

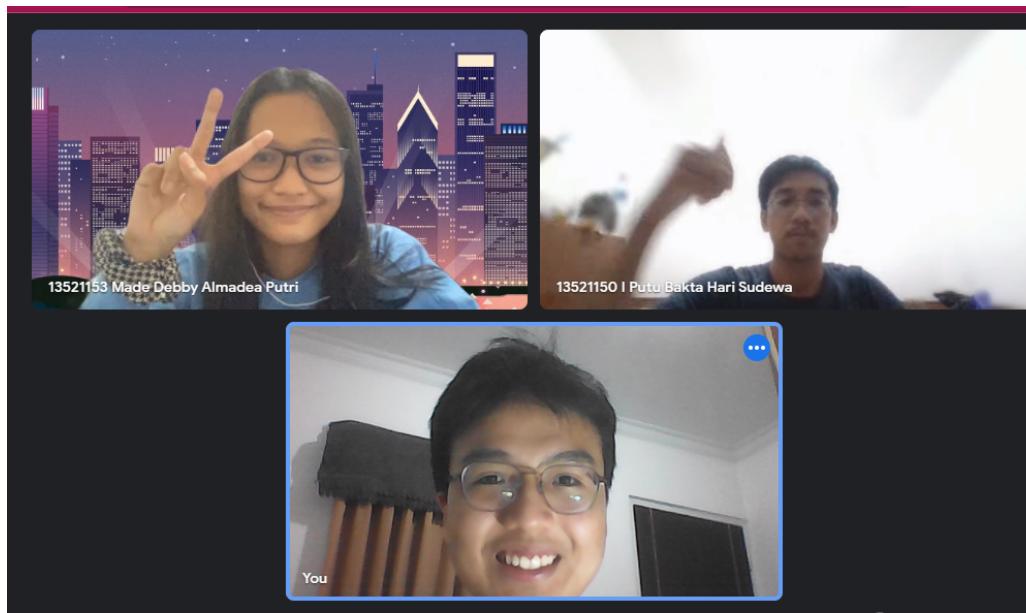
**LAPORAN TUGAS BESAR 1  
IF2123**

**ALJABAR LINIER DAN GEOMETRI SISTEM  
PERSAMAAN LINIER, DETERMINAN, DAN  
APLIKASINYA**

**SEMESTER I TAHUN 2022/2023**

**oleh:**

<b>William Nixon</b>	<b>13521123</b>
<b>I Putu Bakta Hari Sudewa</b>	<b>13521150</b>
<b>Made Debby Almadea Putri</b>	<b>13521153</b>



**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2022**

# **Daftar Isi**

<b>Daftar Isi</b>	<b>2</b>
<b>BAB 1. DESKRIPSI MASALAH</b>	<b>3</b>
I. Interpolasi Polinom	3
II. Bicubic Interpolation	5
III. Regresi Linier Berganda	6
<b>BAB 2. TEORI SINGKAT</b>	<b>7</b>
<b>BAB 3 IMPLEMENTASI PUSTAKA</b>	<b>15</b>
Class Matrix	15
Class Point	21
Class Points	22
Class Coordinate	23
Class Expression	24
Class ExpressionList	25
Class Lineq	26
Class Polinom Interpolation	27
Class Bicubic Interpolation	28
Class MultiLinearReg	29
Class ScaleImage	30
Class FileReader	31
Class FileTulis	33
Class Interface	34
Class Main	35
<b>BAB 4 EKSPERIMEN</b>	<b>36</b>
1. Temukan solusi SPL $Ax = b$ berikut:	36
2. SPL berbentuk matriks augmented	41
3. SPL berbentuk	43
4. Studi Kasus Interpolasi	46
5. Studi Kasus Interpolasi Bicubic	48
6. Studi Kasus Regresi Linier Berganda	49
7. Bonus - Perbesaran Citra (scaling image)	50

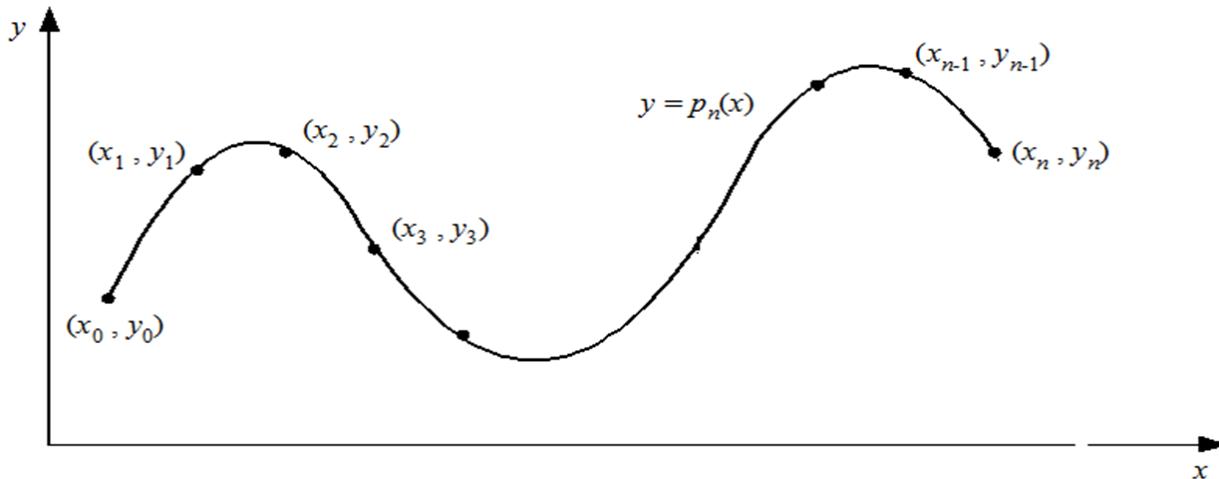
# BAB 1. DESKRIPSI MASALAH

Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. Anda sudah mempelajari berbagai metode untuk menyelesaikan SPL, termasuk menghitung determinan matriks. Sembarang SPL dapat diselesaikan dengan beberapa metode, yaitu metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan ( $x = A^{-1}b$ ), dan kaidah Cramer (khusus untuk SPL dengan  $n$  peubah dan  $n$  persamaan). Solusi sebuah SPL mungkin tidak ada, banyak (tidak berhingga), atau hanya satu (unik/tunggal).

Di dalam Tugas Besar 1 ini, anda diminta membuat satu atau lebih *library* aljabar linier dalam Bahasa Java. Library tersebut berisi fungsi-fungsi seperti eliminasi Gauss, eliminasi Gauss-Jordan, menentukan balikan matriks, menghitung determinan, kaidah Cramer (kaidah Cramer khusus untuk SPL dengan  $n$  peubah dan  $n$  persamaan). Selanjutnya, gunakan *library* tersebut di dalam program Java untuk menyelesaikan berbagai persoalan yang dimodelkan dalam bentuk SPL, menyelesaikan persoalan interpolasi, dan persoalan regresi. Penjelasan tentang interpolasi dan regresi adalah seperti di bawah ini.

## I. Interpolasi Polinom

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



Setelah polinom interpolasi  $p_n(x)$  ditemukan,  $p_n(x)$  dapat digunakan untuk menghitung perkiraan nilai  $y$  di sembarang titik di dalam selang  $[x_0, x_n]$ .

Polinom interpolasi derajat  $n$  yang menginterpolasi titik-titik  $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ . adalah berbentuk  $p_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ . Jika hanya ada dua titik,  $(x_0, y_0)$  dan  $(x_1, y_1)$ , maka polinom yang menginterpolasi kedua titik tersebut adalah  $p_1(x) = a_0 + a_1x$  yaitu berupa persamaan garis lurus. Jika tersedia tiga titik,  $(x_0, y_0), (x_1, y_1)$ , dan  $(x_2, y_2)$ , maka polinom yang menginterpolasi ketiga titik tersebut adalah  $p_2(x) = a_0 + a_1x + a_2x^2$  atau persamaan kuadrat dan kurvanya berupa parabola. Jika tersedia empat titik,  $(x_0, y_0), (x_1, y_1), (x_2, y_2)$ , dan  $(x_3, y_3)$ , polinom yang menginterpolasi keempat titik tersebut adalah  $p_3(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ , demikian seterusnya. Dengan cara yang sama kita dapat membuat polinom interpolasi berderajat  $n$  untuk  $n$  yang lebih tinggi asalkan tersedia  $(n+1)$  buah titik data. Dengan menyulihkan  $(x_i, y_i)$  ke dalam persamaan polinom  $p_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$  untuk  $i=0,1,2,\dots, n$ , akan diperoleh  $n$  buah sistem persamaan lanjar dalam  $a_0, a_1, a_2, \dots, a_n$ ,

$$a_0 + a_1x_0 + a_2x_0^2 + \dots + a_nx_0^n = y_0$$

$$a_0 + a_1x_1 + a_2x_1^2 + \dots + a_nx_1^n = y_1$$

... ...

$$a_0 + a_1x_n + a_2x_n^2 + \dots + a_nx_n^n = y_n$$

Solusi sistem persamaan lanjar ini, yaitu nilai  $a_0, a_1, \dots, a_n$ , diperoleh dengan menggunakan metode eliminasi Gauss yang sudah anda pelajari. Sebagai contoh, misalkan diberikan tiga buah titik yaitu  $(8.0, 2.0794)$ ,  $(9.0, 2.1972)$ , dan  $(9.5, 2.2513)$ . Tentukan polinom interpolasi kuadratik lalu estimasi nilai fungsi pada  $x = 9.2$ . Polinom kuadratik berbentuk  $p_2(x) = a_0 + a_1x + a_2x^2$ . Dengan menyulihkan ketiga buah titik data ke dalam polinom tersebut, diperoleh sistem persamaan lanjar yang terbentuk adalah

$$a_0 + 8.0a_1 + 64.00a_2 = 2.0794$$

$$a_0 + 9.0a_1 + 81.00a_2 = 2.1972$$

$$a_0 + 9.5a_1 + 90.25a_2 = 2.2513$$

Penyelesaian sistem persamaan dengan metode eliminasi Gauss menghasilkan  $a_0 = 0.6762$ ,  $a_1 = 0.2266$ , dan  $a_2 = -0.0064$ . Polinom interpolasi yang melalui ketiga buah titik tersebut adalah  $p_2(x) = 0.6762 + 0.2266x - 0.0064x^2$ . Dengan menggunakan polinom ini, maka nilai fungsi pada  $x = 9.2$  dapat ditaksir sebagai berikut:  $p_2(9.2) = 0.6762 + 0.2266(9.2) - 0.0064(9.2)^2 = 2.2192$ .

## II. Bicubic Interpolation

Bicubic interpolation merupakan teknik interpolasi pada data 2D umumnya digunakan dalam pembesaran citra yang merupakan pengembangan dari interpolasi linear dan cubic yang telah dipelajari pada kuliah metode numerik di aljabar geometri. Diberikan sebuah matrix awal, misal M, kita akan mencari persamaan interpolasi  $f(x,y)$  dengan pemodelan sebagai berikut:

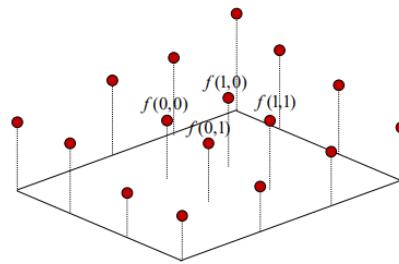
**Normalization:**  $f(0,0), f(1,0)$

$f(0,1), f(1,1)$

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

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

**Solve:**  $a_{ij}$



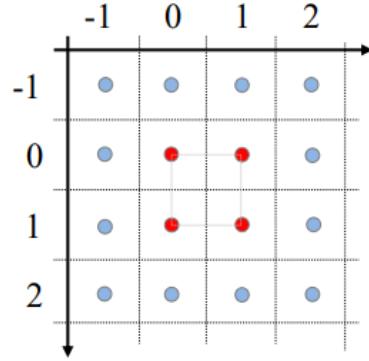
Melakukan substitusi nilai-nilai diketahui pada matriks  $4 \times 4$  tersebut ke persamaan  $f(x,y)$  akan menghasilkan sebuah matriks persamaan:

$$y = Xa$$

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

Elemen pada matrix X adalah koefisien  $a_{ij}$  yang diperoleh dari persamaan  $f(x,y)$  di atas. Sebagai contoh, elemen pada baris 4 kolom ke 10 adalah koefisien dari  $a_{12}$  dan diperoleh dari  $2^1 * (-1)^2 = 2$ , sesuai persamaan  $x^i * y^j$ .

Vektor  $a$  dapat dicari dari persamaan tersebut (menggunakan inverse), lalu vektor  $a$  digunakan sebagai nilai variabel dalam  $f(x,y)$ . Sehingga terbentuk fungsi interpolasi bicubic sesuai model. Tugas Anda adalah menentukan persamaan  $f(x,y)$  lalu melakukan interpolasi berdasarkan  $f(a,b)$  dari masukan matriks  $4 \times 4$ . Nilai masukan  $a$  dan  $b$  dalam rentang  $[0,1]$  (Referensi gambar di bawah, nilai untuk diinterpolasi dalam kotak merah).



### III. Regresi Linier Berganda

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

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

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

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

Sistem persamaan linier tersebut diselesaikan dengan menggunakan metode eliminasi Gauss.

## BAB 2. TEORI SINGKAT

### 2.1. Metode Eliminasi Gauss

Metode Eliminasi Gauss merupakan algoritma yang digunakan untuk menyelesaikan Sistem Persamaan Linier (SPL). Metode Eliminasi Gauss terdiri atas 3 langkah:

1. Nyatakan SPL dalam bentuk matriks *augmented*
2. Terapkan OBE pada matriks *augmented* hingga terbentuk matriks eselon baris.

Matriks eselon baris (*row echelon form*) merupakan matriks yang memiliki 1 utama pada setiap baris, kecuali baris yang seluruhnya nol. Matriks eselon baris berbentuk

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

Ket: \* merupakan sembarang nilai

Sifat-sifat matriks eselon baris:

- a. Jika sebuah baris tidak terdiri dari seluruhnya nol, maka bilangan tidak nol pertama di dalam baris tersebut adalah 1 (disebut 1 utama)
- b. Jika ada baris yang seluruhnya nol, maka semua baris itu dikumpulkan pada bagian bawah matriks.
- c. Di dalam dua baris berurutan yang tidak seluruhnya nol, maka 1 utama pada baris yang lebih rendah terdapat lebih jauh ke kanan daripada 1 utama pada baris yang lebih tinggi.
3. Pecahkan persamaan yang berkorespondensi pada matriks eselon baris dengan teknik penyulihan mundur (*backward substitution*)

### 2.2. Metode Eliminasi Gauss-Jordan

Metode Eliminasi Gauss-Jordan merupakan pengembangan dari Metode Eliminasi Gauss untuk mencari solusi dari SPL. Langkah penyelesaian dengan Metode Eliminasi Gauss-Jordan mirip dengan Metode Eliminasi Gauss yaitu sama-sama memanfaatkan matriks *augmented* dan OBE, tetapi pada Metode Eliminasi Gauss-Jordan, OBE dilakukan dengan tujuan akhir mendapatkan matriks eselon baris tereduksi (*reduced row echelon form*).

Matriks eselon baris tereduksi merupakan matriks yang memiliki nol-nol di bawah dan di atas 1 utama. Sifat-sifat dari matriks eselon baris tereduksi sama dengan sifat dari matriks

eselon baris ditambah sifat terakhir, yaitu setiap kolom yang memiliki 1 utama memiliki nol di kolom lain. Matriks eselon baris tereduksi berbentuk

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

Ket: \* merupakan sembarang nilai

Pada Metode ini, Tidak diperlukan lagi substitusi secara mundur untuk memperoleh nilai-nilai variabel. Nilai variabel langsung diperoleh dari matriks *augmented* akhir. Metode Eliminasi Gauss-Jordan terdiri dari 2 fase:

1. Fase maju (*forward phase*) atau fase eliminasi Gauss, yaitu fase untuk menghasilkan nilai-nilai 0 di bawah 1 utama
2. Fase mundur (*backward phase*), yaitu fase untuk menghasilkan nilai-nilai 0 di bawah 1 utama

### 2.3. Determinan

Determinan merupakan nilai-nilai yang dapat dihitung dari matriks persegi. Matriks persegi adalah matriks yang memiliki jumlah baris dan kolom yang sama. Misalkan ada matriks A berukuran  $n \times n$

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

Determinan matriks A dilambangkan dengan

$$\det(A) = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}$$

Terdapat 2 metode dalam menentukan determinan dari sebuah matriks:

1. Determinan Matriks Segitiga

Pada metode Matriks Segitiga, determinan diperoleh dengan melakukan OBE hingga matriks berbentuk matriks segitiga. Matriks segitiga dibagi lagi menjadi dua buah bentuk:

- a. Matriks segitiga atas (*upper triangular*) yaitu matriks dengan semua elemen di bawah diagonal utama adalah nol

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

maka determinan dari matriks A tersebut adalah

$$\det(A) = a_{11}a_{22}a_{33}a_{44}$$

- b. Matriks segitiga bawah (*lower triangular*) yaitu matriks dengan semua elemen di atas diagonal utama adalah nol

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

maka determinan dari matriks A tersebut adalah

$$\det(A) = a_{11}a_{22}a_{33}a_{44}$$

Dalam melakukan operasi baris elementer, terdapat beberapa aturan determinan yang perlu diperhatikan. Misalkan A adalah matriks  $n \times n$ . Matriks B adalah matriks yang diperoleh dengan memanipulasi matriks A. maka

- Jika baris pada matriks A dikalikan sebuah konstanta K dan B merupakan matriks hasil OBE tersebut, maka  $\det(B) = k \det(A)$
- Jika terdapat pertukaran baris pada matriks A dan B merupakan matriks hasil OBE tersebut, maka  $\det(B) = - \det(A)$
- Jika sebuah baris pada matriks A ditambahkan dengan matriks lainnya dan B merupakan matriks hasil OBE tersebut, maka  $\det(B) = \det(A)$

Jika selama reduksi baris ada OBE berupa perkalian baris-baris matriks dengan  $k_1, k_2, k_3, \dots, k_m$ , maka

$$\det(A) = \frac{(-1)^p a_{11}' a_{22}' a_{33}' \dots a_{mm}'}{k_1 k_2 k_3 \dots k_m}$$

## 2. Determinan dengan Ekspansi Kofaktor

Minor ( $M_{ij}$ ) suatu determinan yang dihasilkan setelah menghapus baris ke-i dan kolom ke-j dari sebuah matriks. Kofaktor adalah minor unsur beserta tanda. Kofaktor memiliki rumus.

$$K_{ij} = (-1)^{i+j} \cdot M_{ij}$$

Ekspansi kofaktor dapat dilakukan dengan mengekspansikan kolom atau baris, dengan rumus sebagai berikut:

$$\det(A) = \sum_{j=1}^n a_{ij} C_{ij} = a_{i1} C_{i1} + a_{i2} C_{i2} + \cdots + a_{in} C_{in}.$$

$$\det(A) = \sum_{i=1}^n a_{ij} C_{ij} = a_{1j} C_{1j} + a_{2j} C_{2j} + \cdots + a_{nj} C_{nj}.$$

Beberapa teorema tentang determinan:

1. Jika A mengandung sebuah baris nol atau kolom nol, maka  $\det(A) = 0$
2. Jika  $A^T$  adalah matriks transpose dari A, maka  $\det(A^T) = \det(A)$
3. Jika  $A = BC$  maka  $\det(A) = \det(B)\det(C)$
4. Sebuah matriks hanya mempunyai balikan jika dan hanya jika  $\det(A) \neq 0$
5.  $\det(A^{-1}) = 1/(\det(A))$

### 2.4. Matriks Balikan

Misalkan A adalah matriks persegi berukuran  $n \times n$ . Balikan (*inverse*) matriks A adalah  $A^{-1}$  sedemikian sehingga  $AA^{-1} = A^{-1}A = I$ . Jika A tidak memiliki balikan, maka A dikatakan matriks singular. Pada SPL  $Ax = b$ , jika A tidak mempunyai balikan, maka  $Ax = b$  tidak memiliki solusi yang tunggal (unik). Namun, jika A mempunyai balikan, maka SPL  $Ax = b$  memiliki solusi unik. Pada SPL homogen  $Ax = 0$ , SPL hanya memiliki solusi trivial, artinya solusi hanyalah  $x_1 = x_2 = x_3 = \dots = x_n = 0$ , jika A memiliki balikan. Jika A tidak memiliki balikan, maka SPL memiliki solusi non-trivial.

Dalam menentukan balikan dari suatu matriks, terdapat 2 metode:

1. Metode Eliminasi Gauss-Jordan

Misalkan A adalah matriks persegi berukuran  $n \times n$ . Balikan (inverse) matriks A adalah  $A^{-1}$  sedemikian sehingga  $AA^{-1} = A^{-1}A = I$ . Untuk matriks A yang berukuran  $n \times n$ , matriks balikan-nya, yaitu  $A^{-1}$  dicari dengan cara berikut:

$$[A | I] \sim [I | A^{-1}]$$

dengan I merupakan matriks identitas berukuran  $n \times n$ . Metode eliminasi gauss-jordan diterapkan secara simultan untuk matriks A dan I.

## 2. Metode Matriks Adjoin

Balikan dari matriks A dapat diperoleh dengan memanfaatkan matriks adjoin dengan rumus:

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

Matriks balikan dapat digunakan untuk mencari solusi dari SPL. Misalkan terdapat SPL

$$Ax = b$$

Kalikan kedua buah ruas dengan  $A^{-1}$  sehingga diperoleh

$$(A^{-1})Ax = (A^{-1})b$$

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

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

## 2.5. Matriks Kofaktor

Misalkan A merupakan matriks berukuran  $n \times n$

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

Didefinisikan:

- $M_{ij}$  sebagai minor entri  $a_{ij}$  yaitu determinan *submatrix* yang elemen-elemennya tidak berada pada baris i dan kolom j.
- $C_{ij} = (-1)^{i+j}M_{ij}$  merupakan kofaktor dari entri  $a_{ij}$

Matriks kofaktor didefinisikan sebagai matriks dengan elemen pada baris i dan kolom j merupakan  $C_{ij}$  atau kofaktor dari entri  $a_{ij}$ . Misalkan A merupakan matriks  $n \times n$  dan  $C_{ij}$  adalah kofaktor entri  $a_{ij}$ , maka matriks kofaktor dari A adalah

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

## 2.6. Matriks Adjoin

Misalkan A merupakan matriks berukuran  $n \times n$ . Matriks Adjoin merupakan transpose matriks kofaktor dari A

$$Adj(A) = \text{transpose matriks kofaktor}$$

## 2.7. Kaidah Cramer

Kaidah Cramer merupakan metode penyelesaian SPL yang memiliki banyak peubah dan banyak persamaan sama. Jika  $Ax = b$  adalah SPL yang terdiri dari n persamaan linier dengan n peubah (variable) sedemikian sehingga  $\det(A) \neq 0$ , maka SPL tersebut memiliki solusi yang unik yaitu

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

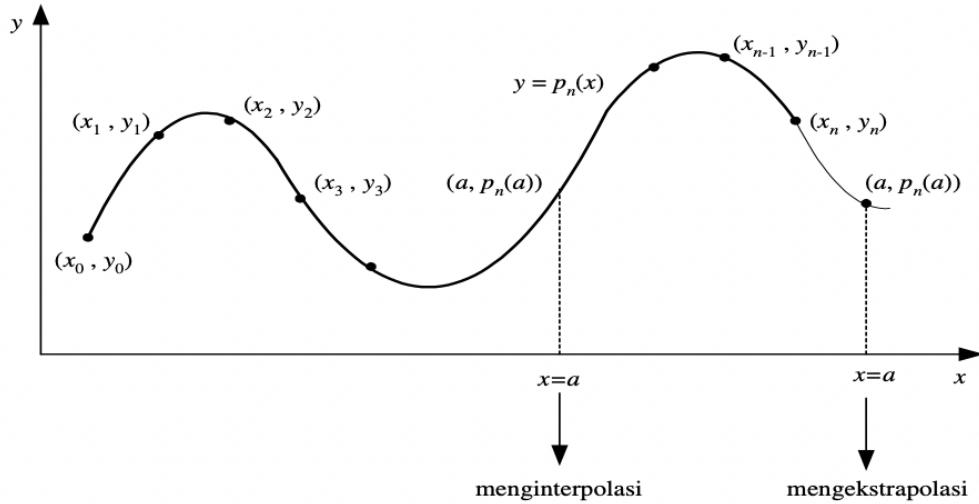
dengan  $A_j$  merupakan matriks yang diperoleh dengan mengganti entri pada kolom ke-j dari A dengan entri dari matriks

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

## 2.8. Interpolasi Polinom

Interpolasi polinom merupakan teknik interpolasi yang mengasumsikan bahwa data yang kita miliki memiliki pola data polinomial baik berderajat satu atau lebih. Misalkan 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  $i = 0, 1, 2, \dots, n$ . Nilai  $y_i$  dapat berasal dari fungsi  $f(x)$  sedemikian sehingga  $y_i = f(x_i)$ , atau  $y_i$  berasal dari nilai empiris yang diperoleh melalui percobaan atau pengamatan.  $p_n(x)$  disebut fungsi hampiran terhadap  $f(x)$ . Setelah polinom interpolasi  $p_n(x)$  ditemukan,  $p_n(x)$  dapat digunakan untuk menghitung perkiraan nilai  $y$  di  $x = a$ , yaitu  $p_n(a)$ .

Bergantung pada letaknya, nilai  $x = a$  mungkin terletak di dalam rentang titik-titik data ( $x_0 < a < x_n$ ) atau di luar rentang titik-titik data ( $a < x_0$  atau  $a > x_n$ ). Jika  $x_0 < a < x_n$  maka  $y_k = p(x_k)$  disebut nilai interpolasi (*interpolated value*). Jika  $x_0 < x_k$  atau  $x_0 < x_n$  maka  $y_k = p(x_k)$  disebut nilai ekstrapolasi (*extrapolated value*).



## 2.9. Interpolasi Bicubic

Bicubic interpolation merupakan teknik interpolasi pada data 2D umumnya digunakan dalam pembesaran citra yang merupakan pengembangan dari interpolasi linear dan cubic. Interpolasi bikubik dapat digunakan untuk menentukan persamaan  $f(x,y)$  dari masukan matriks  $4 \times 4$ . Nilai masukan  $x$  dan  $y$  dalam rentang  $[0,1]$ . Secara umum persamaan interpolasinya dapat dimodelkan sebagai berikut,

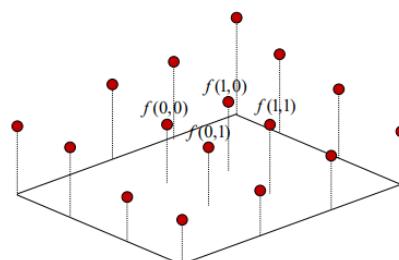
Normalization:  $f(0,0), f(1,0)$

$f(0,1), f(1,1)$

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

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

Solve:  $a_{ij}$



## 2.10. Regresi Linier Berganda

Regresi linier berganda merupakan model regresi yang melibatkan lebih dari satu variabel independen. Regresi linear berganda mengestimasi hubungan antara variabel dependen dan dua atau lebih variabel independen dengan menggunakan garis lurus. Terdapat rumus umum dari regresi linear yang bisa digunakan untuk regresi linear berganda, yaitu.

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

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

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

## BAB 3 IMPLEMENTASI PUSTAKA

### 1. Class Matrix

Kelas matrix merupakan kelas utama di dalam program ini. Kelas matriks berisikan operasi-operasi primitif matriks seperti operasi baris elementer, mencari determinan dengan berbagai metode, invers dengan berbagai metode, transpose, perkalian matriks, dan seterusnya.

#### Attributes

```
public class Matrix {  
    public int VAL_UNDEF = 99999999; // mark untuk nilai undefined  
    public double EPSILON_IMPRECISION = 0.0000000000000001; //impresisi epsilon  
    private double[][] mat;  
    private int rowEff = 0;  
    private int colEff = 0;  
    private boolean isValid; // apakah matriks valid dan layak digunakan  
    private Scanner scanner; // scanner untuk writer / mengambil input  
}
```

#### Methods

```
public class Matrix {  
    /**  
     * Konstruktor matriks  
     * @param row Dimensi baris yang ingin dimiliki matriks  
     * @param col Dimensi kolom yang ingin dimiliki matriks  
     * @param isValid  
     * @param scanner Scanner u/matriks  
     */  
    public Matrix(int row, int col, boolean isValid, Scanner scanner) {}  
  
    /**  
     * @return Index baris terakhir di matriks  
     */  
    public int getRowLastIdx() {}  
  
    /**  
     * @return Index kolom terakhir di matriks  
     */  
    public int getColLastIdx() {}  
  
    /**  
     *  
     * @return Banyak baris efektif  
     */
```

```

*/
public int getRowLength() {}

/**
*
* @return Banyak kolom efektif
*/
public int getColLength() {}

/**
* Menimpa element pada index [row][col] dengan value
*
* @param row
* @param col
* @param value
*/
public void setMatrixElement(int row, int col, double value) {}

/**
* Menimpa element pada index [row][col] dengan value
*
* @param row
* @param col
*/
public double getMatrixElement(int row, int col) {}

/**
* Rubah validitas matrix
* @param isValid
*/
public void changeMatrixValidity(boolean isValid) {}

/**
* Mengecek apakah matriks persegi
*/
public boolean isSquare() {}

/**
* Mengisi matriks sesuai input
*/
public void readMatrix() {}

/**
* Menulis matriks ke layar
*/
public void writeMatrix(int writeChoice, FileTulis fileWriter) {}

```

```

/**
 * Mengcopy this.matriks dan mengembalikan salinannya
 */
public Matrix copyMatrix() {}

/**
 * Mendapatkan matriks A dari AX = B di bentuk matriks augmented (matriks kiri)
 * @return
 */
public Matrix getMatrixAFromAugmented() {}

/**
 * Mendapatkan matriks B dari AX= B dari bentruk matriks augmented (matriks kanan)
 * @return
 */
public Matrix getMatrixBFromAugmented() {}

/**
 * Buat matriks menjadi matriks identitas
 */
public void makeIdentity() {}

/**
 * Melakukan OBE berdasarkan baris
 * @param fromRow  baris yang akan menjadi acuan OBE
 * @param toRow    baris yang akan dilakukan OBE
 * @param multiplier konstanta pengali di dalam OBE
 */
public void doRowOperation(int fromRow, int toRow, double multiplier) {}

/**
 * Melakukan OBE berdasarkan kolom
 * @param fromCol  kolom yang akan menjadi acuan OBE
 * @param toCol    kolom yang akan dilakukan OBE
 * @param multiplier konstanta pengali di dalam OBE
 */
public void doColOperation(int fromCol, int toCol, double multiplier) {}

/**
 * Melakukan operasi perkalian OBE dengan konstant
 * @param row      indeks baris yang akan dikalikan oleh konstanta
 * @param multiplier konstanta pengali di dalam OBE
 */
public void multiplyRow(int row, double multiplier) {}

```

```

/**
 * Membalik dua buah row
 * @param firstRow indeks baris yang akan ditukar dengan baris kedua
 * @param secondRow indeks baris yang akan ditukar dengan baris pertama
 */
public void swapRow(int firstRow, int secondRow) {}

/**
 * Perkalian matriks this dan m2
 * @param m2
 * @return
 */
public Matrix multiplyMatrix(Matrix m2);

/**
 * Mendapatkan indeks kolom elemen tak nol pertama suatu baris
 *
 * @param row indeks baris yang akan dicari indeks letak leading koefisiennya
 * @return indeks kolom dari leading koefisiennya, jika
 *         baris nol semua, maka menghasilkan VAL_UNDEF
 */
public int getLeadingCoeffIdx(int row) {}

/**
 * Mendapat koefisien 1 dilihat dari paling belakang (kanan)
 * Merupakan kebalikan getLeadingCoeffIdx
 */
public int getTrailingCoeffIdx(int row) {}

/**
 * Mendapatkan koordinat (indeks baris, indeks kolom) dari baris
 * [startRow..getRowLastIdx] yang memiliki
 * elemen tak nol paling kiri
 *
 * @param startRow baris awal mulai pengecekan pivot
 * @return Koordinat matriks dari pivot, elemen dengan leading koefisien paling
 *         kiri
 */
public Coordinate getPivot(int startRow) {}

/**
 * Mendapatkan pivot dari belakang (kanan)
 * Merupakan kebalikan getPivot
 */
public Coordinate getPivotReverse(int startRow) {}

```

```

/**
 * Mengubah matriks ke dalam row echelon form
 * Menggunakan eliminasi OBE.
 */
public void toREF() {}

/**
 * Mengubah matriks menjadi bentuk Eselon Baris Tereduksi
 */
public void toRREF() {}

/**
 * Mengubah matriks menjadi REF, memanipulasi matriks inverse juga. Digunakan untuk
mencari inverse menggunakan OBE
 * @param inverse
 */
public void toREFWithInverse(Matrix inverse) {}

/**
 * Mengubah matriks menjadi RREF, memanipulasi invers juga. Digunakan untuk
mencari inverse menggunakan OBE
 * @param inverse
 */
public void toRREFWithInverse(Matrix inverse) {}

/**
 * Converts to upper triangle
 * @return number of swaps done
 */
public int toUpperTriangle() {}

/**
 * Convert matrix to lower triangle,
 * @return the number of swaps done
 */
public int toLowerTriangle() {}

/**
 * Mencari determinant dengan metode segitiga atas/bawah
 * @param isUpper apakah ingin diubah menjadi segitiga atas atau bawah
 * @return nilai determinan
 */
public double getDeterminantWithTriangle(boolean isUpper) {}

/**

```

```

* Menghitung kofaktor dari submatriks
*
* @param refRow baris yang akan dihilangkan dalam perhitungan
* @param refCol kolom yang akan dihilangkan dalam perhitungan
* @return kofaktor dari submatriks
*/
public double getCofactor(int refRow, int refCol) {}

/**
 * Note: menggunakan rekursi sehingga mungkin lambat
 * @return determinan Matriks dengan metode kofaktor
*/
public double getDetWithCofactor() {}

/**
 * Mendapatkan inverse dengan metode OBE
 * @return matriks inverse
*/
public Matrix getInverse() {}

/**
 * Mendapatkan inverse dengan metode OBE, tanpa pengecekan determinan terlebih dahulu.
 * Dilakukan untuk matriks yang determinannya menuju 0 (matriks hilbert)
 * atau jika memerlukan komputasi cepat (soal bonus)
*/
public Matrix getInverseUnsafe() {}

/**
 * Mendapat matriks kofaktor
 * @return kofaktor matriks
*/
public Matrix getMatrixCofactor() {}

/**
 * @return matriks yang telah di transpose
*/
public Matrix transpose() {}

/**
 * @return adjoint dari matriks
*/
public Matrix getAdjoin() {}

/**
 * Inverse dengan adjoint
*/

```

```

* @return
*/
public Matrix getInverseWithAdjoin() {}

/**
 * Substitusi matriks untuk metode cramer
 * @return matriks yang kolomnya telah disubstitusi dengan X (AX=B)
 */
public Matrix substituteCramer(Matrix vector, int kolom) {}

```

## 2. Class Point

Kelas point merepresentasikan struktur data untuk menyimpan x dan y dari (x,y). Dua informasi ini kemudian akan menyimpan nilai poin di dalam interpolasi polinom. Kelas ini hanya helper class kecil, dengan getter dan setter.

### Attributes

```

public class Point {
    private int x;
    private int y;
}

```

### Methods

```

public class Point {
    /**
     * Constructor
     */
    public Point(double x, double y) {}

    /**
     * Untuk mengubah point yang telah terdefinisi
     */
    public void changePoint(double x, double y) {}

    /**
     * Mendapatkan nilai x (absis)
     */
    public double getAbsis() {}

    /**
     * Mendapatkan nilai y (ordinat)
     */
}

```

```
 */
public double getOrdinat() {}
}
```

### 3. Class Points

Kelas points merepresentasikan struktur data yang akan diterima oleh interpolasi polinom. Kelas points juga menyimpan sampel-sampel dari titik-titik yang akan diinterpolasi nilainya.

#### Attributes

```
public class Points {
    private double[][] points; // titik-titik point yang akan interpolasi
    private int rowEff = 0; // inisialisasi dengan 0
    private int colEff = 2; // poin hanya memiliki 2 dimensi, x dan y
    private boolean isValid;
    private Scanner scanner;
    private ArrayList<Double> samples; // sampel disimpan dalam array list
}
```

#### Methods

```
public class Points {
    /**
     * Konstruktur points
     * @param row
     * @param isValid
     * @param scanner
     */
    public Points(int row, boolean isValid, Scanner scanner) {
        this.rowEff = row;
        this.isValid = isValid;
        this.points = new double[row][2];
        this.scanner = scanner;
    }

    /**
     * Getter
     */
    public int getRowEff() {}
    public int getColEff() {}
    public ArrayList<Double> getSamples() {}
```

```

public double[] getPoint(int row) {}

/**
 * Membaca points dari keyboard sesuai banyaknya row efektif.
 */
public void readPoints() {}

/**
 * Setter point
 */
public void setPointsElement(int row, int col, double value) {}

/**
 * Untuk keperluan testing, memprint isi dari points dan juga sampel yang mungkin
 dsiimpan
 */
public void writePoints() {}
}

```

## 4. Class Coordinate

Kelas koordinat merupakan kelas helper kecil yang dipakai untuk merepresentasikan titik pivot di dalam operasi baris elementer. Kelas ini hanya berisi getter,setter, dan juga baris dan kolom.

### Attributes

```

public class Coordinate {
    private int row;
    private int col;
}

```

### Methods

```

public class Coordinate {
    // Konstruktor
    public Coordinate(int row, int col) {}
    // Getter
    public int getRow() {}
    public int getCol() {}
    // Setter
    public void setRow(int newRow) {}
    public void setCol(int newCol) {}
}

```

```
}
```

## 5. Class Expression

Kelas expression merupakan representasi dari variabel. Kelas ini dipergunakan untuk membedakan angka dan variabel.

### Attributes

```
private static final double EPSILON_IMPRECISION = 0.0001;  
private boolean isNumber;  
private double number;  
private String var;
```

### Methods

```
public class Expression {  
  
    // Konstruktor, menerima boolean apakah expresi sebuah angka, konstanta number, dan  
    // string yang digunakan sebagai variabel.  
    public Expression(boolean isNumber, double number, String var) {}  
  
    // Getter dan setter  
    public void setNumber(double number) {}  
    public void setVar(String var) {}  
    public boolean getIsNumber() {}  
    public double getNumber() {}  
    public String getVar() {}  
  
    // Mengalikan number sebuah ekspresi dengan konstanta  
    public void multiplyExpression(double constant) {}  
  
    /**  
     * Menyederhanakan ekspresi jika tipe variabel sama  
     * @param b  
     * @return  
     */  
    public boolean addExpression(Expression b) {}  
  
    /**  
     * Display ekspresi, jika - maka dengan tanda kurung, misalnya (-2)  
     * @return representasi string dari expression  
     */
```

```
public String getDisplayExpression() {}
```

## 6. Class ExpressionList

Kelas expression list merupakan list dari expresi. Kelas ini dibuat untuk menyederhanakan penyimpanan solusi SPL di dalam bentuk parametrik yang mungkin dapat mengandung beberapa variabel/ekspresi. Dengan adanya kelas ini, maka melakukan operasi dan mengeprint nilai hasil substitusi balik di dalam penyelesaian SPL dapat dilakukan dengan mudah.

### Attributes

```
private static final double EPSILON_IMPRECISION = 0.00001;
public ArrayList<Expression> variables = new ArrayList<Expression>();
```

### Methods

```
public class ExpressionList {
    // Konstruktor
    public ExpressionList() {}
    // Getter
    public Expression getVariable(int index) {}

    // Menambahkan sebuah ekspresi ke dalam array of expression.
    public void addExpression(boolean isNumber, double number, String var) {}

    /**
     * Mengalikan semua variabel dalam list substituent dan memasukkannya ke ekspresi ini
     * Mirip operasi OBE, tapi untuk parametrik dan substitusi balik gauss/gauss jordan
     * @param multiplier
     * @param substituent
     */
    public void addAndSubstitute(double multiplier, ExpressionList substituent) {}

    /**
     * Simplifikasi term yang ada di dalam expr list
     * Misalnya a + a = 2a
     */
    public void simplify() {}

    /**
```

```

* Mendapatkan representasi string yang akan di print dari suatu expr list
* @return
*/
public String getStringPrint() { }

}

```

## 7. Class Lineq

Kelas yang bertanggung jawab untuk menyelesaikan SPL dengan berbagai metode. Memanfaatkan berbagai kelas helper yang telah didefinisikan sebelumnya, seperti expressionList, Coordinate, Point, dan lain-lain. Segala penyelesaian SPL dapat dilakukan dengan memanfaatkan kelas ini.

### Attributes

```
private static final double EPSILON_IMPRECISION = 0.00001; // impresisi epsilon
```

### Methods

```

public class Lineq {
    /**
     * Substitusi balik REF / RREF hasil Gauss / Gauss Jordan.
     * Hashmap[1] mengandung expr list untuk x1, hashmap[2] x2, dst.
     * @param m
     * @return
    */
    private HashMap<Integer, ExpressionList> getSolution(Matrix m) {}

    /**
     * Mendisplay solution dari hasil substitusi dengan mengiterasi hashmap
     * @param expression
     * @param writeChoice
     * @param fileWriter
    */
    public void displaySolution(
        HashMap<Integer, ExpressionList> expression,
        int writeChoice,
        FileTulis fileWriter) {}

    /**
     * Generator variabel parametrik
    */
}

```

```

    * @param charCount
    * @return
    */
    public String generateParametricVariable(int charCount) {}

    // Melakukan operasi Gauss dan Gauss-Jordan pada matriks m dan mengembalikan
    solusinya dalam bentuk hashmap of expression
    public HashMap<Integer, ExpressionList> Gauss(Matrix m) {}
    public HashMap<Integer, ExpressionList> GaussJordan(Matrix m) {}

    // Melakukan operasi Cramer pada matriks m, dan menuliskan hasil operasi tersebut ke
    layar / file sesuai kemauan pengguna.
    public void doCramer(Matrix m, int writeChoice, FileTulis fileWriter) {}

    // Versi unsafe dari cramer, melakukan operasi tanpa mengecek determinan terlebih
    dahulu.
    public void doCramerUnsafe(Matrix m, int writeChoice, FileTulis fileWriter) {}

    // Melakukan operasi inverse pada matriks m, dan menuliskan hasil operasi tersebut ke
    layar / file sesuai kemauan pengguna.
    public void doInverse(Matrix m, int writeChoice, FileTulis fileWriter) {}

    // Versi unsafe dari inverse, melakukan operasi tanpa mengecek determinan terlebih
    dahulu.
    public void doInverseUnsafe(Matrix m, int writeChoice, FileTulis fileWriter) {}

}

```

## 8. Class Polinom Interpolation

Kelas yang bertanggung jawab untuk menyelesaikan persoalan interpolasi polinom. Memanfaatkan library Lineq untuk menyelesaikan SPL yang diperolehnya dan Points sebagai tempat menyimpan data yang akan diproses oleh polinom interpolation.

### Attributes

```

public class PolinomInterpolation {
    private int degree; // Derajat polinom tersebut
    private Points points; // Points yang akan dipakai sebagai dataset
    private Scanner scanner;
    private double max = -9999999; // Untuk mengetahui batas bawah dan atas [x,y]
    interpolasi
    private double min = 9999999;
    private Matrix augmented; // Matriks augmented hasil olah poitns
}

```

```
    private HashMap<Integer, Double> solusi; // Solusi polinom interpolasi, hashmap[0]
    mengandung nilai x1, hashmap[1] x2, dst.
}
```

## Methods

```
public class PolinomInterpolation {

    // Konstruktor
    public PolinomInterpolation(Scanner scanner) {}

    // Setter
    public void setPoints(Points read) {}
    public void setAugmented() {}

    // Mencari penyelesaian dari matriks augmented dengan cara SPL
    public void solve(int inputChoice, int writeChoice, FileTulis fileWriter) {}

    // Melakukan interpolasi berdasarkan input file
    private void interpolateFile(int writeChoice, FileTulis fileWriter) {}

    // Melakukan interpolasi berdasarkan input keyboard
    private void interpolateKeyboard(int writeChoice, FileTulis fileWriter) {}

    // Mengubah hashmap yang awalnya memetakan integer ke ekspresi ke double.
    private HashMap<Integer, Double> turnExpressionToDouble(HashMap<Integer,
    ExpressionList> expresi) {}

    // Menampilkan fungsi f(x) ke layar dan ke file sesuai pilihan pengguna
    private void displayAsFunction(int writeChoice, FileTulis fileWriter) {}
    // Mendisplay hasil f(x) dan evaluasi titik-titik sampel ke layar dan juga ke file, sesuai
    pilihan pengguna
    private void displayAndWriteEval(double input, double hasil, int writeChoice, FileTulis
    fileWriter) {}

    // Mengevaluasi fungsi f(x) dengan suatu nilai x
    private double evaluateFunction(double x) {}}
```

## 9. Class Bicubic Interpolation

Kelas yang bertanggung jawab untuk menyelesaikan permasalah interpolasi bicubic. Memanfaatkan library Matrix untuk membangun, mengakses, dan melakukan invers terhadap elemen-elemen awalnya dan menghasilkan suatu nilai hasil interpolasi bicubic.

## Attributes

Class ini tidak memiliki attribute.

## Methods

```
/**  
 * Mengembalikan solusi dari matriks interpolasi (m) berukuran  
 * 4x4 pada suatu titik p(x, y) dengan rentang nilai x dan y  
 * [0,1].  
 *  
 * @param Matrix m -> matriks interpolasi  
 * @param Point p -> titik yang ingin dicari nilai interpolasinya  
 * @param Scanner -> scanner input/output  
 */  
public double bicubic(Matrix m, Point p, Scanner scanner);
```

## 10. Class MultiLinearReg

Kelas yang mengandung penyelesaian masalah regresi linear berganda. Memanfaatkan library Lineq untuk menyelesaikan SPL yang diperoleh dari *Normal Estimation Equation*, library Matrix untuk membangun dan mengakses elemen-elemen, serta library ExpressionList untuk mendapatkan estimasi nilai dari sampel. Input data dari Interface dapat melalui dua metode:

a. Input melalui Keyboard

Input melalui keyboard dilakukan dengan menuliskan matriks *augmented*  $[x_k \mid y]$ , setelah itu dapat memilih untuk melakukan estimasi nilai atau tidak. Jika iya, maka input berupa banyak sampel/data yang diestimasi dan  $x_k$  dari masing-masing sampel/data

b. Input melalui File

Input melalui File memiliki format data yang akan diregresi berbentuk matriks *augmented*  $[x_k \mid y]$  diikuti oleh data-data yang akan diestimasi nilainya (TANPA new line diantara keduanya)

## Attributes

Class ini tidak memiliki attribute.

## Methods

```
/**  
 * Melakukan multiple linear regression untuk mencari estimasi nilai dari  
 * keyboard, menerima input banyak sampel yang akan diestimasi dan nilai-nilai  
 * yang akan diestimasi.  
 *  
 * Menggunakan fungsi getEstimatedValue untuk mencari estimasi nilai dari suatu
```

```

* data
*
* @param data Data yang akan diregresi dalam bentuk matriks augmented [A
* | b] dengan A merupakan nilai-nilai xk dan b merupakan
* nilai y
* @param scanner Scanner untuk matriks dan estimasi nilai
* @param writeChoice Bentuk output. 1 jika ingin ditulis pada FILE
* @param fileWriter FileTulis untuk menuliskan output pada FILE
*/
public void doMultiLinearRegKey(Matrix data, Scanner scanner, int writeChoice,
FileTulis fileWriter);

/**
* Melakukan multiple linear regression untuk mencari estimasi nilai dari data
* yang dibaca dari File
*
* Menggunakan fungsi getEstimatedValue untuk mencari estimasi nilai dari suatu
* data
*
* @param data Data yang akan diregresi dalam bentuk matriks augmented [A
* | b] dengan A merupakan nilai-nilai xk dan b merupakan
* nilai y
* @param estimates Data yang akan diestimasi nilainya. Data bisa lebih dari 1
* @param scanner Scanner untuk matriks dan estimasi nilai
* @param writeChoice Bentuk output. 1 jika ingin ditulis pada FILE
* @param fileWriter FileTulis untuk menuliskan output pada FILE
*/
public void doMultiLinearRegFile(Matrix data, Matrix estimates, Scanner scanner,
int writeChoice, FileTulis fileWriter);

/**
* Mendapatkan solusi dari SPL Normal Estimated Equation untuk Regresi Linier
* Berganda. Menggunakan fungsi getNEE untuk mendapatkan Normal Estimated
* Equation
*
* @param data Data yang akan diregresi dalam bentuk matriks augmented [A
* | b] dengan A merupakan nilai-nilai xk dan b merupakan
* nilai y
* @param scanner Scanner untuk matriks
* @return Solusi dari SPL Normal Estimated Equation untuk Regresi Linier
* Berganda
*/
private HashMap<Integer, ExpressionList> getSolution(Matrix data, Scanner scanner);

/**
* Menuliskan persamaan regresi dari solusi yang telah diperoleh

```

```

/*
 * @param writeChoice Bentuk output. 1 jika ingin ditulis pada FILE
 * @param fileTulis FileTulis untuk menuliskan output pada FILE
 * @param solution Solusi dari SPL Normal Estimated Equation
 */
private void writeRegEquation(int writeChoice, FileTulis fileWriter,
HashMap<Integer, ExpressionList> solution);

/**
 * Menuliskan nilai estimasi sesuai dengan format ke layar/file
 *
 * @param writeChoice Bentuk output. 1 jika ingin ditulis pada FILE
 * @param fileTulis FileTulis untuk menuliskan output pada FILE
 * @param refData Data (xk) yang diestimasi nilainya
 * @param estimatedValue Estimasi nilai dari data tersebut
 */
private void writeEstimate(int writeChoice, FileTulis fileWriter, double[] refData,
double estimatedValue)

/**
 * Mendapatkan sum of product yaitu sigma dari xpi * xqi
 *
 * @param data Data yang akan diregresi dalam bentuk matriks augmented
 * [A | b] dengan A merupakan nilai-nilai xk dan b merupakan
 * nilai y
 * @param firstVarIdx Variabel pertama p [0..k]. Jika p = 0, maka xpi = 1
 * @param secondVarIdx Variabel kedua q [0..k]. Jika q = 0, maka xqi = 1
 * @return Sigma dari xpi * xqi
 */
private double getSOP(Matrix data, int firstVarIdx, int secondVarIdx);

/**
 * Mendapatkan Normal Estimation Equation untuk Multiple Linear Regression dari
 * data
 *
 * @param data Data yang akan dilakukan regresi
 * @param scanner Scanner untuk matriks
 * @return Matriks Normal Estimation Equation untuk Multiple Linear Regression
 */
private Matrix getNEE(Matrix data, Scanner scanner);

/**
 * Mendapatkan estimasi nilai dari refData
 *
 * @param solution Solusi dari multi linear regression yang
 * merupakan Matrix eselon baris tereduksi dari

```

```

*           Normal Estimation Equation
* @param refData data yang akan dicari estimasi nilainya. Prekondisi: banyak
*               peubah (xk) sama dengan banyak peubah (xk) pada data
* @param scanner` Scanner untuk matrix
* @return Estimasi nilai dari refData
*/
private double getEstimatedValue(HashMap<Integer, ExpressionList> solution,
double[] refData, Scanner scanner);

```

## 11. Class ScaleImage

Proses perbesaran suatu citra dilakukan di kelas ini. Dengan memanfaatkan library Matrix, Point, dan Bicubic, nantinya citra awal yang diinput akan diolah dengan mengambil, menambahkan, serta menginterpolasi pixel-pixelnya untuk memperbesar citra tersebut.

### Attributes

Class ini tidak memiliki attribute.

### Methods

```

/**
* Mengembalikan nilai rgb dari img pada titik (x, y).
*
* @param BufferedImage img -> citra awal yang ditinjau
* @param int x -> koordinat arah x dari pixel
* @param int y -> koordinat arah y dari pixel
*
* @return Color -> pixel rgb pada titik x, y
*/
private Color getColorRGB(BufferedImage img, int x, int y);

/**
* Mengembalikan suatu titik (x', y') yang akan digunakan untuk
* interpolasi sesuai segmen yang ditentukan.
*
* @param int x -> koordinat arah x dari pixel
* @param int y -> koordinat arah y dari pixel
*
* @return Point -> point yang akan dinterpolasi
*/
private Point getPoint(int x, int y);

/**
* Melakukan proses perbesaran citra pada file yang dimasukkan.

```

```

* Dimulai dengan proses penambahan pixel (padding) pada citra
* awal, lalu dilakukan interpolasi bicubic dan diakhiri dengan
* pembuatan citra baru yang telah diperbesar 2x dari ukuran
* awal.
*
* @param String file -> nama file + format citra yang akan
* dilakukan perbesaran
*/
public void scaleImage(String file);

```

## 12. Class FileReader

Bertanggung jawab di dalam membaca file sebagai sumber input selain keyboard.  
Memanfaatkan library IO bawaan dari Java seperti File.

### Attributes

```

/**
 * Nama + format dari file yang dibaca
 */
private String fileName;

/**
 * Path dari file yang akan dibaca
 */
private String cwd;

```

### Methods

```

/**
 * Melakukan pengecekan terhadap input file. Jika input file valid maka akan diekstrak
 * nama + format dari file tersebut
 *
 * @param Scanner scanner -> input file
 *
 * @return boolean -> bernilai true jika file valid, false jika tidak
 */
public boolean setFileName(Scanner scanner);

/**
 * Mengembalikan nama + format file
 *
 * @return String -> nama + format file
 */

```

```

public String getFileName();

< /**
 * Menambahkan nama + format file yang telah diekstrak ke dalam cwd
 *
 * @param String fileName -> nama + format file yang telah diekstrak
 */
private void changeCWD(String fileName);

< /**
 * Mengembalikan matriks hasil pembacaan dari file. Dimulai dengan membaca file baris
 * per baris serta melakukan validasi jika terdapat elemen yang tidak valid
 *
 * @return Matrix -> matriks yang valid dari file
 */
public Matrix readMatrix();

< /**
 * Mengembalikan point-point hasil pembacaan dari file. Dimulai dengan membaca file
 * baris per
 * baris serta melakukan validasi jika terdapat elemen yang tidak valid
 *
 * @return Points -> points yang valid dari file
 */
public Points readPointsFromFile();

< /**
 * Mengembalikan ArrayList<Matrix> hasil pembacaan dari file untuk proses interpolasi
 * bicubic. ArrayList<Matrix> ini hanya terdiri dari 2 elemen, yaitu
 * Elemen pertama (index 0) -> matriks 4x4 yang akan diinterpolasi
 * Elemen kedua (index 1) -> titik (x, y) yang menjadi titik target interpolasi
 * Juga dilakukan validasi untuk mengecek apakah format elemen dari file sudah sesuai
 * dengan kriteria yang ditentukan
 *
 * @return ArrayList<Matrix> -> berisi Matrix dan Point yang diperlukan untuk interpolasi
 * bicubic
 */
public ArrayList<Matrix> readBicubicFromFile();

< /**
 * Membaca data untuk Regresi Linier Berganda dari File dengan format data yang
 * diregresi diikuti oleh data yang akan diestimasi nilainya
 *
 * @return Array dari Matrix dengan index 0 data yang akan diregresi dan index 1
 *         data yang akan diestimasi nilainya
 */

```

```
public ArrayList<Matrix> readMLRFromFile();
```

## 13. Class FileTulis

Kelas yang bertanggung jawab untuk menuliskan string ke suatu file. Memanfaatkan library IO bawaan dari Java yaitu FileWriter.

### Attributes

```
private FileWriter fileWriter;  
// Current direktori dari user  
private String cwd = System.getProperty("user.dir");
```

### Methods

```
public class FileTulis {  
    // mengappend cwd dengan /test/output/{filename}  
    private void changeCWD(String fileName) {}  
  
    // Menginisialisasikan sebuah file dengan nama filename di folder  
    // /test/output/{filename}  
    public FileTulis(String filename) {}  
  
    // Menuliskan string ke dalam file yang telah dibuka  
    public void writeFile(String string) {}  
    // Menutup file yang telah selesai ditulis  
    public void closeFile() {}  
}
```

## 14. Class Interface

Kelas yang bertanggung jawab dengan segala tampilan di dalam program. Merupakan driver file yang sesungguhnya, di mana flow program ditentukan oleh fungsi start().

### Attributes

```
public class Interface {  
    private Scanner scanner;  
    private FileTulis fileWriter;  
}
```

## Methods

```
public class Interface {  
    // Menampilkan pilihan untuk main menu, determinan menu, invers menu  
    private int mainMenu() {}  
    private int detMenu() {}  
    private int inversMenu() {}  
    private int splMenu() {}  
  
    // Menampilkan menu yang menanyakan user apakah mereka ingin menuliskan output  
    ke file  
    private int writeMenu() {}  
  
    // Menu yang menanyakan user apakah ingin menginput dari keyboard / file  
    private int inputChoiceMenu() {}  
  
    // Menu yang menanyakan user tentang nama file yang akan ditulis  
    private String writeNameMenu() {}  
  
    // Menu yang berisi flow dari program  
    public void start(Scanner scanner) throws IOException {}  
}
```

## 15. Class Main

Berfungsi sebagai tempat startnya program dan juga agar file .jar dapat dieksekusi. Driver utama terdapat di kelas interface, yaitu pada method start(). Fungsi main akan memanggil metode ini untuk menjalankan program.

```
public class Main {  
    // Main class, memanggil driver interface dan menjalankan program.  
    public static void main(String[] args) throws FileNotFoundException, IOException {}  
}
```

## BAB 4 EKSPERIMENT

1. Temukan solusi SPL  $Ax = b$  berikut:

a.	Solusi
	Metode Gauss <b>SPL tidak memiliki solusi.</b>
$A = \begin{bmatrix} 1 & 1 & -1 & -1 \\ 2 & 5 & -7 & -5 \\ 2 & -1 & 1 & 3 \\ 5 & 2 & -4 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -2 \\ 4 \\ 6 \end{bmatrix}$	Metode Gauss-Jordan <b>SPL tidak memiliki solusi.</b>
	Metode Cramer Determinan matriks 0 sehingga tidak dapat diperoleh solusinya lewat metode Cramer.
	Metode Invers Matriks tidak mempunyai invers! Matrix tidak punya invers (singular) sehingga tidak bisa diperoleh solusinya lewat metode invers.

b.	Solusi
	Metode Gauss $A = \begin{bmatrix} 1 & -1 & 0 & 0 & 1 \\ 1 & 1 & 0 & -3 & 0 \\ 2 & -1 & 0 & 1 & -1 \\ -1 & 2 & 0 & -2 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 6 \\ 5 \\ -1 \end{bmatrix}$ <b>x1 : 3.00 + a x2 : 2.00a x3 : b x4 : (-1.00) + a x5 : a</b>
	Metode Gauss-Jordan

```

x1 : 3.00 + a
x2 : 2.00a
x3 : b
x4 : (-1.00) + a
x5 : a

```

#### Metode Cramer

Determinan matriks u/ metode cramer tidak terdefinisi karena bukan persegi.

#### Metode Invers

Matriks tidak mempunyai invers!  
Matrix tidak punya invers (singular) sehingga tidak bisa diperoleh solusinya lewat metode invers.

c.	Solusi
	<p>Metode Gauss</p> $A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$ <pre> x1 : b x2 : 1.00 + (-a) x3 : c x4 : (-2.00) + (-a) x5 : 1.00 + a x6 : a </pre>
	<p>Metode Gauss-Jordan</p>

```

x1 : b
x2 : 1.00 + (-a)
x3 : c
x4 : (-2.00) + (-a)
x5 : 1.00 + a
x6 : a

```

### Metode Cramer

Determinan matriks u/ metode cramer tidak terdefinisi karena bukan persegi.

### Metode Invers

Matriks tidak mempunyai invers!  
Matrix tidak punya invers (singular) sehingga tidak bisa diperoleh solusinya lewat metode invers.

d.

Matriks hilbert n = 6 dalam bentuk desimal

```

1.0000000000000000 0.5000000000000000 0.3333333333333333 0.2500000000000000 0.2000000000000000 0.1666666666666667 1
0.5000000000000000 0.3333333333333333 0.2500000000000000 0.2000000000000000 0.1666666666666667 0.1428571428571428 0
0.3333333333333333 0.2500000000000000 0.2000000000000000 0.1666666666666667 0.1428571428571428 0.1250000000000000 0
0.2500000000000000 0.2000000000000000 0.1666666666666667 0.1428571428571428 0.1250000000000000 0.1111111111111111 0
0.2000000000000000 0.1666666666666667 0.1428571428571428 0.1250000000000000 0.1111111111111111 0.1000000000000000 0
0.1666666666666667 0.1428571428571428 0.1250000000000000 0.1111111111111111 0.1000000000000000 0.0909090909090909 0

```

Solusi

Metode Gauss

Metode Gauss-Jordan

```

x1 : 36.00
x2 : (-630.00)
x3 : 3360.00
x4 : (-7560.00)
x5 : 7560.00
x6 : (-2772.00)

```

```

x1 : 36.00
x2 : (-630.00)
x3 : 3360.00
x4 : (-7560.00)
x5 : 7560.00
x6 : (-2772.00)

```

Metode Cramer  
(SAFE)

Determinan matriks 0 sehingga tidak dapat diperoleh solusinya lewat metode Cramer.

Metode Invers  
(SAFE)

Matriks tidak mempunyai invers!  
Matrix tidak punya invers (singular) sehingga tidak bisa diperoleh solusinya lewat metode invers.

(UNSAFE)

```

x1 : 36.00
x2 : (-630.00)
x3 : 3360.00
x4 : (-7560.00)
x5 : 7560.00
x6 : (-2772.00)

```

e.

Matriks hilbert n = 10 dalam bentuk desimal

```

1.000000000000000 0.500000000000000 0.333333333333333 0.250000000000000 0.200000000000000 0.166666666666667 0.1428571428571428 0.125000000000000 0.111111111111111 0.100000000000000 1
0.500000000000000 0.333333333333333 0.250000000000000 0.200000000000000 0.166666666666667 0.1428571428571428 0.125000000000000 0.111111111111111 0.100000000000000 0.0909090909090909 0
0.333333333333333 0.250000000000000 0.200000000000000 0.166666666666667 0.1428571428571428 0.125000000000000 0.111111111111111 0.100000000000000 0.0909090909090909 0.0833333333333333
0.250000000000000 0.200000000000000 0.166666666666667 0.1428571428571428 0.125000000000000 0.111111111111111 0.100000000000000 0.0909090909090909 0.0833333333333333
0.200000000000000 0.166666666666667 0.1428571428571428 0.125000000000000 0.111111111111111 0.100000000000000 0.0909090909090909 0.0833333333333333
0.166666666666667 0.1428571428571428 0.125000000000000 0.111111111111111 0.100000000000000 0.0909090909090909 0.0833333333333333
0.1428571428571428 0.125000000000000 0.111111111111111 0.100000000000000 0.0909090909090909 0.0833333333333333
0.125000000000000 0.111111111111111 0.100000000000000 0.0909090909090909 0.0833333333333333
0.111111111111111 0.100000000000000 0.0909090909090909 0.0833333333333333
0.100000000000000 0.0909090909090909 0.0833333333333333

```

Solusi	
Metode Gauss	Metode Gauss-Jordan
<pre> x1 : 99.99 x2 : (-4949.18) x3 : 79182.48 x4 : (-600439.66) x5 : 2521751.05 x6 : (-6304176.72) x7 : 9606103.37 x8 : (-8748210.27) x9 : 4374015.62 x10 : (-923386.67) </pre>	<pre> x1 : 99.99 x2 : (-4949.18) x3 : 79182.48 x4 : (-600439.66) x5 : 2521751.05 x6 : (-6304176.72) x7 : 9606103.37 x8 : (-8748210.27) x9 : 4374015.62 x10 : (-923386.67) </pre>
Metode Cramer  (SAFE)  Determinan matriks 0 sehingga tidak dapat diperoleh solusinya lewat metode Cramer.	Metode Invers  (SAFE)  Matriks tidak mempunyai invers! Matrix tidak punya invers (singular) sehingga tidak bisa diperoleh solusinya lewat metode invers.  (UNSAFE)

	x1 : 99.99
	x2 : -4949.18
	x3 : 79182.48
	x4 : -600439.66
	x5 : 2521751.05
	x6 : -6304176.72
	x7 : 9606103.37
	x8 : -8748210.27
	x9 : 4374015.62
	x10 : -923386.67

Pada eksperimen 1, diberikan beberapa SPL yang tak memiliki solusi (a), solusi parametrik (b), (c) dan memiliki solusi tunggal (d), (e). Pada test case yang tak memiliki solusi, keempat metode menunjukkan kesimpulan yang benar. Metode Gauss dan Gauss Jordan melakukan ini dengan cara membandingkan baris terakhir pada matriks REF dan melihat apabila ada ketidakcocokan. Metode determinan dan invers melakukan ini dengan cara melihat determinan, apakah menuju 0 atau tidak.

Pada test case bersolusi parametrik, keempat metode menunjukkan kesimpulan yang benar. Metode Gauss dan Gauss Jordan melakukannya dengan cara mensubstitusi balik hasil matriks REF ataupun RREF. Metode determinan dan invers tidak dapat memproses karena matriks bukan berbentuk persegi ataupun determinannya menuju 0.

Pada test case bersolusi tunggal, terdapat hasil yang cukup variatif. Matriks yang digunakan menggunakan matriks hilbert, yang memiliki determinan yang menuju 0, semakin besar n yang dipilih. Oleh karena itu, dibuat metode invers yang *unsafe*. Metode ini tidak melakukan validasi nilai determinan sebelum melakukan operasi karena nilai determinan matriks hilbert yang menuju 0. Metode cramer tidak dibuat metode unsafennya karena pembagian dengan nilai determinan menuju 0 pasti tidak akan memberikan hasil yang baik. Matriks hilbert dibangkitkan dengan menggunakan kode generator agar dapat mendapatkan presisi floating point yang maksimal. Generator kode yang digunakan terlampir di lampiran.

Pada nilai n=6, keempat metode menunjukkan hasil yang serupa untuk x. Akan tetapi pada nilai n=10, keempat metode mulai menunjukkan perbedaan hasil. Hal ini dikarenakan adanya *floating point imprecision* pada perhitungan matriks hilbert dengan n = 10, yang determinannya menuju 0. Padahal, penulis telah menggunakan variabel double yang presisi sampai 15 angka di belakang koma. Untuk mendapatkan hasil yang lebih akurat, dapat digunakan konstanta EPSILON yang lebih kecil, ataupun menggunakan class Big Decimal pada Java.

## 2. SPL berbentuk matriks *augmented*

a.	Solusi
----	--------

Metode Gauss

$x_1 : (-1.00) + b$   
 $x_2 : 2.00c$   
 $x_3 : c$   
 $x_4 : b$

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

Metode Gauss-Jordan

$x_1 : (-1.00) + b$   
 $x_2 : 2.00c$   
 $x_3 : c$   
 $x_4 : b$

Metode Cramer

Determinan matriks 0 sehingga tidak dapat diperoleh solusinya lewat metode Cramer.

Metode Invers

Matriks tidak mempunyai invers!  
Matrix tidak punya invers (singular) sehingga tidak bisa diperoleh solusinya lewat metode invers.

b.

Solusi

	Metode Gauss
	<pre>x1 : 0.00 x2 : 2.00 x3 : 1.00 x4 : 1.00</pre>
$\begin{bmatrix} 2 & 0 & 8 & 0 & 8 \\ 0 & 1 & 0 & 4 & 6 \\ -4 & 0 & 6 & 0 & 6 \\ 0 & -2 & 0 & 3 & -1 \\ 2 & 0 & -4 & 0 & -4 \\ 0 & 1 & 0 & -2 & 0 \end{bmatrix}$	Metode Gauss-Jordan
	<pre>x1 : 0.00 x2 : 2.00 x3 : 1.00 x4 : 1.00</pre>
	Metode Cramer
	<p>Determinan matriks u/ metode cramer tidak terdefinisi karena bukan persegi.</p>
	Metode Invers
	<p>Matriks tidak mempunyai invers! Matrix tidak punya invers (singular) sehingga tidak bisa diperoleh solusinya lewat metode invers.</p>

### 3. SPL berbentuk

a.	Solusi
a. $8x_1 + x_2 + 3x_3 + 2x_4 = 0$ $2x_1 + 9x_2 - x_3 - 2x_4 = 1$ $x_1 + 3x_2 + 2x_3 - x_4 = 2$ $x_1 + 6x_3 + 4x_4 = 3$	Metode Gauss

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

Metode Gauss-Jordan

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

Metode Cramer

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

Metode Invers

**x1 : (-0.22)**  
**x2 : 0.18**  
**x3 : 0.71**  
**x4 : (-0.26)**

b.	Solusi
	Metode Gauss
	<b>SPL tidak memiliki solusi.</b>
b.	Metode Gauss-Jordan
	<b>SPL tidak memiliki solusi.</b>
	Metode Cramer
	Determinan matriks u/ metode cramer tidak terdefinisi karena bukan persegi.
	Metode Invers
	Matriks tidak mempunyai invers! Matrix tidak punya invers (singular) sehingga tidak bisa diperoleh solusinya lewat metode invers.

Analisis eksperimen 2 & 3 mirip dengan eksperimen 1. Diberikan beberapa SPL yang tak memiliki solusi, solusi parametrik, dan memiliki solusi tunggal. Secara umum, keempat metode menunjukkan kesimpulan yang benar. Pada kasus 2(b), walaupun SPL memiliki solusi tunggal, akan tetapi metode invers dan cramer tetap tidak dapat digunakan untuk menyelesaikannya. Hal ini karena matriks A ( $AX=B$ ) tidak memiliki dimensi persegi.

#### 4. Studi Kasus Interpolasi

Kasus	Solusi
<pre> 1  0.4  0.043 2  0.7  0.005 3  0.11 0.058 4  0.14 0.072 5  0.17 0.1 6  0.2  0.13 7  0.23 0.147 8  0.2 9  0.55 0  0.85 1  1.28 2 </pre>	<pre> 1 f(x) = -0.1846 + 10.2764x^1 + -163.9157x^2 + 1228.8549x^3 + -4346.3140x^4 + 7102.3992x^5 + -4212.4345x^6 2 3 f(0.20) = 0.130 4 5 f(0.55) = 2.138 6 7 f(0.85) = -66.270 8 9 f(1.28) = -3485.145 10 11 </pre>
<pre> 6.567 12624 7 21.807 7.258 38391 7.451 54517 7.548 51952 7.839 28228 8.161 35764 8.484 20813 8.709 12408 9 10534 7.51613 8.3226 9.166666667 8.709 </pre>	<pre> 1 f(x) = 25387828767373.1560 + -30289673764793.3448x^1 + 16032127510451.4060x^2 + 2 -4940974669395.5430x^3 + 977137403611.9904x^4 + -128592096990.2696x^5 + 3 11261317626.4222x^6 + -632830812.4955x^7 + 28706850.1132x^8 + -300588.3126x^9 4 5 f(7.52) = 53502.863 6 7 f(8.32) = 36977.789 8 9 f(9.17) = -817652.531 0 1 f(8.71) = 12408.641 2 3 </pre>
<p>n = 5</p>	<p>n = 5</p> <pre> f(x) = 0.0000 + 2.0353x^1 + -3.5527x^2 + 3.2371x^3 + -1.4213x^4 + 0.2363x^5 </pre> <p>n = 20</p>

```

0 0.0
0.4 0.418884230141255
0.8 0.5071579685304317
1.2000000000000002 0.5609246748146806
1.6 0.5836856612868684
2.0 0.5766515297517221

```

$n = 20$

```

0.1 0.270690041656283
0.2 0.34276955824301036
0.3000000000000004 0.38653159565945977
0.4 0.418884230141255
0.5 0.44543086822735684
0.6 0.4684314696118773
0.7 0.4888654824302957
0.7999999999999999 0.5071579685304317
0.8999999999999999 0.523479482504966
0.9999999999999999 0.5378828427399902
1.0999999999999999 0.5503697547786223
1.2 0.5609246748146804
1.3 0.5695323934956485
1.4000000000000001 0.576187119761621
1.5000000000000002 0.5808969390639371
1.6000000000000003 0.5836856612868682
1.7000000000000004 0.5845931468464284
1.8000000000000005 0.5836747200777025
1.9000000000000006 0.5810000237884088
2.0000000000000004 0.576651529751722

```

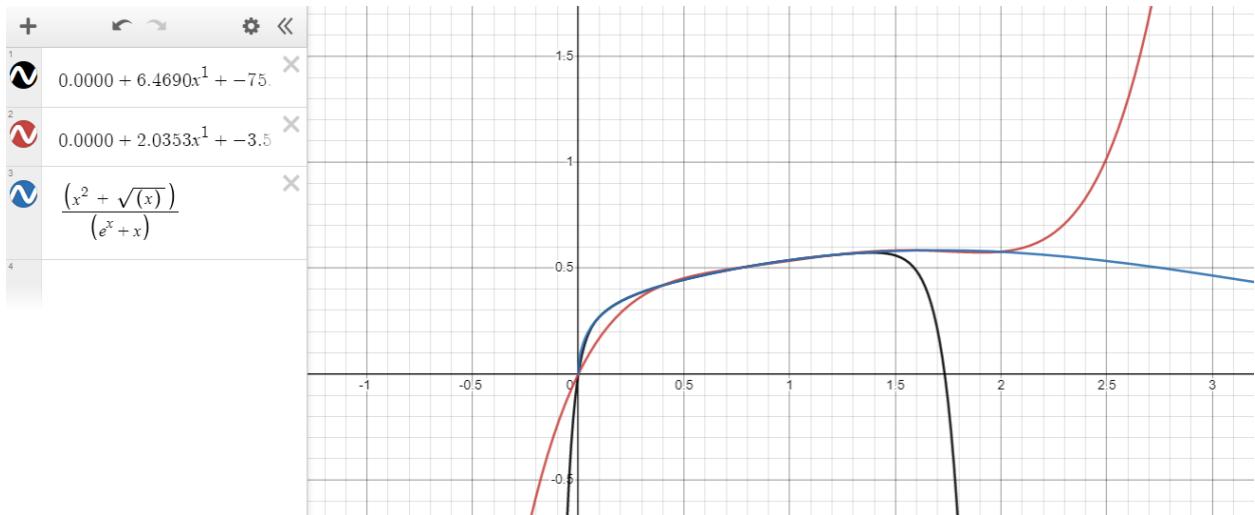
```

f(x) = 0.0000 + 6.4698x^1 + -75.1897x^2 +
627.8184x^3 + -3718.1493x^4 + 16165.5975x^5 +
-53187.6067x^6 + 135519.9488x^7 +
-271910.1058x^8 + 434600.3547x^9 +
-557379.0277x^10 + 575595.2688x^11 + -478543.9022x^12
+ 319026.9982x^13 + -169100.8089x^14 +
70252.7981x^15 + -22364.2518x^16 + 5262.8378x^17 +
-862.0471x^18 + 87.7079x^19 + -4.1720x^20

```

Untuk studi kasus interpolasi bagian (a), dapat dilihat bahwa program berhasil memprediksi nilai  $f(x)$  dengan cukup baik. Akan tetapi di bagian (b), dapat dilihat bahwa pada ketika memprediksi nilai kasus Covid di tanggal (05/09/22), program menghasilkan nilai negatif. Hal ini tidak masuk akal, mengingat bahwa data input semuanya menghasilkan nilai positif. Program berperilaku demikian akibat fakta bahwa nilai 9.15 (05/09/22) sudah berada di luar rentang data input yang diberikan (6.15 - 9). Artinya, nilai yang diberikan merupakan nilai ekstrapolasi dan bukan interpolasi, yaitu memprediksi data di luar range input yang diberikan, sehingga kurang akurat. Idealnya, program hanya digunakan untuk memprediksi nilai di selang 6.15 - 9.

Untuk kasus penyederhanaan fungsi di poin c, dipilih 2 derajat polinom yaitu 5 dan 20. Untuk membangkitkan set point pada range tersebut, digunakan script python yang terlampir di lampiran. Dapat dilihat bahwa semakin tinggi derajat polinom yang digunakan, semakin tinggi pula dataset poin yang harus diberikan. Hal ini mengakibatkan fungsi interpolasi juga menjadi semakin akurat (mendekati fungsi sebenarnya).



Gambar diatas menunjukkan 3 fungsi, fungsi hitam yaitu polinom berderajat 5, merah berderajat 20, dan biru fungsi induk. Pada range interpolasi 0-2, kedua fungsi interpolasi cukup baik mengikuti fungsi  $f(x)$ . Akan tetapi di luar batas tersebut, nilai ekstrapolasi sangatlah buruk. Fungsi yang berderajat 20 jauh lebih baik dalam mengfit fungsi  $f(x)$  daripada polinom berderajat 5, karena lebih banyaknya variabel dan dataset yang diberikan ke fungsi berderajat 20.

## 5. Studi Kasus Interpolasi Bicubic

Kasus	Solusi
a. $f(0,0)$	Masukkan nama file: bicubic1.txt $f(0.00,0.00) = 161.00$
b. $f(0.5,0.5)$	Masukkan nama file: bicubic1.txt $f(0.50,0.50) = 97.73$
c. $f(0.25,0.75)$	Masukkan nama file: bicubic1.txt $f(0.25,0.75) = 105.51$
d. $f(0.1,0.9)$	Masukkan nama file: bicubic1.txt $f(0.10,0.90) = 104.23$
Semua solusi di atas menggunakan pembulatan 2 angka di belakang koma.	

Fungsi bikubik digunakan untuk menginterpolasi kotak tengah pada matriks  $4 \times 4$ . Dapat dilihat bahwa  $f(0,0)$  mengembalikan elemen matriks (1,1). Hal ini karena di pojok-pojok kotak,

nilai tidak perlu diinterpolasi lagi karena telah diberikan di dalam dataset. Nilai  $f(x,y)$  lain merupakan hasil interpolasi dari matriks yang telah diberikan.

## 6. Studi Kasus Regresi Linier Berganda

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.

Agar dapat diproses oleh program, maka data tersebut diubah ke dalam bentuk matriks *augmented*  $[x_k \mid y]$  seperti di bawah

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

Dari data-data tersebut, diterapkan *Normal Estimation Equation for Multiple Linear Regression*, maka diperoleh sistem persamaan linear sebagai berikut.

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

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

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

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

Dengan mengubah SPL tersebut menjadi matriks *augmented* dan menggunakan metode eliminasi Gauss-Jordan, diperoleh solusi dari SPL tersebut sebagai berikut.

$$b_0 = (-3.5078)$$

$$b_1 = (-0.0026)$$

$$b_2 = 0.0008$$

$$b_3 = 0.1542$$

Dengan substitusi solusi tersebut ke dalam persamaan

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

diperoleh persamaan regresi

$$f(x) = -3.5078 + (-0.0026)x_1 + (0.0008)x_2 + (0.1542)x_3$$

Dari persamaan regresi tersebut, terlihat koefisien regresi variabel independen x3 (tekanan udara) paling berpengaruh terhadap variabel dependen (Nitrous Oxide) karena setiap kenaikan x3 sebanyak 1%, terdapat kenaikan Nitrous Oxide sebanyak 1% pula.

Pembacaan input nilai Humidity, temperatur, dan tekanan udara dapat dilakukan melalui file yang sama dengan pembacaan nilai data pada tabel seperti di bawah

72.4	76.3	29.18	0.90
41.6	70.3	29.35	0.91
34.3	77.1	29.24	0.96
35.1	68.0	29.27	0.89
10.7	79.0	29.78	1.00
12.9	67.4	29.39	1.10
8.3	66.8	29.69	1.15
20.1	76.9	29.48	1.03
72.2	77.7	29.09	0.77
24.0	67.7	29.60	1.07
23.2	76.8	29.38	1.07
47.4	86.6	29.35	0.94
31.5	76.9	29.63	1.10
10.6	86.3	29.56	1.10
11.2	86.0	29.48	1.10
73.3	76.3	29.40	0.91
75.4	77.9	29.28	0.87
96.6	78.7	29.29	0.78
107.4	86.8	29.03	0.82
54.9	70.9	29.37	0.95
50	76	29.30	

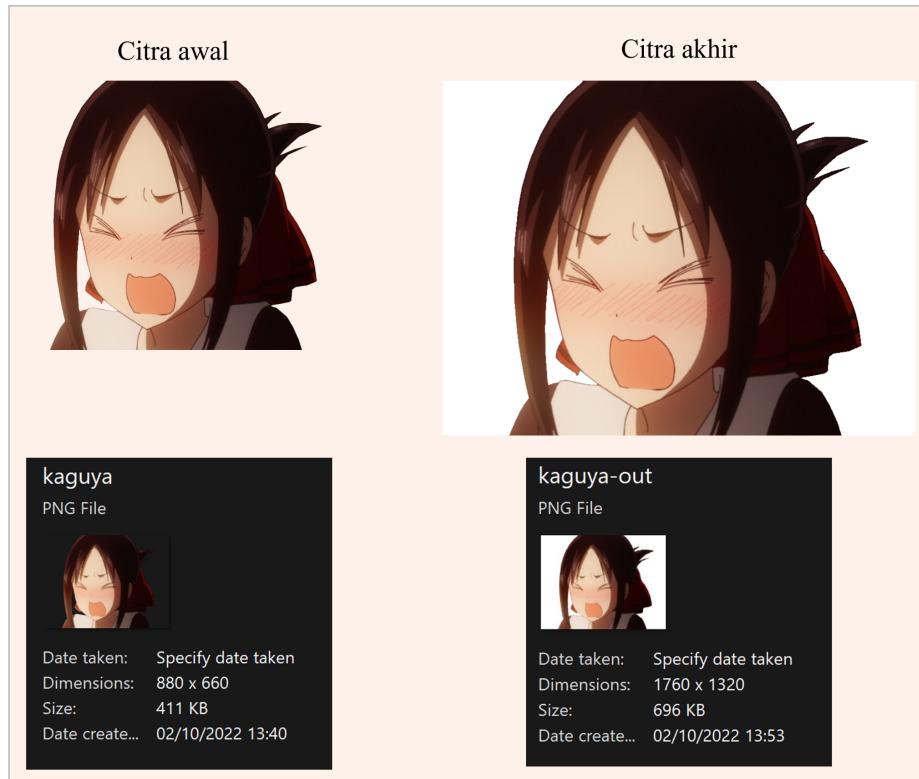
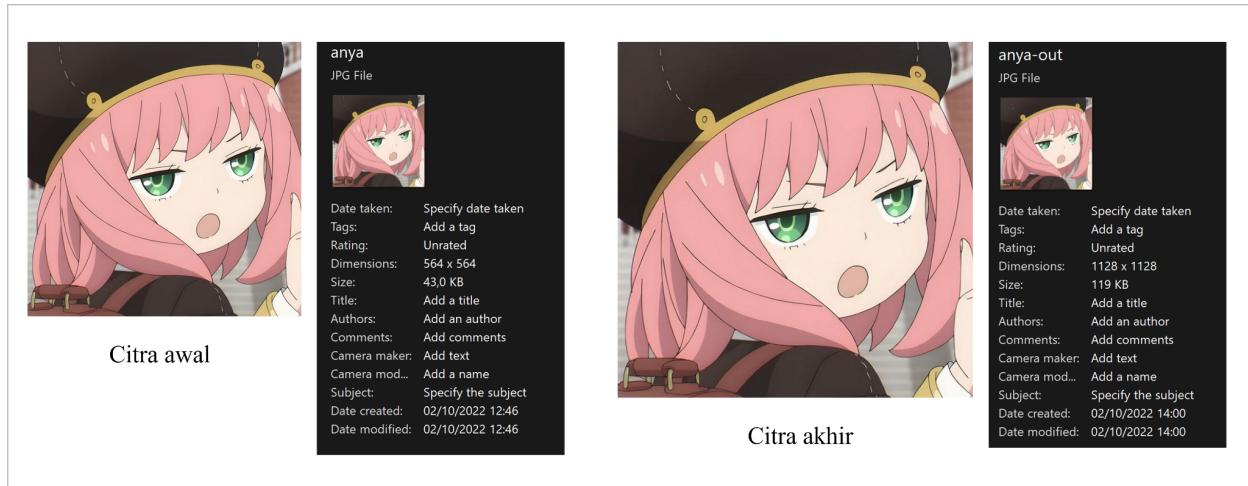
→ nilai yang akan diestimasi

Diperoleh estimasi nilai Nitrous Oxide apabila Humidity bernilai 50%, temperatur 76°F, dan tekanan udara sebesar 29.30 sebagai berikut.

**Estimasi nilai  $x_1 = 50.00$ ,  $x_2 = 76.00$ ,  $x_3 = 29.30$   
 $f(x_k) = 0.9384$**

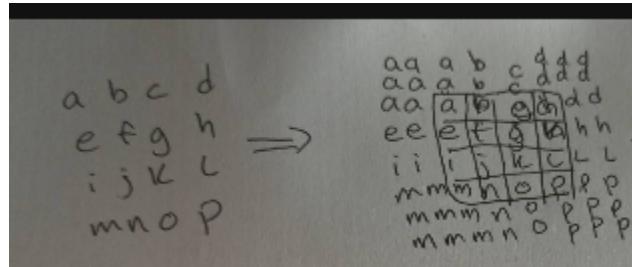
Sehingga apabila Humidity bernilai 50%, temperatur 76°F, dan tekanan udara sebesar 29.30, nilai Nitrous Oxide-nya sebesar 0.9384.

## 7. Bonus - Perbesaran Citra (*scaling image*)



Perbesaran citra yang dilakukan sebesar dua kali dari ukuran awal (dapat dilihat pada gambar di atas). Format file citra yang dapat diolah, yaitu *.jpg* atau *.png*. Agar mendapatkan hasil yang diharapkan (visual citra tetap sama) disarankan menggunakan format *.jpg*. Untuk format *.png* gambar yang awalnya memiliki latar transparan (*no background*) akan memiliki latar putih (*white background*) setelah diperbesar.

Citra input tidak harus diberikan dalam bentuk grayscale, karena program menyimpan nilai RGB tiap pixel. Untuk edge case di tepian gambar, daripada membuat fungsi khusus untuk menginterpolasi pixel tersebut, kami memilih untuk menambahkan padding di pojok-pojok gambar. Padding ini berguna agar dapat memungkinkan interpolasi bikubik untuk tetap dilakukan di tepian. Nilai padding diambil dari pixel yang paling dekat dengannya dengan pola sebagai berikut:



Dalam menentukan titik mana yang harus dilakukan interpolasi kami melakukan analisis pada masing-masing pixel citra. Asumsi, suatu citra memiliki ukuran 3x3 dan telah dilakukan *padding* seperti terlihat pada gambar di bawah.

a	a	a	b	c	c	c
a	a	a	b	c	c	c
a	a	a	b	c	c	c
d	d	d	e	f	f	f
g	g	g	h	i	i	i
g	g	g	h	i	i	i
g	g	g	h	i	i	i

Selanjutnya, pixel awal citra akan diasumsikan berada pada bagian tengah kotak, lalu akan disisipkan masing-masing dua baris dan dua kolom pada kotak pixel tersebut. Untuk melakukan penyisipan baris dan kolom ini kita menggunakan interpolasi bicubic dengan mengambil 16 buah pixel dari citra di atas. Misal, kita mengambil sebuah matriks berukuran 4x4 pada ujung citra,

a	a	a	b
a	a	a	b
a	a	a	b
d	d	q	e

Dari matriks  $4 \times 4$  di atas kita hanya dapat memperoleh dua titik, yaitu  $x$  dan  $y$  untuk diinterpolasi karena titik lainnya  $p, q, r, s, t$ , dan  $u$  berada di luar batas interpolasi, yaitu di luar  $[0,1]$  (ingat kembali bahwa pixel citra terletak pada bagian tengah kotak dan koordinat dimulai dari  $(-1,-1)$  pada ujung kiri matriks).

Observasi ini terus dilakukan hingga ditemukan suatu pola titik-titik apa saja yang harus diinterpolasi pada masing-masing segmen (contoh segmen adalah gambar di atas). Selanjutnya, hasil interpolasi yang diperoleh dalam bentuk RGB dibentuk menjadi suatu citra baru yang berukuran  $2x$  citra awalnya.

## BAB 5 KESIMPULAN

### 1. Kesimpulan

Program sederhana yang mengimplementasikan fungsi-fungsi dalam aljabar geometri dapat dibuat di dalam program Java. Program ini dapat menyelesaikan berbagai persoalan seperti mencari SPL, determinan, ataupun invers. Selain itu, program ini dapat memanfaatkan kelas-kelas lain yang telah dibuat untuk melakukan interpolasi polinom, interpolasi bikubik, dan juga regresi linear. Perbesaran citra menjadi 2x size semula juga berhasil dilakukan dengan memanfaatkan interpolasi bikubik.

### 2. Saran

Saran kami, mungkin bisa diberikan guide / referensi sumber belajar Java yang baik, karena kami belum terlalu familiar. Akan lebih baik jika diberikan sesi hands-on Java, build system, ataupun introduksi ke packages dan library. Hal ini karena kami merasa setup import files dan library cukup sulit untuk di configure, apalagi dengan berbagai command seperti javac, java, dan lain-lain. Kami merasa bahwa kedepannya, pengerjaan tugas akan lebih mudah jika kami tahu tentang keberadaan build tools dari awal.

### 3. Refleksi

Dalam membuat suatu program akan lebih baik jika dibagi ke dalam permasalahan-permasalahan kecil yang terstruktur. Ternyata dekomposisi masalah sulit dilakukan jika tidak direncanakan dengan baik. Contohnya kami bingung kelas mana yang harus membaca dari keyboard, struktur write ke file yang baik, dan lain-lain.

Selain itu, kami juga belajar bahwa setup bahasa pemrograman baru cukup sulit untuk dilakukan. Kami belum memakai build tools di dalam mengerjakan program ini, sehingga masih mengandalkan command javac dan java untuk mengcompile program. Di kemudian hari, kami akan mencoba memakai build tools untuk mencegah masalah ini.

## **DAFTAR PUSTAKA**

- [1] <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/algeo22-23.htm> (Terakhir diakses pada 2 Oktober 2022)
- [2] <https://www.dosenmatematika.co.id/menentukan-determinan-dengan-metode-ekspansi-kofaktor/> (Terakhir diakses pada 2 Oktober 2022)
- [3] [https://en.wikipedia.org/wiki/Linear\\_algebra](https://en.wikipedia.org/wiki/Linear_algebra) (Terakhir diakses pada 29 Oktober 2022)
- [3] [https://www.ece.mcmaster.ca/~xwu/interp\\_1.pdf](https://www.ece.mcmaster.ca/~xwu/interp_1.pdf) (Terakhir diakses pada 29 Oktober 2022)
- [4] <https://docs.oracle.com/en/java/> (Terakhir diakses pada 28 Oktober 2022)

## LAMPIRAN

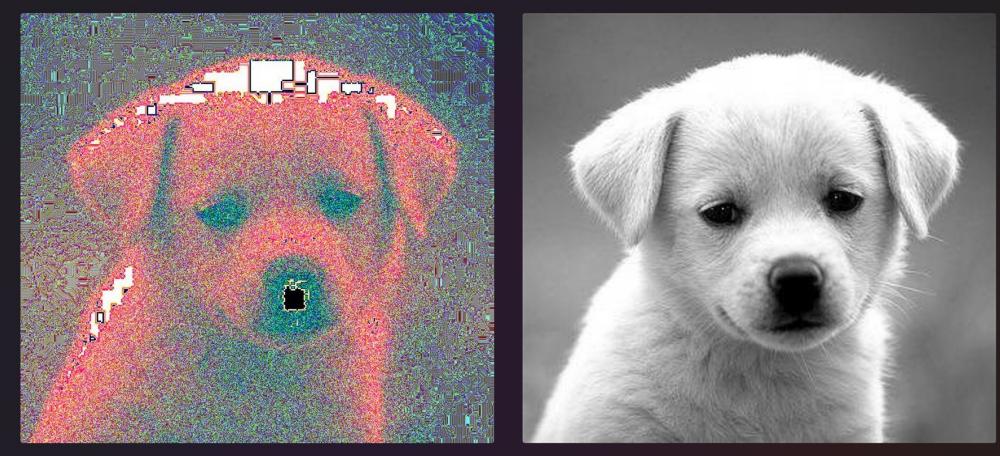
Generator penyederhanaan fungsi studi kasus

```
import math
n = 20
def func(x):
    return (math.pow(x,2) + math.sqrt(x)) / (math.pow((math.e),x) + x)
start = 0
skip = 2/n
for i in range(n + 1):
    print(str(start) + " " + str(func(start)))
    start += skip
```

Generator pencetak matriks hilbert

```
def hilbert(n):
    for i in range(n):
        for j in range(i + 1, n + 1 + i):
            print("{:.16f}".format(1/j), end=" ")
    if (i == 0):
        print(1)
    else:
        print(0)
```

BONUS (Blooper) Attempt Pembesaran Gambar



gukguk guuk guuk guk (watashi knp jadi merah?!) -anjing