LAPORAN TUGAS BESAR I

IF2123 ALJABAR LINIER DAN GEOMETRI

Sistem Persamaan Linier, Determinan, dan Aplikasinya

Disusun Oleh:

M. Rayhan Farrukh	13523035
Hanif Kalyana Aditya	13523041
Athian Nugraha Muarajuang	13523106

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

BAB I
DESKRIPSI MASALAH

1.1. Latar Belakang Masalah

Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. Kami sudah mempelajari berbagai metode untuk menyelesaikan SPL, termasuk menghitung determinan matriks. Sembarang SPL dapat dengan beberapa metode, yaitu metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan (x = A-1b), 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).

Dalam Tugas Besar 1 ini, kami mengimplementasikan yang telah kami pelajari di Aljabar Linier dan Geometri dengan program berbahasa Java. Kami diminta mengimplementasikan pula SPL pada interpolasi polinomial, regresi linier ganda, dan *Bicubic spline interpolation.*

1.2. Spesifikasi

1.2.1. Program dapat menerima masukan (*input*) baik dari *keyboard* maupun membaca masukan dari *file text*. Untuk SPL, masukan dari *keyboard* adalah *m*, *n*, koefisien a_{ij} , dan b_i. Masukan dari *file* berbentuk matriks *augmented* tanpa tanda kurung, setiap elemen matriks dipisah oleh spasi. Misalnya,

3 4.5 2.8 10 12

-3 7 8.3 11 -4

0.5 -10 -9 12 0

1.2.2. Untuk persoalan menghitung determinan dan matriks balikan, masukan dari *keyboard* adalah *n* dan koefisien a_{ij}. Masukan dari *file* berbentuk matriks, setiap elemen matriks dipisah oleh spasi. Misalnya,

3 4.5 2.8

-3 7 8.3

0.5 -10 -9

Luaran (*output*) disesuaikan dengan persoalan (determinan atau invers) dan penghitungan balikan/invers dilakukan dengan metode matriks balikan dan adjoin.

1.2.3. Untuk persoalan invers, metode yang digunakan ada 2 yaitu menggunakan OBE dan Matriks Adjoin

1.2.4. Untuk persoalan interpolasi, masukannya jika dari *keyboard* adalah n, $(x_0, y_0), (x_1, y_1)$, …, (x_n, y_n), dan nilai x yang akan ditaksir nilai fungsinya. Jika masukannya dari *file*, maka titik-titik dinyatakan pada setiap baris tanpa koma dan tanda kurung. Masukan kemudian dilanjutkan dengan satu buah baris berisi satu buah nilai x yang akan ditaksir menggunakan fungsi interpolasi yang telah didefinisikan. Misalnya jika titik-titik datanya adalah (8.0, 2.0794), (9.0, 2.1972), dan (9.5, 2.2513) dan akan mencari nilai y saat x = 8.3, maka di dalam *file text* ditulis sebagai berikut:

```
8.0 2.0794
9.0 2.1972
9.5 2.2513
8.3
```

1.2.5. Untuk persoalan regresi, masukannya jika dari *keyboard* adalah n (jumlah peubah x), m (jumlah sampel), 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.

1.2.6. Untuk persoalan SPL, luaran 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 $x_1 = t$).

1.2.7. Untuk persoalan polinom interpolasi dan regresi, luarannya adalah persamaan polinom/regresi dan taksiran nilai fungsi pada x yang diberikan. Contoh luaran untuk interpolasi adalah

$$f(x) = -0.0064x^2 + 0.2266x + 0.6762, \quad f(5) = \ldots$$

dan untuk regresi adalah

$$f(x) = -9.5872 + 1.0732x_1, \quad f(x_k) = \ldots$$

untuk kasus regresi kuadratik, variabel boleh menggunakan x_1, x_2, dan lain-lain tetapi perlu dijelaskan variabel tersebut merepresentasikan apa. Contoh

```
x1 = X
.
x3 = X²
.
x5 = XY
[Persamaan dan Solusi]
```

1.2.8. Untuk persoalan *bicubic spline interpolation*, masukan dari *file text* (.txt) yang berisi matriks berukuran 4 x 4 yang berisi konfigurasi nilai fungsi dan turunan berarah disekitarnya, diikuti dengan nilai a dan b untuk mencari nilai $f(a, b)$.

Misalnya jika nilai dari $f(0, 0), f(1, 0), f(0, 1), f(1, 1), f_x(0, 0), f_x(1, 0), f_x(0, 1), f_x(1, 1), f_y(0, 0), f_y(1, 0), f_y(0, 1), f_y(1, 1), f_{xy}(0, 0), f_{xy}(1, 0), f_{xy}(0, 1), f_{xy}(1, 1)$ berturut-turut adalah 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 serta nilai *a* dan *b* yang dicari berturut-turut adalah 0.5 dan 0.5 maka isi *file text* ditulis sebagai berikut:

```
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
0.5 0.5
```

Luaran yang dihasilkan adalah nilai dari $f(0.5, 0.5)$.

1.2.9. Luaran program harus dapat ditampilkan pada layar komputer dan dapat disimpan ke dalam *file*.

1.2.10. Bahasa program yang digunakan adalah Java. Anda bebas untuk menggunakan versi java apapun dengan catatan di atas java versi 8 (8/9/11/15/17/19/20).

1.2.11. 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. Interpolasi Bicubic Spline
6. Regresi linier dan kuadratik berganda
7. Interpolasi Gambar (Bonus)
8. Keluar

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

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

Begitu juga untuk pilihan menu nomor 2, 3, dan 6

BAB II
TEORI SINGKAT

2.1. Determinan Matriks

Matrix persegi merupakan matrix dengan jumlah baris dan kolom yang sama banyaknya, sehingga hanya pada matrix persegi lah determinan dapat dihitung. Namun, hal ini tidak berarti bahwa seluruh matrix persegi pasti memiliki determinan. Dalam menghitung determinan sebuah matrix terdapat dua metode yang dapat digunakan dan antara lain adalah dengan ekspansi kofaktor dan dengan eliminasi gauss.

2.1.1. METODE EKSPANSI KOFAKTOR

Jika A adalah sebuah matriks persegi, maka minor entri dinyatakan oleh dan didefinisikan menjadi determinan submatriks yang tetap setelah baris ke *i* dan kolom ke *j* dicoret menjadi determinan submatriks yang tetap setelah baris ke *i* dan kolom ke *j* dicoret dari A.

Bilangan dinyatakan sebagai dan dinamakan kofaktor entri atau . Berikut contoh dari minor entri dan kofaktor.

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

$$M_{11} = \begin{vmatrix} 5 & 6 \\ 4 & 8 \end{vmatrix} = 16,\ C_{11} = (-1)^{1+1}.M_{11} = 16$$

$$M_{32} = \begin{vmatrix} 3 & -4 \\ 2 & 6 \end{vmatrix} = 26,\ C_{32} = (-1)^{3+2}.M_{32} = -26$$

1.1 Kofaktor dari suatu matriks

Berikut tabel yang merepresentasikan tanda positif dan negatif pada kofaktor.

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

1.2 Tabel tanda positif dan negatif dari kofaktor

Berikut contoh perhitungan determinan matriks dengan menggunakan ekspansi kofaktor.

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

$$\det(A) = 3\begin{vmatrix} 5 & 6 \\ 4 & 8 \end{vmatrix} - 1\begin{vmatrix} 2 & 6 \\ 1 & 8 \end{vmatrix} + (-4)\begin{vmatrix} 2 & 5 \\ 1 & 4 \end{vmatrix}$$

$$\det(A) = 48 - 10 - 12 = 26$$

1.3 Pencarian determinan menggunakan ekspansi kofaktor

2.1.2. METODE REDUKSI BARIS

Determinan juga dapat dicari dengan menggunakan Operasi Baris Elementer untuk mendapatkan matriks segitiga atas maupun bawah. Berikut contoh mencari determinan dengan menggunakan matriks segitiga.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22} & a_{23} & a_{24} & a_{25} \\ 0 & 0 & a_{33} & a_{34} & a_{35} \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & a_{55} \end{bmatrix}, \det(A) = a_{11}a_{22}a_{33}a_{44}a_{55}$$

1.4 Mencari determinan dengan menggunakan matriks segitiga

Poin utama dari metode ini adalah penggunaan OBE dalam membuat sebuah matriks menjadi matriks segitiga atas atau matriks segitiga bawah.

$$[A] \sim [segitiga\ atas\ atau\ bawah]$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1k} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2k} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & a_{k3} & \cdots & a_{kk} \end{bmatrix} \sim \begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & \cdots & a'_{1k} \\ 0 & a'_{22} & a'_{23} & \cdots & a'_{2k} \\ 0 & 0 & a_{33} & \cdots & a'_{3k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a'_{kk} \end{bmatrix}$$

1.5 Pencarian determinan menggunakan matriks segitiga dan operasi baris elementer (Nilai p menyatakan banyaknya pertukaran baris dalam Operasi Baris Elementer)

2.2. **Metode Eliminasi Gauss**

Metode eliminasi Gauss sangat erat kaitannya dengan Matriks Eselon baris yang merupakan matriks dengan 1 utama pada setiap baris, kecuali baris yang seluruhnya adalah 0. Berkit contoh matriks eselon baris.

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

1.6 Contoh Matriks eselon baris (* = nilai sembarang)

Matriks Eselon Baris memiliki sifat-sifat yang antara lain adalah:

1. Jika sebuah baris tidak terdiri dari seluruhnya 0, maka bulangan bukan 0 pertama dalam baris tersebut adalah 1 (merupakan 1 utama)

2. Jika ada baris yang seluruhnya 0, maka baris tersebut harus diposisikan pada bagian bawah matriks

3. Pada dua baris berurutan yang tidak seluruhnya 0, maka 1 utama pada baris yang lebih rendah terletak lebih ke kanan daripada 1 utama pada barus yang lebih tinggi.

Langkah-langkah mencari solusi SPL dengan metode Eliminasi Gauss:

1. Nyatakan Sistem Persamaan Linear dalam bentuk matriks *augmented*
2. Terapkan Operasi Baris Elementer agar terbentuk matriks eselon baris
3. Terapkan *backward substitution* untuk mencari solusi dari tiap variabel.

Berikut contoh penyelesaian Sistem Persamaan Linear dengan menggunakan Metode Eliminasi Gauss

$$\begin{aligned} a + 2b + c + 2d &= 2 \\ 2a - b + c + d &= 0 \\ 3a + 2b - c + d &= 1 \\ a + b + c - d &= 9. \end{aligned}$$

Penyelesaian :

Matriks perluasan dari SPL di atas adalah

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 2 & 2 \\ 2 & -1 & 1 & 1 & 0 \\ 3 & 2 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 9 \end{array}\right].$$

Selanjutnya diselesaikan dengan menggunakan metode eleminasi Gauss.

$$\left[\begin{array}{cccc|c} 1 & 2 & 1 & 2 & 2 \\ 2 & -1 & 1 & 1 & 0 \\ 3 & 2 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 9 \end{array}\right] \xrightarrow{b_2-2b_1, b_3-3b_1, b_4-b_1} \left[\begin{array}{cccc|c} 1 & 2 & 1 & 2 & 2 \\ 0 & -5 & -1 & -3 & -4 \\ 0 & -4 & -4 & -5 & -5 \\ 0 & -1 & 0 & -3 & 7 \end{array}\right]$$

$$\xrightarrow{b_2\times(-1), b_3\times(-1), b_4\times(-1)} \left[\begin{array}{cccc|c} 1 & 2 & 1 & 2 & 2 \\ 0 & 5 & 1 & 3 & 4 \\ 0 & 4 & 4 & 5 & 5 \\ 0 & 1 & 0 & 3 & -7 \end{array}\right]$$

$$\xrightarrow{b_2\leftrightarrow b_4} \left[\begin{array}{cccc|c} 1 & 2 & 1 & 2 & 2 \\ 0 & 1 & 0 & 3 & -7 \\ 0 & 4 & 4 & 5 & 5 \\ 0 & 5 & 1 & 3 & 4 \end{array}\right]$$

$$\xrightarrow{b_4 \leftrightarrow 4b_3} \left[\begin{array}{cccc|c} 1 & 2 & 1 & 2 & 2 \\ 0 & 1 & 0 & 3 & -7 \\ 0 & 0 & 1 & -12 & 39 \\ 0 & 0 & 0 & -41 & 123 \end{array}\right]$$

$$\xrightarrow{b_4 \times \frac{-1}{41}} \left[\begin{array}{cccc|c} 1 & 2 & 1 & 2 & 2 \\ 0 & 1 & 0 & 3 & -7 \\ 0 & 0 & 1 & -12 & 39 \\ 0 & 0 & 0 & 1 & -3 \end{array}\right]$$

Diperoleh sistem persamaan baru

$$\begin{aligned} a + 2b + c + 2d &= 2 \\ b \quad\quad + 3d &= -7 \\ c - 12d &= 39 \\ d &= -3. \end{aligned}$$

Dengan mensubstitusikan nilai $d = -3$ ke persamaan ke-2 dan ke-3,

diperoleh nilai $b = 2$ dan $c = 3$.

Selanjutnya, dengan mensubstitusikan nilai $b = 2, c = 3$ dan $d = -3$ ke persamaan ke-1,

diperoleh nilai $a = 1$.

Jadi penyelesian dari SPL

$$\begin{aligned} a + 2b + c + 2d &= 2 \\ 2a - b + c + d &= 0 \\ 3a + 2b - c + d &= 1 \\ a + b + c - d &= 9. \end{aligned}$$

adalah $a = 1, b = 2, c = 3$ dan $d = -3$.

1.7 Contoh pencarian solusi Sistem Persamaan Linear dengan menggunakan Metode Gauss

2.3. **Metode Eliminasi Gauss-Jordan**

Memiliki prinsip utama dengan metode Gauss, metode Eliminasi Gauss Jordan juga dapat digunakan untuk mencari solusi suatu Sistem Persamaan Linear. Secara umum, kedua metode tersebut menggunakan Operasi Baris Elementer untuk mengubah bentuk matriks menjadi *augmented*. Namun, hal yang membedakan keduanya adalah tidak diterapkannya *backward substitution* pada Metode Eliminasi Gauss-Jordan karena bentuk akhir matriksnya merupakan Matriks Eselon Baris Tereduksi (*reduced row echelon form)*.

Sifat Matriks Eselon Baris Tereduksi mirip dengan sifat dari Matriks Eselon Baris, hal yang membedakan adalah bahwa pada Matriks Eselon Baris Tereduksi setiap kolom yang memiliki 1 utama juga memiliki 0 di kolom lainnya. Berikut contoh dari Matriks Eselon Baris Tereduksi.

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

1.8 Contoh Matriks Eselon Baris Tereduksi (* = nilai sembarang)

Langkah-langkah untuk mencari solusi dari Sistem Persamaan Linear dengan Metode Eliminasi Gauss-Jordan, antara lain adalah:

1. Fase maju atau fase eliminasi Gauss (menghasilkan nilai 0 dibawah 1 utama).
2. Fase mundur atau *backward phase* (menghasilkan 0 diatas 1 utama).

Berikut contoh penyelesaian Sistem Persamaan Linear dengan metode Eliminasi Gauss-Jordan

$$2x + 3y - z = 5$$
$$4x + 4y - 3z = 3$$
$$-2x + 3y - z = 1$$

Dari fase eliminasi Gauss:

$$\begin{bmatrix}1&3/2&-1/2&5/2\\0&1&1/2&7/2\\0&0&1&3\end{bmatrix}$$

Eliminasi fase mundur

$$\begin{bmatrix}1&3/2&-1/2&5/2\\0&1&1/2&7/2\\0&0&1&3\end{bmatrix}\begin{matrix}R_1 - 3/2\,R_2\\ \sim\end{matrix}\begin{bmatrix}1&0&-5/4&-11/4\\0&1&1/2&7/2\\0&0&1&3\end{bmatrix}\begin{matrix}R_1 + 5/4\,R_3\\ \sim\\ R_2 - 1/2\,R_3\end{matrix}$$

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

Diperoleh solusi dari Sistem Persamaan linear adalah sebagai berikut:

$$x = 1$$
$$y = 2$$
$$z = 3$$

1.9 Pencarian solusi Sistem Persamaan Linear dengan Metode Gauss-Jordan

2.4. **Matriks Balikan (*Inverse*)**

Balikan atau invers dari suatu matriks akan menghasilkan matriks identitas jika dilakukan perkalian dengan matriks awal ($A = I$). Umumnya, penggunaan matriks balikan bertujuan untuk mencari solusi dari Sistem Persamaan Linear yang memiliki solusi tunggal

$$Ax = B$$

$$A^{-1}Ax = A^{-1}B$$

$$Ix = A^{-1}B$$

$$x = A^{-1}B$$

(I adalah matriks identitas)

Terdapat dua metode untuk mencari matriks balikan, yaitu metode kofaktor dan metode eliminasi Gauss-Jordan

1. Metode Kofaktor

Pada metode kofaktor determinan harus dicari terlebih dahulu. Apabila suatu matriks tidak memiliki determinan, maka matriks tersebut juga tidak memiliki matriks balikan. Namun, jika matriks memiliki determinan maka dapat dilanjutkan pada langkah selanjutnya yaitu mencari matriks kofaktor yang kemudian akan ditranspose sehingga akan menghasilkan matriks adjoint. Berikut rumus dari metode kofaktor dalam mencari matriks balikan.

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

2. Metode Gauss-Jordan

Matriks balikan pada metode Gauss-Jordan diperoleh dengan mengubah matriks *augmented* $[A|I]$~ *Gauss Jordan* ~ $[I|)]$. Dalam hal ini, I merupakan matriks identitas dengan ukuran yang sama dengan ukuran matriks A.

2.5. **Matriks Kofaktor**

Matriks kofaktor merupakan matriks yang terbentuk oleh kofaktor-kofaktor. Susunan elemen pada matriks ini juga sesuai dengan susunan pada matriksnya. Berikut contoh penulisan matriks kofaktor

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1k} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2k} \\ C_{31} & C_{32} & C_{33} & \cdots & C_{3k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{k1} & C_{k2} & C_{k3} & \cdots & C_{kk} \end{bmatrix}$$

1.10 Contoh matriks kofaktor

2.6. **Matriks Adjoint**

Adjoint dari suatu matriks persegi dapat didefinisikan sebagai transpose dari matriks kofaktornya, dengan penulisan adj(Matriks). Adjoint dari suatu matriks akan digunakan dalam mencari matriks balikan dari suatu matriks (jika menggunakan metode kofaktor)

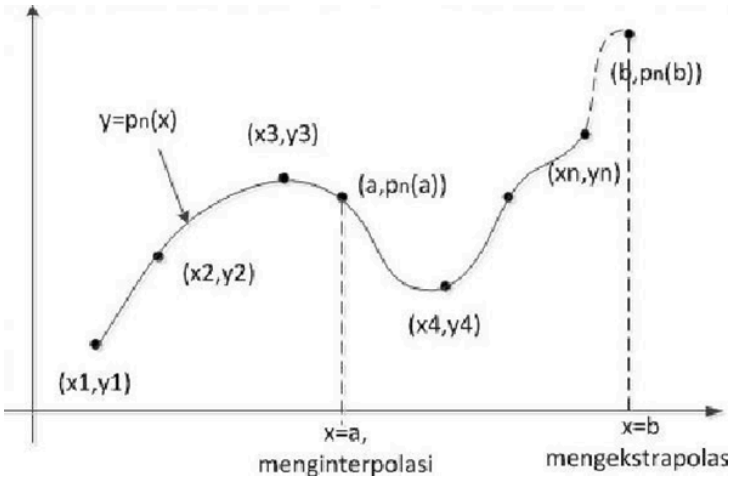
2.7. **Kaidah Cramer**

Kaidah Cramer digunakan untuk mencari solusi dari suatu Sistem Persamaan Linear yang memiliki banyak variabel dan berlaku ketika sistem memiliki solusi tunggal. Pencarian solusi dilakukan dengan menggunakan determinan matriks koefisien (dari sistem persamaan) dan determinan matriks lain yang diperoleh melalui penggantian salah satu kolom matriks koefisien dengan vektor yang terdapat pada sebelah kanan persamaan. Misal Ax = b merupakan SPL yang terdiri dari k persamaan linear dengan k variabel. Jika det(A) ≠ 0, maka setiap variabel akan memiliki solusi unik, yaitu:

$$x_1 = \frac{\det(A_1)}{\det(A)}, x_2 = \frac{\det(A_2)}{\det(A)}, \dots, x_k = \frac{\det(A_k)}{\det(A)}$$

(Matriks didapat dengan mengubah entri pada kolom ke-i dengan matriks b)

2.8. **Interpolasi Polinom**



2.11. Kurva interpolasi polinom

Interpolasi Polinom merupakan suatu metode analisis numerik yang mengaproksimasi suatu nilai fungsi polinom berdasarkan beberapa titik data yang diketahui. Interpolasi Polinom dapat digunakan untuk modelkan bagaimana data berubah-ubah terhadap variabel independennya. Selain membuat prediksi, Interpolasi Polinom juga dapat digunakan untuk

menganalisis hubungan antara variabel-variabel data (berbanding terbalik, tidak berhubungan, berbanding lurus / linear, berbanding kuadratik, dll).

Berikut langkah-langkahnya:

1. Misal diketahui (n+1) buah titik data yang berbeda

$$(x_0, y_0), (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$$

2. Akan terdapat suatu polinom Pn (x) yang memenuhi:

$$y_i = p_n(x_i) \; \forall \; i = 0, 1, 2, \dots, n$$

Polinom tersebut dapat memiliki bentuk:

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

3. Buat Matriks *augmented* dari persamaan yang didapat

$$\begin{bmatrix} 1 & x_0 & {x_0}^2 & \cdots & {x_0}^n & y_0 \\ 1 & x_1 & {x_1}^2 & \cdots & {x_1}^n & y_1 \\ 1 & x_2 & {x_2}^2 & \cdots & {x_2}^n & y_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_n & {x_n}^2 & \cdots & {x_n}^n & y_n \end{bmatrix}$$

4. Lakukan Eliminasi Gauss atau Gauss-Jordan pada matriks *augmented* untuk mendapatkan nilai koefisien dari setiap variabel. contoh menggunakan metode eliminasi Gauss-Jordan:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & k_0 \\ 0 & 1 & 0 & \cdots & 0 & k_1 \\ 0 & 0 & 1 & \cdots & 0 & k_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & k_n \end{bmatrix}$$

Solusi:

$$a_0 = k_0$$
$$a_1 = k_1$$
$$a_2 = k_2$$
$$\vdots$$
$$a_n = k_n$$

2.9. **Regresi Linear Berganda**

Regresi Linear Berganda merupakan metode statistika yang digunakan dalam pemodelan hubungan antara beberapa variabel independen terhadap suatu variabel dependen.

Sesuai dengan namanya, Regresi Linear Berganda hanya dapat digunakan untuk membuat hampiran linear dari beberapa titik data yang berbeda. Tidak seperti Interpolasi Polinom yang dapat digunakan untuk membuat polinomial derajat n. Walau demikian, Regresi Linear Berganda dapat digunakan untuk memodelkan banyak variabel independen secara bersamaan, tidak seperti Interpolasi Polinom yang hanya dapat memodelkan satu variabel independen saja. Namun, keduanya memiliki persamaan dimana Regresi Linear Berganda juga dapat digunakan untuk memperkirakan hubungan antara variabel data independen (hubungan yang dapat ditinjau hanya yang merupakan hubungan berbanding terbalik, berhubungan, atau berbanding lurus / linear).

Berikut langkah-langkah pencarian persamaan Regresi Linear Berganda:

1. Misal terdapat k peubah $(x_1, x_2, x_3, \dots, x_k)$
2. Dibutuhkan k buah titik data yang berbeda untuk melakukan regresi
3. Akan terdapat fungsi $f(x_1, x_2, x_3, \dots, x_k)$ yang memenuhi

$$y_i = f(x_{1i}, x_{2i}, x_{3i}, \dots, x_{ki})$$

Fungsi tersebuat kemudian dapat diasumsikan memiliki bentuk:

$$f(x_1, x_2, x_3, \dots, x_k) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_k x_k + \epsilon_i$$

4. Buat matriks *augmented* dari persamaan tersebut dengan rumus *Normal Estimation Equation for Multiple Linear Regression*

$$\begin{bmatrix} k & \sum_{i=1}^{k} x_{1i} & \sum_{i=1}^{k} x_{2i} & \cdots & \sum_{i=1}^{k} x_{ki} & \sum_{i=1}^{k} y_i \\ \sum_{i=1}^{k} x_{1i} & \sum_{i=1}^{k} x_{1i}^2 & \sum_{i=1}^{k} x_{1i}x_{2i} & \cdots & \sum_{i=1}^{k} x_{1i}x_{ki} & \sum_{i=1}^{k} x_{1i}y_i \\ \sum_{i=1}^{k} x_{2i} & \sum_{i=1}^{k} x_{2i}x_{1i} & \sum_{i=1}^{k} x_{2i}^2 & \cdots & \sum_{i=1}^{k} x_{2i}x_{ki} & \sum_{i=1}^{k} x_{2i}y_i \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum_{i=1}^{k} x_{ki} & \sum_{i=1}^{k} x_{ki}x_{1i} & \sum_{i=1}^{k} x_{ki}x_{2i} & \cdots & \sum_{i=1}^{k} x_{ki}^2 & \sum_{i=1}^{k} x_{ki}y_i \end{bmatrix}$$

5. Lakukan Eliminasi Gauss atau Gauss-Jordan pada matriks *augmented* untuk mendapat nilai koefisien dari setiap variabel

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & n_0 \\ 0 & 1 & 0 & \cdots & 0 & n_1 \\ 0 & 0 & 1 & \cdots & 0 & n_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & n_k \end{bmatrix}$$

Solusi:

$$\beta_0 = n_0$$
$$\beta_1 = n_1$$
$$\beta_2 = n_2$$
$$\vdots$$
$$\beta_k = n_k$$

2.10. **Interpolasi Bikubik**

Interpolasi Bikubik adalah ekstensi dari interpolasi kubik untuk menginterpolasikan poin- poin data dalam kisi-kisi regular. Interpolasi Bikubik juga merupakan Teknik interpolasi pada data 2D umumnya digunakan untuk pembesaran citra.

Model interpolasi bikubik adalah sebagai berikut:

$$f(x, y) = \sum_{j=0}^{3} \sum_{i=0}^{3} a_{ij} x^i y^j$$

Dengan persamaan matriks:

$$\boldsymbol{F = XY \,.\, a}$$

$$\begin{bmatrix} f(-1,-1) \\ f(0,-1) \\ f(1,-1) \\ f(2,-1) \\ f(-1,0) \\ f(0,0) \\ f(1,0) \\ f(2,0) \\ f(-1,1) \\ f(0,1) \\ f(1,1) \\ f(2,1) \\ f(-1,2) \\ f(0,2) \\ f(1,2) \\ f(2,2) \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 2 & 4 & 8 & -1 & -2 & -4 & -8 & 1 & 2 & 4 & 8 & -1 & -2 & -4 & -8 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 4 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 \\ 1 & -1 & 1 & -1 & 2 & -2 & 2 & -2 & 4 & -4 & 4 & -4 & 8 & -8 & 8 & -8 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 8 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 4 & 4 & 4 & 4 & 8 & 8 & 8 & 8 \\ 1 & 2 & 4 & 8 & 2 & 4 & 8 & 16 & 4 & 8 & 16 & 32 & 8 & 16 & 32 & 64 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \\ a_{01} \\ a_{11} \\ a_{21} \\ a_{31} \\ a_{02} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{03} \\ a_{13} \\ a_{23} \\ a_{33} \end{bmatrix}$$

Sehingga akan bisa didapatkan matriks **a** dengan cara mengalikan matriks **F** yang dimiliki dengan invers matriks **XY** (karena matriks **XY** konstan):

$$\boldsymbol{XY}^{-1}.\boldsymbol{F} = \boldsymbol{a}$$

Lalu matriks **a** yang didapat bisa digunakan untuk nilai variable dalam $f(x, y)$ lalu melakukan interpolasi berdasarkan $f(a, b)$ dari matriks 4x4 di dalam rentang indeks 0 sampai 1

BAB III
IMPLEMENTASI

3.1. Package Main

3.1.1. Main.java

1) Method

Nama	Tipe	Parameter	Deskripsi
main	public static void	-	Method utama yang akan dijalankan saat meng-*run* program
inputMain	public static int	-	Menu input utama untuk memilih metode input yang digunakan
inputMatrix	public static Matrix	String tipe	Mengembalikan matrix sesuai metode yang dipilih
outputMain	public static void	-	Menu output utama untuk memilih metode output yang digunakan
border	public static void	-	Menge-*print* border pada terminal untuk syling
clearScreen	public static void	-	Membersihkan terminal dan mengembalikan ke posisi teratas
confirmExit	public static void	-	Agar terminal tidak otomatis ke-*clear*, meminta *user* untuk menekan enter terlebih dahulu

3.1.2. IO.java

1) Method

Nama	Tipe	Parameter	Deskripsi
`keyboardInputMatrix(int row,`	`public static Matrix`		Input matrix melalui keyboard

`int col)`			
`public static String inputFileName(){`			Input nama file .txt
`public static double[] fileInputPoints(String path,int line) {`			Input
`public static void terminalOutputMatrix(Matrix M) {`			
`public static String fileOutputMaster(){`			
`public static void fileOutputSPL(SPL R){`			
`public static void fileOutputDetorBicubic(double val, String tipe){`			
`public static void fileOutputInvers(Matrix invers){`			
`public`			

`static void FileOutputInterpolations(SPL solutions, double Absis, double Ordinat) {`			
`public static void fileOutputRegression`		`SPL persamaanRegresi, int metode, Matrix inputX, int peubah`	

3.2. Package Matrix

Pada *package matrix*, terdapat dua *class* yaitu Matrix.java untuk menampung *constructor* dan *method-method* dasar untuk matrix, dan MatrixAdv.java yang berisikan *method-method advanced* untuk matrix.

3.2.1. Matrix.java

1) Atribut

Nama	Deskripsi
elements	Array of array of double untuk meyimpan elemen-elemen matrix
Row	Integer untuk jumlah baris
Col	Integer untuk jumlah kolom

2) Method

Nama	Tipe	Parameter	Deskripsi
Matrix	public constructor	int nRow, int nCol	Membuat objek matriks dengan jumlah baris dan kolom yang ditentukan.
getRow	Public double	int idx	Mengembalikan baris matriks pada indeks tertentu.
getCol	Public double	int idx	Mengembalikan kolom matriks pada indeks tertentu.
rowCount	Public int	-	Mengembalikan jumlah baris matriks.
colCount	Public int	-	Mengembalikan jumlah kolom matriks.
isSquare	Public boolean	-	Mengecek apakah matriks berbentuk persegi (baris == kolom).
isIdentity	Public boolean	-	Mengecek apakah matriks adalah matriks

			identitas.
setElmt	Public void	int i, int j, double val	Mengatur elemen matriks di posisi tertentu.
getElmt	Public double	int i, int j	Mengambil elemen matriks di posisi tertentu.
copyMatrix	Public Matrix	-	Mengembalikan salinan (copy) dari matriks ini.
swapRows	Public void	int row1, int row2	Menukar dua baris pada matriks.
swapCols	Public void	int col1, int col2	Menukar dua kolom pada matriks.
multiplyRow	Public void	int row, double scalar	Mengalikan sebuah baris matriks dengan skalar tertentu.
addRows	Public void	int row1, int row2, double scalar	Menambahkan baris lain (row2) ke sebuah baris (row1) dengan skalar.
constantMultiply	Public void	double scalar	Mengalikan seluruh elemen matriks dengan suatu skalar.
rowCutter	Public Matrix	int idxRow	Menghapus baris tertentu dan mengembalikan matriks hasil pemotongan.
colCutter	Public Matrix	int idxCol	Menghapus kolom tertentu dan mengembalikan matriks hasil pemotongan.
transposeMatrix	Public Matrix	-	Mengembalikan transpos dari matriks ini.
setIdentityMatrix	Public void	-	Mengubah matriks ini menjadi matriks identitas.
roundToZero	Public void	-	Membulatkan elemen-elemen matriks yang sangat kecil menjadi 0.

hasZeroRow	Public void	-	Mengecek apakah ada baris yang semua elemennya nol.
hasFreeVariables	Public boolean	-	Mengecek apakah matriks memiliki variabel bebas (semua elemen kolom nol).
zeroCheckerRow	public static boolean	Matrix m, int rowIdx, Int manyColumns	Mengecek apakah baris tertentu di matriks memiliki semua elemen nol.
isColumnAllZero	public static boolean	Matrix m, int rowIdx, int colIdx	Mengecek apakah semua elemen di kolom tertentu bernilai nol.
searchNonZeroPivot	public static void	Matrix m, int pivotRow, int pivotCol	Mencari pivot yang tidak nol untuk penukaran baris.
OBE	public static void	Matrix m, int rowOBEIdx, int pivotRow, int pivotCol	Operasi baris elementer (OBE) pada baris tertentu dengan pivot.
OBEReduksi	public static void	Matrix m, int pivotRow, int pivotCol	Operasi baris elementer (OBE) pada semua baris matriks kecuali baris pivot.

3.2.2. MatrixAdv.java

1) Method

Nama	Tipe	Parameter	Deskripsi
multiplyMatrix	public static Matrix	Matrix M1, Matrix M2	Mengalikan dua matriks dan mengembalikan hasilnya.
getMinor	public static Matrix	Matrix M, int row, int col	Mengambil minor matriks, yaitu hasil matriks setelah

			menghapus baris dan kolom tertentu.
getCofactorMatrix	public static Matrix	Matrix M	Menghitung matriks kofaktor dari matriks input.
getAdjoinMatrix	public static Matrix	Matrix M	Mengembalikan matriks adjoin dari matriks input (transpose dari matriks kofaktor).
detByGauss	public static double	Matrix M	Menghitung determinan matriks menggunakan metode eliminasi Gauss.
detByCofactor	public static double	Matrix M	Menghitung determinan matriks menggunakan metode ekspansi kofaktor.
inverseByAdjoin	public static Matrix	Matrix M	Menghitung invers matriks menggunakan metode adjoin dan determinan.
inverseByOBE	public static Matrix	Matrix M	Menghitung invers matriks menggunakan Operasi Baris Elementer (OBE).
getUpperTriangular	public static Matrix	Matrix mProblem	Mengubah matriks menjadi bentuk segitiga atas melalui eliminasi Gauss.
getRREMatrix	public static Matrix	Matrix mProblem	Mengubah matriks menjadi bentuk Row Reduced Echelon (RRE) atau bentuk kanonik dengan eliminasi Gauss.

3.3. Package Functions

3.3.1. SPL.java

1) Atribut

Nama	Deskripsi
solutions	Array of Double untuk menyimpan solusi-solusi yang didapat dari SPL
oneSolution	Boolean untuk SPL yang memiliki solusi unik
infSolution	Boolean untuk SPL yang memiliki banyak solusi
noSolution	Boolean untuk SPL yang tidak ada solusi
paramSolutions	Array of string untuk menyimpan solusi parametrik SPL
variables	Integer untuk jumlah variabel pada SPL

2) Method

Nama	Tipe	Parameter	Deskripsi
SPL	Public Constructor	int varCount	Konstruktor untuk kelas SPL
inverseSPL	public static SPL	Matrix M	Menghasilkan SPL menggunakan metode Invers Matriks
cramerMethodSPL	public static SPL	Matrix M	Menghasilkan SPL dengan metode Cramer
gaussJordanElim	public static SPL	Matrix M	Menghasilkan SPL dari matrix M dengan metode eliminasi gauss-jordan.
gaussElim	public static SPL	Matrix M	Menghasilkan SPL dari matrix M dengan metode eliminasi gauss.

parametricWriter	public static SPL	Matrix M	Menuliskan solusi parametrik dan mengassign ke objek SPL
setOneSolution	public void	-	Set atribut boolean oneSolution objek menjadi true
setNoSolution	public void	-	Set atribut boolean noSolution objek menjadi true
setInfSolution	public void	-	Set atribut boolean infSolution objek menjadi true
setSolution	public void	int Idx, double val	Menaruh nilai *val* ke dalam atribut *solutions* pada index *Idx*
setParamSolutions	public void	int Idx, String val	Assign value ke paramSolutions
getSolutions	public double	int Idx	Mengembalikan suatu nilai pada array solusi
getParamSolutions	public string	int Idx	Mengembalikan suatu nilai pada array solusi parametrik
varCount	public int	-	Mengembalikan jumlah variabel atau panjang array solusi
displaySolutions	public void	-	Menampilkan solusi pada terminal

3.3.2. PolyInterpolation.java

1) Method

Nama	Tipe	Parameter	Deskripsi

KeyboardInputPoints	public static Matrix	-	Menghasilkan matrix n x 2 dari input points dari user melalui keyboard
PointstoMatrix	public static Matrix	Matrix mPoints	Menghasilkan matrix dari titik-titik yang diinput menjadi matrix yang akan diinterpolasi
InterpolationFunction	public static SPL	Matrix otwSolved	Menghasilkan SPL dari matrix yang telah diinterpolasikan
OutputInterpolation	public static void	SPL solusi	Memberi output persamaan polinomial dari solusi SPL yang dimiliki
InterpolationFX	public static void	SPL solusi	Memberi output nilai ordinat polinomial (P(x)) berdasarkan persamaan polinomial dan input absis dari user
FileInputPoints	public static void	Matrix mPoints, double Absis	

3.3.3. Regression.java

1) Method

Nama	Tipe	Parameter	Deskripsi
MultipleLinearReg	public static SPL	Matrix matrixData, int nPeubah, int totalSampel	Menghasilkan SPL hasil dari regresi secara linear yang

			inputnya adalah nilai-nilai *x1i, x2i*, …, *xNi* dan *Yi*; jumlah peubah; jumlah sampel;
MultipleCuadraticReg	public static SPL	Matrix matrixData, int nPeubah, int totalSampel	Menghasilkan SPL hasil dari regresi secara kuadratik yang inputnya adalah nilai-nilai *x1i, x2i*, …, *xNi* dan *Yi*; jumlah peubah; jumlah sampel;
sumValue	public static double	Matrix matrixData, int x1, int x2, int totalSampel	Menjumlahkan hasil kali antara nilai-nilai *xi* sesuai dengan teori regresi
MatrixExtender	public static Matrix	Matrix matrix	Membuat matriks baru dengan kolom paling kiri seluruhnya berisi angka satu dan sisanya adalah matriks data
combination	public static int	int n, int r	Menghitung nilai kombinasi dari nC2 (n adalah jumlah peubah)
factorial	public static int	int angka	Menghitung nilai faktorial untuk method combination

3.3.4. Bicubic.java

1) Method

Nama	Tipe	Parameter	Deskripsi
bicubicInterpolation	public static void	Matrix M, double a, double b	Mengembalikan nilai interpolasi bicubic spline pada suatu titik
getBasisMatrix	public static Matrix	-	Mengembalikan matriks 16x16 sebagai basis perhitungan
getAlphaMatrix	public static Matrix	Matrix X, Matrix y	Mengembalikan matriks koefisien dari matriks basis dan matriks inputan
reshapeTo16x1	public static Matrix	Matrix M	Mengubah matriks input 4x4 menj

BAB IV
EKSPERIMEN

4.1. Tes Kasus Sistem Persamaan Linier

a. Kasus:

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

```
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

SPL tidak memiliki solusi / solusi tidak ditemukan

+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

b. Kasus:

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

```
+=+=+=+=+=+=+=+=+=+=+=

X1 = 3.0 + b
X2 = 2.0b
X3 = a
X4 = -1.0 + b
X5 = b

+=+=+=+=+=+=+=+=+=+=+=
```

c. Kasus:

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

```
+=+=+=+=+=+=+=+=+=+=+=+=+

X1 = a
X2 = 1.0 - c
X3 = b
X4 = -2.0 - c
X5 = 1.0 + c
X6 = c

+=+=+=+=+=+=+=+=+=+=+=+=+
```

d.

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

n = 6

```
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

X1 = 36.00000000097987
X2 = -630.0000000292648
X3 = 3360.000000203484
X4 = -7560.000000539234
X5 = 7560.000000603351
X6 = -2772.000000240222

+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

n = 10

```
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

X1 = 99.9963465634064
X2 = -4949.6826599849455
X3 = 79193.2148318453
X4 = -600538.1369470586
X5 = 2522224.258680422
X6 = -6305485.547186596
X7 = 9608261.783228852
X8 = -8750305.21420386
X9 = 4375119.565502384
X10 = -923630.2349573893

+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

4.2. Studi Kasus SPL Augmented

a. Kasus:

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

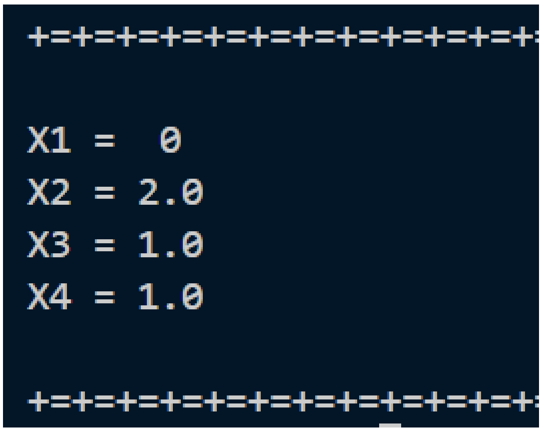
```
+=+=+=+=+=+=+=+=+=+=+=+=+

X1 = -1.0 + b
X2 = 2.0a
X3 = a
X4 = b

+=+=+=+=+=+=+=+=+=+=+=+=+
```

b. Kasus:

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



4.3. Studi Kasus SPL lainnya

a. Kasus:

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

```
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

X1 = -0.2243243243243243
X2 = 0.18243243243243246
X3 = 0.7094594594594594
X4 = -0.25810810810810797

+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

b. Kasus:

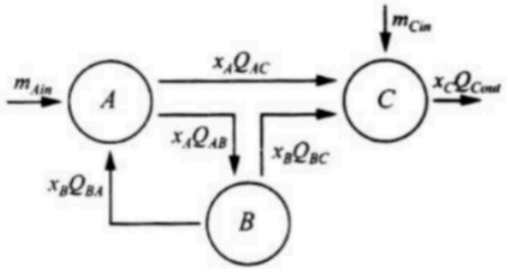
$$
\begin{aligned}
x_7 + x_8 + x_9 &= 13.00 \\
x_4 + x_5 + x_6 &= 15.00 \\
x_1 + x_2 + x_3 &= 8.00 \\
0.04289(x_3 + x_5 + x_7) + 0.75(x_6 + x_8) + 0.61396x_9 &= 14.79 \\
0.91421(x_3 + x_5 + x_7) + 0.25(x_2 + x_4 + x_6 + x_8) &= 14.31 \\
0.04289(x_3 + x_5 + x_7) + 0.75(x_2 + x_4) + 0.61396x_1 &= 3.81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

```
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

SPL tidak memiliki solusi / solusi tidak ditemukan

+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

4.4. Studi Kasus sistem reaktor



Dengan laju volume Q dalam m^3/s dan input massa min dalam *mg/s*. Konservasi massa pada tiap inti reaktor adalah sebagai berikut:

$$\text{A:} \quad m_{A_{in}} + Q_{BA}x_B - Q_{AB}x_A - Q_{AC}x_A = 0$$

$$\text{B:} \quad Q_{AB}x_A - Q_{BA}x_B - Q_{BC}x_B = 0$$

$$\text{C:} \quad m_{C_{in}} + Q_{AC}x_A + Q_{BC}x_B - Q_{C_{out}}x_C = 0$$

Tentukan solusi x_A, x_B, x_C dengan menggunakan parameter berikut : 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*.

```
+=+=+=+=+=+=+=+=+=+=+=+=+
X1 = 14.444
X2 = 7.222
X3 = 10
+=+=+=+=+=+=+=+=+=+=+=+=+
```

4.5. Studi Kasus Interpolasi

a. Gunakan tabel di bawah ini untuk mencari polinom interpolasi dari pasangan titik-titik yang terdapat dalam tabel. Program menerima masukan nilai x yang akan dicari nilai fungsi $f(x)$.

x	0.1	0.3	0.5	0.7	0.9	1.1	1.3
$f(x)$	0.003	0.067	0.148	0.248	0.370	0.518	0.697

Lakukan pengujian pada nilai-nilai berikut:

$x = 0.2$ $f(x) = ?$

$x = 0.55$ $f(x) = ?$

$x = 0.85$ $f(x) = ?$

$x = 1.28$ $f(x) = ?$

```
Masukkan banyak seluruh titik: 7
Masukkan titik x1 dan y1 :
x1: 0,1
y1: 0,003
Masukkan titik x2 dan y2 :
x2: 0,3
y2: 0,067
Masukkan titik x3 dan y3 :
x3: 0,5
y3: 0,148
Masukkan titik x4 dan y4 :
x4: 0,7
y4: 0,248
Masukkan titik x5 dan y5 :
x5: 0,9
y5: 0,370
Masukkan titik x6 dan y6 :
x6: 1,1
y6: 0,518
Masukkan titik x7 dan y7 :
x7: 1,3
y7: 0,697
P(X) = (-5.910407676398067E-15)x^6 + (2.550133859867565E-14)x^5 + (0.026041666666624073)x^4 + (3.441176380306895E-14)x^3 + (0.1973958333333196)x^2 + (0.24000000000000246)x + (-0.02297656250000014)

Masukkan absis titik : 0,2
P(0.2) = 0.032960937500000002
```

```
Masukkan banyak seluruh titik: 7
Masukkan titik x1 dan y1 :
x1: 0,1
y1: 0,003
Masukkan titik x2 dan y2 :
x2: 0,3
y2: 0,067
Masukkan titik x3 dan y3 :
x3: 0,5
y3: 0,148
Masukkan titik x4 dan y4 :
x4: 0,7
y4: 0,248
Masukkan titik x5 dan y5 :
x5: 0,9
y5: 0,370
Masukkan titik x6 dan y6 :
x6: 1,1
y6: 0,518
Masukkan titik x7 dan y7 :
x7: 1,3
y7: 0,697
 + (3.441176380306895E-14)x^3 + (0.1973958333333196)x^2 + (0.24000000000000246)x + (-0.0229
7656250000014)

Masukkan absis titik : 0,55
P(0.55) = 0.17111865234374998
```

b. Jumlah kasus positif baru Covid-19 di Indonesia semakin fluktuatif dari hari ke hari. Di bawah ini diperlihatkan jumlah kasus baru Covid-19 di Indonesia mulai dari tanggal 17 Juni 2022 hingga 31 Agustus 2022:

Tanggal	Tanggal (desimal)	Jumlah Kasus Baru
17/06/2022	6,567	12.624
30/06/2022	7	21.807
08/07/2022	7,258	38.391
14/07/2022	7,451	54.517
17/07/2022	7,548	51.952
26/07/2022	7,839	28.228
05/08/2022	8,161	35.764
15/08/2022	8,484	20.813
22/08/2022	8,709	12.408
31/08/2022	9	10.534

Tanggal (desimal) adalah tanggal yang sudah diolah ke dalam bentuk desimal 3 angka di belakang koma dengan memanfaatkan perhitungan sebagai berikut:

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

Sebagai contoh, untuk tanggal 17/06/2022 (dibaca: 17 Juni 2022) diperoleh tanggal(desimal) sebagai berikut:

Tanggal (desimal) = 6 + (17/30) = 6,567

Gunakanlah data di atas dengan memanfaatkan **interpolasi polinomial** untuk melakukan prediksi jumlah kasus baru Covid-19 pada tanggal-tanggal berikut:

a. 16/07/2022
b. 10/08/2022
c. 05/09/2022
d. Masukan user lainnya berupa **tanggal (desimal) yang sudah diolah** dengan asumsi prediksi selalu dilakukan untuk tahun 2022.

5b a.

```
P(X) = (-140993.71224863594)x^9 + (9372849.23910132)x^8 + (-2.7547453942066944E8)x^7 + (4.695806315428793E9)x^6
 + (-5.113187676013281E10)x^5 + (3.68550807175535E11)x^4 + (-1.7568101863613564E12)x^3 + (5.334203055240578E12)
x^2 + (-9.34699307917343 8E12)x + (7.187066071661201E12)

Masukkan absis titik : 7,516
P(7.516) = 53566.80859375
```

5c

```
Matrix untuk interpolasi polinomial  :
1.0 0.4 0.16000000000000003 0.06400000000000002 0.025600000000000005 0.419
1.0 0.8 0.6400000000000001 0.5120000000000001 0.4096000000000001 0.507
1.0 1.2 1.44 1.7279999999999998 2.0736 0.5609
1.0 1.6 2.5600000000000005 4.096000000000001 6.553600000000001 0.5837
1.0 2.0 4.0 8.0 16.0 0.577

+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
Persamaan polinomial yang diperoleh:
P(X) = (-0.002278645833333521)x^4 + (0.016927083333334352)x^3 + (-0.1380729166666686)x^2 + (0.3689166666666681)x + (0.29249999999999965)

+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
Masukkan absis titik : Masukkan absis titik : 1,15
P(1.15) = 0.555911346435547
```

4.6. Studi Kasus Regresi Berganda Linear dan Kuadratik

Diberikan sekumpulan data sesuai pada tabel berikut ini.

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.

Gunakan Normal Estimation Equation for Multiple Linear Regression untuk mendapatkan regresi linear berganda dari data pada tabel di atas, kemudian estimasi nilai Nitrous Oxide apabila Humidity bernilai 50%, temperatur 76°F, dan tekanan udara sebesar 29.30. Dari data-data tersebut, apabila diterapkan *Normal Estimation Equation for Multiple Linear Regression*, maka diperoleh sistem persamaan linear sebagai berikut.

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

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

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

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

```
50.0
76.0
29.3

20.0 + 841.4999999999999 X1 + 1516.7 X2 + 588.24 X3 = 19.52

841.4999999999999 + 51170.33 X1 + 64872.74 X2 + 24645.010000000002 X3 = 770.1289999999999

1516.7 + 64872.74 X1 + 115653.43 X2 + 44604.37699999999 X3 = 1477.4399999999998

588.24 + 24645.010000000002 X1 + 44604.37699999999 X2 + 17302.418200000004 X3 = 574.5149000000001
```

```
35.1 68.0 29.27 0.89
10.7 79.0 29.78 1.0
12.9 79.7 29.8 1.1
8.3 66.8 29.69 1.15
20.1 76.9 29.82 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 79.6 29.63 1.1
10.6 68.7 29.43 1.1
11.2 80.6 29.48 1.1
73.3 73.6 29.38 0.91
75.4 74.9 29.28 0.87
75.0 78.7 29.09 0.88
107.4 86.8 29.03 0.82
54.9 70.9 29.37 0.95

Y = -2.624981067912226 + -0.0024498159127554413X1 + 5.203826453038095E-4X2 + 0.12459519429262875X3Hasil dari f(X) :
-2.7959198657742377
```

4.7. Studi Kasus Interpolasi *Bicubic Spline*

$$\begin{pmatrix} 21 & 98 & 125 & 153 \\ 51 & 101 & 161 & 59 \\ 0 & 42 & 72 & 210 \\ 16 & 12 & 81 & 96 \end{pmatrix}$$

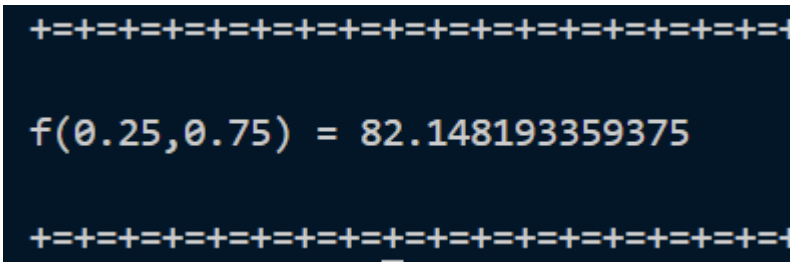
a. f(0,0)

```
+=+=+=+=+=+=+=+=+=+=+=+=
f(0.0,0.0) = 21.0
+=+=+=+=+=+=+=+=+=+=+=+=
```

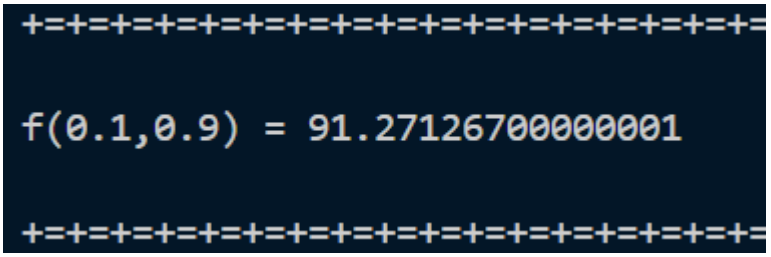
b. f(0.5,0.5)


```
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
f(0.5,0.5) = 87.796875
```

c. f(0.25,0.75)



d. f(0.1,0.9)



BAB V
PENUTUP

5.1. Kesimpulan

Program telah dikembangkan untuk menyelesaikan permasalahan terkait materi yang diajarkan di kelas. Mulai dari sistem persamaan linear, interpolasi polinom, interpolasi bikubik, dan regresi linear dengan bantuan beberapa library yang telah diimplementasikan.

5.2. Saran

1. Program masih dapat dikembangkan dengan berbagai hal seperti menambahkan graphical user interface (GUI).
2. Program dapat ditambah dengan pesan-pesan interaktif terhadap kasus yang tidak memenuhi kriteria atau salah mengenai masukan program.
3. Error yang terjadi seharusnya masih bisa ditanggulangi dengan cara tertentu.

5.3. Refleksi

Pada tugas besar ini kami memperoleh pengalaman berharga yang akan menjadi bekal di masa depan. Terutama bagi kami yang baru pertama kali menggunakan bahasa pemrograman Java dan mengaplikasikan OOP. Meskipun demikian, masih banyak hal yang dapat diperbaiki baik dalam pengerjaan maupun perencanaan dari tugas besar ini. Perencanaan yang matang akan memudahkan pengerjaan tugas besar sehingga bisa lebih baik dan selesai tidak mendekati deadline.

5.4.

LAMPIRAN

Link repository : https://github.com/Starath/Algeo01-23035

https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-02-Matriks-Eselon.pdf (Diakses pada 19 Oktober, 2024)

https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-03-Sistem-Persamaan-Linier.pdf (Diakses pada 19 Oktober, 2024)

https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-05-Sistem-Persamaan-Linier-2.pdf (Diakses pada 19 Oktober, 2024)