 & * \\ 0 & 0 & 1 & * \end{array}\right]$$

reduced row echelon form

Selain matriks eselon baris, terdapat juga istilah matriks eselon baris tereduksi, yakni matriks eselon baris yang memenuhi: untuk setiap kolom yang terdapat *leading one*, elemen lainnya pada kolom tersebut adalah 0. Metode Jordan memanfaatkan OBE untuk mengubah sebuah matriks eselon baris menjadi matriks eselon baris tereduksi, sehingga metode ini disebut juga metode Gauss-Jordan. Pada metode ini, matriks langsung dapat ditentukan solusinya.

Determinan dari sebuah matriks M berukuran n × n adalah

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

Determinan matriks M berukuran n × n dapat dihitung dengan beberapa cara: reduksi baris dan ekspansi kofaktor. Pada metode reduksi baris, matriks M direduksi menjadi sebuah matriks segitiga atas, yakni matriks dengan elemen aij = 0 untuk i > j. Nilai determinan M dihitung dari hasil kali seluruh elemen diagonal M. Sementara itu, untuk metode ekspansi kofaktor, nilai determinan ditentukan dari penjumlahan hasil kali dari nilai elemen aij dengan kofaktornya, cij, untuk suatu baris atau kolom tertentu pada M. Kofaktor suatu elemen aij dari sebuah matriks M adalah $(-1)^{i+j}m_{ij}$, dengan m_{ij} adalah minor entri dari a_{ij}, yakni determinan dari submatriks M yang tidak terletak pada baris ke-i dan kolom ke-j. Oleh karena itu, metode ekspansi kofaktor ini dapat diselesaikan secara rekursif, karna determinan dari submatriks M tersebut dapat juga diselesaikan dengan kofaktor. Basis dari determinan submatriks ini adalah saat matriks berukuran 2 × 2, yang nilainya $a_{00} \times a_{11} - a_{01} \times a_{10}$. Untuk optimalisasi, dapat dilakukan kombinasi dari kedua metode ini untuk menentukan determinan matriks. Pada matriks M dilakukan reduksi baris sedemikian sehingga terdapat beberapa baris atau kolom yang elemennya terdapat 0, kemudian dilakukan ekspansi kofaktor. Hal ini bertujuan untuk mengeliminasi perhitungan

kofaktor yang berkorespondensi dengan suatu elemen yang bernilai 0, karena hasil kalinya sudah pasti 0.

Invers dari suatu matriks M berukuran n × n biasa dinotasikan dengan M-1 dan memenuhi M.M-1 = M-1.M = I, di mana I adalah matriks identitas.

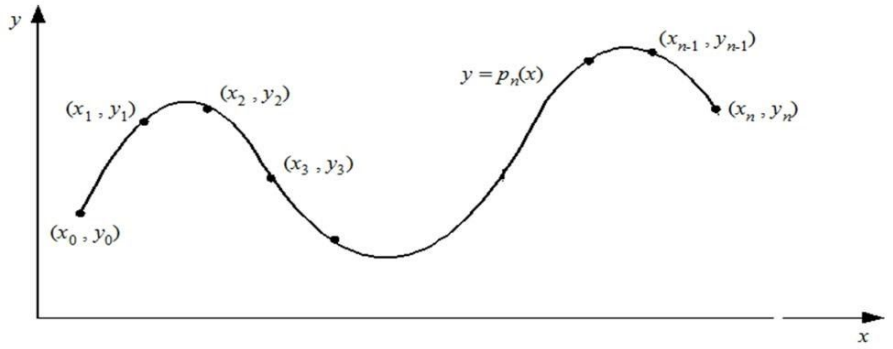
Untuk reduksi baris, dibuat sebuah matriks MAugmented berukuran n × 2n yang beranggotakan elemen-elemen matriks asal M untuk n kolom pertama dan elemen-elemen matriks identitas ukuran n × n pada n kolom terakhir. Selanjutnya, dengan menggunakan OBE dalam implementasi Gauss-Jordan, MAugmented diubah menjadi matriks eselon baris tereduksi. Dengan kata lain, diterapkan OBE sedemikian sehingga elemen-elemen pada n kolom matriks pertama menjadi matriks identitas berukuran n. Elemen-elemen pada n kolom terakhir pada MAugmented yang telah dioperasikan dengan metode Gauss-Jordan tersebut adalah invers dari matriks M. Namun ada kalanya n kolom pertama pada MAugmented tidak dapat dijadikan matriks identitas. Dalam hal ini, matriks M tidak memiliki invers.

Untuk metode ekspansi kofaktor, setiap elemen pada invers matriks M dihitung dengan menghitung kofaktornya. Kofaktor suatu elemen aij dari sebuah matriks M adalah $(-1)^{i+j}m_{ij}$, dengan m_{ij} adalah minor entri dari a_{ij}, yakni determinan dari submatriks M yang tidak terletak pada baris ke-i dan kolom ke-j. Oleh karena itu, metode ekspansi kofaktor ini dapat diselesaikan secara rekursif, karna determinan dari submatriks M tersebut dapat juga diselesaikan dengan kofaktor. Basis dari determinan submatriks ini adalah saat matriks berukuran 2 × 2, yang nilainya $a_{00} \times a_{11} - a_{01} \times a_{10}$.

SPL memiliki banyak aplikasi dalam bidang sains dan rekayasa, dua diantaranya diterapkan pada tugas besar ini, yaitu interpolasi polinom dan regresi linier.

II. 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 $a_0, a_1, a_2, ..., a_n$)

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

Solusi sistem persamaan lanjar ini, yaitu nilai a_0, a_1, ..., a_n, diperoleh dengan menggunakan metode eliminasi Gauss 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 sisten 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: $p_2(9.2) = 0.6762 + 0.2266(9.2) -0.0064(9.2)2= 2.2192$.

III. Regresi Linear Berganda

Regresi Linear (akan dipelajari lebih lanjut di Probabilitas dan Statistika) merupakan salah satu metode untuk memprediksi nilai selain menggunakan Interpolasi Polinom. Meskipun sudah ada rumus jadi untuk menghitung regresi linear sederhana, terdapat rumus umum dari regresi linear yang bisa digunakan untuk regresi linear berganda, yaitu.

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

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

$$\begin{array}{lllll}
nb_0 + b_1 \sum_{i=1}^{n} x_{1i} & + b_2 \sum_{i=1}^{n} x_{2i} & + \cdots + b_k \sum_{i=1}^{n} x_{ki} & = \sum_{i=1}^{n} y_i \\
b_0 \sum_{i=1}^{n} x_{1i} + b_1 \sum_{i=1}^{n} x_{1i}^2 & + b_2 \sum_{i=1}^{n} x_{1i}x_{2i} + & \cdots + b_k \sum_{i=1}^{n} x_{1i}x_{ki} & = \sum_{i=1}^{n} x_{1i}y_i \\
\vdots \quad \vdots & \vdots & \vdots & \vdots \\
b_0 \sum_{i=1}^{n} x_{ki} + b_1 \sum_{i=1}^{n} x_{ki}x_{1i} & + b_2 \sum_{i=1}^{n} x_{ki}x_{2i} + & \cdots + b_k \sum_{i=1}^{n} x_{ki}^2 & = \sum_{i=1}^{n} x_{ki}y_i
\end{array}$$

Dengan merepresentasikan persamaan tersebut ke dalam matriks, sistem persamaan linier tersebut dapat diselesaikan dengan menggunakan salah satu metode dalam subbab pertama bagian ini, seperti metode eliminasi Gauss, metode Gauss-Jordan, metode invers, dan metode Cramer.

BAB III

IMPLEMENTASI PROGRAM DALAM JAVA

- **ADT Matriks**

  Diimplementasikan dengan nama Matriks.java. Untuk memecahkan masalah yang sudah disebutkan di awal, maka hal pertama dibuat abstract data type untuk Matriksnya, yang memiliki atribut array 2 dimensi untuk menyimpan elemen serta panjang baris dan kolomnya. Setiap class menggunakan ADT Matriks untuk memproses solusi. Memiliki setter,getter umum, serta KaliMatriks dan Transposematriks.

  Tambahan lain yaitu InputMatrix.java untuk melakukan proses input data oleh user baik melalui console ataupun file. Kemudian OutputMatrix.java yang digunakan untuk menampilkan matriks.

- **Main Menu**

  Diimplementasikan dengan nama Main.java. Class ini adalah fungsi utama yang mengimplementasikan seluruh class java yang lain. Main.java ini merupakan class yang dijalankan user ketika ingin menyelesaikan permasalahan yang sudah disebutkan di Bab I. Pertama user akan dihadapkan pada menu seperti berikut

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

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

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

User diminta memilih menu dengan inputan angka sesuai dengan yang tersedia di MENU. Selanjutnya akan diminta memasukan jenis input data, apakah melalui file(.txt) atau melalui console. Lalu akan diproses sesuai permintaan kemudian menampilkannya ke console juga mendapatkan hasilnya ke dalam file baru bernama “result.txt”.

- **SPL Matriks**

Pada class SPLMatriks.java ini terdapat beberapa fungsi dan prosedur untuk mencari solusi dari suatu SPL (Sistem Persamaan Linear). Ada 3 kondisi yang bisa terjadi ketika menghitung SPL. Yang pertama adalah solusinya tunggal artinya solusi SPLnya tepat ada 1 solusi. Selanjutnya ada SPL Parametrik yang artinya solusi SPL nya ada sangat banyak kemungkinan yang memenuhinya. Dan yang ketiga solusinya tidak ada artinya SPL tersebut tidak memilik satu solusi. Di class SPL Matriks ini ada 2 algoritma utama yang digunakan untuk mencari solusi dari suatu SPL.

Terdapat beberapa prosedur/fungsi pada SPLMatriks:

- Fungsi reduksiOBE(Matriks matriks) → fungsi ini bertujuan untuk mengubah augmented SPL matriks menjadi matriks segitiga bawah atau segitiga atas.
- Fungsi reduksiOBEJordan(Matriks matriks) → fungsi ini bertujuan untuk mengubah augmented SPL matriks menjadi matriks identitas sehingga bisa dicaria solusi dari SPLnya.
- Fungsi Determinan(Matriks matriks) → fungsi ini bertujuan untuk mencari determinan dari suatu matriks dengan bantuan fungsi reduksiOBE. Pertama kita lakukan operrasi reduksiOBE habis itu kita panggil fungsi Determinan yang akan menghitung determinannya dengan mengkalikan seluruh elemen diagonalnya.
- isParametrik(Matriks matriks) → fungsi ini bertujuan untuk mendeteksi hasil dari reduksiOBE atau reduksiOBEJordan apakah solusinya berbentuk parametrik.
- isNotHaveSolution(Matriks matriks) → fungsi ini bertujuan untuk mendeteksi hasil dari reduksiOBE atau reduksiOBEJordan apakah solusinya tidak ada..

- **SPL Invers**

Diimplementasikan dengan nama SPLinvers.java dan inverseKofaktor.java. Kelas ini berfungsi untuk menyelesaikan sistem persamaan linear dengan menggunakan metode invers/balikan. Cara untuk membuat matriks balikannya digunakan metode OBE gauss-jordan. Selanjutnya solusi diperoleh dengan cara mengalikan matriks invers dari koefisien dengan matriks Y pada persamaan. Jika determinan koefisien bernilai 0, maka SPL yang diberikan tidak dapat diselesaikan oleh kelas ini karena solusi tidak unik.

Terdapat beberapa prosedur/fungsi pada SPLinvers :

- Prosedur SplitMatriks(Matriks MKiri, Matriks MKanan) → Untuk memisahkan matriks keofisien dan hasil dari matriks. MKiri merupakan matriks X berukuran n x m-1, MKanan merupakan matriks Y berukuran n x 1
- Fungsi inverseMatriks(Matriks matrik) → Mengembalikan matriks baru yang merupakan balikannya
- Fungsi solusiSPLInverse(Matriks MAugmented) → mengembalikan solusi dari SPL dalam bentuk matriks
- Prosedur printSolusiSPLInvers(Matriks MatriksX) → Digunakan untuk menampilkan solusi ke console sekaligus menyimpan kedalam file

Adapun fungsi/prosedur yang terdapat pada inverseKofaktor.java adalah

- ❖ prosedur hitungKofaktor → prosedur untuk menghitung kofaktor tiap minor matriks
- ❖ fungsi hitunDeterminan → Fungsi untuk menghitung tiap determinan kofaktor
- ❖ Fungsi hitungAdjoin → Fungsi untuk menghitung Adjoin hasil transpose dari kofaktor dan minor
- ❖ hitungInverse → Fungsi untuk menghitung inverse dari suatu Matriks, Inverse Matriks dirumuskan Inverse = 1/Determinan * Adjoin

- **SPL Cramer**

Diimplementasikan dengan nama SPLCramer.java. Kelas ini pada dasarnya hanya menampilkan hasil dari solusi SPL. Proses untuk mendapatkan hasil terdapat pada kelas Cramer.java. Pada kelas tersebut terdapat fungsi untuk mengembalikan nilai determinan dari matriks dengan Cramer's rule. Jika determinan koefisien bernilai 0, maka SPL yang diberikan tidak dapat diselesaikan oleh kelas ini karena solusi tidak unik.

Terdapat satu prosedur pada kelas SPLCramer.java

- ❖ Solusi(Matriks matriks) → Untuk menghitung solusi sekaligus menampilkan hasilnya ke console juga file

Adapun fungsi pada Cramer.java adalah

- ❖ Determinan(Matriks matriks, int variable) → Mengembalikan nilai determinan dari suatu matriks, pada prinsipnya dengan cramer's rule, bergantung pada parameter **variable**

- **Invers Matriks**

Diimplementasikan dalam kelas SPLinvers.java. Kelas ini memiliki atribut isInversValid untuk menentukan apakah sebuah matriks memiliki invers atau tidak. Pada kelas ini terdapat fungsi inverseMatriks yang mengembalikan matriks balikan dengan metode matriks augmented dan Gauss-Jordan, yakni dengan menggabungkan dahulu matriks dengan matriks identitasnya. Kemudian digunakan OBE gauss-jordan. Sehingga didapat matriks invers yaitu ambil baris serta kolom dari posisi matriks identitas. Kondisi awal yang harus dipenuhi adalah matriks sudah dicek determinannya sehingga dapat dicari balikannya.

Digunakan fungsi invers yang sudah dijelaskan sebelumnya.

- **Determinan Matriks**

Cara untuk menghitung matriks ini terbagi menjadi 2 yaitu dengan metode mencari kofaktor dan dengan reduksiOBE. Diimplementasikan dalam kofaktor.java dan fungsi Determinan pada kelas SPLMatriks.java. Prekondisi ukuran matriks harus NxN atau matriks persegi. Cara menghitung determinan sudah dijelaskan pada teori singkat.

Adapun fungsi yang digunakan adalah

- ❖ Pada kofaktor.java, fungsi cofactor(Matriks matriks) → mengembalikan nilai determinan dengan metode cofactor

- Pada SPLMatriks.java fungsi Determinan(Matriks matriks) -- .mengembalikan nilai determinan dengan metode reduksi OBE

- **Interpolasi**

  Diimplementasikan dalam kelas interpolasi.java, kelas ini digunakan untuk menghitung permasalahan interpolasi berderajat n. Rumus untuk menghitung terdapat pada teori singkat interpolasi. Adapun fungsi/prosedur yang terdapat dalam interpolasi.java adalah sebagai berikut

  - fungsi MakeInterpolasi(Matriks matriks) → mengembalikan matriks yang sudah dibentuk menjadi augmented menjadi matriks SPL.
  - Fungsi HasilTaksiran(Matriks MatriksInterpolasi, double taksiranX) → Menghasilkan hasil taksiran P(x) dengan metode Cramer
  - Fungsi HasilTaksiranInverse(Matriks matriksInterpolasi, double taksiranX) → Menghasilkan hasil taksiran P(x) dengan metode SPL inverse
  - Fungsi HasilTaksiranGaussJordan(Matriks matriksInterpolasi, double taksiranX) → Menghasilkan hasil taksiran P(x) dengan metode SPL gauss-jordan
  - Fungsi HasilTaksiranGauss(Matriks matriksInterpolasi, double taksiranX) → Menghasilkan hasil taksiran P(x) dengan metode SPL gauss

- **Regresi**

  Diimplementasikan dalam kelas regresi.java dan sigma.java. Kelas ini menentukan hasil *Multiple Linear Regression* dalam bentuk persamaan regresi menurut *Normal Estimation Equation for Multiple Linear Regression*, serta menentukan hasil dari persamaan yang didapat berdasarkan input. Terdapat 4 fungsi/prosedur dalam kelas ini.

  - prosedur ComputeRegresi(Matriks MAugmentedReg). Merangkum seluruh proses penghitungan regresi setelah melakukan input MAugmentedReg.
  - fungsi makeMatriksRegresi(Matriks MAugmentedReg) → mengembalikan MReg yang merepresentasikan persamaan *Normal Estimation Equation for Multiple Linear Regression*. Dalam menghitung koefisien, fungsi ini memanggil fungsi dari kelas sigma.java, yakni sigma1, sigma23, dan sigma4, yang berturut-turut menghitung sigma dari x_i, x_i*x_j, dan x_i*y_i (selengkapnya lihat landasan teori)
  - prosedur PrintMatriksRegresi(Matriks MReg). Menampilkan MReg dalam format persamaan *Normal Estimation Equation for Multiple Linear Regression*.
  - fungsi KoefisienRegresi(Matriks MReg) → mengembalikan matriks B yang elemennya merupakan koefisien regresi solusi dari representasi MReg. Digunakan metode invers dalam menentukan matriks B
  - prosedur PrintPersamaanRegresi(Matriks B). Menampilkan persamaan regresi akhir berdasarkan koefisien regresi yang telah dihasilkan ke dalam format persamaan y = b_0 + b_1*x_1 + … + b_k*x_k
  - fungsi ResultRegresi(Matriks MTest, Matriks B) → mengembalikan nilai y berdasarkan persamaan regresi akhir yang telah ditentukan serta input user MTest yang merupakan matriks yang berisi variabel-variabel independen x_j

- prosedur WriteRegresiToFile(Matriks MReg, Matriks B). Menuliskan hasil seperti yang dihasilkan pada PrintMatriksRegresi dan PrintPersamaanRegresi ke dalam file result.txt

- **Write File**

  Diimplementasikan dalam kelas WriteFile.java. Kelas ini digunakan untuk menyimpan hasil dari proses kedalam file bernama result.txt. Terdapat 4 fungsi/prosedur dalam kelas ini.

  - Prosedur Savefile(input : String) → Untuk menulis string input kedalam file
  - prosedur DelFileExist() → Untuk menghapus jika ada file sebelumnya, sehingga file baru akan berisi hasil dari proses yang baru saja dilakukan, tidak ada spam file.
  - prosedur SaveSuccess() → Berisi *output console* bahwa file berhasil disimpan
  - prosedur addNewLine() → Digunakan untuk menuliskan *newline* pada *file*

BAB IV
Eksperiment

1. Studi Kasus 1

a. Tidak menghasilkan solusi untuk semua metode SPL
b. Hanya dapat digunakan metode gauss dan gauss-Jordan

   Untuk metode Gauss :
   Solusi Parametrik:
   (1.0x1) + (-1.0x2) + (1.0x5) = (3.0)
   (2.0x2) + (-3.0x4) + (-1.0x5) = (3.0)
   (2.5x4) + (-2.5x5) = (-2.5)

   Untuk metode gauss-Jordan didapatkan
   Solusi Parametrik:
   (1.0x1) + (-1.0x5) = (3.0)
   (1.0x2) + (-2.0x5) = (0.0)
   (1.0x4) + (-1.0x5) = (-1.0)

   Keduanya menghasilkan solusi yang sama jika disederhanakan.

c. Hanya dapat menggunakan metode gauss dan gauss-jordan
   Dengan metode gauss-Jordan didapatkan
   Solusi SPL adalah Solusi Parametrik:
   (1.0x2) + (1.0x5) + = (2.0)
   (1.0x4) + (1.0x5) + = (-1.0)
   (-1.0x5) + (1.0x6) = (-1.0)

d. Menghasilkan hasil yang sama untuk setiap metode yakni
   Solusi Tunggal:
   x1 = 18.533978474457854
   x2 = -152.76171089685886
   x3 = 208.4723906841682
   x4 = 511.28057507632184
   x5 = -1253.268164807756
   x6 = 673.1281225157244

2. Studi Kasus 2

a. Hanya dapat diselesaikan oleh metode Gauss dan Gauss Jordan karena solusi parametrik/ tidak unik

Hasil dengan metode Gauss :
Solusi Parametrik:
(1.0x1) + (-1.0x2) + (2.0x3) + (-1.0x4) = (-1.0)
(3.0x2) + (-6.0x3) = (0.0)
Hasil dengan metode Gauss Jordan
Solusi Parametrik:
(1.0x1) + (-1.0x4) = (-1.0)
(1.0x2) + (-2.0x3) = (0.0)
Keduanya menghasilkan hasil yang sama namun pada Gauss belum sederhana

b. Hanya dapat diselesaikan dengan metode Gauss dan Gauss-Jordan
Menghasilkan solusi tunggal yang sama
Solusi Tunggal:
x1 = 0.0 x2 = 2.0 x3 = 1.0 x4 = 1.0

3. Studi Kasus 3

Dengan 4 metode SPL diperoleh hasil yang sama yaitu

x1 = -0.22432432432432436
x2 = 0.18243243243243246
x3 = 0.7094594594594594
x4 = -0.25810810810810797

4. Studi Kasus 4

Dengan metode Gauss, Gauss-Jordan, Balikan didapatkan hasil solusi tunggal seperti berikut
X1 = I12 = 6.9565217391304355
X2 = I52 = -5.217391304347829
X3 = I32 = -1.7391304347826064
X4 = I64 = -6.9565217391304355
X5 = I54 = -1.7391304347826064
X6 = I43 = -1.7391304347826064
X7 = V2 = 165.2173913043478
X8 = V3 = 147.82608695652175
X9 = V4 = 139.13043478260872
X10 = V5 = 113.04347826086959

Namun dengan metode cramer terdapat kesalahan (perbedaan tanda), yang jika lebih jauh lagi diteliti sepertinya ada pada fundamental akses matriks pada salah satu fungsi didalam program yang *salah paham*. Dengan cramer didapat :
X1 = 6.9565217391304355
X2 = -5.217391304347826
X3 = 1.7391304347826086
X4 = -6.9565217391304355

X5 = -1.7391304347826086
X6 = 1.7391304347826086
X7 = 165.21739130434784
X8 = -147.82608695652175
X9 = -139.13043478260872
X10 = -113.04347826086959

5. Studi Kasus 5

a. x = 0.2
   Dengan 4 metode didapatkan hasil sama seperti berikut :
   Hasil taksiran P(x) adalah 0.032960937499987214
   Untuk X = 0.2
   Terdapat sedikit perbedaan pada ketelitian hasil.
   Untuk metode gauss belum berhasil menemukan algoritma yang pas.

b. x = 0.55

   Dengan 3 metode yakni

   Hasil taksiran P(x) adalah 0.1711186523437442 , Untuk X = 0.55

   Namun, terdapat sedikit perbedaan pada ketelitian hasil dikarenakan mungkin ada perbedaan proses menghitung kali,bagi dsb. Untuk metode gauss belum berhasil menemukan algoritma yang pas.

c. x = 0.85

   Dengan 3 metode yakni

   Hasil taksiran P(x) adalah 0.33723583984376776 Untuk X = 0.85

   Namun, terdapat sedikit perbedaan pada ketelitian hasil dikarenakan mungkin ada perbedaan proses menghitung kali,bagi dsb. Untuk metode gauss belum berhasil menemukan algoritma yang pas.

d. x = 1.28

   Dengan 3 metode yakni

   Hasil taksiran P(x) adalah 0.6775418375000526 Untuk X = 1.28

   Namun, terdapat sedikit perbedaan pada ketelitian hasil dikarenakan mungkin ada perbedaan proses menghitung kali,bagi dsb. **Untuk metode gauss belum berhasil menemukan algoritma yang benar untuk angka yang sangat kecil.**

6. Studi Kasus 6

Matriks augmented-nya adalah

4.8 8211

5 10118
5.516 17025
5.710 20796
6.5 39294
7.194 64958
8.097 113134
8.258 123503
9.033 177571
9.333 145510

Dengan metode gauss-Jordan, Cramer dan balikan didapatkan :

a. 25/05/20 = 5 + (25 / 31) = 5.806451612903226

   Cramer : Hasil taksiran P(x) adalah 22613.505717992783

   Gauss-Jordan : Hasil taksiran P(x) adalah 22804.244754195213

   Balikan : Hasil taksiran P(x) adalah 22804.24475902319

b. 30/08/20 = 8 + (30/31) = 8.967741935483872

   Cramer : Hasil taksiran P(x) adalah 175301.88438034058

   Gauss-Jordan : Hasil taksiran P(x) adalah 175759.3491382599

   Balikan : Hasil taksiran P(x) adalah 175759.34913253784

c. 15/09/20 = 9 + (15/30) = 9.5

   Cramer : Hasil taksiran P(x) adalah 67703.74975967407

   Gauss-Jordan : Hasil taksiran P(x) adalah 68216.4264831543

   Balikan : Hasil taksiran P(x) adalah 68216.4264831543

d. 01/7/20 = 7 + (1/31) = 7.032258

   Cramer : Hasil taksiran P(x) adalah 57546.23740243912

   Gauss-Jordan : Hasil taksiran P(x) adalah 57826.97116112709

   Balikan : Hasil taksiran P(x) adalah 57826.97116112709

Ketiganya memberikan hasil yang relatif sama, hanya berbeda dalam ketelitiannya.

Sama seperti sebelumnya, metode gauss belum berhasil untuk digunakan pada testcase ini.

7. Studi Kasus 7

Pilih n = 8 untuk rentang [0,2]
h = ((2-0) / 8) = 0.25
Titik X yang dipilih : {0, 0.25, 0.5 , 0.75, 1, 1.25, 1.5 , 1.75, 2}

sehingga f(x) untuk x = {0, 0.25, 0.5 , 0.75, 1, 1.25, 1.5 , 1.75, 2}
f(x) = { 0.36668 , 0.44543, 0.498265, 0.53788, 0.565473, 0.580896, 0.584358, 0.576651}
Atau dalam bentuk matriks 2 dimensi untuk x dan f(x)
0 0
0.25 0.36668
0.5 0.44543
0.75 0.498265
1 0.53788
1.25 0.565473
1.5 0.580896
1.75 0.584358
2 0.576651

Dengan menggunakan interpolasi berderajat n = 8 didapatkan
$\mathbf{f(x) = ax^0 + bx + cx^2 + dx^3 + ex^4 + fx^5 + gx^6 + hx^7 + ix^8}$
Dengan
a = 0.0
b = 3.214690423809526
c = -11.528000658730184
d = 24.721931244444523
e = -31.920249377777917
f = 25.24653937777792
g = -11.979655111111196
h = 3.1276194539682827
i = -0.3449953523809564
atau dengan metode cramer (dibulatkan):
a = 0.0
b = 3.21
c = -11.53
d = 24.72
e = -31.92
f = 25.25
g = -11.98
h = 3.13
i = -0.34

8. Studi Kasus 8

Digunakan tabel data di bawah ini untuk menentukan persamaan regresi. Sebagaimana ditampilkan pada tabel, variabel independen *humidity*, *temperature*, *pressure*, dan variabel dependen *nitrous oxide* masing-masing berturut-turut dilambangkan dengan x1, x2, x3, dan y.

Table 12.1: Data for Example 12.1

Nitrous Oxide, y	Humidity, x_1	Temp., x_2	Pressure, x_3	Nitrous Oxide, y	Humidity, x_1	Temp., x_2	Pressure, x_3
0.90	72.4	76.3	29.18	1.07	23.2	76.8	29.38
0.91	41.6	70.3	29.35	0.94	47.4	86.6	29.35
0.96	34.3	77.1	29.24	1.10	31.5	76.9	29.63
0.89	35.1	68.0	29.27	1.10	10.6	86.3	29.56
1.00	10.7	79.0	29.78	1.10	11.2	86.0	29.48
1.10	12.9	67.4	29.39	0.91	73.3	76.3	29.40
1.15	8.3	66.8	29.69	0.87	75.4	77.9	29.28
1.03	20.1	76.9	29.48	0.78	96.6	78.7	29.29
0.77	72.2	77.7	29.09	0.82	107.4	86.8	29.03
1.07	24.0	67.7	29.60	0.95	54.9	70.9	29.37

Source: Charles T. Hare, "Light-Duty Diesel Emission Correction Factors for Ambient Conditions," EPA-600/2-77-116. U.S. Environmental Protection Agency.

Dari tabel di atas dihitung *Normal Estimation Equation for Multiple Linear Regression* sebagai berikut:

$20b_0 + 863.1b_1 + 1530.4b_2 + 587.84b_3 = 19.42$

$863.1b_0 + 54876.89b_1 + 67000.09b_2 + 25283.395b_3 = 779.477$

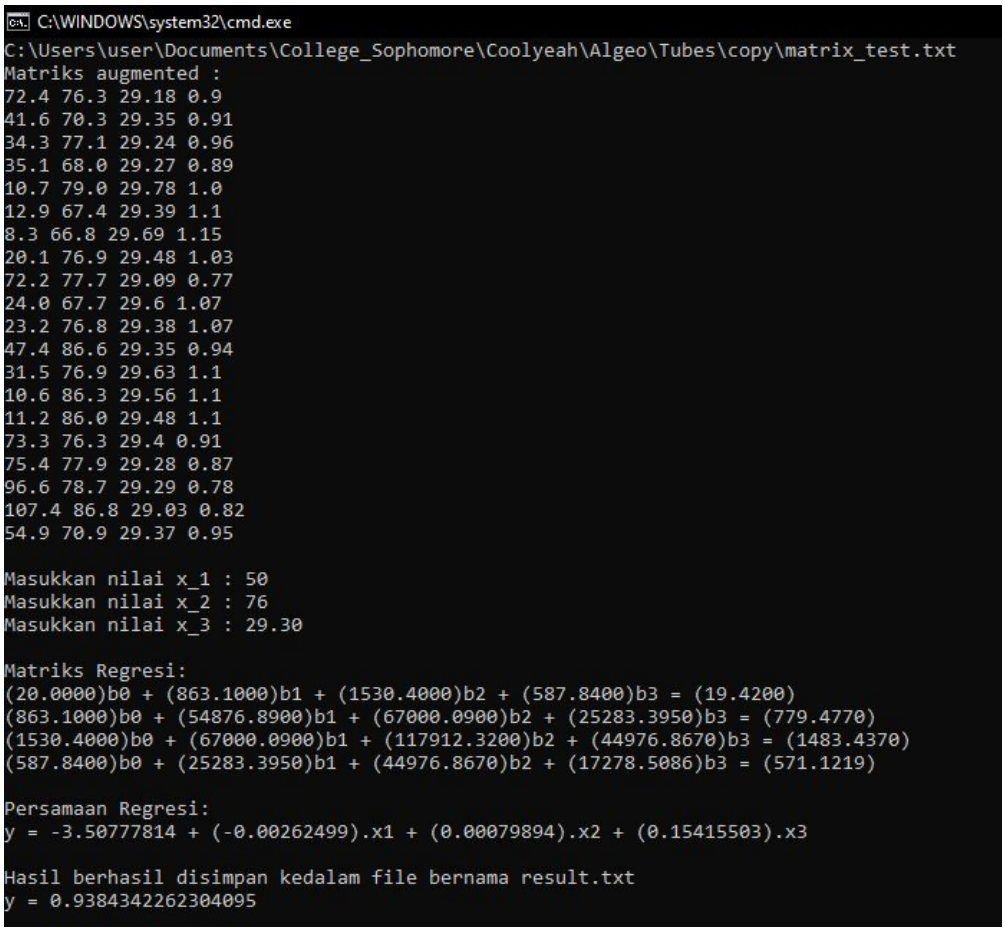
$1530.4b_0 + 67000.09b_1 + 117912.32b_2 + 44976.867b_3 = 1483.437$

$587.84b_0 + 25283.395b_1 + 44976.867b_2 + 17278.5086b_3 = 571.1219$

Dari SPL di atas, didapatkan bahwa hubungan antara variabel independen x1, x2, dan x3 dengan variabel dependen y dapat ditaksir dengan persamaan:
y = -3.50777814 + (-0.00262499)x1 + (-0.00079894)x2 + (0.15415503)x3

Dilakukan input variabel independen x1, x2, dan x3 untuk menentukan variabel dependen y. Ditetapkan x1 = 50, x2 = 76, dan x3 = 29.30, didapatkan y = 0.93843422.

Berikut pensimulasian proses di atas di dalam program:


C:\WINDOWS\system32\cmd.exe
C:\Users\user\Documents\College_Sophomore\Coolyeah\Algeo\Tubes\copy\matrix_test.txt
Matriks augmented :
72.4 76.3 29.18 0.9
41.6 70.3 29.35 0.91
34.3 77.1 29.24 0.96
35.1 68.0 29.27 0.89
10.7 79.0 29.78 1.0
12.9 67.4 29.39 1.1
8.3 66.8 29.69 1.15
20.1 76.9 29.48 1.03
72.2 77.7 29.09 0.77
24.0 67.7 29.6 1.07
23.2 76.8 29.38 1.07
47.4 86.6 29.35 0.94
31.5 76.9 29.63 1.1
10.6 86.3 29.56 1.1
11.2 86.0 29.48 1.1
73.3 76.3 29.4 0.91
75.4 77.9 29.28 0.87
96.6 78.7 29.29 0.78
107.4 86.8 29.03 0.82
54.9 70.9 29.37 0.95
Masukkan nilai x_1 : 50
Masukkan nilai x_2 : 76
Masukkan nilai x_3 : 29.30
Matriks Regresi:
(20.0000)b0 + (863.1000)b1 + (1530.4000)b2 + (587.8400)b3 = (19.4200)
(863.1000)b0 + (54876.8900)b1 + (67000.0900)b2 + (25283.3950)b3 = (779.4770)
(1530.4000)b0 + (67000.0900)b1 + (117912.3200)b2 + (44976.8670)b3 = (1483.4370)
(587.8400)b0 + (25283.3950)b1 + (44976.8670)b2 + (17278.5086)b3 = (571.1219)
Persamaan Regresi:
y = -3.50777814 + (-0.00262499).x1 + (0.00079894).x2 + (0.15415503).x3
Hasil berhasil disimpan kedalam file bernama result.txt
y = 0.9384342262304095

BAB V
KESIMPULAN

Untuk menyelesaikan permasalahan Menghitung solusi SPL dengan metode eliminasi Gauss, metode Eliminasi Gauss-Jordan, metode matriks balikan, dan kaidah Cramer (kaidah Cramer khusus untuk SPL dengan n peubah dan n persamaan) Menyelesaikan persoalan interpolasi dan regresi linier, menghitung matriks balikan, menghitung determinan matriks dengan berbagai metode (reduksi baris dan ekspansi kofaktor) dapat dibuat program dalam bahasa Java dengan mengikuti kaidah/metode yang berlaku. Hanya saja tingkat ketelitian pada setiap perhitungan sedikit berbeda dikarenakan mungkin ada rounding untuk angka-angka yang sangat besar/sangat kecil.

Keseluruhan program berjalan dengan baik, hanya saja terdapat masalah pada metode Gauss untuk elemen-elemen yang nilainya sangat kecil/besar, contoh masukan untuk tipe data *double* tidak dapat menerima 31 digit dibelakang koma. Sehingga dibutuhkan tipe data lain yang ukurannya lebih besar untuk menampung kepresisiannya. Efektivitas pembuatan kelas juga dapat dimaksimalkan lagi, karena beberapa fungsi ada yang dapat disederhanakan contoh pada kelas interpolasi.java, juga dari segi pembuatan fungsi masih bingung dengan *pass by value* di java sehingga nilai *original* dari suatu matriks yang di *pass* ke fungsi berubah-ubah yang seharusnya disimpan. Pada kelas SPLCramer terdapat anomali, yakni ketika mengakses kolom ke-0 maka ketika akses selanjutnya menjadi berubah tanda seperti halnya *overflow*, belum diketahui dimana *bug*nya masih dalam pencarian. Kelas SPLCramer berhasil digunakan untuk interpolasi namun tidak untuk SPL biasa.

Terakhir harapan kami adalah program ini dapat dikembangkan lagi sehingga kemungkinan adanya *bug* semakin kecil atau bahkan sama sekali tidak ada juga sampai dapat digunakan pada *real-world*. Program ini juga menjadi pelajaran kami sebagai *software engineer* di masa depan nanti, bahwa komunikasi dan kerja sama tim adalah yang terpenting sehingga dalam pengerjaan program tidak ada kesalahpahaman arsitektur fungsi/prosedur.