

TUGAS BESAR 1

**IF2123 ALJABAR LINIER DAN GEOMETRI SISTEM PERSAMAAN LINIER
DETERMINAN, DAN APLIKASINYA
SEMESTER 1 TAHUN 2022/2023**



Disusun Oleh

Kelompok 29 AverageAlgeoEnjoyers

13521110 Yanuar Sano Nur Rasyid

13521112 Rayhan Hanif Maulana Pradana

13521173 Dewana Gustavus Haraka Otang

**PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK ELEKTRO DAN
INFORMATIKA**

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

BAB I

DESKRIPSI MASALAH

Tugas Besar 1 ini merupakan pengaplikasian konsep Sistem persamaan linear (SPL). Metode yang dipakai untuk menyelesaikan SPL meliputi 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).

Konsep SPL tersebut akan diaplikasikan untuk 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, *library* tersebut akan digunakan di dalam program Java untuk menyelesaikan berbagai persoalan yang dimodelkan dalam bentuk SPL, menyelesaikan persoalan interpolasi, dan persoalan regresi.

Spesifikasi programnya Java yang dibuat meliputi

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
```

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
```

3. 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. Misalnya jika titik-titik datanya adalah $(8.0, 2.0794)$, $(9.0, 2.1972)$, dan $(9.5, 2.2513)$, maka di dalam file text ditulis sebagai berikut:

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

4. Untuk persoalan regresi, masukannya jika dari *keyboard* adalah n (jumlah peubah x), m (jumlah sampel), semua nilai-nilai $x_{1i}, x_{2i}, \dots, x_{ni}$, nilai y_i , dan nilai-nilai x_k yang akan ditaksir nilai fungsinya. Jika masukannya dari file, maka titik-titik dinyatakan pada setiap baris tanpa koma dan tanda kurung.
5. Untuk persoalan SPL, luaran (*output*) program adalah solusi SPL. Jika solusinya tunggal, tuliskan nilainya. Jika solusinya tidak ada, tuliskan solusi tidak ada, jika solusinya banyak, maka tuliskan solusinya dalam bentuk parametrik (misalnya $x_4 = -2$, $x_3 = 2s - t$, $x_2 = s$, dan $x_1 = t$.)
6. Untuk persoalan determinan dan matriks balikan, maka luarannya sesuai dengan persoalan masing-masing
- ```
-3 7 8.3 11 -4
3 4.5 2.8 10 12
0.5 -10 -9 12 0
0.1 0.2
```
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  $f(x) = -0.0064x + 0.2266 + 0.6762 \cdot 2x$   $f(5) = \dots$  dan untuk regresi adalah  $f(x) = -9.5872 + 1.0732x$  ,  $= \dots \cdot 1 f(x_k)$
8. Untuk persoalan interpolasi bicubic, masukan dari file text (.txt) yang berisi matriks berukuran  $4 \times 4$  yang berisi nilai  $f(i,j)$  dengan  $i$  dan  $j$  adalah indeks matriks diikuti dengan nilai  $a$  dan  $b$  untuk mencari nilai  $f(a,b)$ . misalnya jika nilai dari  $f(-1,-1)$ ,  $f(-1,0)$ ,  $f(-1,1)$ ,  $f(-1,2)$ ,  $f(0,-1)$ ,  $f(0,0)$ ,  $f(0,1)$ ,  $f(0,2)$ ,  $f(1,-1)$ ,  $f(1,0)$ ,  $f(1,1)$ ,  $f(1,2)$ ,  $f(2,-1)$ ,  $f(2,0)$ ,  $f(2,1)$ ,  $f(2,2)$  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 $f(0.5,0.5)$. Masukannya adalah matriks 4×4 , diikuti oleh nilai a dan b , maka luarannya adalah nilai $f(a,b)$.
9. Luaran program harus dapat ditampilkan **pada layar komputer dan dapat disimpan ke dalam file**.
10. Bahasa program yang digunakan adalah Java.
11. Program **tidak harus** berbasis GUI, cukup text-based saja, namun boleh menggunakan GUI (memakai kakas *Eclipse* misalnya).

12. Program dapat dibuat dengan pilihan menu. Urutan menu dan isinya dipersilakan dirancang masing-masing. Misalnya, menu:

MENU

1. Sistem Persamaan Linier
2. Determinan
3. Matriks balikan
4. Interpolasi Polinom
5. Interpolasi Bicubic
6. Regresi linier berganda
7. Keluar

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

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

BAB II

TEORI SINGKAT

2.1 Operasi Baris Elementer

Operasi Baris Elementer (OBE) adalah operasi pada matriks yang mengikuti langkah sebagai berikut :

1. Mengalikan sebuah baris dengan konstanta tidak nol.
2. Mempertukarkan dua buah baris.
3. Menjumlahkan sebuah baris dengan kelipatan baris lainnya.

2.2 Sistem Persamaan Linier dan Matriks Augmented

Sebuah Sistem Persamaan Linier (SPL) berbentuk :

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

Atau $\mathbf{Ax} = \mathbf{b}$

Matriks Augmented merupakan matriks dari SPL itu sendiri yang berbentuk :

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

2.3 Matriks Eselon Baris

Matriks Eselon Baris biasa berbentuk :

$$\left[\begin{array}{ccc} 1 & * & * \\ 0 & 1 & * \\ 0 & 0 & 1 \end{array} \right] \quad \left[\begin{array}{cccc} 1 & * & * & * \\ 0 & 0 & 1 & * \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array} \right] \quad \left[\begin{array}{ccccc} 0 & 1 & * & * & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right] \quad \text{dst}$$

Keterangan: * adalah sembarang nilai

Sedangkan matriks Eselon Baris tereduksi berbentuk :

$$\begin{bmatrix} 1 & * & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ atau } \begin{bmatrix} 0 & 1 & 0 & 0 & * \\ 0 & 0 & 1 & 0 & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ dst}$$

2.4 Metode Eliminasi Gauss

Metode Eliminasi Gauss merupakan sebuah algoritma atau langkah khusus untuk mendapatkan sebuah solusi dari SPL dengan menggunakan OBE pada matriks augmented hingga mencapai matriks eselon baris.

$$\left[\begin{array}{ccccc} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} & b_n \end{array} \right] \sim_{\text{OBE}} \left[\begin{array}{cccccc} 1 & * & * & \dots & * & * \\ 0 & 1 & * & \dots & * & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \vdots & 1 & * \end{array} \right]$$

2.5 Metode Eliminasi Gauss-Jordan

Metode Eliminasi Gauss-Jordan merupakan pengembangan dari metode Eliminasi Gauss, yang terdiri dari dua langkah untuk mencapai matriks eselon baris tereduksi, yaitu :

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

$$\left[\begin{array}{cccc} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{array} \right] \sim_{\text{OBE}} \dots \sim \left[\begin{array}{cccc} 1 & 3/2 & -1/2 & 5/2 \\ 0 & 1 & 1/2 & 7/2 \\ 0 & 0 & 1 & 3 \end{array} \right]$$

2. Fase mundur (*backward phase*)

$$\left[\begin{array}{cccc} 1 & 3/2 & -1/2 & 5/2 \\ 0 & 1 & 1/2 & 7/2 \\ 0 & 0 & 1 & 3 \end{array} \right] \sim_{\substack{\text{R1}-(3/2)\text{R2} \\ \text{R2}-(1/2)\text{R3}}} \left[\begin{array}{cccc} 1 & 0 & -5/4 & -11/4 \\ 0 & 1 & 1/2 & 7/2 \\ 0 & 0 & 1 & 3 \end{array} \right] \sim_{\substack{\text{R1}+(5/4)\text{R3} \\ \text{R2}-(1/2)\text{R3}}} \left[\begin{array}{cccc} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{array} \right]$$

↑
Matriks eselon baris tereduksi

2.6 Solusi pada Sistem Persamaan Linier

Sistem persamaan linier memiliki tiga jenis kemungkinan solusi :

1. Solusi unik/tunggal

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 2 & 3 & 1 & 1 \\ 3 & 1 & 2 & 1 \end{array} \right] \xrightarrow{\text{Eliminasi Gauss}} \left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{array} \right]$$

Solusi: $x_1 = 1, x_2 = 0, x_3 = -1$

2. Solusi parametrik/tak berhingga

$$\left[\begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 2 & -1 & 1 & 2 \\ 1 & 2 & 3 & 6 \end{array} \right] \xrightarrow{\text{Eliminasi Gauss}} \left[\begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 \end{array} \right]$$

Misalkan $x_3 = k$,
maka $x_2 = 2 - k$ dan $x_1 = 4 - x_2 - 2x_3 = 4 - (2 - k) - 2k = 2 - k$,
dengan $k \in R$. Terdapat tidak berhingga nilai k .

3. Tidak memiliki solusi

$$\left[\begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 2 & -1 & 1 & 2 \\ 1 & 2 & 3 & 7 \end{array} \right] \xrightarrow{\text{Eliminasi Gauss}} \left[\begin{array}{ccc|c} 1 & 1 & 2 & 4 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{array} \right]$$

2.7 Determinan

Determinan merupakan nilai skalar dari sebuah matriks persegi atau berdimensi $n \times n$.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Untuk matriks A determinan dapat dihitung dengan $\det(A) = bc - cd$.

Selain itu, terdapat cara lain untuk mencari determinan sebuah matriks, seperti

1. Reduksi baris

$$\left[\begin{array}{cccc} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{array} \right] \underset{\text{OBE}}{\sim} \left[\begin{array}{cccc} a'_{11} & a'_{12} & \dots & a'_{1n} \\ 0 & a'_{22} & \dots & a'_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a'_{nn} \end{array} \right]$$

Menggunakan operasi baris elementer, sebuah matriks dapat diubah menjadi matriks segitiga sehingga determinannya adalah $(-1)^p$ dikalikan diagonal utama / k_n dengan p banyak operasi pertukaran baris dan k_n perkalian baris selama operasi berlangsung.

2. Ekspansi Kofaktor

$$A = \begin{bmatrix} 3 & -1 & 2 \\ 5 & 0 & 4 \\ 8 & 2 & -3 \end{bmatrix}$$

Determinan dari matriks A diatas dapat dihitung dengan rumus

$$\det(A) = a_{12}C_{12} + a_{22}C_{22} + a_{23}C_{23}$$

Dengan keterangan a elemen dan C kofaktor di baris dan kolom tersebut.

2.8 Matriks Balikan

Matriks balikan atau matriks inverse adalah suatu matriks yang jika dikalikan dengan matriks asalnya akan membentuk matriks identitas.

$$A \cdot A^{-1} = A^{-1}A = I$$

Dengan A sebuah matriks, A^{-1} inverse matriks A ,dan I adalah matriks identitas.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Determinan dari matriks A adalah $bc - ad$ dan nilainya tidak boleh sama dengan 0 jika matriks tersebut memiliki sebuah balikan.

Ada beberapa cara untuk menghitung matriks balikan

1. Metode Eliminasi Gauss Jordan

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

2. Metode adjoint

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

Penyelesaian sistem persamaan linier dapat diselesaikan dengan memanfaatkan matriks balikan, dan solusi sistem persamaan linier dapat didapat dengan mengali matriks balikan dengan matriks nilai sistem persamaan linier.

- Tinjau SPL $Ax = b$. Kalikan kedua ruas persamaan dengan A^{-1}

$$\begin{aligned} (A^{-1})Ax &= (A^{-1})b \\ Ix &= A^{-1}b \quad (\text{karena } A^{-1}A = I) \\ x &= A^{-1}b \quad (\text{karena } Ix = x) \end{aligned}$$

- Jadi, solusi SPL $Ax = b$ adalah $x = A^{-1}b$

2.9 Matriks Kofaktor

Kofaktor adalah elemen dari sebuah matriks yang didapatkan dengan rumus berikut.

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

Dari rumus di atas, simbol C melambangkan nilai kofaktor, M melambangkan nilai minor matriks, sedangkan i dan j melambangkan nilai baris dan kolom berturut-turut. Nilai minor itu sendiri dapat dicari dengan cara berikut.

$$A = \begin{bmatrix} 6 & 3 & 1 \\ 2 & 2 & -4 \\ 1 & 5 & 3 \end{bmatrix} \quad M_{11} = \begin{vmatrix} 2 & -4 \\ 5 & 3 \end{vmatrix} = (2)(3) - (-4)(5) = 26$$

Matriks kofaktor merupakan matriks yang dibuat dengan nilai kofaktor tiap elemennya.

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

$$\mathbf{C} = \begin{bmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix},$$

2.10 Matriks Adjoin

Matriks adjoin adalah transpose dari matriks kofaktor.

$$\text{adj}(\mathbf{A}) = \mathbf{C}^T = \begin{bmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} \\ +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}.$$

2.11 Kaidah Cramer

Solusi sistem persamaan linier dapat dicari menggunakan kaidah Crammer yaitu dengan cara mengganti tiap baris dengan nilai sistem persamaan linier lalu solusi didapat dari hasil pembagian antara determinan matriks baru dengan determinan matriks awal.

Penyelesaian: $\mathbf{Ax} = \mathbf{b}$

$$\mathbf{A} = \begin{bmatrix} -1 & 2 & -3 \\ 2 & 0 & 1 \\ 3 & -4 & 4 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$$

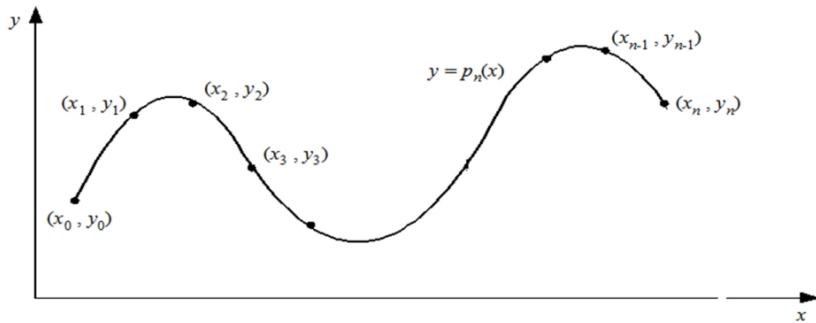
$$\mathbf{A}_1 = \begin{bmatrix} \color{red}{1} & 2 & -3 \\ \color{red}{0} & 0 & 1 \\ \color{red}{2} & -4 & 4 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} -1 & \color{red}{1} & -3 \\ 2 & \color{red}{0} & 1 \\ 3 & \color{red}{2} & 4 \end{bmatrix} \quad \mathbf{A}_3 = \begin{bmatrix} -1 & 2 & \color{red}{1} \\ 2 & 0 & \color{red}{0} \\ 3 & -4 & \color{red}{2} \end{bmatrix}$$

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

2.12 Interpolasi Polinom

Interpolasi merupakan sebuah metode numerik yang digunakan untuk menaksir sebuah titik yang berada pada fungsi tertentu, dengan persoalan :

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



Akan didapat sebuah Sistem Persamaan Linier :

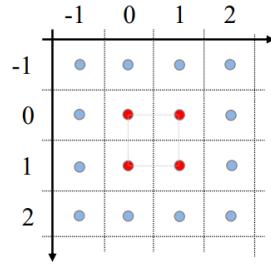
$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \dots + a_n x_0^n &= y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \dots + a_n x_1^n &= y_1 \\ \dots & \dots \\ a_0 + a_1 x_n + a_2 x_n^2 + \dots + a_n x_n^n &= y_n \end{aligned}$$

Dengan menggunakan metode Eliminasi Gauss, Gauss-Jordan, ataupun kaidah Crammers, akan didapat nilai $a_0, a_1, a_2, \dots, a_n$

2.13 Interpolasi Bicubic

Bicubic interpolation merupakan teknik interpolasi pada data 2D umumnya digunakan dalam pembesaran citra yang merupakan pengembangan dari interpolasi linear dan cubic.

Nilai yang digunakan didapat dari nilai koordinat matriks 4×4 yang nantinya akan diselesaikan dalam sistem persamaan linier.



$$y = Xa$$

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

Koefisien sistem persamaan linier didapat dengan cara :

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

$$x, y = -1, 0, 1, 2$$

Dengan mencari solusi untuk vektor a kita dapat meng-interpolasi nilai dengan persamaan yang kita dapat dengan cara :

Interpolate pixel values.

$$f(x, y) = \sum_{j=0}^3 \sum_{i=0}^3 a_{ij} x^i y^j \quad 0 < x < 1, 0 < y < 1$$

2.14 Regresi Linier Berganda

Regresi linier berganda atau biasa disebut juga multiple regression merupakan metode untuk memprediksi nilai suatu variabel dependen dari beberapa variabel independen.

$$Y = a + b_1X_1 + b_2X_2 + \dots + b_nX_n$$

Pada rumus di atas, Y menggambarkan variabel tak bebas sedangkan $X_1 \dots X_n$ melambangkan variabel bebas sebanyak n sedangkan b melambangkan koefisien setiap variabel independen dan a sebagai perpotongan Y saat semua $X = 0$.

Untuk mencari nilai b, dapat digunakan metode *Normal Estimation Equation for Multiple Linear Regression* dengan menggambarkan persamaan double linier regresi menjadi persamaan dalam matriks.

$$\begin{aligned} a\sum X_1 &+ b_1\sum X_1^2 + b_2\sum X_2^2 = \sum Y \\ a\sum X_1 + b_1\sum X_1^2 &+ b_2\sum X_1 X_2 = \sum X_1 Y \\ a\sum X_2 + b_1\sum X_2 X_1 &+ b_2\sum X_2^2 = \sum X_2 Y \end{aligned}$$

Sistem persamaan linier tersebut dapat diselesaikan dengan metode eliminasi gauss.

BAB III

IMPLEMENTASI

3.1 Operasi Primitif

3.1.1 Prosedur copyMatrix

```
public static void copyMatrix(double[][] matrix, double[][] copy){  
    int n = matrix.length;  
    int m = matrix[0].length;  
    copy = new double[n][m];  
    int i, j;  
  
    for(i = 0; i < n; i++){  
        for(j = 0; j < m; j++){  
            copy[i][j] = matrix[i][j];  
        }  
    }  
}
```

Membuat deep copy dari matrix

Initial State copy terdefinisi dan nilainya akan disamakan dengan matrix

Final State seluruh nilai copy sama dengan nilai matrix

3.1.2 Fungsi Perkalian Matrix

```
public static double[][] PerkalianMatrix(double[][] matrix1, double[][] matrix2){  
    int n = matrix1.length;  
    int m = matrix2[0].length;  
    int o = matrix1[0].length;  
    double[][] matrix3 = new double[n][m];  
    for(int i=0;i<n;i++){  
        for(int j=0;j<m;j++){  
            for(int k=0;k<o;k++){  
                matrix3[i][j] += matrix1[i][k] * matrix2[k][j];  
            }  
        }  
    }  
    return matrix3;  
}
```

Melakukan operasi perkalian matrix

3.1.3 Fungsi Perkalian Matrix Konstanta

```

public static double[][] PerkalianMatrixKonstanta(double[][] matrix, double konstanta){
    int n = matrix.length;
    int m = matrix[0].length;
    double[][] matrix2 = new double[n][m];
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            matrix2[i][j] = matrix[i][j] * konstanta;
        }
    }
    return matrix2;
}

```

Melakukan operasi perkalian matrix dengan konstanta

3.1.4 Fungsi SolusiSPL

```

public static double[] SolusiSPL(double[][] matrix, double[] nilai){
    int dimensi = matrix.length, col = matrix[0].length;
    double[] solusi = new double[col];
    for(int i=0;i<dimensi;i++){
        for(int j=0;j<dimensi;j++){
            solusi[i] += matrix[i][j] * nilai[j];
        }
    }
    return solusi;
}

```

Melakukan operasi perkalian matrix dengan vector untuk mendapat solusi sistem persamaan linier

3.1.5 Fungsi Transpose

```

public static double[][] transpose(double[][] m) {
    // transpose matrix
    int row = m.length;
    int col = m[0].length;
    double[][] trans = new double[col][row];

    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            trans[j][i] = m[i][j];
        }
    }
    return trans;
}

```

Menerima input sebuah matriks dan mengembalikan output matriks tersebut yang sudah di-transpose.

3.1.6 Prosedur getCofactor

```

public static void getCoFactor(double[][] m, double[][] temp, int a, int b, int n) {
    // Mendapatkan kofaktor dari m dan menyimpannya di temp

    int row , col;
    row = col = 0;

    for (int i = 0 ; i < n ; i++){
        for (int j = 0 ; j < n ; j++){
            if (i != a && j != b){
                temp[row][col++] = m[i][j];
                if (col == n - 1){
                    col = 0;
                    row++;
                }
            }
        }
    }
}

```

Initial State matriks m dan temp terdefinisi. a,b,n terdefinisi

Final State matriks temp menjadi kofaktor matriks m

3.1.7 Prosedur gauss

```

public static void gauss(double[][] matrix){
    int i, j, k;
    int m = matrix.length, n = matrix[0].length;
    double ratio = 0d;
    int zero_row1 = 0, zero_row2 = 0;
    // Forward Elimination
    for(i = 0; i < m-1; i++){
        pivot(matrix);
        leadingOne(matrix, i);
        for(j = 0; j < n; j++){
            zero_row1 = j;
            if(matrix[i][j] != 0){
                break;
            }
            if(j == n-1 && matrix[i][n-1] == 0){
                zero_row1++;
            }
        }
        for(k = i+1; k < m; k++){
            for(j = 0; j < n; j++){
                zero_row2 = j;
                if(matrix[k][j] != 0){
                    break;
                }
            }
            if(zero_row1 == zero_row2){
                ratio = matrix[k][zero_row2] / matrix[i][zero_row1];
                for(j = 0; j < n; j++){
                    matrix[k][j] -= ratio * matrix[i][j];
                }
            }
        }
        leadingOne(matrix, i);
    }
}

```

Menerima input berupa matriks dan output matriks yang telah dilakukan operasi Eliminasi Gauss.

3.1.8 Prosedur gauss_jordan

```
public static void gauss_jordan(double[][] matrix){
    gauss(matrix);

    int i, j, k;
    int m = matrix.length, n = matrix[0].length;
    int lead_one = 0;
    double ratio = 0d;
    // Backward Elimination
    for(i = 0; i < m; i++){
        for(j = 0; j < n; j++){
            lead_one = j;
            if(matrix[i][j] == 1){
                break;
            }
            if(j == n-1 && matrix[i][n-1] == 0){
                lead_one++;
            }
        }
        if(lead_one != n){
            for(k = 0; k < m; k++){
                if(k != i && matrix[k][lead_one] != 0){
                    ratio = matrix[k][lead_one] / matrix[i][lead_one];
                    for(j = 0; j < n; j++){
                        matrix[k][j] -= ratio * matrix[i][j];
                    }
                }
            }
        }
    }
}
```

Menerima input berupa matriks dan output matriks yang telah dilakukan operasi Eliminasi Gauss. Di dalam prosedur juga memanfaatkan prosedur gauss.

3.1.9 Prosedur pivot

```

public static void pivot(double[][] matrix){
    int i, j, k;
    int m = matrix.length, n = matrix[0].length;
    // Operasi Pertukaran Baris
    int zero_row1 = 0, zero_row2 = 0;
    double[] temp_array = new double[n];

    for(k = 0; k < m; k++){
        for(i = 0; i < m-1; i++){ // Mengkomparasikan indeks kolom matriks
            for(j = 0; j < n; j++){ // Mengecek indeks elemen tidak nol
                zero_row1 = j;
                if(matrix[i][j] != 0){
                    break;
                }
                if(j == n-1 && matrix[i][n-1] == 0){ // Jika tidak ada elemen yang tidak nol
                    zero_row1++;
                }
            }
            for(j = 0; j < n; j++){ // Mengecek indeks elemen tidak nol
                zero_row2 = j;
                if(matrix[i+1][j] != 0){
                    break;
                }
            }

            if(zero_row1 > zero_row2){ // Pertukaran baris
                for(j = 0; j < n; j++){
                    temp_array[j] = matrix[i][j];
                    matrix[i][j] = matrix[i+1][j];
                    matrix[i+1][j] = temp_array[j];
                }
            }
        }
    }
}

```

Menerima input berupa matriks dan output matriks yang sudah dipertukarkan berdasarkan aturan eselon baris.

3.2.0 Prosedur leadingOne

```

public static void leadingOne(double[][] matrix, int i){
    int j, zero_row = 0;
    int n = matrix[0].length;
    double ratio;

    for(j = 0; j < n; j++){
        zero_row = j;
        if(matrix[i][j] != 0){
            break;
        }
        if(j == n-1 && matrix[i][n-1] == 0){
            zero_row++;
        }
    }
    if(zero_row != n){
        ratio = matrix[i][zero_row];
        for(j = 0; j < n; j++){
            matrix[i][j] /= ratio;
        }
    }
}

```

Menerima input berupa matriks dan integer i (untuk merepresentasikan indeks baris) dan output berupa matriks yang telah dilakukan operasi pembagian pada baris i untuk mendapatkan leading one/satu utama.

Terdapat prosedur-prosedur yang serupa dengan prosedur gauss hingga leadingOne, yaitu prosedur gauss_inv, gauss_jordan_inv, pivot_inv, dan leadingOne_inv untuk melakukan operasi yang serupa, namun setiap manipulasi/modifikasi pada matriks masukan dilakukan juga pada sebuah matriks identitas untuk mendapatkan invers dengan menggunakan metode identitas. Digunakan pada prosedur inverseIdentity.

3.2 Operasi SPL

3.2.1 Fungsi SPLgauss_jordan

```

public static double[] SPLgauss_jordan(double[][] A, double[] b){
    int i, j;
    int m = A.length, n = A[0].length;
    int zero_row = 0;
    double[] solution = new double[n];
    double[][] augmented = new double[m][n+1];

    // Set Solutions to undefined
    for(i = 0; i < n; i++){
        solution[i] = Double.NaN;
    }

    // Create Matrix Augmented
    for(i = 0; i < m; i++){
        for(j = 0; j < n+1; j++){
            if(j == n){
                augmented[i][j] = b[i];
            } else{
                augmented[i][j] = A[i][j];
            }
        }
    }

    // Gaussian Elimination
    OperasiPrimitif.gauss_jordan(augmented);
    // Check Solution
    for(j = 0; j < n+1; j++){
        zero_row = j;
        if(augmented[m-1][j] != 0){
            break;
        }
    }

    if(zero_row == n && augmented[m-1][n] == 0){ // Solusi parametrik
        return solution;
    } else if(zero_row == n){ // Tidak memiliki solusi
        return solution;
    } else{
        for(i = 0; i < m; i++){
            solution[i] = augmented[i][n];
        }
    }
    return solution;
}

```

Menerima input sebuah matriks yang menyatakan A dalam persamaan $\mathbf{Ax} = \mathbf{b}$, dan array b yang menyatakan b untuk dijadikan matriks augmented dan dilakukan eliminasi Gauss-Jordan (dengan memanggil prosedur gauss). Output berupa array berisi solusi SPL.

Terdapat fungsi yang serupa yaitu SPLgauss dengan menggunakan metode eliminasi Gauss.

3.2.2 Fungsi SolusiCrammer

```

public static double[] SolusiCramer(double[][] matrix, double[] nilai){
    int n = nilai.length;
    double[] solusi = new double[n];
    double determinanUtama = OperasiDeterminan.DeterminanOBE(matrix);
    if(determinanUtama == 0){
        return solusi; // jika determinan matrix utama 0 maka tidak ada solusi
    }

    double[][] matrictmp = new double[n][n];
    for(int i=0;i<n;i++){ // iterate seluruh kolom yang akan diganti
        OperasiPrimitif.copyMatrix(matrix, matrictmp); // copy matrix
        for(int j=0;j<n;j++){
            matrictmp[j][i] = nilai[j]; // ganti nilai kolom matrix
        }
        double determinanKecil = OperasiDeterminan.DeterminanOBE(matrictmp);
        solusi[i] = determinanKecil/determinanUtama;
    }
    return solusi;
}

```

Menerima input berupa matrix dan input array nilai. Output berupa array dari solusi kaidah Cramer.

3.2.3 Fungsi SolusiSPLInverse

```

public static double[] SolusiSPLInverse(double[][] matrix, double[] nilai){
    double[][] matrixinverse = new double[matrix.length][matrix[0].length];
    OperasiPrimitif.copyMatrix(matrix, matrixinverse);
    OperasiInverse.inverseIdentity(matrixinverse);
    double[] solusi = OperasiPrimitif.SolusiSPL(matrixinverse, nilai);
    return solusi;
}

```

Menerima input berupa matrix dan input array nilai. Output berupa array dari solusi SPL metode invers.

3.3 OperasiDeterminan

3.3.1 Fungsi DeterminanCofactor

```

public static double DeterminanCofactor(double[][] m, int n) {
    // Mendapatkan determinan dari m secara rekursif menggunakan metode kofaktor
    double det = 0;
    if (n == 1) {
        return m[0][0];
    }
    double[][] temp = new double[n][n];
    int sign = 1;
    for (int i = 0; i < n; i++) {
        operasiPrimitif.getCofactor(m, temp, a: 0, i, n);
        det += sign * m[0][i] * DeterminanCofactor(temp, n - 1);
        sign = -sign;
    }
    return det;
}

```

Mencari determinan matrix dengan menggunakan metode cofactor

3.3.2 Fungsi DeterminanOBE

```

public static double DeterminanOBE(double[][] Matrix){
    int dimensi = Matrix.length;
    double[][] matrixOBE = new double[dimensi][dimensi];
    OperasiPrimitif.copyMatrix(Matrix, matrixOBE);
    int tukarcounter = 0;

    for(int i=0;i<dimensi;i++){ // kolom yang akan di nol kan
        if(matrixOBE[i][i] == 0 && i!=dimensi-1){ // apabila diagonal utama nol, tukar barus
            boolean found = false;
            for(int j=i+1;j<dimensi;j++){
                if(matrixOBE[j][i] != 0){
                    found = true;
                    for(int k=0;k<dimensi;k++){
                        double temp = matrixOBE[j][k];
                        matrixOBE[j][k] = matrixOBE[i][k];
                        matrixOBE[i][k] = temp;
                    }
                    break;
                }
            }
            if(!found){ // jika tidak bisa menukar baris
                return 0;
            }else{
                tukarcounter++;
            }
        }
        for(int j=i+1;j<dimensi;j++){ // meng-nol kan kolom dibawahnya
            double kali = matrixOBE[j][i]/matrixOBE[i][i];

            for(int k=0;k<dimensi;k++){
                matrixOBE[j][k] -= kali* matrixOBE[i][k];
            }
        }
    }
    double determinan = Math.pow(-1, tukarcounter);
    for(int i=0;i<dimensi;i++){
        determinan *= matrixOBE[i][i];
    }
    return determinan;
}

```

Mencari determinan matrix dengan metode Operasi Baris Elementer

3.4 Operasi Inverse

3.4.1 Fungsi Adjoint

```
public static double[][] adjoint(double[][] m){
    // Mendapatkan matriks adjoin dari m secara rekursif
    int n = m.length;
    double[][] adj = new double[n][n];
    // if (n == 1){
    //     adj[0][0] = 1;
    //     return adj;
    // }
    int sign = 1;
    double[][] temp = new double[n][n];
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            OperasiPrimitif.getCofactor(m, temp, i, j, n);
            sign = ((i + j) % 2 == 0) ? 1 : -1;
            adj[j][i] = (sign) * (OperasiDeterminan.DeterminanCofactor(temp, n - 1));
        }
    }
    return adj;
}
```

Menerima input matriks m dan mengeluarkan adjoint dari matriks m tersebut.

3.4.2 Fungsi inverseCofactor

```
public static double[][] inverseCofactor(double[][] m){
    // Mendapatkan matriks invers dari m dari adjoint dan determinan
    int n = m.length;
    double[][] inv = new double[n][n];
    double det = OperasiDeterminan.DeterminanCofactor(m, n);
    if (det == 0){
        System.out.println("Determinan 0, tidak memiliki invers");
        return inv;
    }
    double[][] adj = adjoint(m);
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            inv[i][j] = adj[i][j] / det;
        }
    }
    return inv;
}
```

Merupakan sebuah fungsi yang menerima matriks dan mengeluarkan inverse dari matriks tersebut.

3.4.3 Prosedur inverseIdentity

```

public static void inverseIdentity(double[][] matrix){
    int i, j, k;
    int m = matrix.length;
    double[][] copy = new double[m][m];
    double[][] identity = new double[m][m];

    // Check if matrix has inverse
    if(OperasiDeterminan.DeterminanOBE(matrix) == 0){
        System.out.println("Matriks tidak memiliki balikan");
    } else{
        // Copy Matrix
        OperasiPrimitif.copyMatrix(matrix, copy);

        // Create Identity Matrix
        for(i = 0; i < m; i++){
            for(j = 0; j < m; j++){
                if(i == j){
                    identity[i][j] = 1d;
                } else{
                    identity[i][j] = 0d;
                }
            }
        }

        // Elimination
        OperasiPrimitif.gauss_jordan_inv(copy, identity);

        // Store inverse value to matrix
        for(i = 0; i < m; i++){
            for(j = 0; j < m; j++){
                matrix[i][j] = identity[i][j];
            }
        }
    }
}

```

Menerima input berupa matriks dan mengeluarkan output berupa matriks yang telah dilakukan operasi invers identitas.

3.5 Solusi Parametrik

3.5.1 Prosedur solusi

```

public static void solusi(double[][] A, double[] b){
    int i, j, k = 0, l, p, q;
    int nonzero_count;
    int hex = 0x70;
    int UNDEF = -9999;
    double value;
    double[] matrix = new double[A.length][A[0].length + 1];

    // Create Matrix Augmented
    for(i = 0; i < A.length; i++){
        for(j = 0; j < A[0].length + 1; j++){
            if(j == A[0].length){
                matrix[i][j] = b[i];
            } else{
                matrix[i][j] = A[i][j];
            }
        }
    }

    int m = matrix.length, n = matrix[0].length - 1;
    double[] solution = new double[n];
    String[] parvariable = new String[n];
    String[] parsolution = new String[n];
    // Set solutions to undefined
    for(i = 0; i < n-1; i++){
        solution[i] = UNDEF;
    }

    // Gauss-Jordan Elimination
    OperasiPrimitif.gauss_jordan(matrix);
}

```

```

// Parametrik
g = n-1;
for(i = m-1; i >= 0; i--){
    nonzero_count = 0;
    // Count the number of nonzero elements in a row (without the last column)
    for(j = 0; j < n; j++){
        if(matrix[i][j] != 0){
            nonzero_count++;
            k = j; // The col index of the last nonzero element
        }
    }
    // If only one element, then the solution is in the last column
    if(nonzero_count == 1){
        solution[k] = matrix[i][n];
        // Look for the parametric solution (that is, the column of the zero element after the only nonzero)
        for(l = k; l < n; l++){
            if(matrix[i][l] == 0 && solution[l] == UNDEF){
                parvariable[l] = Character.toString((char)hex);
                hex++;
            }
        }
    }
    // OR if there are no nonzero elements in the row, the parametric solution is clear
    else if(nonzero_count == 0){
        parvariable[g] = Character.toString((char)hex);
        hex++;
        g--;
    }
}

// The solution with more than one element in a row
for(i = m-1; i >= 0; i--){
    nonzero_count = 0;
    // Count the nonzero elements in a row
    for(j = n-1; j >= 0; j--){
        if(matrix[i][j] != 0){
            nonzero_count++;
            k = j; // Index of the first nonzero element
        }
    }
    // If more than one element, then subtract the rightmost column of the row (it is the solution)
    if(nonzero_count > 1){
        solution[k] = matrix[i][n];
        for(l = k+1; l < n; l++){
            // If the solution of the current index not parametric
            if(parvariable[l] == null && solution[l] != UNDEF){
                solution[k] -= matrix[i][l] * solution[l];
            }
        }
    }
    // Check if there are any parametric solutions left
    for(i = 0; i < n; i++){
        if(solution[i] == UNDEF && parvariable[i] == null && matrix[i][i] == 0){
            parvariable[i] = Character.toString((char)hex);
            hex++;
        }
    }
}

// Check if there are any parametric solutions left
for(i = 0; i < n; i++){
    if(solution[i] == UNDEF && parvariable[i] == null && matrix[i][i] == 0){
        parvariable[i] = Character.toString((char)hex);
        hex++;
    }
}

// Complete the parametric solution
for(i = 0; i < m; i++){
    // Check the first index of nonzero element in a row
    for(j = 0; j < n; j++){
        k = j;
        if(matrix[i][j] != 0){
            break;
        }
    }
    // Completion
    if(solution[i] != UNDEF && solution[i] != 0){
        parsolution[i] = solution[i] + " ";
    } else if(solution[i] != UNDEF && solution[i] == 0){
        parsolution[i] = "";
    }
    for(l = k; l < n; l++){
        if(matrix[i][l] > 0 && parvariable[l] != null && solution[i] != 0){
            parsolution[i] += "- " + Double.toString(matrix[i][l]) + " " + parvariable[l] + " ";
        } else if(matrix[i][l] < 0 && parvariable[l] != null && solution[i] != 0){
            parsolution[i] += "+" + Double.toString(-matrix[i][l]) + " " + parvariable[l] + " ";
        } else if(matrix[i][l] > 0 && parvariable[l] != null && solution[i] == 0){
            parsolution[i] += " " + Double.toString(-matrix[i][l]) + " " + parvariable[l] + " ";
        } else if(matrix[i][l] < 0 && parvariable[l] != null && solution[i] == 0){
            parsolution[i] += Double.toString(-matrix[i][l]) + " " + parvariable[l] + " ";
        }
    }
}

```

```

    // Complete the remaining solutions
    for(i = 0; i < n-1; i++){
        if(parsolution[i] == null && parvariable[i] != null){
            parsolution[i] = parvariable[i];
        } else if(parsolution[i] == null && solution[i] != UNDEF){
            parsolution[i] = solution[i] + "";
        }
    }

    if(parvariable[n-1] == null){
        parsolution[n-1] = solution[n-1] + "";
    } else{
        parsolution[n-1] = parvariable[n-1];
    }

    // Output Parametric Solution
    for(i = 0; i < n; i++){
        System.out.println("x" + (i+1) + " = " + parsolution[i]);
    }
}

```

Menerima input berupa matriks A dalam persamaan $\mathbf{Ax} = \mathbf{b}$, array b. Output berupa solusi parametrik.

3.5.2 Fungsi isParametrik

```

public static boolean isParametrik(double [][]matrix){
    int i, j, zero_row = 0;
    int m = matrix.length, n = matrix[0].length;
    OperasiPrimitif.gauss_jordan(matrix);

    for(j = 0; j < n; j++){
        zero_row = j;
        if(matrix[m-1][j] != 0){
            break;
        }
    }
    if((zero_row == n-1 && matrix[m-1][n-1] == 0) || (n - 1 > m)){ // Solusi parametrik
        return true;
    } else{
        return false;
    }
}

```

Menerima input matriks dengan output berupa nilai boolean apakah matriks memiliki solusi parametrik atau tidak.

3.6 Interpolasi Polinom rey

3.6.1 Prosedur estimate

```

public static void estimate(double[][] A, double[] b, double x){
    int i,j,l;
    int m = A.length, n = A[0].length;
    double y = 0d;
    double[] solution = new double[n];

    // Solve SPL
    solution = OperasiSPL.SPLgauss_jordan(A, b);

    // Taksiran
    for(i = 0; i < n; i++){
        y += solution[i] * Math.pow(x, i);
    }

    // Output Interpolation Result
    System.out.print("f(x) = ");
    System.out.printf("%.4f", solution[n-1]);
    System.out.print("x^" + (n-1));
    for(i = n-2; i >= 2; i--){
        if(solution[i] >= 0){
            System.out.print(" + ");
            System.out.printf("%.4f", solution[i]);
            System.out.print("x^" + i);
        } else{
            System.out.print(" - ");
            System.out.printf("%.4f", solution[i]);
            System.out.print("x^" + i);
        }
    }
    if(solution[1] >= 0){
        System.out.print(" + ");
        System.out.printf("%.4f", solution[1]);
        System.out.print("x");
    } else{
        System.out.print(" - ");
        System.out.printf("%.4f", solution[1]);
        System.out.print("x");
    }
    if(solution[0] >= 0){
        System.out.print(" + ");
        System.out.printf("%.4f,\n", solution[0]);
    } else{
        System.out.print(" - ");
        System.out.printf("%.4f,\n", solution[0]);
    }
    System.out.print("f(" + x + ") = ");
    System.out.printf("%.4f\n", y);
}

```

Menerima input berupa matriks A dalam persamaan $\mathbf{Ax} = \mathbf{b}$, array b, dan nilai yang akan ditaksir x. Output berupa fungsi dari hasil interpolasi dan hasil taksiran x.

3.7 InterpolasiBicubic

3.7.1 Fungsi BarisMatrixInterpolasiBicubicNormal dan sejenisnya

```

public static double[] BarisMatrixInterpolasiBicubicNormal(double x, double y){
    double[] baris = new double[16];
    for(int j=0;j<4;j++){
        for(int i=0;i<4;i++){
            baris[i+j*4] = Math.pow(x, i) * Math.pow(y, j);
        }
    }
    return baris;
}

```

Mencari baris pada matrix model untuk interpolasi bicubic

Fungsi ini mirip dengan BarisMatrixInterpolasiBicubicDx, BarisMatrixInterpolasiBicubicDy, BarisMatrixInterpolasiBicubicDxy

3.7.2 Fungsi solusiInterpolasiBicubic

```

public static double[] solusiInterpolasiBicubic(double[] nilai){
    double[][] MatrixModel = new double[16][16];
    for(int y=0;y<4;y++){
        for(int x=0;x<4;x++){
            MatrixModel[x+y*4] = BarisMatrixInterpolasiBicubicNormal(x-1, y-1);
        }
    }
    double[] solusi = OperasiSPL.SolusiCrammer(MatrixModel, nilai);
    return solusi;
}

```

Mencari vektor solusi untuk interpolasi bicubic, fungsi ini dibuat agar perhitungan mencari vektor solusi tidak dilakukan berulang-ulang ketika menerima nilai untuk di interpolasi yang berbeda tetapi memiliki persamaan yang sama

Terdapat fungsi serupa yaitu solusiInterpolasiBicubicSpline

3.7.3 Fungsi hasilInterpolasiBicubic

```

public static double hasilInterpolasiBicubic(double[] solusi, double ax, double ay){
    double interpolasi = 0;
    double[] baris = BarisMatrixInterpolasiBicubicNormal(ax, ay);
    for(int i=0;i<16;i++){
        interpolasi += baris[i] * solusi[i];
    }
    return interpolasi;
}

```

Mencari nilai interpolasi bicubic dengan prasyarat vektor solusi didapat dari fungsi solusiInterpolasiBicubic

Terdapat fungsi serupa yaitu hasilInterpolasiBicubicSpline

3.7.4 interpolasiBicubic

```

public static double interpolasiBicubic(double[] nilai, double ax, double ay){
    double[] solusi = solusiInterpolasiBicubic(nilai);
    double interpolasi = hasilInterpolasiBicubic(solusi, ax, ay);
    return interpolasi;
}

```

Melakukan operasi interpolasi bicubic secara lengkap yaitu membuat vektor solusi dan mencari nilai yang ingin diinterpolasi

Terdapat fungsi serupa yaitu interpolasiBicubicSpline

3.8 DoubleLinearReg

3.8.1 Fungsi doubRegSolveCof

```

public static double[][] doubRegSolveCof(double[][] x, double[][] y){
    double[][] xt ; // [x.length][x[0].length]
    double[][] invxxt;
    double [][] xxt ;
    xt = OperasiPrimitif.transpose(x);
    xxt = OperasiPrimitif.PerkalianMatrix(xt,x);
    invxxt = OperasiInverse.inverseCofactor(xxt);

    // b = (XX^T)^-1 X^T Y
    double[][] solve = OperasiPrimitif.PerkalianMatrix(OperasiPrimitif.PerkalianMatrix(invxxt, xt),y);
    return solve;
}

```

Fungsi yang menerima dua buah matriks dan mengeluarkan hasil perhitungan double linier regression dari kedua matriks tersebut.

Terdapat fungsi yang serupa yaitu doubRegSolveIdent.

3.8.2 Prosedur estimateDoubReg

```

public static void estimateDoubReg(double[][] x, double[][] y, double[][] a){
    double[][] solution = doubRegSolveIdent(x,y);
    int m = solution.length, n = solution[0].length;

    // Output Interpolation Result
    System.out.print("f(x) = ");
    for (int i = 0 ; i < m; i++){
        if (i != m-1){
            if (i != 0){
                System.out.printf("%.4f", solution[i][0]);
                System.out.print("x"+i+" + ");
            }else{
                System.out.printf("%.4f + ", solution[i][0]);
            }
        }else{
            System.out.printf("%.4f", solution[i][0]);
            System.out.print("x"+i);
        }
    }
    System.out.println();
    double est = solution[0][0];
    System.out.print("f(xk) = ");
    for (int i = 1 ; i < m ; i++){
        est += solution[i][0] * a[i-1][0];
    }
    System.out.printf("%.4f", est);
}

```

Initial State tiga matriks terdefinisi.

Final state meng-output persamaan regresi dan perkiraan regresi dengan variabel tertentu.

3.9 ScalingCitra

3.9.1 Fungsi takeImage

```
public static int[][] takeImage(String inputImagePath) {
    BufferedImage img = null;
    // BufferedImage img = new BufferedImage(4, 4, BufferedImage.TYPE_4BYTE_ABGR); // image file
    // Read the source image or throw an exception
    try {
        img = ImageIO.read(new File(inputImagePath));
    } catch(Exception e) {
        System.out.println("Terjadi kesalahan.");
    }

    // Get the image width and height dimensions
    int width = img.getWidth();
    int height = img.getHeight();
    int[][] matrix = new int[height][width];

    // Convert to grayscale by looping over pixels, beginning at top-most left coordinate (0,0)
    for (int y = 0; y < height; y++) { // y = rows
        for (int x = 0; x < width; x++) { // x = columns

            // Get the pixel value at this (x,y) coordinate
            int p = img.getRGB(x,y);

            // Extract the alpha, R, G, B values from pixel p
            int a = (p>>24) & 0xff; // Shift bits and unsign
            int r = (p>>16) & 0xff;
            int g = (p>>8) & 0xff;
            int b = p & 0xff;

            // Calculate average color (grayscale it)
            int avg = (r+g+b)/3;

            // Replace RGB value with avg
            p = (a<<24) | (avg<<16) | (avg<<8) | avg;
            matrix[y][x] = p;
        }
    }
    return matrix;
}
```

Mengubah gambar menjadi matrix sehingga bisa diproses

3.9.2 Prosedur saveImage

```

public static void saveImage(String output, int[][] matrix) {
    int height = matrix.length;
    int width = matrix[0].length;

    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_BYTE_GRAY);
    for (int y = 0; y < height; y++) { // y = rows
        for (int x = 0; x < width; x++) { // x = columns
            image.setRGB(x, y, matrix[y][x]);
        }
    }
    // Save or throw exception
    try {
        System.out.println("... Saving image to " + output);
        ImageIO.write(image, formatName: "jpg", new File(output));
    } catch(Exception e) {
        System.out.println(x: "Terjadi kesalahan.");
        return;
    }
    System.out.println(x: "... Image saved.");
}

```

Menyimpan matrix menjadi gambar

Initial State output merupakan path gambar akan disimpan dan sudah benar, nilai matrix terdefinisi dan siap diubah menjadi gambar

Final State gambar tersimpan berdasarkan nilai matrix

3.9.3 Fungsi IntToDoubleMat

```

public static double[][] IntToDoubleMat(int[][] Matrix){
    int n = Matrix.length;
    int m = Matrix[0].length;
    double[][] Matrixout = new double[n][m];

    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            Matrixout[i][j] = Matrix[i][j];
        }
    }
    return Matrixout;
}

```

Mengubah matrix integer menjadi matrix double

Terdapat fungsi serupa yaitu DoubleToIntMat

3.9.4 Fungsi Inbound

```

public static boolean inbound(double[][] Matrix, int i, int j){
    // check if index i,j is inside matrix
    int n = Matrix.length;
    int m = Matrix[0].length;
    boolean cond = (i < n) && (j < m);
    return cond;
}

```

Mengecek apakah index yang diberikan masih berada di dalam matrix

3.9.5 Fungsi prosesBicubic

```

public static double[][] prosesBicubic(double[][] Matrix){
    int n = Matrix.length;
    int m = Matrix[0].length;
    double[][] hasil = new double[2*n][2*m];
    boolean[][] processed = new boolean[2*n][2*m];

    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            hasil[i*2][j*2] = Matrix[i][j];
            processed[i*2][j*2] = true;
        }
    }

    // interpolasi sisi
    for(int i=1;i<2*n;i+=2){
        hasil[i][0] = hasil[i-1][0];
        processed[i][0] = true;
    }
    for(int j=1;j<2*m;j+=2){
        hasil[0][j] = hasil[0][j-1];
        processed[0][j] = true;
    }
    for(int i=1;i<2*n;i+=2){
        hasil[i][1] = (hasil[i-1][1] + hasil[i][0])/2;
        processed[i][1] = true;
    }
    for(int j=1;j<2*m;j++){
        hasil[1][j] = (hasil[0][j] + hasil[1][j-1])/2;
        processed[1][j] = true;
    }

    for(int i=0;i<2*n;i+=2){ // iterate all idx
        for(int j=0;j<2*m;j+=2){
            double last = hasil[i][j];
            double[] value = new double[16];
            for(int k=0;k<4;k++){
                for(int l=0;l<4;l++){
                    if(inbound(hasil, i+2*k, j+2*l) && processed[i+2*k][j+2*l]){
                        last = hasil[i+2*k][j+2*l];
                    }
                    value[4*k + l] = last;
                }
            }
            double[] solusiInterpolasi = InterpolasiBicubic.solusiInterpolasiBicubicSpline(value);
            for(int k=2;k<5;k++){
                for(int l=2;l<5;l++){
                    if(!inbound(hasil, i+k, j+l) || processed[i+k][j+l]) continue;
                    double interpolasi = InterpolasiBicubic.hasilInterpolasiBicubicSpline(solusiInterpolasi, (k-2)/2, (l-2)/2);
                    hasil[i+k][j+l] = interpolasi;
                    processed[i+k][j+l] = true;
                }
            }
        }
    }
    return hasil;
}

```

Melakukan pembesaran citra menggunakan interpolasi bicubic spline

3.9.6 Prosedur scalingCitra

```

public static void scalingCitra(){
    String input, output;
    Scanner scan = new Scanner(System.in);
    System.out.print(s: "Masukkan path image yang ingin diperbesar : ");
    input = scan.nextLine();
    System.out.print(s: "Masukkan path dimana image akan disimpan : ");
    output = scan.nextLine();

    int[][] convertedImage = takeImage(input);
    double[][] doublematrix = IntToDoubleMat(convertedImage);
    double[][] hasilproses = prosesBicubic(doublematrix);
    int[][] outputmatrix = DoubleToIntMat(hasilproses);
    saveImage(output, outputmatrix);

    System.out.println(x: "\nScaling selesai.");
}

```

Melakukan algoritma pembesaran citra dari menerima input gambar sampai menghasilkan output gambar

3.10 IOTerminal

3.10.1 Fungsi InputMatrix

```

public static double[][] InputMatrix(){
    int n,m;
    Scanner scan = new Scanner(System.in);
    System.out.println(x: "Masukkan dimensi matrix n,m : ");
    System.out.print(s: "n : ");
    n = scan.nextInt();
    System.out.print(s: "m : ");
    m = scan.nextInt();
    System.out.println(x: "Berikan input matrix : ");
    double[][] MatrixOut = new double[n][m];
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            MatrixOut[i][j] = scan.nextDouble();
        }
    }
    return MatrixOut;
}

```

Menerima input matrix dari terminal

Terdapat fungsi serupa yaitu InputSPLAugmented

3.10.2 Fungsi InputRegresiX

```

public static double[][] InputRegresiX(int n, int m , Scanner scan){
    System.out.println(x: "Berikan input regresi: ");
    double[][] MatrixSPL = new double[m][n];
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            MatrixSPL[i][j] = scan.nextDouble();
        }
    }
    return MatrixSPL;
}

```

Menerima input persamaan regresi untuk sumbu x dari terminal

Terdapat fungsi serupa yaitu InputRegresiY

3.10.3 Prosedur DisplayArray

```

public static void DisplayArray(double[] array){
    for(int j=0;j<array.length;j++){
        System.out.print(array[j] + " ");
    }
    System.out.println();
}

```

Menampilkan array ke terminal

Terdapat prosedur serupa yaitu DisplayMatrix

3.10.4 Fungsi persamaanSPLAugmented

```

public static double[][] persamaanSPLAugmented(double[][] MatrixAugmented){
    int n = MatrixAugmented.length;
    int m = MatrixAugmented[0].length;
    double[][] Matrix = new double[n][m-1];
    for(int i=0;i<n;i++){
        for(int j=0;j<m-1;j++){
            Matrix[i][j] = MatrixAugmented[i][j];
        }
    }
    return Matrix;
}

```

Memecah matrix augmented menjadi matrix persamaan SPL saja

Terdapat fungsi serupa yaitu nilaiSPLAugmented

3.10.5 Fungsi cekDeterminan

```

public static boolean cekDeterminan(double[][] Matrix){
    // return true jika boolean != 0, false jika boolean 0;
    if(Matrix.length != Matrix[0].length) return true;
    double determinan = OperasiDeterminan.DeterminanOBE(Matrix);
    return determinan != 0;
}

```

Mengecek apakah determinan matrix adalah 0 atau determinan matrix tidak ada karena bukan matrix persegi

3.10.6 Fungsi adaSolusiSPL

```

public static boolean adaSolusiSPL(double[][] Matrix, double[] nilai){
    boolean adaDeterminan = cekDeterminan(Matrix);
    boolean solusiParametrik = SolusiParametrik.isParametrik(Matrix);
    if(!adaDeterminan){
        System.out.println(x: "Determinan 0, Tidak ada solusi.");
        return false;
    }else if(solusiParametrik) {
        SolusiParametrik.solusi(Matrix, nilai);
        return false;
    }
    return true;
}

```

Menentukan apakah SPL yang diberikan memiliki solusi parametrik, solusi unik, atau tidak ada solusi

3.10.7 Prosedur PrintSolusiSPL

```

public static void PrintSolusiSPL(double[] solusi){
    int n = solusi.length;
    System.out.println(x: "Solusi persamaan tersebut adalah : ");
    for(int i=0;i<n;i++){
        System.out.println("x" + i + " : " + solusi[i]);
    }
}

```

Menampilkan vektor solusi SPL ke terminal

3.10.8 Prosedur MenuSPLGauss

```

public static void MenuSPLGauss(int o){
    double[][] MatrixAugmented = InputSPLAugmented();
    double[][] Matrix = persamaanSPLAugmented(MatrixAugmented);
    double[] nilai = nilaiSPLAugmented(MatrixAugmented);
    if (o == 1){
        boolean adaSolusi = adaSolusiSPL(Matrix, nilai);
        if(adaSolusi){
            double[] solusi = OperasiSPL.SPLgauss(Matrix, nilai);
            PrintSolusiSPL(solusi);
        }else if(o == 2) {
            System.out.println(x: "Masukkan nama file yang akan di-read : ");
            Scanner input = new Scanner(System.in);
            String name = input.nextLine();
            try {
                writeSPLGauss(name, Matrix, nilai);
            } catch (IOException e) {
                System.out.println(x: "Terjadi kesalahan.");
            }
        }
    }
}

```

Melakukan prosedur penyelesaian SPL dengan menggunakan metode gauss

Terdapat prosedur serupa yaitu MenuSPLGaussJordan, MenuSPLInverse, MenuSPLCrammer

3.10.9 Prosedur MenuDeterminanCofactor

```

public static void MenuDeterminanCofactor(int i){
    double[][] Matrix = InputMatrix();
    if (i == 1){
        if(Matrix.length != Matrix[0].length){
            System.out.print(s: "Matrix bukan matrix segitiga, tidak memiliki determinan");
        }else{
            double determinan = OperasiDeterminan.DeterminanCofactor(Matrix, Matrix.length);
            System.out.println("Determinan Matrix : " + determinan);
        }
    }else if(i == 2){
        System.out.println(x: "Masukkan nama file yang akan di-read : ");
        Scanner input = new Scanner(System.in);
        String name = input.nextLine();
        try{
            writeDeterminanCofactor(name,Matrix);
        }catch (IOException e){
            System.out.println(x: "Terjadi kesalahan.");
        }
    }
}

```

Melakukan prosedur mencari determinan matrix dengan metode cofactor

Terdapat prosedur serupa yaitu MenuDeterminanOBE

3.10.10 Prosedur MenuInversAdjoin

```

public static void MenuInversAdjoin(int o){
    double[][] Matrix = InputMatrix();
    double determinan = OperasiDeterminan.DeterminanCofactor(Matrix, Matrix.length);
    if (o == 1){
        if(determinan == 0 || Matrix.length != Matrix[0].length){
            System.out.println(x: "Matrix tidak memiliki Invers");
        }else{
            Matrix = OperasiInverse.inverseCofactor(Matrix);
            System.out.println(x: "Invers Matrix: ");
            DisplayMatrix(Matrix);
        }
    }else if(o == 2){
        System.out.println(x: "Masukkan nama file yang akan di-read : ");
        Scanner input = new Scanner(System.in);
        String name = input.nextLine();
        try{
            writeInverseAdjoint(name,Matrix,determinan);
        }catch (IOException e){
            System.out.println(x: "Terjadi kesalahan.");
        }
    }
}

```

Melakukan prosedur mencari matrix balikan dengan metode adjoint

Terdapat prosedur serupa yaitu MenuInversOBE

3.10.11 Prosedur MenuInterpolasiPolinom

```

public static void MenuInterpolasiPolinom(int o){
    int i, j;
    Scanner sc = new Scanner(System.in);
    double x, y;
    int point;

    System.out.print(s: "Masukkan jumlah titik : ");
    point = sc.nextInt();

    double[][] Matrix = new double[point][point];
    double[] nilai = new double[point];

    for(i = 0; i < point; i++){
        System.out.print("Masukkan titik x" + i + " y" + i + " : ");
        x = sc.nextDouble();
        y = sc.nextDouble();
        for(j = 0; j < point; j++){
            Matrix[i][j] = Math.pow(x, j);
        }
        nilai[i] = y;
    }
    System.out.print(s: "Masukkan nilai x yang ingin di taksir : ");
    double xtaksir = sc.nextDouble();
    if (o == 1){
        InterpolasiPolinom.estimate(Matrix, nilai, xtaksir);
    }else if (o == 2){
        System.out.println(x: "Masukkan nama file yang akan disimpan : ");
        Scanner input = new Scanner(System.in);
        String name = input.nextLine();
        try{
            writePolinom(name,Matrix, nilai, xtaksir);
        }catch (IOException e){
            System.out.print(s: "Terjadi kesalahan.");
        }
    }
}

```

Melakukan prosedur menginterpolasi nilai persamaan polinom

3.10.12 Prosedur MenuInterpolasiBicubic

```
public static void MenuInterpolasiBicubic(int o){
    double[] nilai = new double[16];
    double a;
    Scanner scan = new Scanner(System.in);
    for(int y=1;y<3;y++){
        for(int x=1;x<3;x++){
            System.out.print("Masukkan nilai f(%d,%d) : ", x, y);
            a = scan.nextDouble();
            nilai[x+y*4+5] = a;
        }
    }
    System.out.print(s: "Masukkan nilai yang ingin di interpolasi : ");
    double ax, ay;
    ax = scan.nextDouble();
    ay = scan.nextDouble();
    if (o == 1){
        double interpolasi = InterpolasiBicubic.interpolasiBicubic(nilai, ax, ay);
        System.out.printf(format: "Nilai f(%f,%f) hasil interpolasi adalah : %f\n", ax, ay, interpolasi);
    }else if(o == 2){
        System.out.println(x: "Masukkan nama file yang akan disimpan : ");
        Scanner input = new Scanner(System.in);
        String name = input.nextLine();
        try{
            writeBicubic(name, nilai, ax,ay);
        }catch (IOException e){
            System.out.println(x: "Terjadi kesalahan.");
        }
    }
}
```

Melakukan prosedur menginterpolasi nilai dengan interpolasi bicubic

3.10.13 Prosedur MenuRegresiLinierBerganda

```
public static void MenuRegresiLinierBerganda(int o){
    int n,m;
    double[][] Matrix;
    double[][] nilai;
    double[][] var ;
    Scanner scan = new Scanner(System.in);
    System.out.println(x: "Masukkan banyak peubah x (n) : ");
    System.out.print(s: "n : ");
    n = scan.nextInt();
    System.out.println(x: "Masukkan banyak sampel (m) : ");
    System.out.print(s: "m : ");
    m = scan.nextInt();
    Matrix = InputRegresiX(n,m,scan);
    System.out.println(x: "Berikan input nilai Y : ");
    nilai = InputRegresiY(m,scan);
    System.out.println(x: "Masukkan semua nilai x yang ingin di taksir : ");
    var = InputRegresiY(n-1, scan);
    if (o == 1){
        DoubleLinearReg.estimateDoubReg(Matrix, nilai, var);
    }else if( o == 2){
        System.out.println(x: "Masukkan nama file yang akan disimpan : ");
        Scanner input = new Scanner(System.in);
        String name = input.nextLine();
        try {
            writeDoubleReg(name, Matrix, nilai, var);
        } catch (IOException e) {
            System.out.println(x: "file tidak ada");
        }
    }
}
```

Melakukan prosedur mentaksir nilai dengan teknik regresi linier berganda

3.11 IOfiles

3.11.1 Fungsi readMatrix

```
public static double[][] readMatrix(String file) throws FileNotFoundException {
    int row, col;
    row = col = 0;

    File matrix = new File(file);
    Scanner scan_row = new Scanner(matrix);
    while (scan_row.hasNextLine()) {
        row++;
        Scanner scan_col = new Scanner(scan_row.nextLine());
        int n = 0;
        while (scan_col.hasNextDouble()) {
            n++;
            scan_col.nextDouble();
        }
        col = n;
    }
    Scanner scan = new Scanner(matrix);
    double[][] mat;
    mat = new double[row][col];

    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            if (scan.hasNextLine()) {
                mat[i][j] = scan.nextDouble();
            }
        }
    }
    return mat;
}
```

Dengan input string nama file akan meng-outputkan matriks di dalam file tersebut.

3.11.2 Prosedur filesSPLCrammer dan sejenisnya

```
public static void filesSPLCrammer(String file) {
    double[][] matrix = new double[1][1];
    try{
        matrix = readMatrix(file);
    }catch (FileNotFoundException e){
        System.out.println("File tidak ditemukan.");
        return;
    }
    double[][] Matrix = persamaanSPLAugmented(matrix);
    double[] nilai = nilaiSPLAugmented(Matrix);
    System.out.println("Masukkan nama file yang akan disimpan : ");
    Scanner input = new Scanner(System.in);
    String name = input.nextLine();
    try {
        writeSPLCrammer(name, Matrix, nilai);
    }catch (IOException e){
        System.out.println("Terjadi kesalahan.");
    }
}
```

Initial State string yang terdefinisi.

Final State terbuat file dengan nama string yang berisi hasil perhitungan operasi matrix.

Fungsi ini mirip dengan filesSPLGauss, filesSPLGaussJordan, filesSPLInverse ,filesDeterminanOBE, filesDeterminanCofactor , filesInverseOBE dan filesInverseAdjoint.

3.11.3 Prosedur writeSPLGauss dan sejenisnya

```
public static void writeSPLGauss(String file, double[][] Matrix, double[] nilai)
    throws IOException {
    File myObj = new File(file);
    if (myObj.createNewFile()) {
        System.out.println("File dibuat: " + myObj.getName());
    } else {
        System.out.println("File sudah ada.");
        System.out.println("File akan di-overwrite.");
    }
    PrintStream o = new PrintStream(myObj);
    PrintStream console = System.out;
    System.setOut(o);
    boolean adaSolusi = adaSolusiSPL(Matrix, nilai);
    if (adaSolusi) {
        double[] solusi = OperasiSPL.SPLgauss(Matrix, nilai);
        try {
            writeArray(file, solusi);
        } catch (IOException e) {
            System.out.println("Terjadi kesalahan.");
        }
        System.setOut(console);
    }
}
```

Initial State string, matrix, dan array yang terdefinisi.

Final State terbuat file dengan nama string yang berisi hasil perhitungan operasi matrix.

Fungsi ini mirip dengan filesSPLCrammer, filesSPLGaussJordan, filesSPLInverse , filesDeterminanOBE, filesDeterminanCofactor , filesInverseOBE dan filesInverseAdjoint.

3.11.4 Fungsi readBicubic

```
public static double[][] readBicubic(String file) throws FileNotFoundException {
    int row, col;
    row = col = 4;
    File matrix = new File(file);

    double[][] mat;
    mat = new double[row][col];
    Scanner scan = new Scanner(matrix);
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            if (scan.hasNextLine()) {
                mat[i][j] = scan.nextDouble();
            }
        }
    }
    return mat;
}
```

Fungsi yang menerima string nama file dan me-return matriks bicubic yang sudah dibaca.

3.11.5 Fungsi readBicubicValue

```

public static double[][] readBicubicValue(String file) throws FileNotFoundException {
    int row, col;
    row = 1;
    col = 2;
    double[][] mat;
    File matrix = new File(file);
    mat = new double[col][row];
    Scanner scan = new Scanner(matrix);
    for (int i = 0; i < 4; i++) {
        scan.nextLine();
    }
    for (int i = 0; i < col; i++) {
        mat[i][0] = scan.nextDouble();
    }
    return mat;
}

```

Fungsi yang menerima string berupa nama file dan meng-output nilai yang akan ditafsir polinom bicubic.

3.11.6 Prosedur writeBicubic

```

public static void writeBicubic(String file, double[] nilai, double ax, double ay) throws IOException{
    File myObj = new File(file);
    if (myObj.createNewFile()) {
        System.out.println("File dibuat: " + myObj.getName());
    } else {
        System.out.println("File sudah ada.");
        System.out.println("File akan di-overwrite.");
    }
    PrintStream o = new PrintStream(myObj);
    PrintStream console = System.out;
    System.setOut(o);
    double interpolasi = InterpolasiBicubic.interpolasiBicubic(nilai, ax, ay);
    System.out.printf("Nilai f(%f,%f) hasil interpolasi adalah : %f\n", ax, ay, interpolasi);
    System.setOut(console);
}

```

Initial state string, array, double terdefinisi

Final state nilai dari operasi bicubic tertulis dalam file dengan nama string

3.11.7 Prosedur inputFilePolinom

```

public static void inputFilePolinom(String file){
    double[][] MatrixAugmented;
    try {
        MatrixAugmented = readMatrix(file);
    } catch (FileNotFoundException e) {
        System.out.println("File tidak ditemukan.");
        System.out.println("Kembali ke menu sebelumnya.");
        return;
    }
    int n = MatrixAugmented.length;
    int m = MatrixAugmented[0].length;
    double[][] Matrix = new double[n][m-1];
    double[] nilai = new double[n];
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            Matrix[i][j] = MatrixAugmented[i][j];
        }
    }
    for(int j=0;j<n;j++){
        nilai[j] = MatrixAugmented[j][m-1];
    }

    Scanner scan = new Scanner(System.in);
    System.out.print("Masukkan nilai x yang ingin di taksir : ");
    double x = scan.nextDouble();
    System.out.println("Masukkan nama file yang akan disimpan : ");
    Scanner input = new Scanner(System.in);
    String name = input.nextLine();
    try{
        writePolinom(name, Matrix, nilai, x);
    } catch (IOException e){
        System.out.print("Terjadi kesalahan.");
    }
}

```

Initial State string terdefinisi

Final State terdapat file yang berisi hasil operasi polinom.

Terdapat fungsi serupa yaitu inputFilesDoubleReg.

3.11.8 Prosedur writeDoubleReg

```
public static void writeDoubleReg(String file, double[][] x, double[][] y, double[][] a)
throws IOException {

    File myObj = new File(file);
    if (myObj.createNewFile()) {
        System.out.println("File dibuat " + myObj.getName());
    } else {
        System.out.println("File sudah ada.");
        System.out.println("File akan di-overwrite.");
    }
    PrintStream o = new PrintStream(myObj);
    PrintStream console = System.out;
    System.setOut(o);

    double[][] solution = doubRegSolveIdent(x, y);

    int m = solution.length, n = solution[0].length;

    // Output Interpolation Result
    System.out.print("f(x) = ");
    for (int i = 0; i < m; i++) {
        if (i != m - 1) {
            if (i != 0) {
                System.out.printf("%4f", solution[i][0]);
                System.out.print("x" + i + " + ");
            } else {
                System.out.printf("%4f + ", solution[i][0]);
            }
        } else {
            System.out.printf("%4f", solution[i][0]);
            System.out.print("x" + i);
        }
    }
}
```

Initial State string, matriks terdefinisi.

Final State tertulis hasil double linier regression di file.

Terdapat fungsi serupa yaitu writePolinom.

3.12 Menu

3.12.1 Prosedur daftarMenu

```

public static void daftarMenu(){
    System.out.print("""
        Menu
        1. Sistem Persamaan Linier
        2. Determinan
        3. Matriks balikan
        4. Interpolasi Polinom
        5. Interpolasi Bicubic
        6. Regresi Linier berganda
        7. Perbesaran Citra
        8. Keluar
        Masukan nomor menu :
        """);
}

```

Meng-output menu mengenai fitur yang tersedia di dalam program.

Terdapat prosedur serupa seperti subMenuSPL, subMenuDet, subMenuInv, iMenu, dan oMenu.

3.12.2 Prosedur inverse

```

public static void inverse(int menu2){
    int i = iMenu();
    if (menu2 == 1) {
        if (i == 1) {
            int o = oMenu();
            MenuInversOBE(o);
            return;
        } else if (i == 2) {
            Scanner input = new Scanner(System.in);
            System.out.println("Masukkan nama file yang akan di-read : ");
            String file = input.nextLine();
            filesInverseOBE(file);
            return;
        } else {
            System.out.println("Input tidak dikenal.");
            return;
        }
    } else if (menu2 == 2) {
        if (i == 1) {
            int o = oMenu();
            MenuInversAdjoin(o);
            return;
        } else if (i == 2) {
            Scanner input = new Scanner(System.in);
            System.out.println("Masukkan nama file yang akan di-read : ");
            String file = input.nextLine();
            filesInverseAdjoint(file);
            return;
        } else {
            System.out.println("Input tidak dikenal.");
            return;
        }
    }else{
        System.out.println("Input tidak dikenal.");
    }
}

```

Initial State int terdefinisi

Final State akan keluar output inverse sesuai dengan menu yang sudah dipilih.

Terdapat prosedur serupa yaitu spl dan determinan.

3.12.3 Prosedur operasiMatriks

```

public static void operasiMatriks(int menu1) {
    int menu2;
    if (menu1 == 1) {
        menu2 = subMenuSPL();
        if (menu2 > 0 && menu2 < 5){
            spl(menu2);
        }else return;
    } else if (menu1 == 2) {
        menu2 = subMenuDet();
        if (menu2 > 0 && menu2 < 3) {
            determinan(menu2);
        }else return;
    } else if (menu1 == 3) {
        menu2 = subMenuInv();
        if (menu2 > 0 && menu2 < 3) {
            inverse(menu2);
        }else return;
    } else{
        System.out.println("Input tidak dikenal.");
    }
}

```

Initial State int menu1 terdefinisi

Final State user akan ditunjukkan ke fitur yang sudah dipilih

Terdapat fungsi yang serupa yaitu aplikasiMatriks.

3.12.4 Prosedur interPol

```

public static void interPol(){
    int i = iMenu();
    if (i == 1){
        int o = oMenu();
        MenuInterpolasiPolinom(o);
        return;
    }else if(i == 2){
        Scanner input = new Scanner(System.in);
        System.out.println("Masukkan nama file yang akan di-read : ");
        String file = input.nextLine();
        inputFilesPolinom(file);
        return;
    }else{
        System.out.println("Input tidak dikenal.");
    }
}

```

Inisial State –

Final State muncul output sesuai dengan pilihan user.

Terdapat prosedur yang serupa yaitu interBic , doubReg, scalingImage.

3.12.5 Prosedur menuLoop

```

public static void menuLoop() {
    daftarMenu();
    int menu1, menu2;
    Scanner input = new Scanner(System.in);
    menu1 = input.nextInt();
    while (true) {
        if (menu1 >= 1 && menu1 <= 3) {
            operasiMatriks(menu1);
            System.out.println("Kembali ke menu utama.");
        } else if (menu1 >= 4 && menu1 <= 7) {
            aplikasiMatriks(menu1);
            System.out.println("Kembali ke menu utama.");
        }
        if (menu1 < 1 || menu1 > 8) {
            System.out.println("Masukan menu yang valid.");
        }
        if (menu1 == 8) {
            quit();
        }
        daftarMenu();
        menu1 = input.nextInt();
    }
}

```

Initial State –

Final State user akan diteruskan sesuai dengan pilihan menu.

3.13 Main

3.13.1 Fungsi main

```

public class Main {
    ♦ yansans
    public static void main(String[] Args) { menuLoop(); }
}

```

Menerima input pengguna dan mengeluarkan output sesuai dengan masukan input yang tersedia.

BAB IV

EKSPERIMENT

Berikut adalah hasil eksekusi program terhadap contoh-contoh kasus yang sudah diberikan.

4.1 Temukan solusi SPL $Ax = b$

4.1.1

Input

```
Masukkan dimensi matrix n,m :  
n : 4  
m : 5  
Berikan input matrix augmented :  
1 1 -1 -1 1  
2 5 -7 -5 -2  
2 -1 1 3 4  
5 2 -4 2 6
```

Output

```
Solusi persamaan tersebut adalah :  
x0 : 2.0  
x1 : -1.0  
x2 : -4.0  
x3 : 1.0
```

4.1.2

Input

```
Berikan input matrix augmented :
```

```
1 -1 0 0 1 3  
1 1 0 -3 0 6  
2 -1 0 1 -1 5  
-1 2 0 -2 1 -1
```

Output

Error out of bounds

Tidak menangani kasus jumlah variabel yang jauh lebih banyak daripada baris matriks

4.1.3

Input

```
Masukkan dimensi matrix n,m :
```

```
n : 3
```

```
m : 7
```

```
Berikan input matrix augmented :
```

```
0 1 0 0 1 0 2  
0 0 0 1 1 0 -1  
0 1 0 0 0 1 1
```

Output

Error out of bounds

Tidak menangani kasus jumlah variabel yang jauh lebih banyak daripada baris matriks

4.1.4

Input

```
Masukkan dimensi matrix n,m :  
n : 6  
m : 7  
Berikan input matrix augmented :  
1 0.5 0.33 0.25 0.2 0.17 1  
0.5 0.33 0.25 0.2 0.17 0.14 0  
0.33 0.25 0.2 0.17 0.14 0.125 0  
0.25 0.2 0.17 0.14 0.125 0.11 0  
0.2 0.17 0.14 0.125 0.11 0.1 0  
0.17 0.14 0.125 0.11 0.1 0.09 0
```

Output

```
x0 : 0.5  
x1 : 0.25  
x2 : 0.17  
x3 : 0.125  
x4 : 0.1  
x5 : 1.0
```

4.2 SPL berbentuk matriks *augmented*

4.2.1

Input

```
Masukkan dimensi matrix n,m :  
n : 4  
m : 5  
Berikan input matrix augmented :  
1 -1 2 -1 -1  
2 1 -2 -2 -2  
-1 2 -4 1 1  
3 0 0 -3 -3
```

Output

```
x1 = 1.0p  
x2 = 2.0q  
x3 = q  
x4 = p
```

4.2.2

Input

```
Masukkan dimensi matrix n,m :  
n : 6  
m : 5  
Berikan input matrix augmented :  
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
```

Output

```
x1 =  
x2 =  
x3 = - -1.0q  
x4 = p
```

4.3 SPL berbentuk

4.3.1

Input

```
Berikan input matrix augmented :  
8 1 3 2 0  
2 9 -1 -2 1  
1 3 2 -1 2  
1 0 6 4 3
```

Output

```
Solusi persamaan tersebut adalah :  
x0 : 0.0  
x1 : 1.0  
x2 : 2.0  
x3 : 3.0
```

4.3.2

Input

```
Berikan input matrix augmented :  
0 0 0 0 0 1 1 1 13.00  
0 0 0 1 1 1 0 0 0 15.00  
0 0 0.04289 0 0.04289 0.75 0.04289 0.75 0.61396 14.79  
0 0.25 0.91421 0.25 0.91421 0.25 0.91421 0.25 0 14.31  
0.61396 0.75 0.04289 0.75 0.04289 0 0.04289 0 0 3.81  
0 0 1 0 0 1 0 0 1 18.00  
0 1 0 0 1 0 0 1 0 12.00  
1 0 0 1 0 0 1 0 0 6.00  
0.04289 0.75 0.61396 0 0.04289 0.75 0 0 0.04289 10.51  
0.91421 0.25 0 0.25 0.91421 0.25 0 0.25 0.91421 16.13  
0.04289 0 0 0.75 0.04289 0 0.61396 0.75 0.04289 7.04|
```

Output

```
x1 =  
x2 =  
x3 =  
x4 =  
x5 =  
x6 =  
x7 =  
x8 = - -1.0q  
x9 = p
```

Hal ini terjadi karena kami membatasi loop yang terjadi agar tidak error out of bound. Akan tetapi, hal itu menyebabkan beberapa variabel tidak muncul pada output walapun hasil sudah dihitung.

4.4 Interpolasi Polinom

4.4.1

Input

```
Masukkan jumlah titik : 7  
Masukkan titik x0 y0 : 0.4 0.043  
Masukkan titik x1 y1 : 0.7 0.005  
Masukkan titik x2 y2 : 0.11 0.058  
Masukkan titik x3 y3 : 0.14 0.072  
Masukkan titik x4 y4 : 0.17 0.1  
Masukkan titik x5 y5 : 0.2 0.13  
Masukkan titik x6 y6 : 0.23 0.147  
Masukkan nilai x yang ingin di taksir : 0.2
```

```
Masukkan jumlah titik : 7
Masukkan titik x0 y0 : 0.4 0.043
Masukkan titik x1 y1 : 0.7 0.005
Masukkan titik x2 y2 : 0.11 0.058
Masukkan titik x3 y3 : 0.14 0.072
Masukkan titik x4 y4 : 0.17 0.1
Masukkan titik x5 y5 : 0.2 0.13
Masukkan titik x6 y6 : 0.23 0.147
Masukkan nilai x yang ingin di taksir : 0.55
```

```
Masukkan jumlah titik : 7
Masukkan titik x0 y0 : 0.4 0.043
Masukkan titik x1 y1 : 0.7 0.005
Masukkan titik x2 y2 : 0.11 0.058
Masukkan titik x3 y3 : 0.14 0.072
Masukkan titik x4 y4 : 0.17 0.1
Masukkan titik x5 y5 : 0.2 0.13
Masukkan titik x6 y6 : 0.23 0.147
Masukkan nilai x yang ingin di taksir : 0.85
```

```
Masukkan jumlah titik : 7
Masukkan titik x0 y0 : 0.4 0.043
Masukkan titik x1 y1 : 0.7 0.005
Masukkan titik x2 y2 : 0.11 0.058
Masukkan titik x3 y3 : 0.14 0.072
Masukkan titik x4 y4 : 0.17 0.1
Masukkan titik x5 y5 : 0.2 0.13
Masukkan titik x6 y6 : 0.23 0.147
Masukkan nilai x yang ingin di taksir : 1.28
```

Output

```
f(x) = -4212.4345x^6 + 7102.3992x^5 -4346.3140x^4 + 1220.8549x^3 -163.9157x^2 + 10.2764x -0.1846,
f(0.2) = 0.1300
```

```
f(x) = -4212.4345x^6 + 7102.3992x^5 -4346.3140x^4 + 1220.8549x^3 -163.9157x^2 + 10.2764x -0.1846,
f(0.55) = 2.1376
```

```
f(x) = -4212.4345x^6 + 7102.3992x^5 -4346.3140x^4 + 1220.8549x^3 -163.9157x^2 + 10.2764x -0.1846,
f(0.85) = -66.2696
```

```
f(x) = -4212.4345x^6 + 7102.3992x^5 -4346.3140x^4 + 1220.8549x^3 -163.9157x^2 + 10.2764x -0.1846,
f(1.28) = -3485.1449
```

4.4.2

Input

```
Masukkan jumlah titik : 10
Masukkan titik x0 y0 : 6.567 12624
Masukkan titik x1 y1 : 7 21807
Masukkan titik x2 y2 : 7.258 38391
Masukkan titik x3 y3 : 7.451 54517
Masukkan titik x4 y4 : 7.548 51952
Masukkan titik x5 y5 : 7.839 28228
Masukkan titik x6 y6 : 8.161 35764
Masukkan titik x7 y7 : 8.484 20813
Masukkan titik x8 y8 : 8.709 12408
Masukkan titik x9 y9 : 9 10534
Masukkan nilai x yang ingin di taksir : 7.516
```

```
Masukkan jumlah titik : 10
Masukkan titik x0 y0 : 6.567 12624
Masukkan titik x1 y1 : 7 21807
Masukkan titik x2 y2 : 7.258 38391
Masukkan titik x3 y3 : 7.451 54517
Masukkan titik x4 y4 : 7.548 51952
Masukkan titik x5 y5 : 7.839 28228
Masukkan titik x6 y6 : 8.161 35764
Masukkan titik x7 y7 : 8.484 20813
Masukkan titik x8 y8 : 8.709 12408
Masukkan titik x9 y9 : 9 10534
Masukkan nilai x yang ingin di taksir : 8.323
```

```
Masukkan jumlah titik : 10
Masukkan titik x0 y0 : 6.567 12624
Masukkan titik x1 y1 : 7 21807
Masukkan titik x2 y2 : 7.258 38391
Masukkan titik x3 y3 : 7.451 54517
Masukkan titik x4 y4 : 7.548 51952
Masukkan titik x5 y5 : 7.839 28228
Masukkan titik x6 y6 : 8.161 35764
Masukkan titik x7 y7 : 8.484 20813
Masukkan titik x8 y8 : 8.709 12408
Masukkan titik x9 y9 : 9 10534
Masukkan nilai x yang ingin di taksir : 9.167
```

```
Masukkan jumlah titik : 10
Masukkan titik x0 y0 : 6.567 12624
Masukkan titik x1 y1 : 7 21807
Masukkan titik x2 y2 : 7.258 38391
Masukkan titik x3 y3 : 7.451 54517
Masukkan titik x4 y4 : 7.548 51952
Masukkan titik x5 y5 : 7.839 28228
Masukkan titik x6 y6 : 8.161 35764
Masukkan titik x7 y7 : 8.484 20813
Masukkan titik x8 y8 : 8.709 12408
Masukkan titik x9 y9 : 9 10534
Masukkan nilai x yang ingin di taksir : 10
```

Output

$$f(7.516) = 53566.8086$$

```
f(8.323) = 36331.7227
```

```
f(9.167) = -667646.2188
```

```
f(10.0) = -216931708.9375
```

4.4.3

Input

```
Masukkan jumlah titik : 6
Masukkan titik x0 y0 : 0 0
Masukkan titik x1 y1 : 0.4 0.4189
Masukkan titik x2 y2 : 0.8 0.5072
Masukkan titik x3 y3 : 1.2 0.5609
Masukkan titik x4 y4 : 1.6 0.5837
Masukkan titik x5 y5 : 2 0.5767
Masukkan nilai x yang ingin di taksir : 0.5
```

Output

```
f(x) = 0.2358x^5 -1.4188x^4 + 3.2329x^3 -3.5499x^2 + 2.0347x + 0.0000,
f(0.5) = 0.4527
```

4.5 Interpolasi Bicubic

```
Masukkan nilai f(-1,-1) : 153
Masukkan nilai f(0,-1) : 59
Masukkan nilai f(1,-1) : 210
Masukkan nilai f(2,-1) : 96
Masukkan nilai f(-1,0) : 125
Masukkan nilai f(0,0) : 161
Masukkan nilai f(1,0) : 72
Masukkan nilai f(2,0) : 81
Masukkan nilai f(-1,1) : 98
Masukkan nilai f(0,1) : 101
Masukkan nilai f(1,1) : 42
Masukkan nilai f(2,1) : 12
Masukkan nilai f(-1,2) : 21
Masukkan nilai f(0,2) : 51
Masukkan nilai f(1,2) : 0
Masukkan nilai f(2,2) : 16
Masukkan nilai yang ingin di interpolasi : 0 0
```

Output

```
Nilai f(0.000000,0.000000) hasil interpolasi adalah : 161.000000
```

4.6 Regresi Linier Berganda

Input

```
Berikan input regresi:
20 863.1 1530.4 587.84
863.1 54876.89 67000.09 25283.395
1530.4 67000.09 117912.32 44976.867
587.84 25283.395 44976.867 17278.5086
Berikan input nilai Y :

19.42
779.477
1483.437
571.1219
Masukkan semua nilai x yang ingin di taksir :
50
76
29.30
```

Output

```
f(x) = -3.4883 + -0.0026x1 + 0.0008x2 + 0.1537x3
f(xk) = 0.9441
```

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Kesimpulan

Setelah melakukan pengerjaan dan pengujian, program berbahasa Java yang memiliki fitur input dan output dari terminal atau file untuk menghitung operasi matriks dan aplikasinya berhasil dibuat. Fitur-fitur tersebut terdiri atas

1. Menyelesaikan SPL dengan metode eliminasi Gauss, eliminasi Gauss-Jordan, matriks balikan , dan kaidah Cramer.
2. Menentukan determinan dengan metode OBE dan kofaktor.
3. Menentukan matriks inverse dengan metode OBE dan adjoint.
4. Menyelesaikan interpolasi polinom.
5. Menghitung hasil fungsi interpolasi bicubic.
6. Menghitung perkiraan dari double linier regression.
7. Memperbesar citra dengan aplikasi konsep interpolasi bicubic.

5.2 Saran

Melihat program yang dibuat, tim penulis memeliki beberapa saran

1. Lebih mempertimbangkan kembali kasus-kasus yang dapat ditemui karena ada beberapa algoritma yang terpaku hanya pada satu kasus sehingga tidak dapat digunakan pada kasus yang lain.
2. Terkait kode yang dibuat masih bisa dikembangkan lebih lanjut seperti dengan mengurangi pengulangan kode dan membuat fitur yang lebih lanjut seperti GUI.
3. Menjadikan program ini terbukan untuk umum agar dapat berguna untuk orang banyak.
4. Setiap membuat program atau fungsi atau prosedur, harus dicek secara mendalam, agar tidak fatal nantinya. Yang dimaksud fatal adalah kasus program terdeteksi error saat telah terintegrasi dengan program lain yang dibuat.
5. Membaca spesifikasi dengan detail di hari pertama agar dapat memikirkan sebuah solusi dengan leluasa dan jauh sebelum deadline. Contoh dalam kasus kami : Program output solusi parametrik, yang kami buat H-1 menjelang deadline (sehingga timbul beberapa masalah pada program).

5.3 Refleksi

Tim penulis merasakan pertama kali berkuliah secara luring pada semester 3 ini yang bertolak-belakang jika dibandingkan saat pertama kali masuk kuliah yang dilaksanakan secara daring. Pengalaman yang dirasakan saat melaksanakan kuliah secara offline lebih terasa jika dibandingkan saat melaksanakan kuliah daring. Manfaat yang dirasakan dari kuliah luring lebih besar dibandingkan dengan kuliah daring ,tetapi terdapat efek lain dari kuliah luring tersebut.

Hal-hal tersebut meliputi suasana baru dan lingkungan baru yang dapat menyebabkan seorang mahasiswa terbebani terutama yang pertama kali merasakan luring setelah sebelumnya daring. Hal tersebut dapat dilalui seiring dengan berjalanannya waktu. Tim penulis berharap dapat menjalani masa-masa awal kuliah ini dan dapat melaksanakan kuliah dengan lebih maksimal.

Link repository : <https://github.com/yansans/Algeo01-21110>

DAFTAR PUSTAKA

- Anton, Howard, and Chris Rorres. Elementary Linear Algebra: Applications Version. John Wiley & Sons, 2010.
- Munir, Rinaldi. 2022. IF2123 Aljabar Geometri <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/algeo22-23.htm>, diakses 29 September 2022 hingga 3 Oktober 2022 <https://stackoverflow.com/>, diakses 24 September 2022 hingga 3 Oktober 2022 <https://www.geeksforgeeks.org/>, diakses 24 September 2022 hingga 2 Oktober 2022