

# LAPORAN TUGAS BESAR 1

Mata Kuliah Aljabar Linear dan Geometri IF2123

Dosen Pengampu :

Dr. Ir. Rinaldi Munir, M.T.

Nugraha Priya Utama S.T.,M.A.,Ph.D.

Dr. Judhi Santoso M.Sc.

Ir. Rila Mandala M.Eng.,Ph.D.



Disusun Oleh :

Denilsen Axel Candiasa (13519059)

Jesson Gosal Yo (13519079)

Jeremia Axel Bachtera (13519188)

PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2020

# BAB I DESKRIPSI MASALAH

## 1. Abstraksi

Sistem persamaan linier (SPL)  $Ax = b$  dengan  $n$  peubah (variable) dan  $m$  persamaan adalah berbentuk

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

Gambar I.1. Bentuk SPL

yang dalam hal ini  $x_i$  adalah peubah,  $a_{ij}$  dan  $b_i$  adalah koefisien  $\in \mathbb{R}$ . Sembarang SPL dapat diselesaikan dengan beberapa metode, yaitu metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan ( $x = A^{-1}b$ ), dan kaidah Cramer (khusus untuk SPL dengan  $n$  peubah dan  $n$  persamaan). Solusi sebuah SPL mungkin tidak ada, banyak, atau hanya satu (unik/tunggal).

Sebuah matriks  $M$  berukuran  $n \times n$

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

Gambar I.2. Matriks ukuran  $n \times n$

Determinannya adalah

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

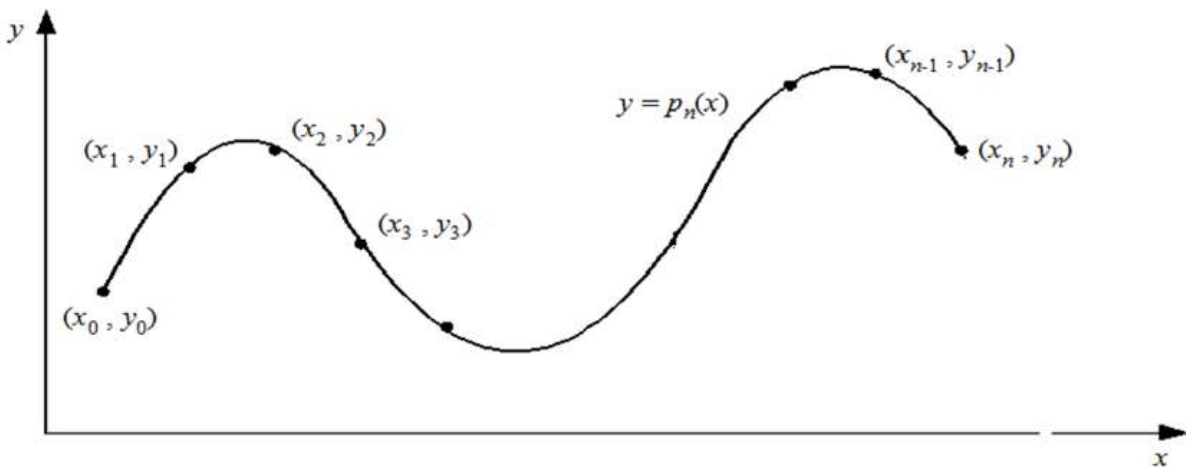
Gambar I.3. Determinan matriks berukuran  $n \times n$

Determinan matriks  $M$  berukuran  $n \times n$  dapat dihitung dengan beberapa cara: reduksi baris dan ekspansi kofaktor.

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

## 2. 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  $i = 0, 1, 2, \dots, n$ .



Gambar I.4. Ilustrasi Interpolasi Polinom dari  $n + 1$  Buah Titik

Setelah polinom interpolasi  $p_n(x)$  ditemukan,  $p_n(x)$  dapat digunakan untuk menghitung perkiraan nilai  $y$  di sembarang titik di dalam selang  $[x_0, x_n]$ . Polinom interpolasi derajat  $n$  yang menginterpolasi titik-titik  $(x_0, y_0)$ ,  $(x_1, y_1)$ , ...,  $(x_n, y_n)$ . adalah berbentuk  $p_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ . 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 linier dalam  $a_0, a_1, \dots, a_n$ .

$$\begin{aligned} a_0 + a_1x_0 + a_2x_0^2 + \dots + a_nx_0^n &= y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + \dots + a_nx_1^n &= y_1 \\ &\vdots \\ a_0 + a_1x_n + a_2x_n^2 + \dots + a_nx_n^n &= y_n \end{aligned}$$

Gambar I.5. SPL untuk Mencari Interpolasi Polinom

Solusi sistem persamaan linier 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 linier yang terbentuk adalah

$$\begin{aligned} a_0 + 8.0a_1 + 64.00a_2 &= 2.0794 \\ a_0 + 9.0a_1 + 81.00a_2 &= 2.1972 \\ a_0 + 9.5a_1 + 90.25a_2 &= 2.2513 \end{aligned}$$

Gambar I.6. Hasil SPL Contoh Soal

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

### 3. 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.

$$y_i = \beta_0 + \beta_1x_{1i} + \beta_2x_{2i} + \dots + \beta_kx_{ki} + \epsilon_i$$

Gambar I.7. Rumus Umum Regresi Linear

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

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

Gambar I.8. *Normal Estimation Equation for Multiple Linear Regression*

Sistem persamaan linier tersebut diselesaikan dengan menggunakan metode eliminasi Gauss.

## BAB II TEORI SINGKAT

### 1. Penyelesaian Sistem Persamaan Linear n-Variabel

#### a. Metode Eliminasi Gauss

Metode eliminasi Gauss dapat digunakan untuk menyelesaikan sistem persamaan linear n-variabel dengan melakukan operasi baris elementer terhadap matriks augmentasi persamaan dan mengubahnya menjadi bentuk eselon baris. Solusi dari SPL dapat dicari dengan substitusi mundur terhadap matriks hasil eliminasi gauss (matriks eselon baris).

#### b. Metode Eliminasi Gauss-Jordan

Eliminasi Gauss-Jordan ini dilakukan dengan melakukan OBE pada matriks *augmented* sampai terbentuk matriks eselon baris tereduksi. Ilustrasinya dapat dilihat pada gambar II.1 di bawah ini

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

Gambar II.1. Ilustrasi Eliminasi Gauss

Metode eliminasi ini tidak membutuhkan lagi substitusi secara mundur untuk memperoleh nilai - nilai variabel. Nilai variabel langsung dapat diperoleh dari matriks *augmented* akhir.

#### c. Metode Invers Balikan

Misal suatu SPL dinyatakan dalam  $Ax = b$ , dimana A dan b adalah suatu matriks. Solusi SPL ini dapat diperoleh dengan mengalikan matriks balikan A dengan matriks b. Adapun alasan dari mengalikan matriks A dengan matriks b adalah sebagai berikut :

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

$$Ix = A^{-1}b \quad (\text{karena } A^{-1}A = I)$$

$$x = A^{-1}b \quad (\text{karena } Ix = x)$$

Gambar II.2 ilustrasi pembuktian

Sehingga solusi SPL tersebut adalah  $x = A^{-1}b$ . Adapun perkondisi dari metode ini adalah matriks  $A$  harus berupa matriks persegi, dan determinannya tidak 0. Sehingga dapat diketahui bahwa metode ini akan selalu menghasilkan solusi SPL yang unik.

#### d. Metode Cramer

Jika terdapat SPL dengan  $n$  persamaan dengan  $n$  peubah dengan determinannya tidak sama dengan nol, maka SPL tersebut memiliki solusi unik. Solusi unik tersebut dapat dicari dengan menggunakan Kaidah Cramer. Untuk SPL dengan bentuk

$$Ax = b$$

SPL dapat dicari menggunakan

$$x_i = \frac{\det(A_i)}{\det(A)}$$

$A_i$  adalah matriks yang diperoleh dengan mengganti kolom ke- $i$  dengan  $b$

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

## 2. Determinan

### a. Metode Row Reduction

Determinan matriks  $n \times n$  dapat dicari dengan metode reduksi baris/row reduction. Reduksi baris akan diaplikasikan terhadap matriks hingga matriks menjadi matriks segitiga atas. Melakukan penambahan kelipatan suatu persamaan ke persamaan lain tidak mengubah nilai determinan matriks. Menukar baris pada matriks membuat nilai determinan matriks perlu dikalikan negatif 1. Melakukan perkalian skalar terhadap salah satu baris mengakibatkan nilai determinan matriks membesar sebesar faktor pengalinya. Nilai determinan adalah hasil perkalian angka pada diagonal utama dengan mempertimbangkan operasi di atas, contohnya jika kita menukar baris maka nilai determinan harus dikalikan dengan negatif 1 untuk mendapat nilai determinan aslinya dan juga jika melakukan perkalian skalar sebesar  $k$  maka nilai determinan harus dibagi dengan  $k$  untuk mendapat nilai determinan aslinya.

### b. Metode Cofactor

Determinan matriks  $n \times n$  dapat didefinisikan dengan mencari determinan matriks persegi yang memiliki ukuran dimensi sebesar 1 di bawah matriks tersebut. Contohnya determinan matriks  $3 \times 3$  dapat didefinisikan secara induktif dari 4 buah matriks determinan  $2 \times 2$  yang menyusunnya. Secara umum, determinan sebuah matriks, misalkan  $A$  dapat ditulis sebagai berikut jika kita melakukan ekspansi terhadap kolom  $j$ .  $C$  melambangkan matriks minor.

$$\det(A) = \sum_{i=1}^n A_{i,j} C_{i,j}.$$

Gambar II.3 Salah satu cara ekspansi kofaktor



### 3. Invers Matriks

a. Metode adjoint

Untuk mencari invers balikan dari suatu matriks, dapat menggunakan adjoin dari matriks tersebut, misal matriks A yang berukuran  $n \times n$ , matriks balikkannya dapat dihitung dengan menggunakan rumus :

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

Gambar II.4. Rumus invers balikan menggunakan adjoint

Adjoin A adalah transpose dari matriks kofaktor A, dimana kofaktor matriks A dapat diilustrasikan sebagai berikut :

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

Gambar II.5. Ilustrasi matriks kofaktor A

Dimana :

$$C_{11} = (-1)^{1+1} M_{11} = M_{11} = 26$$

$$C_{12} = (-1)^{1+2} M_{12} = -M_{12} = -10$$

$$C_{13} = (-1)^{1+3} M_{13} = M_{13} = 8$$

$$C_{22} = (-1)^{2+2} M_{22} = M_{22} = 17$$

Gambar II.6. Penjabaran dari  $C_{11}$  -  $C_{nn}$

$M_{ij}$  pada gambar II.4 merupakan minor entri dari suatu elemen  $ij$  pada matriks  $A$ , dimana minor entri tersebut dapat dihitung dengan menghitung determinan matriks yang tidak melibatkan baris  $i$  dan  $j$ .

b. Metode Eliminasi Gauss-Jordan

Untuk suatu matriks, misal matriks  $A$  yang berukuran  $n \times n$ , matriks balikkannya, yaitu  $A^{-1}$  dapat dicari dengan menggunakan metode eliminasi Gauss-Jordan sesuai ilustrasi pada gambar II.3 :

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

Gambar II.7 Ekuivalensi

Gambar II.5. Ilustrasi Metode Eliminasi Gauss-Jordan untuk mencari matriks balikan  $A$   
Perhatikan bahwa metode eliminasi Gauss-Jordan ini dapat diterapkan secara simultan / bersamaan pada matriks  $A$  maupun  $I$ .

## 4. Interpolasi Polinom

Interpolasi polinom adalah metode mencocokkan kumpulan titik data dengan kurva menggunakan menginterpolasi titik-titik data dengan suatu polinom. Interpolasi sendiri didefinisikan sebagai metode untuk menghasilkan titik-titik data baru dalam suatu jangkauan dari suatu set diskrit data-data yang diketahui. Titik data akan diubah menjadi bentuk SPL dan direpresentasikan sebagai matriks. Misalnya jika ada  $n$  buah titik  $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ , maka bentuk matriksnya adalah sebagai berikut.

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

Gambar II.8 SPL  $n$  variabel

Persamaan yang menginterpolasi SPL tersebut adalah polinom  $p_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ . Nilai  $a_0$  hingga  $a_n$  dapat dicari menggunakan eliminasi gauss maupun eliminasi gauss jordan. Solusi  $a_0$  hingga  $a_n$  disubstitusikan ke persamaan  $p_n(x)$  untuk mendapat solusi persamaan interpolasi titik.

## 5. Regresi Linier Berganda

Regresi Linier Berganda adalah model regresi linear dengan melibatkan lebih dari satu variabel bebas. Regresi linier berganda ini berfungsi untuk memprediksi suatu nilai yang dipengaruhi oleh beberapa variabel bebas. Adapun rumus umum regresi linier yang dapat digunakan untuk regresi linier berganda, yaitu :

$$y_i = \beta_0 + \beta_1x_{1i} + \beta_2x_{2i} + \dots + \beta_kx_{ki} + \epsilon_i$$

Gambar II.9. Rumus Umum Regresi Linear

Keterangan :

y = Variabel terikat atau *response*

x = Variabel bebas atau *predictor*

$\beta$  = *Slope* atau koefisien estimasi

$\epsilon$  = Residu

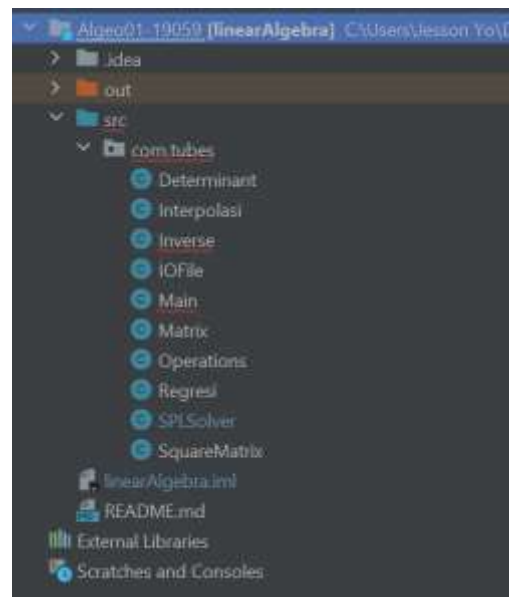
Untuk memperoleh nilai setiap  $\beta_i$ , dapat digunakan *Normal Estimation Equation for Multiple Linear Regression*, sebagai berikut :

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

Gambar II.10. Normal Estimation Equation for Multiple Linear Regression

# BAB III IMPLEMENTASI PROGRAM DALAM BAHASA JAVA

Implementasi program Java memanfaatkan Object Oriented Programming untuk melakukan modularitas dalam program. Program ini memanfaatkan IntelliJ IDEA untuk melakukan setup dan code editing. Di bawah ini adalah tampilan directory program saat development.



Gambar III.1

Semua class bentukan disimpan dalam package `com.tubes` yang merupakan sub directory dari `src` dan bytecode tersimpan dalam folder `out`. Program memanfaatkan library java seperti `Math` dan `java.util` terutama `java.util.Scanner` untuk membaca input dari command line. Dalam package `com.tubes` ada beberapa class yang dimanfaatkan untuk modularitas.

## 1. Main

Main adalah class utama dan semua class maupun method akan dipanggil oleh class ini. Dalam main terdapat method `public static void main` yang merupakan pusat/driver program ini. Di dalam fungsi ini ada loop yang tidak akan berhenti kecuali user exit. Selama loop berjalan, user akan diberikan pilihan operasi apa yang ingin dilakukan. Setelah memilih operasi, user harus menginput matriks yang antar elemennya dipisahkan spasi atau enter. Kemudian user memilih metode (jika ada) agar dapat mengeluarkan solusi

## 2. Matrix

Matrix adalah class public yang memiliki 3 atribut private yaitu

- a. Private int n\_row
- b. Private int n\_col
- c. Private double[][] Matrix

Class ini lengkap dengan constructor, getter, dan setter. Object ini menyimpan angka dalam representasi double. Matrix bisa memiliki besar row dalam col yang berbeda.

### 3. SquareMatrix

SquareMatrix adalah class public yang memiliki 2 atribut private yaitu

- a. Private int dimension
- b. Private double[][] Matrix

Class ini lengkap dengan constructor, getter, dan setter. Object ini sama dengan matrix namun memiliki row dan col yang sama yang direpresentasikan dalam atribut dimension.

### 4. Operations

Operations adalah class public yang dimanfaatkan untuk meletakkan method-method yang sering dipakai. Class ini tidak memiliki atribut dan semua method merupakan static agar Object Operations tidak perlu dikonstruksi untuk dapat memakai method( bisa langsung di invoke dengan Operations.<method>). Operations juga memanfaatkan method overloading untuk mempermudah penulisan program ketika parameternya adalah Square Matrix atau Matrix.

Method yang tersedia dalam class ini:

- a. `static void swapRow(int a,int b,SquareMatrix matrix)`
- b. `static void swapRow(int a, int b, Matrix matrix)`
- c. `static void printMatrix(Matrix matrix)`
- d. `static void printMatrix(SquareMatrix matrix)`
- e. `static void fillMatrix(Matrix matrix)`

Method untuk melakukan pengisian manual terhadap matrix

- f. `static void fillMatrix(SquareMatrix matrix)`

Method untuk melakukan pengisian manual terhadap square matrix

- g. `static SquareMatrix cofactor(SquareMatrix matrix,int row,int column)`
- h. `static SquareMatrix transpose(SquareMatrix matrix)`
- i. `static void scalarMultiplication(SquareMatrix matrix, double x)`
- j. `static SquareMatrix getAdjoint(SquareMatrix matrix)`

Method ini dipanggil untuk mencari adjoint dari suatu matriks persegi, adapun cara mencari adjointnya adalah dengan mengtranspos matriks kofaktor. Untuk mencari

matriks kofaktornya sendiri, kami menggunakan teori yang sudah dijelaskan pada bagian sebelumnya.

k. `static double roundAvoid(double value,int digits)`

Method ini untuk menghandle floating point precision dengan digits adalah banyaknya digit di belakang koma yang diinginkan. Contohnya 1 kadang diinterpretasikan sebagai 0.999...999 yang akan diubah menjadi 1.000 (jika digits = 3). Masalah yang ditimbulkan jika hal ini tidak ditangani bisa kecil hingga besar. Contoh masalah kecil adalah output program yang tidak tepat ketika menghitung determinan, 124 dicetak sebagai 123.99...9997. Masalah besar dapat timbul contohnya ketika 0 ditulis sebagai 9E-18 sehingga elemen ini akan diinterpretasikan algoritma sebagai elemen leading one dan akan melakukan perkalian skalar dengan orde  $10^{18}$ .

5. IOFile

IOFile akan menghandle input output jika user memilih untuk menginput melalui file maupun ingin menyimpan hasil output program ke dalam file.

6. SPLSolver

SPL Solver adalah class yang diciptakan dengan tujuan utama menyelesaikan permasalahan 1 dalam tugas ini yaitu sistem persamaan linear. Class ini tidak memiliki atribut.

Method yang tersedia dalam class ini:

a. `public void gaussDriver(Matrix matrix)`

Method yang memanggil method lain untuk menyelesaikan SPL dengan metode eliminasi gauss.

b. `public void gaussDriverInterpolation(Matrix matrix)`

Method spesial yang akan dipanggil oleh class Interpolasi.

c. `public void gaussDriverRegression(Matrix matrix, int n)`

Method ini dipanggil setelah class Regresi berhasil meng-*generate Normal Estimation Equation for Multiple Linear Regression*. Method ini lalu akan memanggil method gauss untuk mencari nilai  $\beta_i$ , lalu mencetak persamaan regresi liniernya, lalu menerima input  $x_i$  untuk mencari nilai  $y$ .

d. `static Matrix gauss(Matrix matrix)`

Method untuk menyelesaikan persamaan gauss dengan operasi baris dan sulih mundur.

e. `static int validateGaussMatrix(Matrix matrix)`

Method ini melakukan pengecekan terhadap Matrix hasil SPL, jika matrix tidak memiliki solusi maka akan return 0, jika matrix memiliki solusi tunggal akan return 1, dan jika matrix memiliki banyak solusi(parametrik) akan return 2. Method ini akan dimanfaatkan

untuk percabangan dalam menentukan fungsi mana yang tepat untuk digunakan dalam memunculkan solusi.

f. `static void singleSolution(Matrix matrix)`

Method ini dipakai terhadap matrix yang memiliki solusi tunggal, fungsi ini akan mengevaluasi dan mencetak output ke command line.

g. `static void extractAug(Matrix m, SquareMatrix matriksA, Matrix matriksB)`

h. `static void cramerSPL(Matrix m)`

i. `static void inversSPL(Matrix m)`

Method ini dipanggil jika user ingin menyelesaikan SPL dengan menggunakan metode matriks balikan, dimana syarat metode ini adalah bentuk matriks A yang berupa persegi, dan determinan matriks A tidak boleh 0.

j. `static Vector<Double> singleSolutionReturn(Matrix matrix)`

Method ini sama dengan `singleSolution` namun solusi disimpan dalam representasi Vector dan Vector akan direturn oleh method.

k. `static void translator(Matrix matrix)`

Method ini dipanggil untuk menangani kasus saat matrix memiliki banyak solusi. Cara kerjanya adalah pertama method akan memisahkan variabel independent dan dependent kemudian dilakukan substitusi mundur. Kaidah penamaan independent dan dependent dalam program ini bukan mengacu pada kaidah penamaan matematis namun kepada ketergantungan variabel terhadap matriks pembantu(matriks padding). Pertama substitusi mundur akan dilakukan terhadap variabel dependent yaitu variabel-variabel yang nantinya akan menjadi variabel bebas, substitusi mundur dilakukan tidak dengan mengubah  $b_i$  (persamaan) namun dengan menyimpan koefisien variabel bebas dalam matrix pembantu untuk ditangani `paddingParser`. Setelah disimpan dalam matrix pembantu, koefisien di matrix asli akan dijadikan 0. Substitusi variabel independent dilakukan sama seperti ketika solusi bisa ditentukan( tunggal ) dengan pura-pura mengabaikan variabel-variabel dependent tadi. Setelah itu barulah kita menambahkan variabel-variabel dependent tadi dengan mengubah semua menjadi string dan melakukan concatenation. Concatenation dilakukan dengan koefisien-koefisien yang ada dalam matrix pembantu (padding) yang diberikan nama variabel terlebih dahulu dengan bantuan `padding parser`.

l. `static String paddingParser(Matrix matrix, int row, Matrix padding)`

Padding parser membantu dalam proses concatenation untuk mencetak solusi parametrik dengan “mengubah” isi dari baris tertentu pada padding menjadi sebuah string yang sesuai. Misal sebuah baris dalam padding berisi [ 3 , 0 , 2 ] dan solusi yang didapat dari matrix sebelum melakukan concatenation adalah  $x_2 = 3$ . Maka [3,0,-2] akan diubah menjadi  $+3.0w - 2.0u$  dan setelah concatenation string lengkap akan menjadi  $x_2 = 3 + 3.0w - 2.0u$

m. `static void gaussJordanSolver (Matrix matrix)`

Method ini dipanggil jika user ingin menyelesaikan SPL dengan menggunakan metode eliminasi Gauss-Jordan, dimana akan dilakukan OBE pada matrix *augmented* yang sudah dimasukkan oleh pengguna sehingga terbentuk matriks eselon baris tereduksi. Lalu method ini akan menentukan apakah SPL tersebut memiliki solusi unik, memiliki solusi banyak, atau tidak memiliki solusi. Jika SPL memiliki solusi unik atau solusi banyak, maka method ini akan mencetak nilai  $x_i$ .

## 7. Determinant

Class ini tidak memiliki atribut dan hanya memiliki 2 method static:

a. `static double RowReductionDeterminant(SquareMatrix matrix)`

Method ini dipanggil ketika pengguna memilih untuk mencari determinan dengan metode reduksi baris. Method ini akan mereturn double.

b. `static double CofactorExpansionDeterminant(SquareMatrix matrix)`

Method ini dipanggil ketika pengguna ingin mencari determinan dengan metode kofaktor. Untuk method ini ada constraint karena penggunaan rekursif. Ketika ukuran matriks terlalu besar maka ada kemungkinan timbul masalah karena kedalaman rekursif dibatasi sistem untuk menghindari stack overflow.

## 8. Invers

Class ini tidak memiliki atribut dan hanya memiliki 2 method static:

a. `static void AdjointInverse(SquareMatrix matrix)`

Method ini dipanggil ketika pengguna ingin mengetahui balikan dari suatu matriks. Adapun cara mencari matriks balikannya dengan menggunakan metode adjoint yang sudah dijelaskan pada bab sebelumnya.

b. `static void RowOperationInverse(SquareMatrix matrix)`

Method ini dipanggil ketika pengguna ingin mengetahui balikan dari suatu matriks. Adapun cara mencari matriks balikannya dengan menggunakan metode eliminasi Gauss-Jordan yang sudah dijelaskan pada bab sebelumnya.



## 9. Interpolasi

Class ini tidak memiliki atribut dan memiliki 2 method

a. `public void driverManual(int n)`

Driver manual dibuat untuk menangani kasus saat matrix diciptakan manual melalui input user. Driver akan otomatis mengubah titik data menjadi bentuk matriks yang diinginkan kemudian juga akan memanggil SPLSolver untuk menyelesaikan sistem persamaan sekaligus memunculkan persamaan yang terbentuk. Setelah persamaan terbentuk, user akan diminta input nilai x dan program akan mengevaluasi hasil pemetaan nilai x dengan fungsi polinom yang diciptakan dengan interpolasi.

b. `static void createPowMatrix(Matrix matrixIn, Matrix matrixOut)`

Method createPowMatrix digunakan untuk membaca kumpulan titik dari file dan mengubahnya menjadi sistem persamaan linear dalam representasi matriks yang diinginkan dan melakukan hal-hal yang sama seperti driverManual.

## 10. Regresi

Class ini tidak memiliki atribut dan memiliki 1 method

a. `public void generateMatrixEquation(Matrix tabel, Matrix dataMatrix, int var, int nData)`

Method ini dibuat untuk meng-*generate* matriks *Normal Estimation Equation for Multiple Linear Regression* dari tabel data yang diterima dari input user. Setelah berhasil di-*generate* dan ditampung dalam variabel dataMatrix, selanjutnya akan dilakukan eliminasi Gauss

## BAB IV EKSPERIMEN

### Soal 1

a. Input :

```
Baris: 4  
Kolom: 5  
1 1 -1 -1 1  
2 5 -7 -5 -2  
2 -1 1 3 4  
5 2 -4 2 6
```

Output :

```
1.0 1.0 -1.0 -1.0 1.0  
0.0 1.0 -1.6666666666666665 -1.0 -1.3333333333333333  
0.0 0.0 1.0 -1.0 1.0  
0.0 0.0 0.0 0.0 1.0  
No possible solution
```

Analisis :

Program mengeksekusi SPL di atas dengan melakukan metode eliminasi Gauss. Dimana setelah melakukan OBE untuk memperoleh matriks eselon baris, diperoleh matriks *augmented* akhir seperti diatas, namun terlihat pada baris terakhir,  $0x_1 + 0x_2 + 0x_3 = 1$ , sehingga tidak ada nilai  $x_i$  yang memenuhi SPL tersebut, begitu pula jika kita menggunakan metode eliminasi Gauss-Jordan, kita akan memperoleh matriks *augmented* akhir yang sama yang menyebabkan SPL tidak memiliki solusi. Jika kita menggunakan metode invers balikan ataupun cramer, program akan otomatis berhenti melakukan operasi, karena didapat bahwa determinan matriks A 0, sehingga tidak mungkin memperoleh solusi melalui metode tersebut.

b. Input :

```
Baris: 4
Kolom: 6
1.0 -1.0 0.0 0.0 1.0 3.0
1.0 1.0 0.0 -3.0 0.0 6.0
2.0 -1.0 0.0 1.0 -1.0 5.0
-1.0 2.0 0.0 -2.0 -1.0 -1.0
```

Output :

```
1.0 0.0 0.0 0.0 -1.0 3.0
0.0 1.0 0.0 0.0 -2.0 0.0
0.0 0.0 0.0 1.0 -1.0 -1.0
0.0 0.0 0.0 0.0 0.0 0.0
x3 = w
x5 = v
x1 = 3.0 + 1.0v
x2 = 2.0v
x4 = -1.0 + 1.0v
```

Analisis :

Program mengeksekusi SPL di atas dengan melakukan metode eliminasi Gauss-Jordan. Dimana setelah melakukan OBE untuk memperoleh matriks eselon baris, diperoleh matriks *augmented* akhir seperti diatas, namun terlihat pada baris terakhir,  $0x_1 + 0x_2 + 0x_3 = 0$ , sehingga SPL tersebut memiliki banyak solusi. Dapat dilihat pula pada kolom ke-3 berisi angka 0 semua, sehingga dapat diketahui bahwa berapapun nilai  $x_3$ , SPL akan selalu terpenuhi. Setelah mengganti variabel independen  $x_3$  dan  $x_5$  dengan huruf w dan v, dapat dibentuk solusi dari SPL tersebut sesuai dengan gambar di atas. Jika kita menggunakan metode invers balikan ataupun cramer, program akan otomatis berhenti melakukan operasi, karena matriks A bukanlah matriks persegi, sehingga determinannya tidak dapat ditentukan.

c. Input :

```

Baris: 3
Kolom: 7
0 1 0 0 1 0 2
0 0 0 1 1 0 -1
0 1 0 0 0 1 1

```

Output :

```

0.0 1.0 0.0 0.0 0.0 1.0 1.0
0.0 0.0 0.0 1.0 0.0 1.0 -2.0
-0.0 -0.0 -0.0 -0.0 1.0 -1.0 1.0
x1 = w
x3 = v
x6 = u
x2 = 1.0 - 1.0u
x4 = -2.0 - 1.0u
x5 = 1.0 + 1.0u

```

Analisis :

Program mengeksekusi SPL di atas dengan melakukan metode eliminasi Gauss-Jordan. Dimana setelah melakukan OBE untuk memperoleh matriks eselon baris tereduksi, diperoleh matriks *augmented* akhir seperti diatas, namun terlihat pada kolom ke-1 dan ke-3 berisi angka 0 semua, sehingga dapat diketahui bahwa berapapun nilai  $x_1$  dan  $x_3$ , SPL akan selalu terpenuhi. Setelah mengganti variabel independen  $x_1$ ,  $x_3$ , dan  $x_6$  dengan huruf w, v, dan u, dapat dibentuk solusi dari SPL tersebut sesuai dengan gambar di atas. Jika kita menggunakan metode invers balikan ataupun cramer, program akan otomatis berhenti melakukan operasi, karena matriks A bukanlah matriks persegi, sehingga determinannya tidak dapat ditentukan.

- d. Untuk Matriks Hilbert dengan  $n = 6$  :

Input :

```
matriksHilbert.txt
1.0 0.5 0.33333 0.25 0.2 0.16667 1.0
0.5 0.33333 0.25 0.2 0.16667 0.14286 0.0
0.33333 0.25 0.2 0.16667 0.14286 0.125 0.0
0.25 0.2 0.16667 0.14286 0.125 0.11111 0.0
0.2 0.16667 0.14286 0.125 0.11111 0.1 0.0
0.16667 0.14286 0.125 0.11111 0.1 0.09091 0.0
```

```
ain.java x matriksHilbert.txt x SPL
1 1/2 1/3 1/4 1/5 1/6 1
1/2 1/3 1/4 1/5 1/6 1/7 0
1/3 1/4 1/5 1/6 1/7 1/8 0
1/4 1/5 1/6 1/7 1/8 1/9 0
1/5 1/6 1/7 1/8 1/9 1/10 0
1/6 1/7 1/8 1/9 1/10 1/11 0
```

Output :

```
8.428489850821173E-18
x1 = 31.658258528519585
x2 = -462.95819954944665
x3 = 2101.7210263750785
x4 = -4111.767552226272
x5 = 3641.3235901118746
x6 = -1200.387942888392
```

Analisis :

Pertama - tama program menerima input berupa file dengan nama “matriksHilbert.txt” kemudian mengeksekusi SPL tersebut dengan melakukan metode Cramer. Dimana setelah memperoleh determinan matriks A yakni 8.4284...E-18 dengan menggunakan metode ekspansi kofaktor, dimana untuk memperoleh determinannya cukup dengan menjumlahkan seluruh hasil perkalian elemen matriks A dengan kofaktor A pada baris ke-1. Program lalu mengganti nilai kolom 1 dengan matriks b lalu mencari determinannya dengan metode yang sama, x1 kemudian diperoleh dengan membagi determinan matriks A yg kolom 1 nya telah diganti tadi dengan determinan matriks A sebelumnya, begitu pula untuk x<sub>2</sub> sampai x<sub>n</sub> diperoleh dengan cara yang sama. Untuk metode invers balikan juga diperoleh hasil yang sama, hanya saja prosesnya sedikit berbeda, yakni dengan mengalikan matriks A-1 dengan matriks b sehingga diperoleh matriks yang berisi nilai x<sub>1</sub> sampai x<sub>n</sub> yang merupakan solusi SPL, adapun A-1 didapat dengan menggunakan metode

adjoint. Untuk metode eliminasi Gauss dan Gauss-Jordan juga diperoleh dengan hasil yang sama, dengan menggunakan OBE untuk memperoleh matriks eselon baris untuk metode eliminasi Gauss dan eselon baris tereduksi untuk metode eliminasi Gauss-Jordan.

Untuk Matriks Hilbert dengan  $n = 10$  :

Input :

```
matriksHilbert.txt
1.0 0.5 0.33333 0.25 0.2 0.16667 0.14286 0.125 0.11111 0.1 1.0
0.5 0.33333 0.25 0.2 0.16667 0.14286 0.125 0.11111 0.1 0.09091 0.0
0.33333 0.25 0.2 0.16667 0.14286 0.125 0.11111 0.1 0.09091 0.08333 0.0
0.25 0.2 0.16667 0.14286 0.125 0.11111 0.1 0.09091 0.08333 0.07692 0.0
0.2 0.16667 0.14286 0.125 0.11111 0.1 0.09091 0.08333 0.07692 0.07143 0.0
0.16667 0.14286 0.125 0.11111 0.1 0.09091 0.08333 0.07692 0.07143 0.06667 0.0
0.14286 0.125 0.11111 0.1 0.09091 0.08333 0.07692 0.07143 0.06667 0.0625 0.0
0.125 0.11111 0.1 0.09091 0.08333 0.07692 0.07143 0.06667 0.0625 0.05882 0.0
0.11111 0.1 0.09091 0.08333 0.07692 0.07143 0.06667 0.0625 0.05882 0.05556 0.0
0.1 0.09091 0.08333 0.07692 0.07143 0.06667 0.0625 0.05882 0.05556 0.05263 0.0
```

Output :

```
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 35.73766578769422
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -526.5708850183961
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 2069.7894563379473
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 -2617.70454687687
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 287.08070766957246
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 549.5091688759367
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 369.8634077167321
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 -45.441676689163614
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 950.7439931596857
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -1087.4748267111306
x1 = 35.73766578769422
x2 = -526.5708850183961
x3 = 2069.7894563379473
x4 = -2617.70454687687
x5 = 287.08070766957246
x6 = 549.5091688759367
x7 = 369.8634077167321
x8 = -45.441676689163614
x9 = 950.7439931596857
x10 = -1087.4748267111306
```

Analisis :

Pertama program menerima input berupa file dengan nama “matriksHilbert.txt” kemudian mengeksekusi SPL tersebut dengan melakukan metode eliminasi Gauss-Jordan. Dimana setelah

melakukan OBE untuk memperoleh matriks eselon baris tereduksi, diperoleh matriks *augmented* akhir seperti diatas, terlihat dari matriksnya bahwa SPL tersebut memiliki solusi yang unik. Begitu pula jika kita menggunakan metode eliminasi Gauss, akan diperoleh hasil yang sama, hanya saja dalam pengerjaannya OBE hanya dilakukan sampai memperoleh matriks eselon baris saja, lalu dilakukan substitusi mundur dari baris paling bawah. Metode matriks balikan dan Cramer juga memberi hasil yang sama, tapi dengan waktu pemrosesan yang cukup lama, karena pada metode Cramer, dalam mencari determinan tiap - tiap matriks, digunakan metode ekspansi kofaktor yang merupakan fungsi rekursif, sehingga cukup memakan waktu, untuk metode matriks balikan waktu pemrosesannya cukup cepat karena dalam mencari matriks invers A digunakan metode *row reduction* / OBE, sehingga waktu yang dimakan jauh lebih cepat dari metode Cramer, tetapi tetap sedikit lebih lambat dari metode eliminasi Gauss dan eliminasi Gauss-Jordan.

Namun setelah dilakukan sulih ke persamaan 1 pada SPL, hasil yang didapat tidak tepat 1.0 melainkan 1.005067230579229 . Ketidakakuratan ini diakibatkan oleh floating point precision yang terjadi ketika angka sangat kecil atau angka sangat besar. Tidak ada cara untuk menghindari ketidakakuratan ini selain menggunakan tipe bentukan baru yang kompleks.

## Soal 2

a. Input :

```
Baris: 4
Kolom: 5
1 -1 2 -1 -1
2 1 -2 -2 -2
-1 2 -4 1 1
3 0 0 -3 -3
```

Output :

```
1.0 0.0 0.0 -1.0 -1.0
0.0 1.0 -2.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
x4 = w
x3 = v
x1 = -1.0 + 1.0w
x2 = 2.0v
```

Analisis:

Program mengeksekusi SPL di atas dengan melakukan metode eliminasi Gauss-Jordan. Dimana setelah melakukan OBE untuk memperoleh matriks eselon baris, diperoleh matriks *augmented* akhir seperti diatas, namun terlihat pada baris 3 dan 4,  $0x_1 + 0x_2 + 0x_3 + 0x_4 = 0$ , sehingga SPL tersebut memiliki banyak solusi. Setelah mengganti variabel independen  $x_4$  dan  $x_3$  dengan huruf w dan v, dapat dibentuk solusi dari SPL tersebut sesuai dengan gambar di atas. Jika kita menggunakan metode invers balikan ataupun cramer, program akan otomatis berhenti melakukan operasi, karena matriks A memiliki determinan yang bernilai 0, sehingga tidak mungkin memperoleh solusi melalui metode tersebut.

b. Input :

```
Baris: 6
Kolom: 5
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 :

```
1.0 0.0 4.0 0.0 4.0
0.0 1.0 0.0 4.0 6.0
0.0 0.0 1.0 0.0 1.0
0.0 0.0 0.0 1.0 1.0
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
Solutions can be determined
x1=0.0
x2=2.0
x3=1.0
x4=1.0
```

Analisis :

Program mengeksekusi SPL di atas dengan melakukan metode eliminasi Gauss, Dimana setelah melakukan OBE untuk memperoleh matriks eselon baris, diperoleh matriks *augmented* akhir seperti diatas, namun meskipun terlihat pada baris 5 dan 6,  $0x_1 + 0x_2 + 0x_3 + 0x_4 = 0$ , SPL tersebut tetap memiliki solusi unik karena pada baris 1 hingga 4 nya terbentuk matriks eselon baris, sehingga ketika dilakukan substitusi mundur dari baris 4, dapat diperoleh  $x_1$ ,  $x_2$ ,  $x_3$ , dan  $x_4$ .



Begitu pula jika kita menggunakan metode eliminasi Gauss-Jordan, akan mendapat hasil yang sama. Namun jika kita menggunakan metode invers balikan ataupun cramer, program akan otomatis berhenti melakukan operasi, karena matriks A bukanlah matriks persegi, sehingga determinannya tidak dapat ditentukan.

### Soal 3

a. Input :

```
Baris: 4
Kolom: 5
8 1 3 2 0
2 9 -1 -2 1
1 3 2 -1 2
1 0 6 4 3
```

Output :

```
0.13783783783783782 -0.018918918918918906 0.010810810810810811 -0.07567567567567568
-0.03378378378378378 0.14189189189189189 -0.08108108108108109 0.06756756756756757
-0.020270270270270268 -0.11486486486486486 0.3513513513513513 0.04054054054054054
-0.0040540540540540525 0.177027027027027 -0.5297297297297298 0.20810810810810812
x1 = -0.2243243243243243
x2 = 0.18243243243243243
x3 = 0.7094594594594593
x4 = -0.25810810810810814
```

Analisis :

Program mengeksekusi SPL di atas dengan melakukan metode matriks balikan. Dimana Pertama - tama program meng-*generate* matriks A-1, yang hasilnya tampak pada gambar di atas, lalu mengalikannya dengan matriks b, sehingga diperoleh solusi  $x_1$ ,  $x_2$ ,  $x_3$ , dan  $x_4$  di atas. Untuk metode Cramer, solusi yang dihasilkan sama, namun dengan cara yang berbeda, yakni dengan mencari determinan matriks A terlebih dahulu, lalu mengganti kolom 1 matriks A dengan matriks b, lalu mencari determinannya,  $x_1$  kemudian diperoleh dengan membagi determinan matriks A yang kolom 1 nya telah diganti dengan determinan matriks A. begitu pula dengan  $x_2$  hingga  $x_n$  diperoleh dengan cara yang sama. Sedangkan untuk metode eliminasi Gauss dan Gauss-Jordan juga diperoleh dengan hasil yang sama, dengan menggunakan OBE untuk memperoleh matriks eselon baris untuk metode eliminasi Gauss dan eselon baris tereduksi untuk metode eliminasi Gauss-Jordan.

b. Input :

```
Baris: 12
Kolom: 10
0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 13.0
0.0 0.0 0.0 1.0 1.0 1.0 0.0 0.0 0.0 15.0
1 1 1 0 0 0 0 0 0 8
0.0 0.0 0.04289 0.0 0.04289 0.75 0.04289 0.75 0.61396 14.79
0.0 0.25 0.91421 0.25 0.91421 0.25 0.91421 0.25 0.0 14.31
0.61396 0.75 0.04289 0.75 0.04289 0.0 0.04289 0.0 0.0 3.81
0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 1.0 18.0
0.0 1.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 12.0
1.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 6.0
0.04289 0.75 0.61396 0.0 0.04289 0.75 0.0 0.0 0.04289 10.51
0.91421 0.25 0.0 0.25 0.91421 0.25 0.0 0.25 0.91421 16.13
0.04289 0.0 0.0 0.75 0.04289 0.0 0.61396 0.75 0.04289 7.04
```

Output :

```
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 24.00406675250905
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 768.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 256.0
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 -256.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 -252.81164523299117
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 -256.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 285.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 -16.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 532.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1792.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -4432.0
SPL tidak memiliki solusi
```

Analisis :

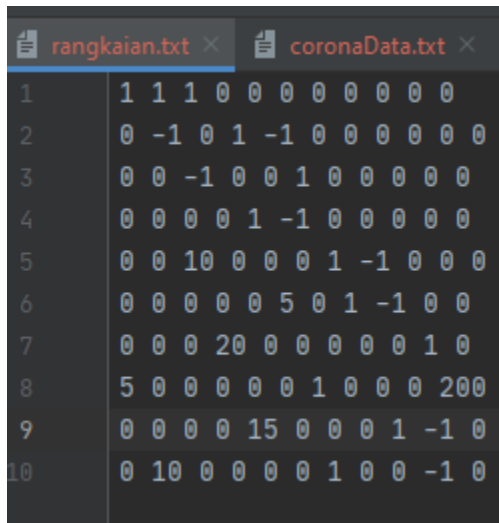
Program mengeksekusi SPL di atas dengan melakukan metode eliminasi Gauss-Jordan. Dimana setelah melakukan OBE untuk memperoleh matriks eselon baris tereduksi, diperoleh matriks *augmented* akhir seperti diatas, namun terlihat pada baris ke-10 hingga ke-12,  $0x_1 + 0x_2 + 0x_3 + \dots + 0x_9 \neq 0$ , sehingga SPL tidak memiliki solusi, begitu pula jika kita menggunakan metode eliminasi Gauss-Jordan, akan diperoleh matriks eselon baris tereduksi yang kondisinya sama, yang menyebabkan SPL tidak memiliki solusi. Jika kita menggunakan metode invers balikan

ataupun cramer, program akan otomatis berhenti melakukan operasi, karena matriks A bukanlah matriks persegi, sehingga determinannya tidak dapat ditentukan.

## Soal 4

Input :

```
Masukkan nama file :  
rangkaian.txt
```



1	1	1	1	0	0	0	0	0	0	0	0
2	0	-1	0	1	-1	0	0	0	0	0	0
3	0	0	-1	0	0	1	0	0	0	0	0
4	0	0	0	0	1	-1	0	0	0	0	0
5	0	0	10	0	0	0	1	-1	0	0	0
6	0	0	0	0	0	5	0	1	-1	0	0
7	0	0	0	20	0	0	0	0	0	1	0
8	5	0	0	0	0	0	1	0	0	0	200
9	0	0	0	0	15	0	0	0	1	-1	0
10	0	10	0	0	0	0	1	0	0	-1	0

Output :

```
1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 6.153846153846155  
0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -4.615384615384617  
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -1.5384615384615383  
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -6.153846153846155  
0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 -1.5384615384615383  
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 -1.5384615384615383  
0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 169.23076923076923  
0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 153.84615384615384  
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 146.15384615384616  
-0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 1.0 123.0769230769231  
x1 = 6.153846153846155  
x2 = -4.615384615384617  
x3 = -1.5384615384615383  
x4 = -6.153846153846155  
x5 = -1.5384615384615383  
x6 = -1.5384615384615383  
x7 = 169.23076923076923  
x8 = 153.84615384615384  
x9 = 146.15384615384616  
x10 = 123.0769230769231
```

Analisis :

Pertama - tama program menerima input berupa file dengan nama “rangkaian.txt” kemudian mengeksekusi SPL tersebut dengan melakukan metode eliminasi Gauss-Jordan. Dimana setelah melakukan OBE untuk memperoleh matriks eselon baris tereduksi, diperoleh matriks *augmented* akhir seperti diatas, terlihat dari matriksnya bahwa SPL tersebut memiliki solusi yang unik. Begitu pula jika kita menggunakan metode eliminasi Gauss, akan diperoleh hasil yang sama, hanya saja dalam pengerjaannya OBE hanya dilakukan sampai memperoleh matriks eselon baris saja, lalu dilakukan substitusi mundur dari baris paling bawah. Metode matriks balikan dan Cramer juga memberi hasil yang sama, tapi dengan waktu pemrosesan yang cukup lama, karena pada metode Cramer, dalam mencari determinan tiap - tiap matriks, digunakan metode ekspansi kofaktor yang merupakan fungsi rekursif, sehingga cukup memakan waktu, untuk metode matriks balikan waktu pemrosesannya cukup cepat karena dalam mencari matriks invers A digunakan metode *row reduction* / OBE, sehingga waktu yang dimakan jauh lebih cepat dari metode Cramer, tetapi tetap sedikit lebih lambat dari metode eliminasi Gauss dan eliminasi Gauss-Jordan. Solusi  $x_1$  merepresentasikan nilai  $i_{12}$ ,  $x_2$  merepresentasikan nilai  $i_{52}$ ,  $x_3$  merepresentasikan nilai  $i_{32}$ ,  $x_4$  merepresentasikan nilai  $i_{65}$ ,  $x_5$  merepresentasikan nilai  $i_{54}$ ,  $x_6$  merepresentasikan nilai  $i_{43}$ ,  $x_7$  merepresentasikan  $V_2$ ,  $x_8$  merepresentasikan  $V_3$ ,  $x_9$  merepresentasikan  $V_4$ ,  $x_{10}$  merepresentasikan  $V_5$ .

## Soal 5

Input Titik :

```
Jumlah titik: 7
Masukkan titik ke-1
0.1 0.005
Masukkan titik ke-2
0.3 0.067
Masukkan titik ke-3
0.5 0.148
Masukkan titik ke-4
0.7 0.248
Masukkan titik ke-5
0.9 0.370
Masukkan titik ke-6
1.1 0.518
Masukkan titik ke-7
1.3 0.697
```

Input x yang ingin diperkirakan nilai  $f(x)$  nya :

a.  $P_6(x) = -0.023 + 0.24x + 0.1974x^2 + 0.026x^4$   
Input x: 0.2

Output :

$P_6(0.2) = 0.032937600000000004$

b.  $P_6(x) = -0.023 + 0.24x + 0.1974x^2 + 0.026x^4$   
Input x: 0.55

Output :

$P_6(0.55) = 0.17109266250000002$

c.  $P_6(x) = -0.023 + 0.24x + 0.1974x^2 + 0.026x^4$   
Input x: 0.85

Output :

$P_6(0.85) = 0.3371936625$

d.  $P_6(x) = -0.023 + 0.24x + 0.1974x^2 + 0.026x^4$   
Input x: 1.28

Output :

$P_6(1.28) = 0.67741337856$

Analisis :

Program pertama - tama menerima masukan 7 titik, kemudian membentuk 7 buah persamaan sesuai dengan teori yang telah dikemukakan pada bab sebelumnya, program kemudian mencari nilai  $a_0$ ,  $a_1$ , hingga  $a_6$  dengan menggunakan metode eliminasi Gauss, kemudian mencetak persamaannya seperti yang tampak pada gambar - gambar diatas, kemudian program menerima input x yang akan dicari nilai  $f(x)$  nya sesuai dengan persamaan yang sudah diproses oleh program, dengan menyulihkan nilai x yang ingin dicari  $f(x)$  nya ke dalam persamaan  $P_6(x)$ . Sehingga untuk beberapa kasus x yang ada pada soal, didapatkan  $P_6(x)$  yang tertera pada gambar - gambar di atas.

## Soal 6

Input :

Untuk  $n = 5$  :

```
Masukkan nama file :  
nilaiFungsi.txt
```

```
nilaiFungsi.txt x corona  
1 0 0  
2 0.4 0.4188842301  
3 0.8 0.5071579685  
4 1.2 0.5609246748  
5 1.6 0.5836856613  
6 2.0 0.5766515298
```

Output :

```
P5(x)= 0.0+2.0353x-3.5527x^2+3.2371x^3-1.4213x^4+0.2363x^5
```

Analisis :

Program pertama - tama menerima masukan 6 titik, kemudian membentuk 6 buah persamaan sesuai dengan teori yang telah dikemukakan pada bab sebelumnya, program kemudian mencari nilai  $a_0$ ,  $a_1$ , hingga  $a_5$  dengan menggunakan metode eliminasi Gauss, kemudian mencetak persamaan  $P_6(x)$  seperti yang tampak pada gambar - gambar diatas.  $P_6(x)$  inilah yang merupakan penyederhanaan dengan derajat polinom 5 dari fungsi

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

## Soal 7

Input :

```
Masukkan nama file :  
coronaData.txt
```

	coronaData.txt	matr
1	4.8	8211
2	5	10118
3	5.516	17025
4	5.71	20796
5	6.5	39294
6	7.194	64958
7	8.097	113134
8	8.258	123503
9	9.033	177571
10	9.333	145510

Output persamaan yang terbentuk :

$$P9(x) = 2.270963744503E8 - 4.158426380969E8x + 3.18150063216E8x^2 - 1.360032836955E8x^3 + 3.61760389242E7x^4 - 6249554.4666x^5 + 704212.2778x^6 - 50061.9546x^7 + 2041.9197x^8 - 36.471x^9$$

Input tanggal untuk prediksi jumlah kasus Covid-19 :

- a. 25/05/20 (x = 5.806) :

```
Input x: 5.806
P9(5.806) = 22427.97900336981
```

- b. 30/08/20 (x = 8.967)

```
Input x: 8.967
P9(8.967) = 158231.59097480774
```

- c. 15/09/20 (x = 9.500)

```
Input x: 9.5
P9(9.5) = 38932.19631576538
```

- d. 18/06/20 (x = 6.6)

```
Input x: 6.6
P9(6.6) = 41077.9001108408
```

- e. 12/04/20 (x = 4.387)

```
Input x: 4.387
P9(4.387) = -5060.706604283303
```

Analisis :

Program pertama - tama menerima masukan 10 titik dari file “coronaData.txt” yang merupakan data jumlah kasus Covid-19 pada tanggal tertentu yang sudah dikonversi ke dalam bilangan desimal, program kemudian membentuk 10 buah persamaan sesuai dengan teori yang telah dikemukakan pada bab sebelumnya, program kemudian mencari nilai  $a_0$ ,  $a_1$ , hingga  $a_9$  dengan menggunakan metode eliminasi Gauss, kemudian mencetak persamaannya seperti yang tampak pada gambar - gambar diatas, kemudian program menerima input  $x$  yang merupakan tanggal (desimal) yang sudah diolah yang akan diprediksi jumlah kasus Covid-19 nya pada tanggal itu. Tampak pada gambar di atas program sudah dapat memprediksi jumlah kasus Covid-19 pada tanggal - tanggal tertentu dengan cukup baik. Akan tetapi, terdapat anomali pada input tanggal 15/09/20 ( $x = 9.500$ ), dimana terjadi penurunan jumlah kasus yang cukup besar yang tidak masuk akal jika dibandingkan dengan data yang diberikan, begitu pula hal yang sama terjadi pada input tanggal 12/04/20 ( $x = 4.387$ ), dimana persamaannya memberi hasil negatif yang tidak mungkin terjadi. Hal ini disebabkan karena interpolasi polinom ini hanya dapat digunakan untuk memprediksi nilai  $y$  di sembarang titik pada selang  $[x_0, x_n]$ , untuk kasus ini  $x_0$  nya adalah 4,8 (tanggal 24/04/20), dan  $x_n$  nya adalah 9,333 (tanggal 10/09/20). Sehingga, kita tidak dapat memprediksi dengan baik jumlah kasus Covid-19 pada tanggal - tanggal sebelum tanggal 24/04/20 atau sesudah tanggal 10/09/20.

## Soal 8

Input :



```

Jumlah variabel penentu: 3
Jumlah sampel data: 20
72.4 76.3 29.18 0.9
41.6 78.3 29.35 0.91
34.3 77.1 29.24 0.96
35.1 68.8 29.27 0.89
10.7 79.8 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 29.89 0.77
24.8 67.7 29.6 1.07
23.2 76.8 29.38 1.07
47.4 86.6 29.35 0.94
31.5 76.9 29.63 1.10
10.6 86.3 29.56 1.10
11.2 86.8 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 29.83 0.82
54.9 78.9 29.37 0.95

```

Output :

```

y = -3.5078 - 0.0026x1 + 8.0E-4x2 + 0.1542x3
Input x1: 50
Input x2: 76
Input x3: 29.3
y(x1, x2, x3) = 0.94106

```

Analisis :

Program pertama - tama menerima masukan data sampel dengan format per baris  $x_{1i}$   $x_{2i}$   $x_{3i}$   $y_i$ . Kemudian program akan membentuk *Normal Estimation Equation for Multiple Linear Regression* dan menggunakan metode eliminasi Gauss untuk mencari nilai  $\beta_0$ ,  $\beta_1$ ,  $\beta_2$ , dan  $\beta_3$ . Kemudian program mencetak persamaan yang terbentuk seperti pada gambar diatas, lalu menerima input  $x_1$ ,  $x_2$ , dan  $x_3$  lalu mencari estimasi  $y$  (nilai Nitrous Oxide pada soal ini) pada kondisi  $x_1$ ,  $x_2$ , dan  $x_3$ . Program lalu menyulihkan nilai  $x_1$ ,  $x_2$ , dan  $x_3$  ke persamaan yang sudah dicetak sebelumnya, sehingga diperoleh  $y(x_1, x_2, x_3) = 0.94106$  seperti yang tampak pada gambar.

# BAB V SIMPULAN, SARAN, DAN REFLEKSI

## Sistem Persamaan Linear

Sembarang Sistem Persamaan Linear dengan persamaan

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2$$

:

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m$$

dapat diselesaikan dengan beberapa cara yaitu metode Eliminasi Gauss, metode Eliminasi Gauss-Jordan, metode matriks balikan, dan metode Kaidah Cramer. Pertama bentuk persamaan linear harus diubah ke bentuk matriks

$$Ax = b$$

Dengan A adalah matriks koefisien dari x, x adalah matriks x dan b adalah matriks konstanta. Metode tertentu memiliki syarat-syarat tertentu yang harus dipenuhi. Untuk metode eliminasi Gauss dan eliminasi Gauss Jordan tidak ada syarat khusus untuk menggunakan metode tersebut. Sedangkan, untuk metode matriks balikan ada syarat yaitu matriks A harus berupa matriks persegi dengan ordo  $n \times n$  agar bisa didapat matriks inversnya, dan untuk metode Kaidah Cramer memiliki syarat yaitu determinan matriks A tidak boleh nol. Dari hasil yang didapat sebelumnya, dapat disimpulkan bahwa keempat metode tersebut dapat memberikan hasil yang benar.

## Interpolasi Polinom

Jika terdapat  $n+1$  buah titik berbeda  $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$ , untuk mengetahui perkiraan nilai y di titik dalam selang  $[x_0, x_n]$  dapat dilakukan dengan menentukan polinom  $p_n(x)$ . Polinom  $p_n(x)$  didapat dengan cara menginterpolasi semua titik-titik tersebut sedemikian sehingga  $y_i = p_n(x_i)$  untuk  $i = 0, 1, 2, \dots, n$ . Polinom berderajat n yang menginterpolasi titik-titik  $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$  adalah

$$a_0 + a_1x_0 + a_2x_0^2 + \dots + a_n x_0^n = y_0$$

$$a_0 + a_1x_1 + a_2x_1^2 + \dots + a_n x_1^n = y_1$$

:

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

Setelah dilakukan penyulihan terdapat tiap titiknya, dan melakukan operasi, akan didapat persamaan polinom :

$$p_n(x) = y_n = a_0 + a_1x^1 + a_2x^2 + \dots + a_nx^n$$

Dengan persamaan tersebut, dapat ditaksir nilai y dari suatu titik dalam selang  $[x_0, x_n]$ . Dari hasil yang telah didapat, penyelesaian interpolasi polinom dapat menggunakan eliminasi Gauss. Interpolasi polinom pada program yang dibuat dapat memberi jawaban dengan benar dan program yang dibuat telah bekerja.

## Regresi Linear Berganda

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

Regresi Linear Berganda adalah salah satu metode untuk memprediksi nilai selain Interpolasi Polinom.

Gambar V.1. *Normal Estimation Equation for Multiple Linear Regression*

Dengan memasukkan n peubah x, i sampel data, dan masing-masing nilai  $x_{ni}$ ,  $y_i$ , dan nilai x yang akan ditaksir nilainya, dapat membentuk matriks yang kemudian dapat diselesaikan dengan menggunakan eliminasi Gauss. Dari hasil yang didapat, dapat disimpulkan bahwa Regresi Linear Berganda dapat menggunakan metode eliminasi Gauss yang telah terimplementasikan pada program yang telah dibuat.

Saran untuk pengembangan berikutnya, test case yang diberikan bisa ditambahkan test case dengan angka yang lebih besar sehingga dapat dilakukan pemeriksaan akurasi dari nilai floating point.

Melalui Tugas Besar 1 Aljabar Linear dan Geometri ini, kami dapat lebih memahami implementasi konkrit ilmu-ilmu dalam mata kuliah ini pada kasus riil seperti melakukan perhitungan arus pada rangkaian menggunakan SPL dan prediksi kasus positif COVID-19. Hal ini menunjukkan bahwa digitalisasi berbagai hal dalam kehidupan manusia sangat memungkinkan jika diketahui ilmunya. Dengan hal tersebut berbagai permasalahan dalam kehidupan dapat lebih mudah dan cepat diselesaikan.

# DAFTAR REFERENSI

<http://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-05-Sistem-Persamaan-Linier-2.pdf>. Diakses tanggal 29 September 2020 pukul 21.45 WIB.

<http://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-08-Determinan-bagian1.pdf>. Diakses tanggal 29 September 2020 pukul 21.53 WIB.

<http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm>. Diakses tanggal 29 September 2020 pukul 22.35 WIB.

Howard Anton and Chris Rorres, Elementary Linear Algebra, 12th ed, John Wiley and Sons, 2019