LAPORAN TUGAS BESAR I

“Sistem Persamaan Linier, Determinan, dan Aplikasinya”

Laporan Ini Dibuat Untuk Memenuhi Tugas Perkuliahan

Mata Kuliah Aljabar Linier dan Geometri (IF2123)

KELAS 02

Dosen : Dr. Ir. Rinaldi Munir, MT.

DISUSUN OLEH:

Kelompok 12

Anggota:

Rava Naufal A (13520077)

Jeremy Rionaldo P (13520082)

Yakobus Iryanto P (13520104)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER I TAHUN 2021-2022

Daftar Isi

Daftar Tabel

Daftar Gambar

BAB I

Deskrpisi Masalah

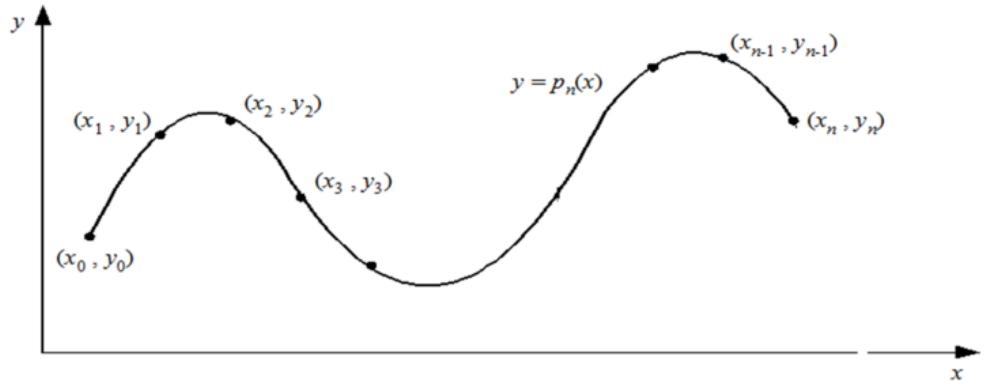
Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. Anda sudah mempelajari berbagai metode untuk menyelesaikan SPL, termasuk menghitung determinan matriks. Sembarang SPL dapat diselesaikan dengan beberapa metode, yaitu metode eliminasi Gauss, metode eliminasi Gauss-Jordan, 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, anda diminta membuat satu atau lebih *library* aljabar linier dalam Bahasa Java. Library 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, gunakan *library* tersebut di dalam program Java untuk menyelesaikan berbagai persoalan yang dimodelkan dalam bentuk SPL, menyelesaikan persoalan interpolasi, dan persoalan regresi. Penjelasan tentang interpolasi dan regresi adalah seperti di bawah ini.

Beberapa tulisan cara membuat library di Java:
1. https://www.programcreek.com/2011/07/build-a-java-library-for-yourself/
2. https://developer.ibm.com/tutorials/j-javalibrary/
3. https://stackoverflow.com/questions/3612567/how-to-create-my-own-java-libraryapi

1.1. 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 $i = 0, 1, 2, \ldots, n$.



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 $pn(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$. 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 $p_n(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ untuk $i = 0, 1, 2, \ldots, n$, akan diperoleh n buah sistem persamaan lanjar dalam $a_0 , a_1 , a_2 , \ldots, a_n$,

$$\begin{aligned}
a_0 + a_1x_0 + a_2x_0^2 + \ldots + a_n x_0^n &= y_0 \\
a_0 + a_1x_1 + a_2x_1^2 + \ldots + a_n x_1^n &= y_1 \\
\ldots \qquad\qquad & \ldots \\
a_0 + a_1x_n + a_2x_n^2 + \ldots + a_n x_n^n &= y_n
\end{aligned}$$

Solusi sistem persamaan lanjar ini, yaitu nilai $a_0 , a_1 , \ldots, a_n$, 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

$$a_0 + 8.0a_1 + 64.00a_2 = 2.0794$$
$$a_0 + 9.0a_1 + 81.00a_2 = 2.1972$$
$$a_0 + 9.5a_1 + 90.25a_2 = 2.2513$$

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

1.2. Regresi Linier 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:

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

Sistem persamaan linier tersebut diselesaikan dengan menggunakan metode eliminasi Gauss.

BAB II

Teori Singkat

2.1. Metode Eliminasi Gauss

Eliminasi Gauss adalah suatu cara mengoperasikan nilai-nilai di dalam matriks menjadi matriks yang lebih sederhana dan banyak digunakan dalam penyelesaian sistem persamaan linier (SPL). Prosedur penyelesaian dari metode ini adalah dengan melakukan operasi baris menjadi matriks eselon-baris. Metode ini mengubah persamaan linear tersebut ke dalam matriks *augmented* dan mengoperasikannya.

SPL dalam bentuk matriks sebagai berikut.

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

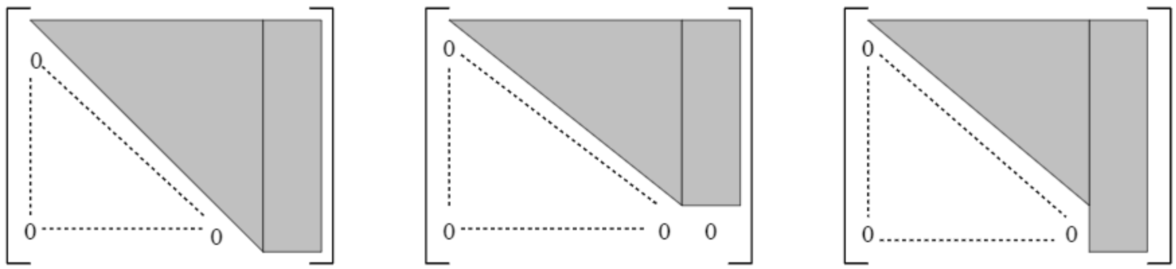
Setelah itu SPL dinyatakan secara ringkas dengan matriks *augmented* sebagai berikut.

$$[A \mid \mathbf{b}] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]$$

Terdapat tiga jenis operasi yang dapat dilakukan dalam metode ini:

1. Mengganti urutan dua baris
2. Mengalikan baris dengan angka yang bukan nol
3. Menambah suatu baris dengan baris yang lainnya

Metode eliminasi Gauss ini dilakukan jika terindikasi matriks *augmented* berakhir pada bentuk matriks eselon-baris. Untuk kemungkinan solusi dari SPL yang ada, dapat diindikasikan jika matriks *augmented* berakhir dengan pola sebagai berikut.



Solusi unik   Solusi banyak   Tidak ada solusi

Selanjutnya, solusi dari SPL didapatkan dengan terlebih dahulu menyelesaikan salah satu persamaan untuk satu variabel dan kemudian mensubstitusi ekspresi ini ke dalam persamaan yang tersisa, proses ini dilanjutkan sampai semua variabel asli telah dievaluasi.

2.2. Metode Eliminasi Gauss-Jordan

Eliminasi Gauss-Jordan adalah pengembangan dari eliminasi Gauss dengan hasil yang lebih sederhana. Caranya adalah dengan meneruskan operasi baris dari eliminasi Gauss yang diterapkan pada matriks augmented sehingga menghasilkan matriks eselon baris tereduksi. Hal ini juga dapat digunakan sebagai salah satu metode penyelesaian persamaan linear dengan menggunakan matriks. Metode ini juga dapat digunakan untuk mencari invers dari sebuah matriks.

Pada metode eliminasi Gauss-Jordan tidak diperlukan lagi substitusi secara mundur untuk memperoleh nilai-nilai variabel. Nilai variabel langsung diperoleh dari matriks augmented akhir, dapat dilihat pola sebagai berikut.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} & b_m \end{bmatrix} \sim \text{OBE} \sim \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & * \\ 0 & 1 & 0 & \dots & 0 & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \vdots & 1 & * \end{bmatrix}$$

Metode eliminasi Gauss-Jordan terdiri dari dua fase:

1. Fase maju (*forward phase*) atau fase eliminasi Gauss

   Fase ini akan menghasilkan nilai-nilai 0 di bawah 1 utama

$$\begin{bmatrix} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{bmatrix} \overset{\text{OBE}}{\sim \dots \sim} \begin{bmatrix} 1 & 3/2 & -1/2 & 5/2 \\ 0 & 1 & 1/2 & 7/2 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

2. Fase mundur (*backward phase*)

   Fase ini akan menghasilkan nilai-nilai 0 diatas 1 utama

$$\begin{bmatrix} 1 & 3/2 & -1/2 & 5/2 \\ 0 & 1 & 1/2 & 7/2 \\ 0 & 0 & 1 & 3 \end{bmatrix} \overset{R1-(3/2)R2}{\sim} \begin{bmatrix} 1 & 0 & -5/4 & -11/4 \\ 0 & 1 & 1/2 & 7/2 \\ 0 & 0 & 1 & 3 \end{bmatrix} \underset{R2-(1/2)R3}{\overset{R1+(5/4)R3}{\sim}} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

Matriks eselon baris tereduksi

2.3. Determinan

Determinan suatu matriks didefinisikan sebagai selisih antara perkalian elemen-elemen pada diagonal utama dengan perkalian elemen-elemen pada diagonal sekunder. Determinan matriks hanya dapat ditentukan pada matriks persegi. Determinan dari matriks A dapat dituliskan det(A) atau |A|.

Untuk menentukan determinan matriks dengan ordo 2x2 kita dapat menggunakan cara sebagai berikut,

$$\det(A) = |A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = a.d - b.c$$

untuk matriks dengan ordo 3x3 determinan dapat ditentukan dengan kaidah Sarrus ataupun ekspansi kofaktor, berikut penyelesaian dengan kaidah Sarruss.

$$|A| = \left|\begin{array}{ccc|cc} a & b & c & a & b \\ d & e & f & d & e \\ g & h & i & g & h \end{array}\right.$$

$$|A| = (a.e.i) + (b.f.g) + (c.d.h) - (c.e.g) - (a.f.h) - (b.d.i)$$

$$|A| = (a.e.i + b.f.g + c.d.h) - (c.e.g + a.f.h + b.d.i)$$

Pada matriks dengan ordo *n* x *n* yang lebih rumit dapat digunakan cara menentukan determinan matriks dengan matriks segitiga, yaitu dengan cara membuat matriks menjadi matriks segitiga atas atau matriks segitiga bawah, kemudian kalikan semua elemen diagonal matriks. Untuk contoh sebagai berikut.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \longrightarrow \det(A) = a_{11}a_{22}\,a_{33}a_{44}$$

$$A = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \longrightarrow \det(A) = a_{11}a_{22}\,a_{33}a_{44}$$

.

Sehingga secara umum, untuk matriks segitika berukuran *n* x *n*, maka det(A) = $a_{11}a_{22}a_{33}..a_{nn}$.

2.4. Matriks Balikan

Matriks balikan adalah kebalikan (*inverse*) dari sebuah matriks yang apabila matriks tersebut dikalikan dengan *invers*-nya, akan menjadi matriks identitas. *Inverse* matriks dilambangkan dengan A^{-1}. Suatu matriks dikatakan memiliki invers jika determinan dari matriks tersebut tidak sama dengan nol. Jika A adalah matriks persegi berukuran *n* x *n*, maka sedemikian sehingga $AA^{-1} = A^{-1}A = I$, dalam hal ini kita dapat memanfaatkan metode eliminasi Gauss-Jordan untuk menghitung matriks balikan.

Untuk matriks A yang berukuran *n* x *n*, A^{-1} dapat dicari dengan cara berikut,

$$[A|I] \overset{\text{G-J}}{\sim} [I|A^{-1}]$$

yang dalam hal ini *I* adalah matriks identitas berukuran *n* x *n*, dan metode eliminasi Gauss-Jordan diterapkan secara simultan untuk *A* maupun *I*.

Jika pada akhir operasi didapatkan matriks sebagai berikut,

$$\left(\begin{array}{ccc|ccc} 1 & 6 & 4 & 1 & 0 & 0 \\ 0 & 1 & 9/8 & 2/8 & -1/8 & 0 \\ 0 & 0 & 0 & -1 & 1 & 1 \end{array}\right)$$

Ada baris bernilai 0

maka dapat disimpulkan bahwa matriks *A* tidak memiliki balikan.

2.5. Matriks Kofaktor

Jika A adalah matriks berukuran *n* x *n*:

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

Didefinisikan:

M_{ij} = minor entri a_{ij}

= determinan upa-matriks (submatrix) yang elemen-elemennya tidak berada pada baris *i* dan kolom *j*

C_{ij} = $(-1)^{i+j}M_{ij}$ = kofaktor entri aij

Kofaktor C*ij* berkoresponden dengan minor entri M_{ij}, hanya berbeda tanda (positif atau negatif, tergantung nilai *i* dan *j*).

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

Maka matriks kofaktor dari A adalah:

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

2.6. Matriks Adjoin

Adjoin dari matriks persegi A berukuran *n* x *n* didefinisikan sebagai *transpose* dari matriks kofaktor A. Adjoin dari matriks A dilambangkan dengan adj(A). Untuk mencari adjoin dari sebuah matriks, cari terlebih dahulu kofaktor dari matriks yang diberikan, kemudian temukan *transpose* dari matriks kofaktor tersebut.

adj(A) = *transpose* dari matriks kofaktor A

2.7. Kaidah Cramer

Jika Ax = b adalah SPL yang terdiri dari n persamaan linier dengan *n* peubah (variabel) sedemikian sehingga det(A) ≠ 0, maka SPL tersebut memiliki solusi yang unik yaitu,

$$x_1 = \frac{\det(A_1)}{\det(A)}, \quad x_2 = \frac{\det(A_2)}{\det(A)}, \quad \ldots, \quad x_n = \frac{\det(A_n)}{\det(A)}$$

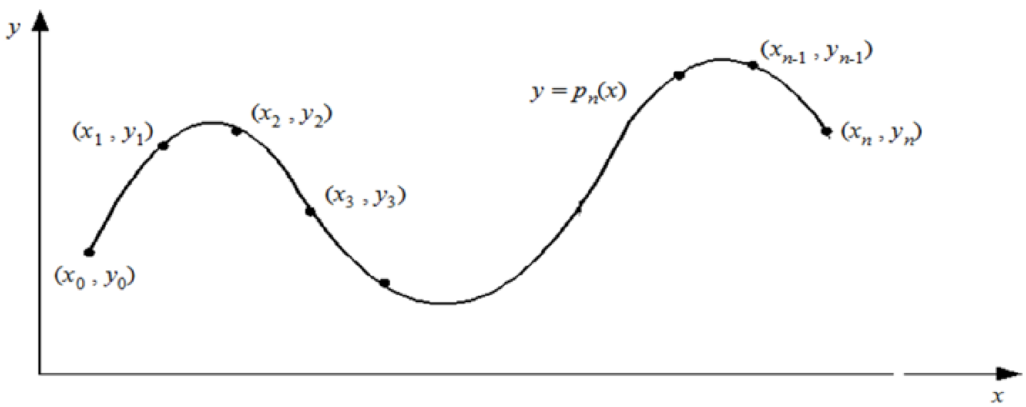
yang dalam hal ini, A*j* adalah matriks yang diperoleh dengan mengganti entri pada kolom ke-*j* dari A dengan entri dari matriks

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

2.8. Interpolasi Polinom

Interpolasi polinom adalah metode memperkirakan nilai antara titik data yang diketahui. Ketika data grafis mengandung celah, tetapi data tersedia di kedua sisi celah atau pada beberapa titik tertentu di dalam celah, perkiraan nilai di dalam celah dapat dibuat dengan interpolasi.

Metode interpolasi yang paling sederhana adalah dengan menggambar garis lurus antara titik-titik data yang diketahui dan mempertimbangkan fungsinya sebagai kombinasi dari garis-garis lurus tersebut. Metode ini, yang disebut interpolasi linier, biasanya menimbulkan kesalahan yang cukup besar. Pendekatan yang lebih tepat menggunakan fungsi polinomial untuk menghubungkan titik-titik. Polinomial adalah ekspresi matematika yang terdiri dari jumlah istilah, setiap istilah termasuk variabel atau variabel yang dipangkatkan dan dikalikan dengan koefisien. Polinomial paling sederhana memiliki satu variabel. Polinomial bisa ada dalam bentuk faktor atau ditulis secara penuh.



Jika sekumpulan data berisi n titik yang diketahui, maka terdapat tepat satu polinomial berderajat n-1 atau lebih kecil yang melalui semua titik tersebut. Sebagai contoh, 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 $p_3(x) = a_0 + a_1x + a_2x^2 + a_3x^3$. Grafik polinomial dapat dianggap sebagai "pengisi kurva" untuk memperhitungkan data antara titik-titik yang diketahui. Metodologi ini, yang dikenal sebagai interpolasi polinomial, seringkali (tetapi tidak selalu) memberikan hasil yang lebih akurat daripada interpolasi linier. Dalam persamaan polinom $p_n(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ untuk $i = 0, 1, 2, \ldots, n$, akan diperoleh n buah sistem persamaan lanjar dalam $a_0, a_1, a_2, \ldots, a_n$.

$$
\begin{aligned}
a_0 + a_1x_0 + a_2x_0^2 + \ldots + a_n x_0^n &= y_0 \\
a_0 + a_1x_1 + a_2x_1^2 + \ldots + a_n x_1^n &= y_1 \\
\ldots \qquad\qquad & \ldots \\
a_0 + a_1x_n + a_2x_n^2 + \ldots + a_n x_n^n &= y_n
\end{aligned}
$$

2.9. Regresi Linier Berganda

Regresi linear 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:

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

Sistem persamaan linier tersebut diselesaikan dengan menggunakan metode eliminasi Gauss.

BAB III

Implementasi Pustaka dan Program Java

Pada tugas besar ini, kami mendeklarasikan 7 buah class, yang terdiri dari,

1. Matrix.java

   Class `Matrix.java` terdapat tiga atribut, yaitu

   - `double[][] M` : sebagai kontainer nilai pada matriks
   - `int row` : sebagai ukuran jumlah baris efektif matriks
   - `int col` : sebagai ukuran jumlah kolom efektif matriks

Tabel 1. Deskripsi Method Pada *Class* Matrix.java

Method	Deskripsi
`Public Matrix(int r, int c)`	Konstuktor pembentuk elemen Matrix
`Public int getRow()`	Fungsi untuk mengembalikan jumlah baris efektif matriks
`Public int getCol()`	Fungsi untuk mengembalikan jumlah kolom efektif matriks
`Public double getELMT(int i, int j)`	Fungsi untuk mengembalikan elemen matriks pada baris dan kolom tertentu
`Public void setELMT(int i, int j, double value)`	Prosedur untuk *assign* nilai elemen matriks pada baris dan kolom tertentu
`Public void readMatrix()`	Membaca input matriks dari user/keyboard
`Public void readMatrixRegresi(int N)`	Prosedur untuk membaca input matriks data titik dari user/keyboard
`Public void displayMatrix()`	Menampilkan matriks ke layar
`Public int Size()`	Mengembalikan ukuran matriks *n* x *m*
`Public void operationRow(int i1, int`	Prosedur untuk melakukan operasi baris

`i2, double k)`	elementer
`Public void divideRow(int i, double k)`	Prosedur untuk membagi elemen pivot pada Matrix dengan koefisien dirinya sendiri jika elemen pivot ≠ 1
`Public void switchRow(int i1, int i2)`	Fungsi untuk melakukan pertukaran baris antara baris i1 dengan baris i2
`Public void copyMatrix(Matrix outM)`	Menduplikasi suatu matriks ke variabel bertipe matriks lainnya
`Public void setIdentity()`	Mengubah matriks menjadi matriks identitas
`Public static void switchRowEmpty(Matrix m)`	Menempatkan suatu baris kosong ke bagian bawah matriks tersebut
`Public static void changeZerovalue(Matrix m)`	Fungsi untuk mengubah nilai -0.0 menjadi 0.0
`Public Matrix transpose(Matrix M)`	Fungsi untuk melakukan transpose matriks
`Public static Matrix getMatKoef(Matrix m)`	Fungsi untuk mengembalikan matriks koefisien
`Public static Matrix createAug(Matrix koef, Matrix cons)`	Fungsi untuk membuat matriks augmented
`Public static Matrix getMatCons(Matrix m)`	Fungsi untuk mengembalikan matriks konstanta
`Public Matrix multiplyMatrix(Matrix m1, Matrix m2)`	Mengalikan dua matriks dengan prekondisi Baris(M1) = Baris(M2)
`Public static boolean isRowEmpty(Matrix M, int row)`	Fungsi untuk mengecek apakah seluruh elemen bernilai nol pada baris tertentu
`Public static boolean isNRowZero(Matrix M, int row)`	Fungsi untuk mengecek apakah elemen pada matriks normal (tidak ada kolom terakhir) bernilai nol pada baris tertentu
`Public static boolean isUnderEmpty(Matrix m, int i, int j)`	Fungsi untuk mengecek apakah elemen-elemen di bawah elemen terkini

	bernilai nol
`Public static boolean isDiagonalOne(Matrix M)`	Fungsi untuk mengecek apakah elemen di indeks baris i dan kolom i adalah 1
`Public static int getLeadingOne(Matrix M, int row)`	Fungsi untuk mencari indeks letak 1 utama sesuai baris yang diminta
`Public String[] StripNonDigits(String x)`	Fungsi untuk memisahkan antara bagian angka dan huruf dalam sebuah variabel parametrik

2. SPL.java

Class `SPL.java` hanya memiliki *method-method* dan tidak memiliki atribut.

Tabel 2. Deskripsi Method Pada *Class* SPL.java

Method	**Deskripsi**
`Public static Matrix inverseSPL(Matrix M)`	Fungsi untuk menghasilkan nilai variabel SPL dengan cara *inverse* dalam bentuk matriks
`Public static Matrix Cramer(Matrix M)`	Fungsi untuk melakukan metode Cramer
`Public static Matrix getRowEchelon(Matrix M)`	Fungsi untuk mengubah matriks *augmented* menjadi matriks eselon baris
`Public static Matrix elimGaussJordan(Matrix m)`	Fungsi untuk mengubah matriks *augmented* menjadi matriks eselon tereduksi
`Public static void GaussElimination(Matrix M)`	Fungsi untuk mengecek status solusi matriks Gauss ataupun Gauss-Jordan
`Public static void getGaussSolutions(Matrix M)`	Fungsi untuk mendapatkan hasil persamaan atau jawaban dari matriks Gauss ataupun Gauss-Jordan

3. Determinan.java

   Class `Determinan.java` hanya memiliki *method-method* dan tidak memiliki atribut.

   Tabel 3. Deskripsi Method Pada *Class* Determinan.java

Method	**Deskripsi**
`Public static double detCofactor(Matrix M)`	Fungsi untuk mendapatkan nilai determinan dengan cara kofaktor
`Public static double detReduksi(Matrix M)`	Fungsi untuk mendapatkan nilai determinan dengan cara baris tereduksi

4. Inverse.java

   Class `Inverse.java` hanya memiliki *method-method* dan tidak memiliki atribut.

   Tabel 4. Deskripsi Method Pada *Class* Inverse.java

Method	**Deskripsi**
`Public static Matrix getCofMatrix(Matrix M, int pivotrow, int pivotcol)`	Fungsi untuk mendapatkan matriks kofaktor dari sebuah matriks
`Public static Matrix getAdjoin(Matrix M)`	Fungsi untuk mendapatkan matriks adjoin
`Public static Matrix adjoinInverse(Matrix M)`	Fungsi untuk mendapatkan matriks *inverse* menggunakan cara adjoin
`Public static Matrix gaussInverse(Matrix M)`	Fungsi untuk mendapatkan matriks *inverse* menggunakan cara Gauss-Jordan

5. InterpolasiRegresi.java

Class `InterpolasiRegresi.java` hanya memiliki *method-method* dan tidak memiliki atribut.

Tabel 5. Deskripsi Method Pada *Class* InterpolasiRegresi.java

Method	Deskripsi
`Public static double interpolasiSPL(Matrix M, double x)`	Fungsi untuk menghitung rumus interpolasi serta mendapatkan nilai taksiran dari titik-titik data yang tersedia
`Public static double regresiGandaSPL(Matrix M, Matrix X)`	Fungsi untuk menghitung rumus regresi linier berganda serta mendapatkan nilai taksiran dari data sampel

6. IOFile.java

Class `IOFile.java` memiliki beberapa atribut, yaitu

- `Scanner File` : sebagai kontainer `file.txt` yang akan dibaca
- `String fileName` : sebagai nama file yang akan dibaca

Tabel 6. Deskripsi Method Pada *Class* IOFile.java

Method	Deskripsi
`Public IOFile(String filename)`	Konstruktor pembentuk variabel namaFile untuk menyimpan nama file **.txt** yang akan dibaca
`Public void openFile()`	Prosedur untuk membuka file **.txt**
`Public void closeFile()`	Prosedur untuk menutup file **.txt**
`Public int readRow()`	Fungsi untuk mendapatkan banyak baris Matrix dalam file **.txt**
`Public int readCol()`	Fungsi untuk mendapatkan banyak kolom Matrix dalam file **.txt**

`Public Matrix readFile()`	Fungsi untuk mendapatkan data Matrix dari suatu file **.txt**
`Public static void displaySave()`	Prosedur untuk menampilkan opsi save ke layar
`Public static void writeMatrix(Matrix M, String filename)`	Prosedur untuk menulis hasil Matrix ke dalam file **.txt**
`Public static void saveFileInverse(Matrix hasil)`	Prosedur untuk menulis hasil Matrix balikan ke dalam file **.txt**
`Public static void saveFileSPL(Matrix hasil)`	Prosedur untuk menulis hasil solusi SPL ke dalam file **.txt**
`Public static void saveFilePolinom(String m1, String m2)`	Prosedur untuk menulis persamaan polinom dan hasil taksiran polinom ke dalam file **.txt**
`Public static void saveFileRegresi(String m1, String m2)`	Prosedur untuk menulis persamaan regresi dan hasil taksiran regresi ke dalam file **.txt**
`Public static void saveFileParametric(int N, String[] Eq)`	Prosedur untuk menulis hasil solusi SPL parametrik ke dalam file **.txt**
`Public static void saveFile(String m)`	Prosedur untuk menulis hasil keluaran bertipe data *String* ke dalam file **.txt**

7. MenuUI.java

Class `MenuUI.java` diperuntukan sebagai *driver* dari seluruh program yang ada, sehingga hanya memiliki *method-method* dan tidak ada atribut yang dideklarasikan.

Tabel 7. Deskripsi Method Pada *Class* MenuUI.java

Method	**Deskripsi**
`Public static void displayTipe()`	Prosedur untuk menampilkan ke layar menanyakan apakah data yang akan dimasukan dari *keyboard* atau file
`Public static void main(String[] args)`	Main program untuk menjalankan seluruh program yang ada

BAB IV

Eksperimen

4.1. Studi Kasus 1

```
============= SPL =============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============= SPL =============
>Masukan: 1

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_1a.txt

SPL tidak memiliki solusi
```

Gambar 1. Studi kasus 1a

```
============= SPL =============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============= SPL =============
>Masukan: 2

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_1b.txt

Solusi SPL:
X1 = 3.0+p
X2 = +2.0p
X3 = q
X4 = -1.0+p
X5 = p
```

Gambar 2. Studi kasus 1b

```
============== SPL ==============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============== SPL ==============
>Masukan: 2

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_1c.txt

Solusi SPL:
X1 = q
X2 = 1.0-p
X3 = r
X4 = -2.0-p
X5 = 1.0+p
X6 = p
```

Gambar 3. Studi kasus 1c. dengan eliminasi Gauss-Jordan

```
============== SPL ==============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============== SPL ==============
>Masukan: 3

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_1c.txt

Metode tidak dapat digunakan karena determinan tidak dapat dikalkulasi
```

Gambar 4. Studi kasus 1c. dengan matriks balikan

```
============= SPL =============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============= SPL =============
>Masukan: 3

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_1d_1.txt

Solusi Inverse:
X1 = 8.629064
X2 = -36.089483
X3 = 62.398896
X4 = -208.428101
X5 = 404.400545
X6 = -235.023758
```

Gambar 5. Studi kasus 1d. Matriks Hilbert n = 6

```
============= SPL =============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============= SPL =============
>Masukan: 1

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_1d_2.txt

Solusi SPL:
X1 = 11.328755866068327
X2 = -58.713689585877724
X3 = 59.81893349428947
X4 = -101.86109531016814
X5 = 376.36789217677847
X6 = -207.57970541926068
X7 = -106.29160370287855
X8 = -183.6671300439334
X9 = 42.283264955248285
X10 = 173.09388079950128
```

Gambar 6. Studi kasus 1d. Matriks Hilbert n = 10

Hasil analisis:

1a. Dengan menggunakan metode eliminasi Gauss, untuk studi kasus 1a SPL tidak memiliki solusi karena setelah dilakukannya OBE terdapat baris dari matriks koefisien yang seluruh elemennya bernilai nol, sedangkan elemen di kolom sebelah kanannya (yang berarti menunjuk pada konstanta) tidak bernilai nol, yang mana hal ini mengindikasikan bahwa SPL tidak memiliki solusi.

1b. Pada studi kasus 1b dengan menerapkan metode eliminasi Gauss-Jordan yang melakukan *forward phase* dan *backward phase*, didapatkan solusi SPL yang berupa parametrik dengan nilai,

$$x_1 = 3 + p$$
$$x_2 = 2p$$
$$x_3 = q$$
$$x_4 = (-1) + p$$
$$x_5 = p$$

dengan *p* dan *q* merupakan elemen bilangan riil.

1c. Pada studi kasus 1c diterapkan dua metode, yaitu eliminasi Gauss-Jordan dan matriks balikan. Dengan menggunakan eliminasi Gauss-Jordan didapatkan solusi parametrik, yaitu

$$x_1 = q$$
$$x_2 = 1 - p$$
$$x_3 = r$$
$$x_4 = (-2) - p$$
$$x_5 = 1 + p$$
$$x_6 = p$$

yang mana disini *p*, *q*, dan *r* merupakan elemen bilangan riil.

Namun, jika diterapkan dengan metode matriks balikan, maka tidak akan mengeluarkan solusi. Hal ini dikarenakan metode matriks balikan menggunakan nilai dari determinan, yang mana pada kasus ini determinan tidak dapat dikalkulasi.

1d. Studi kasus pola Matriks Hilbert dengan $n = 6$ dan $n = 10$ dilakukan dengan dua metode, saat $n = 6$ digunakan Kaidah Cramer, sedangkan saat $n = 10$ dilakukan dengan metode eliminasi Gauss. Dan dari kedua cara tersebut, didapatkan bahwa Matriks Hilbert dengan $n = 6$ dan $n = 10$ memiliki solusi yang unik.

4.2. Studi Kasus 2

```
============== SPL ==============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============== SPL ==============
>Masukan: 2

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_2a.txt

Solusi SPL:
X1 = -1.0+q
X2 = +2.0p
X3 = p
X4 = q
```

Gambar 7. Studi kasus 2a

```
============== SPL ==============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============== SPL ==============
>Masukan: 4

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_2b.txt

Metode tidak dapat digunakan karena determinan tidak dapat dikalkulasi
```

Gambar 8. Studi kasus 2b. dengan menggunakan Kaidah Cramer

```
============= SPL =============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============= SPL =============
>Masukan: 1

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_2b.txt

Solusi SPL:
X1 = 0.0
X2 = 2.0
X3 = 1.0
X4 = 1.0
```

Gambar 9. Studi kasus 2b. dengan menggunakan eliminasi Gauss

Hasil analisis:

2a. Dari data matriks *augmented* yang disajikan, dengan diterapkannya metode eliminasi Gauss-Jordan didapatkan solusi bersifat parametrik, yaitu

$$x_1 = (-1) + q$$
$$x_2 = 2p$$
$$x_3 = p$$
$$x_4 = q$$

yang mana dalam hal ini *p* dan *q* merupakan elemen bilangan riil.

2b. Pada kasus ini dilakukan dua cara, yaitu Kaidah Cramer dan metode eliminasi Gauss. Dengan menggunakan Kaidah Cramer solusi tidak keluar karena metode ini membutuhkan nilai dari determinan, namun dalam kasus ini determinan tidak dapat dikalkulasi. Sedangkan dengan metode eliminasi Gauss didapatkan solusi SPL yang unik, yaitu

$$x_1 = 0.0$$
$$x_2 = 2.0$$
$$x_3 = 1.0$$
$$x_4 = 1.0$$

4.3. Studi Kasus 3

```
============== SPL ==============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============== SPL ==============
>Masukan: 3

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_3a.txt

Solusi Inverse:
X1 = -0.224324
X2 = 0.182432
X3 = 0.709459
X4 = -0.258108
```

Gambar 10. Studi kasus 3a

```
============== SPL ==============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============== SPL ==============
>Masukan: 1

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_3b.txt

SPL tidak memiliki solusi
```

Gambar 11. Studi kasus 3b

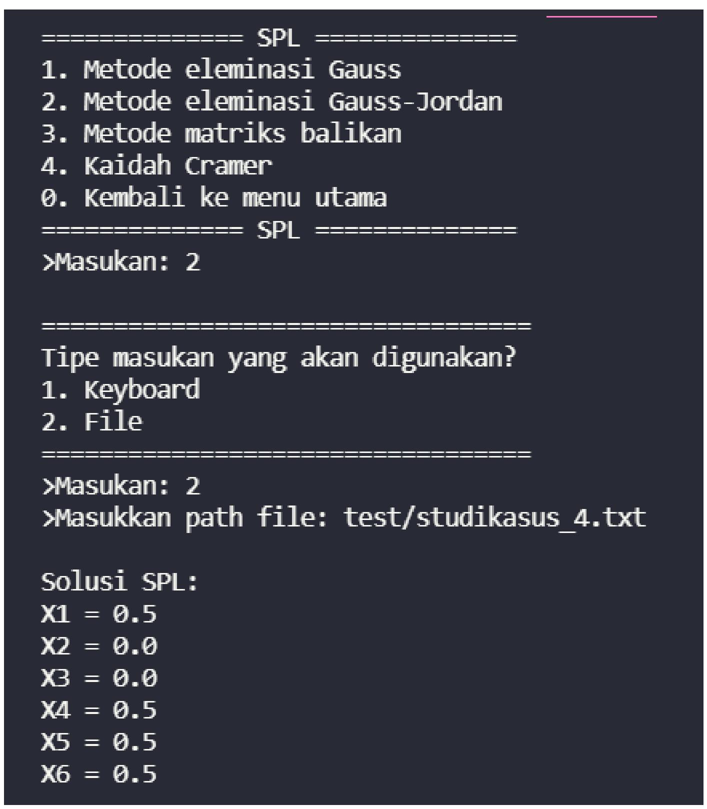
Hasil analisis:

3a. Sama halnya dengan kasus-kasus sebelumnya, kita menggunakan salah satu dari empat metode yang dapat digunakan untuk mendapatkan solusi suatu SPL Pada kasus ini digunakan metode matriks balikan, dan dengan metode tersebut didapatkan solusi unik, yaitu

$$x_1 = -0.224324$$
$$x_2 = 0.182432$$
$$x_3 = 0.709459$$
$$x_4 = -0.258108$$

3b. Pada studi kasus ini diterapkan metode eliminasi Gauss, sama halnya dengan kasus 1a, SPL tidak memiliki solusi dikarenakan alasan yang sama pula. Yaitu terdapat baris dari matriks koefisien yang seluruh elemennya bernilai nol, sedangkan elemen di kolom sebelah kanannya (yang berarti menunjuk pada konstanta) tidak bernilai nol, yang mana hal ini mengindikasikan bahwa SPL tidak memiliki solusi.

4.4. Studi Kasus 4

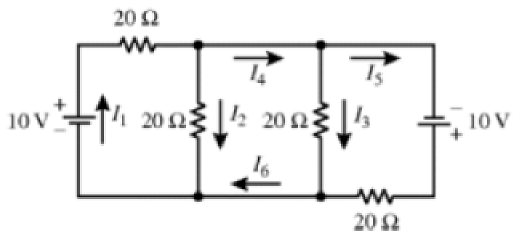


```
============== SPL ==============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============== SPL ==============
>Masukan: 2

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_4.txt

Solusi SPL:
X1 = 0.5
X2 = 0.0
X3 = 0.0
X4 = 0.5
X5 = 0.5
X6 = 0.5
```

Gambar 12. Studi kasus 4

Hasil analisis:

Pada studi kasus ini kita diminta untuk menentukan arus-arus listrik yang mengalir dalam suatu rangkaian listrik, dengan metode eliminasi Gauss-Jordan diperoleh solusi SPL yang unik sebgai berikut.



$i_1 = x_1 = 0.5$

$i_2 = x_2 = 0.0$

$i_3 = x_3 = 0.0$

$i_4 = x_4 = 0.5$

$i_5 = x_5 = 0.5$

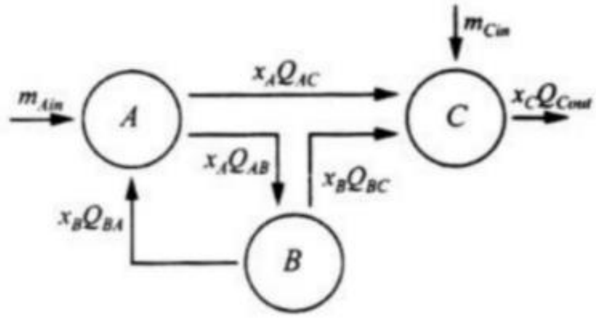
$i_6 = x_6 = 0.5$

4.5. Studi Kasus 5

```
============= SPL =============
1. Metode eleminasi Gauss
2. Metode eleminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
0. Kembali ke menu utama
============= SPL =============
>Masukan: 4

==================================
Tipe masukan yang akan digunakan?
1. Keyboard
2. File
==================================
>Masukan: 2
>Masukkan path file: test/studikasus_5.txt

Solusi Cramer:
X1 = 14.444444444444445
X2 = 7.222222222222222
X3 = 10.0
```

Gambar 13. Studi kasus 5

Hasil analisis:



Disajikan data yang memperlihatkan parameter Q_{AB} = 40, Q_{AC} = 80, Q_{BA} = 60, Q_{BC} = 20 dan Q_{Cout} = 150 m^3/s dan m_{Ain} = 1300 dan m_{Cin} = 200 mg/s. Dengan menggunakan Kaidah Cramer, didapatkan solusi SPL sebagai berikut.

$$x_1 = 14.4444$$
$$x_2 = 7.2222$$
$$x_3 = 10.0$$

4.6. Studi Kasus 6

```
>Masukan: 2
>Masukkan path file: test/studikasus_6a.txt
>Masukkan nilai yang ditaksir: 0.2

Persamaan polinom yang terbentuk:
P6(X) = -0.022976414138603657 + 0.23999728679289456 X^1 + 0.19741187024158435 X^2 + -4.3293148
067391485E-5 X^3 + 0.026100370490780733 X^4 + -3.8740759507049916E-5 X^5 + 9.870444007020751E-
6 X^6

Hasil taksiran polinom:
P6(0.2) = 0.03296092051190471
```

Gambar 14. Studi kasus 6a. dengan nilai yang ditaksir x = 0.2

```
>Masukan: 2
>Masukkan path file: test/studikasus_6a.txt
>Masukkan nilai yang ditaksir: 0.55

Persamaan polinom yang terbentuk:
P6(X) = -0.022976414138603657 + 0.23999728679289456 X^1 + 0.19741187024158435 X^2 + -4.3293148
067391485E-5 X^3 + 0.026100370490780733 X^4 + -3.8740759507049916E-5 X^5 + 9.870444007020751E-
6 X^6

Hasil taksiran polinom:
P6(0.55) = 0.17111865193359974
```

Gambar 15. Studi kasus 6a. dengan nilai yang ditaksir x = 0.55

```
>Masukan: 2
>Masukkan path file: test/studikasus_6a.txt
>Masukkan nilai yang ditaksir: 0.85

Persamaan polinom yang terbentuk:
P6(X) = -0.022976414138603657 + 0.23999728679289456 X^1 + 0.19741187024158435 X^2 + -4.3293148
067391485E-5 X^3 + 0.026100370490780733 X^4 + -3.8740759507049916E-5 X^5 + 9.870444007020751E-
6 X^6

Hasil taksiran polinom:
P6(0.85) = 0.3372358581562203
```

Gambar 16. Studi kasus 6a. dengan nilai yang ditaksir x = 0.85

```
>Masukan: 2
>Masukkan path file: test/studikasus_6a.txt
>Masukkan nilai yang ditaksir: 1.28

Persamaan polinom yang terbentuk:
P6(X) = -0.022976414138603657 + 0.23999728679289456 X^1 + 0.19741187024158435 X^2 + -4.3293148
067391485E-5 X^3 + 0.026100370490780733 X^4 + -3.8740759507049916E-5 X^5 + 9.870444007020751E-
6 X^6

Hasil taksiran polinom:
P6(1.28) = 0.6775418758284182
```

Gambar 17. Studi kasus 6a. dengan nilai yang ditaksir x = 1.28

```
>Masukan: 2
>Masukkan path file: test/studikasus_6b.txt
>Masukkan nilai yang ditaksir: 7.516

Persamaan polinom yang terbentuk:
P9(X) = 7.192310492172672E12 + -9.353084414593117E12 X^1 + 5.337343472679124E12 X^2 + -1.75775
3430333325E12 X^3 + 3.6873270304490826E11 X^4 + -5.115523180809919E10 X^5 + 4.697802959238463E
9 X^6 + -2.755841344620495E8 X^7 + 9376353.988026122 X^8 + -141043.46330044218 X^9

Hasil taksiran polinom:
P9(7.516) = 53551.400390625
```

Gambar 18. Studi kasus 6b. dengan nilai yang ditaksir x = 7.516

```
>Masukan: 2
>Masukkan path file: test/studikasus_6b.txt
>Masukkan nilai yang ditaksir: 8.322

Persamaan polinom yang terbentuk:
P9(X) = 7.192310492172672E12 + -9.353084414593117E12 X^1 + 5.337343472679124E12 X^2 + -1.75775
3430333325E12 X^3 + 3.6873270304490826E11 X^4 + -5.115523180809919E10 X^5 + 4.697802959238463E
9 X^6 + -2.755841344620495E8 X^7 + 9376353.988026122 X^8 + -141043.46330044218 X^9

Hasil taksiran polinom:
P9(8.322) = 36365.23828125
```

Gambar 19. Studi kasus 6b. dengan nilai yang ditaksir x = 8.322

```
>Masukan: 2
>Masukkan path file: test/studikasus_6b.txt
>Masukkan nilai yang ditaksir: 9.166

Persamaan polinom yang terbentuk:
P9(X) = 7.192310492172672E12 + -9.353084414593117E12 X^1 + 5.337343472679124E12 X^2 + -1.75775
3430333325E12 X^3 + 3.6873270304490826E11 X^4 + -5.115523180809919E10 X^5 + 4.697802959238463E
9 X^6 + -2.755841344620495E8 X^7 + 9376353.988026122 X^8 + -141043.46330044218 X^9

Hasil taksiran polinom:
P9(9.166) = -659047.1328125
```

Gambar 20. Studi kasus 6b. dengan nilai yang ditaksir x = 9.166

```
>Masukan: 2
>Masukkan path file: test/studikasus_6b.txt
>Masukkan nilai yang ditaksir: 6.967

Persamaan polinom yang terbentuk:
P9(X) = 7.192310492172672E12 + -9.353084414593117E12 X^1 + 5.337343472679124E12 X^2 + -1.75775
3430333325E12 X^3 + 3.6873270304490826E11 X^4 + -5.115523180809919E10 X^5 + 4.697802959238463E
9 X^6 + -2.755841344620495E8 X^7 + 9376353.988026122 X^8 + -141043.46330044218 X^9

Hasil taksiran polinom:
P9(6.967) = 26647.490234375
```

Gambar 21. Studi kasus 6b. dengan nilai yang ditaksir x = 6.967

```
>Masukan: 2
>Masukkan path file: test/studikasus_6c.txt
>Masukkan nilai yang ditaksir: 1.5

Persamaan polinom yang terbentuk:
P9(X) = 0.17551117894032475 + 1.3112852393030145 X^1 + -3.3918926803761593 X^2 + 6.47949102168
9995 X^3 + -8.217597423479504 X^4 + 6.940259340055604 X^5 + -3.8845286371227985 X^6 + 1.383581
6607927 X^7 + -0.28369918820574824 X^8 + 0.025472352876951376 X^9

Hasil taksiran polinom:
P9(1.5) = 0.5808938057071094
```

Gambar 22. Studi kasus 6c. dengan nilai yang ditaksir x = 1.5

Hasil analisis:

6a. Pada studi kasus nomor 6a dilakukan pengujian interpolasi polinom pada pasangan titik-titik dalam tabel studi kasus tersebut. Rumus interpolasi polinom diperoleh:

$P_6(x) = -0.022976 + 0.239997\ x + 0.197412\ x^2 - 4.329315\ x^3 + 0.0261\ x^4 + (-3.874075 * 10^{-5})\ x^5 + (9.870444 * 10^{-6})\ x^6$

Hasil taksiran pada titik-titik berikut:

$x = 0.2$; $P_6(0.2) = 0.032960$

$x = 0.55$; $P_6(0.55) = 0.171118$

$x = 0.85$; $P_6(0.85) = 0.337235$

$x = 1.28$; $P_6(1.28) = 0.677541$

6b. Pada studi kasus nomor 6b dilakukan pengujian interpolasi polinom pada jumlah data kasus Covid-19. Persamaan polinom yang didapatkan:

$P_9(x) = 7.192310 * 10^{12} + (-9.353084 * 10^{2})\, x + (5.337343 * 10^{12})\, x^2 + (-1.757753 * 10^{12})\, x^3 + (3.687327 * 10^{11})\, x^4 – (5.115523 * 10^{10})\, x^5 + (4.697803 * 10^{9})\, x^6 – (2.755841 * 10^{8})\, x^7 + 9376353.988026\, x^8 + (-141043.463300)\, x^9$

Prediksi jumlah kasus Covid-19 pada tanggal-tanggal berikut menggunakan program interpolasi polinom:

tanggal(desimal) = bulan + (tanggal / jumlah hari pada bulan tersebut)

a. Tanggal 16/07/2021, didapatkan sebanyak 53.551 jumlah kasus baru
b. Tanggal 10/08/2021, didapatkan sebanyak 36.365 jumlah kasus baru
c. Tanggal 05/09/2021, didapatkan sebanyak -659.047 jumlah kasus baru. Jumlah kasus baru ini bernilai negatif karena tanggal desimal yang digunakan berada di luar range data yang diuji menggunakan interpolasi polinom.
d. Masukan user lainnya yaitu tanggal 29/06/2021, didapatkan sebanyak 26.647 jumlah kasus baru.

6c. Pada kasus ini kami mencoba menaksir nilai saat $x = 1.5$, dengan menggunakan program interpolasi polinom yang telah dibuat, didapat hasil taksiran polinom sebagai berikut

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

$$P_9(1.5) = 0.5808938057071094$$

dan prediksi ini sangatlah dekat bahkan hampir sama nilainya jika dibandingkan dengan nilai sebenarnya. Didapatkan hasil dengan menggunakan kalkulator yaitu 0.5808969391.

4.7. Studi Kasus 7

```
>Masukan: 2
>Masukkan path file: test/studikasus_7.txt
>Masukkan nilai X yang akan ditaksir:
>X1: 50
>X2: 76
>X3: 29.30

Persamaan Regresi yang terbentuk:
Y = -3.50775318658998 + -0.0026249968167544324 X1 + 7.989462450378182E-4 X2 + 0.15415417614394
755 X3

Hasil taksiran regresi: 0.938434248212836
```

Gambar 23. Studi kasus 7

Hasil analisis:

Pada studi kasus nomor 7 dilakukan pengujian regresi linier berganda mengenai data pada tabel pada kasus tersebut. Data tabel tersebut diubah menjadi matriks terlebih dahulu menggunakan fungsi *Normal Estimation Equation for Multiple Linear Regression* kemudian dapat diperoleh solusi persamaan regresinya. Persamaan regresi yang diperoleh:

$$y = -3.507753 - 0.00262\ x_1 + (7.989462 * 10^{-4})\ x_2 + 0.154154\ x_3$$

y	: Nitro Oxide
x_1	: Humidity
x_2	: Temperatur
x_3	: Tekanan Udara

Dengan menggunakan program regresi linier berganda yang telah dibuat, hasil taksiran estimasi nilai Nitro Oxide dengan Humidity bernilai 50%, temperatur bernilai 76°F, dan tekanan udara bernilai 29.30 adalah sebesar 0.93843425.

BAB V

Kesimpulan, Saran, dan Refleksi

5.1. Kesimpulan

1. Kami telah berhasil membuat program yang bisa menghitung nilai sebuah Sistem Persamaan Linier (SPL) dengan Metode Eliminasi Gauss, Metode Eliminasi Gauss-Jordan, Metode Matriks Balikan, dan Kaidah Cramer
2. Kami telah berhasil membuat program yang bisa menghitung nilai determinan sebuah matriks persegi dengan Metode Eliminasi Gauss dan Metode Ekspansi Kofaktor
3. Kami telah berhasil membuat program yang bisa membuat dan menghitung matriks balikan (*inverse*) dengan Metode Eliminasi Gauss-Jordan dan Metode Ekspansi Kofaktor dan Adjoin
4. Kami telah berhasil membuat program yang bisa menyelesaikan sebuah interpolasi polinom
5. Kami telah berhasil membuat program yang bisa menyelesaikan sebuah regresi linier berganda

5.2. Saran

Kedepannya program bisa dikembangkan lebih lagi, jika diberikan waktu pengumpulan yang lebih lama. Salah satu contoh perkembangan yang bisa dilakukan adalah dengan membuat sebuah *user interface* dengan GUI, untuk meningkatkan kemudahan penggunaan program. Dengan sebuah GUI, pengguna akan lebih mudah untuk melakukan *input* ke dalam program dan juga lebih mudah untuk membaca hasil perhitungan dari program. Dalam pembuatan matriks eselon dan eselon tereduksi juga bisa diubah dari bentuk desimal ke bentuk pecahan, untuk meningkatkan keakuratan dan terlihat lebih bagus.

5.3. Refleksi

Di awal *development* program, kami menggunakan tipe data *float*. Namun, seiring waktu berjalan, kami menemukan masalah dengan penggunaan tipe *float*, yaitu kurangnya presisi dalam jawaban yang dikeluarkan. Oleh karena itu, kami mengubahnya ke tipe *double*, yang memiliki presisi yang lebih tinggi dibandingkan *float*. Permasalahan *time-management* juga menjadi masalah bagi kami, karena kami harus membagi waktu dengan mata kuliah lain, sehingga kadang Tugas Besar ini terbengkalai. Masalah ketiga adalah penggunaan bahasa pemrograman Java, yang notabene berorientasi objek, sehingga kami harus membiasakan diri, mempelajari cara membuat program dengan orientasi objek, yang memakan waktu cukup lama.

Kesalahan tersebut murni datang dari diri kami sendiri, dan karena itu kami bertekad untuk mengembangkan kemampuan kami untuk kedepannya.

BAB VI

Pembagian Tugas

Tabel 8. Pembagian Tugas

No.	Program	Sub-Program	Penanggung Jawab
1.	Sistem Persamaan Linier	Metode Eliminasi Gauss	Yakobus I
		Metode Eliminasi Gauss-Jordan	Rava N
		Metode Matriks Balikan	Jeremy R
		Kaidah Cramer	Jeremy R
2.	Determinan	Metode Eliminasi Gauss	Jeremy R
		Metode Ekspansi Kofaktor	Jeremy R
3.	Matriks Balikan (*Invers*)	Metode Eliminasi Gauss-Jordan	Rava N
		Metode Ekspansi Kofaktor dan Adjoin	Yakobus I
4.	Interpolasi Polinom	Aplikasi SPL	Jeremy R
5.	Regresi Linier Berganda	Aplikasi SPL	Jeremy R

Referensi

https://stackoverflow.com/

https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/algeo.htm

https://id.wikipedia.org/wiki/Eliminasi_Gauss

https://www.britannica.com/science/Gauss-elimination

https://blog.ub.ac.id/mandegani/2014/06/07/metode-eliminasi-gauss-jordan/

https://akupintar.id/info-pintar/-/blogs/matriks-pengertian-operasi-determinan-invers-dan-contoh-soal

https://whatis.techtarget.com/definition/polynomial-interpolation

https://www.statmat.net/regresi-linier-berganda/