LAPORAN TUGAS BESAR 1
IF2123 ALJABAR LINEAR DAN GEOMETRI

Kelompok 54 Kopi Jawa

Mayla Yaffa Ludmilla 13523050

Diyah Susan Nugrahani 13523080

Muhammad Izzat Jundy 13523092

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. Andstrea 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, *library* tersebut akan diterapkan di dalam program Java untuk menyelesaikan berbagai persoalan yang dimodelkan dalam bentuk SPL, menyelesaikan persoalan interpolasi, dan persoalan regresi.

BAB II
TEORI SINGKAT

2.1 Sistem Persamaan Linear

2.1.1 Metode Eliminasi Gauss

Metode eliminasi gauss adalah metode yang ditemukan oleh Carl Friedrich Gauss untuk memecahkan sistem persamaan linear dengan merepresentasikan (mengubah) menjadi bentuk matriks. Matriks tersebut lalu diubah ke bentuk Eselon Baris melalui Operasi Baris Elementer. Cara membuat eselon baris adalah dengan mengubah bentuk matriksnya menjadi lebih sederhana yaitu dengan membuat elemen tak 0 paling kiri dalam tiap barisnya bernilai 1, nilai ini disebut sebagai 1 utama . Setelah itu sistem diselesaikan dengan substitusi balik. Bentuk umum dari penyelesaian SPL dengan metode eliminasi Gauss adalah sebagai berikut, dengan tanda * menunjukkan bahwa nilai tersebut bebas diisi oleh angka berapapun.

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

2.1.2 Metode Eliminasi Gauss-Jordan

Metode Eliminasi Gauss-Jordan adalah salah satu teknik yang dapat digunakan dalam menyelesaikan persoalan Solusi Persamaan Linear dalam bentuk matriks. Metodenya hampir sama dengan metode eliminasi gauss jordan, perbedaannya adalah tidak hanya elemen-elemen yang berada dibawah 1 utama saja yang dibuat 0, namun juga elemen-elemen yang berada diatas 1 utama dibuat menjadi 0 menggunakan OBE. Kelebihan dari metode ini adalah tidak perlu melakukan substitusi mundur karena solusinya dapat dicari dengan cara melihat elemen-elemen

pada kolom terakhir. Bentuk umum penyelesaian SPL dengan metode eliminasi Gauss-Jordan adalah sebagai berikut, dengan tanda * menunjukkan elemen kolom terakhir yang dapat digunakan untuk menentukan solusi.

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

2.2 Determinan

2.2.1 Determinan Metode OBE

Metode Operasi Baris Elementer (OBE) merupakan salah satu metode yang dapat digunakan untuk menghitung determinan dari sebuah matriks. Metode ini menggunakan pendekatan matematis untuk mengubah matriks awal menjadi bentuk segitiga atas atau segitiga bawah dengan mengaplikasikan operasi baris elementer seperti pertukaran baris, perkalian baris dengan suatu konstanta, atau penambahan/subtraksi baris. Ketika matriks mencapai bentuk segitiga atas atau segitiga bawah, determinan dapat dihitung dengan mengalikan elemen-elemen diagonal utama.

$$[A] \overset{\text{OBE}}{\sim} [\text{matriks segitiga bawah}]$$

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \overset{\text{OBE}}{\sim} \begin{bmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} \\ 0 & a'_{22} & \dots & a'_{2n} \\ \vdots & \vdots & \dots & a'_{3n} \\ 0 & 0 & 0 & a'_{nn} \end{bmatrix}$$

$$\det(A) = (-1)^p \, a'_{11} a'_{22} \dots a'_{nn}$$

2.2.2 Determinan Metode Kofaktor

Metode kofaktor dapat dimanfaatkan untuk mencari nilai determinan dari sebuah matriks. Konsepnya adalah kita memilih satu baris atau kolom sebagai elemen utama dalam ekspansi

kofaktor. Kemudian elemen tersebut kita kalikan dengan kofaktor dari sub matriks yang lebih kecil yang diperoleh dengan menghapus baris dan kolom yang mengandung elemen tersebut. Lalu dikalikan dengan mempertimbangkan tanda positif dan negatif sesuai dengan posisi elemen tersebut. Terakhir jumlahkan semua hasil perkalian tersebut untuk mendapatkan nilai determinan matriks.

$$\det(A) = a_{11}C_{11} + a_{12}C_{12} + \dots + a_{1n}C_{1n}$$
$$\det(A) = a_{21}C_{21} + a_{22}C_{22} + \dots + a_{2n}C_{2n}$$
$$\vdots$$
$$\det(A) = a_{n1}C_{n1} + a_{n2}C_{n2} + \dots + a_{nn}C_{nn}$$

secara baris

$$\det(A) = a_{11}C_{11} + a_{21}C_{21} + \dots + a_{n1}C_{n1}$$
$$\det(A) = a_{12}C_{12} + a_{22}C_{22} + \dots + a_{n2}C_{n2}$$
$$\vdots$$
$$\det(A) = a_{1n}C_{1n} + a_{2n}C_{2n} + \dots + a_{nn}C_{nn}$$

secara kolom

2.3 Matriks Balikan

Matriks balikan adalah matriks yang apabila dikalikan dengan matriks asalnya akan menghasilkan matriks identitas, yaitu matriks yang semua diagonal utamanya bernilai 1 dan elemen sisanya bernilai 0. Untuk mendapatkan matriks balikan dari suatu matriks, dibuat sebuah matriks augmented dengan matriks awal di sebelah kiri dan matriks identitas di sebelah kanan. Setelah itu, dilakukan metode eliminasi Gauss-Jordan hingga matriks sebelah kiri menjadi matriks identitas. Matriks di sebelah kanan setelah dilakukan metode eliminasi Gauss Jordan adalah matriks balikan atau invers dari matriks awal,

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

2.4 Matriks Kofaktor

Matriks kofaktor merupakan hasil perkalian minor dengan suatu angka yang nilainya dikalikan dengan -1^{i+j} dimana i adalah baris dan j adalah kolom. Minor sendiri merupakan determinan matriks bagian yang diperoleh dengan cara menghilangkan elemen-elemen pada baris ke-i dan kolom ke-j. Nilai dari kofaktor dari baris ke-i dan kolom ke-j dapat dituliskan sebagai berikut.

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

2.5 Matriks Adjoin

Matrik adjoin adalah transpose dari matriks kofaktor. Transpose matriks adalah pertukaran elemen elemen pada baris matriks menjadi di kolom atau sebaliknya dari kolom matriks menjadi di baris matriks.

2.6 Kaidah Cramer

Kaidah Cramer adalah salah satu metode untuk menyelesaikan sistem persamaan linear. Setiap variabel dinyatakan dengan hasil determinan dari matriks koefisien (dari persamaan)yang salah satu kolomnya diganti dengan vektor hasil persamaan linier, dibagi dengan determinan matriks koefisien tersebut.

$$x_i = \frac{\det(A_i)}{\det(A)} \qquad i = 1, \ldots, n$$

2.7 Interpolasi Polinom

Interpolasi Polinom adalah metode aproksimasi nilai fungsi berdasarkan titik titik yang diketahui. Untuk mendapatkan aproksimasi nilai fungsi, persamaan polinomial berdasarkan data yang dimiliki dibuat terlebih dahulu. Setelah itu, persamaan polinomial yang terbentuk dimanfaatkan untuk menginterpolasi dari nilai titik yang diketahui. Persamaan polinomial tersebut dapat dibentuk menjadi matriks.

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & \dots & x_1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ x_n^n & x_n^{n-1} & x_n^{n-2} & \dots & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

2.8 Interpolasi Bicubic Spline

Interpolasi bicubic spline adalah perluasan dari interpolasi kubik spline yang mengapromaksikan nilai fungsi dari nilai titik titik yang diketahui. Interpolasi bicubic spline memanfaatkan 16 titik, di mana 4 titik yang berada di tengah akan menjadi titik titik referensi utama dan 12 titik lainnya adalah aproksimasi turunan dari keempat titik referensi untuk membagun permukaan bikubik. Interpolasi ini juga memanfaatkan turunan berarah di x, y, dan xy dengan persamaan sebagai berikut

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

$$f_x(x,y)=\sum_{j=0}^{3}\sum_{i=1}^{3}a_{ij}ix^{i-1}y^{j}$$

$$f_y(x,y)=\sum_{j=1}^{3}\sum_{i=0}^{3}a_{ij}jx^{i}y^{j-1}$$

$$f_{xy}(x,y)=\sum_{j=0}^{3}\sum_{i=0}^{3}a_{ij}ijx^{i-1}y^{j-1}$$

.

Dengan menggunakan nilai fungsi dan turunan berarah tersebut, dapat terbentuk sebuah matriks solusi *X* yang membentuk persamaan penyelesaian.

$$y = Xa$$

$$\begin{bmatrix} f(0,0) \\ f(1,0) \\ f(0,1) \\ f(1,1) \\ f_x(0,0) \\ f_x(1,0) \\ f_x(0,1) \\ f_x(1,1) \\ f_y(0,0) \\ f_y(1,0) \\ f_y(0,1) \\ f_y(1,1) \\ f_{xy}(0,0) \\ f_{xy}(1,0) \\ f_{xy}(0,1) \\ f_{xy}(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 2 & 4 & 6 & 0 & 3 & 6 & 9 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \\ a_{01} \\ a_{11} \\ a_{21} \\ a_{31} \\ a_{02} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{03} \\ a_{13} \\ a_{23} \\ a_{33} \end{bmatrix}$$

2.9 Regresi Linear dan Kuadratik Berganda

2.9.1 Regresi Linear Berganda

Regresi linear berganda adalah model regresi linear dengan melibatkan lebih dari satu variabel bebas atau independen. Regresi linear berganda digunakan untuk memodelkan hubungan antara satu variabel dependen terhadap beberapa variabel independen. Dalam penerapan konsep SPL, data yang akan dihitung regresinya akan dinyatakan dalam bentuk matriks yang berisi semua variabel independen, termasuk intercept, dan vektor parameter yang

menyimpan koefisien regresi. Bentuk umum persamaan dari regresi linear berganda adalah sebagai berikut.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k + \varepsilon$$

Untuk mendapatkan nilai dari βi dapat digunakan *Normal Estimation Equation For Multiple Linear Regression* sebagai berikut

$$\begin{aligned}
n\hat{\beta}_0 + \hat{\beta}_1 \sum_{i=1}^{n} x_{i1} + \hat{\beta}_2 \sum_{i=1}^{n} x_{i2} + \cdots + \hat{\beta}_k \sum_{i=1}^{n} x_{ik} &= \sum_{i=1}^{n} y_i \\
\hat{\beta}_0 \sum_{i=1}^{n} x_{i1} + \hat{\beta}_1 \sum_{i=1}^{n} x_{i1}^2 + \hat{\beta}_2 \sum_{i=1}^{n} x_{i1} x_{i2} + \cdots + \hat{\beta}_k \sum_{i=1}^{n} x_{i1} x_{ik} &= \sum_{i=1}^{n} x_{i1} y_i \\
\vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots & \\
\hat{\beta}_0 \sum_{i=1}^{n} x_{ik} + \hat{\beta}_1 \sum_{i=1}^{n} x_{ik} x_{i1} + \hat{\beta}_2 \sum_{i=1}^{n} x_{ik} x_{i2} + \cdots + \hat{\beta}_k \sum_{i=1}^{n} x_{ik}^2 &= \sum_{i=1}^{n} x_{ik} y_i
\end{aligned}$$

2.9.2 Regresi Kuadratik Berganda

Regresi kuadratik berganda merupakan model regresi yang melibatkan lebih dari satu variabel bebas atau independen. Proses mengubah data-data dalam regresi kuadratik berganda cukup berbeda dengan regresi linear berganda. Bentuk persamaan dalam regresi kuadratik berganda ada tiga, yaitu :

a. Variabel Linear : variabel dengan derajat satu, seperti X, Y, dan Z
b. Variabel Kuadrat: Variabel dengan derajat dua seperti X^2
c. Variabel Interaksi: 2 Variabel dengan derajat satu yang dikalikan dengan satu sama lain seperti XY, YZ, dan XZ

Setiap n-peubah, jumlah variabel linier, kuadrat, dan interaksi akan

berbeda-beda. Perhatikan contoh regresi kuadratik 2 variabel peubah sebagai berikut!

$$\begin{pmatrix}
N & \sum u_i & \sum v_i & \sum u_i^2 & \sum u_i v_i & \sum v_i^2 \\
\sum u_i & \sum u_i^2 & \sum u_i v_i & \sum u_i^3 & \sum u_i^2 v_i & \sum u_i v_i^2 \\
\sum v_i & \sum u_i v_i & \sum v_i^2 & \sum u_i^2 v_i & \sum u_i v_i^2 & \sum v_i^3 \\
\sum u_i^2 & \sum u_i^3 & \sum u_i^2 v_i & \sum u_i^4 & \sum u_i^3 v_i & \sum u_i^2 v_i^2 \\
\sum u_i v_i & \sum u_i^2 v_i & \sum u_i v_i^2 & \sum u_i^3 v_i & \sum u_i^2 v_i^2 & \sum u_i v_i^3 \\
\sum v_i^2 & \sum u_i v_i^2 & \sum v_i^3 & \sum u_i^2 v_i^2 & \sum u_i v_i^3 & \sum v_i^4
\end{pmatrix}
\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix}
=
\begin{pmatrix}
\sum y_i \\ \sum y_i u_i \\ \sum y_i v_i \\ \sum y_i u_i^2 \\ \sum y_i u_i v_i \\ \sum y_i v_i^2
\end{pmatrix}$$

N menandakan jumlah peubah, terdapat 2 variabel linier yaitu u_i dan v_i, 2 variabel kuadrat yaitu u_i^2 dan v_i^2, dan 1 variabel interaksi yaitu uv. Untuk setiap n-peubah, akan terdapat 1 konstan N (Terlihat di bagian atas kiri gambar), n variabel linier, n variabel kuadrat, dan C_2^n variabel linier (dengan syarat n > 1). Tentu dengan bertambahnya peubah n, ukuran matriks akan bertumbuh lebih besar dibandingkan regresi linier berganda tetapi solusi tetap bisa didapat dengan menggunakan SPL.

BAB III
IMPLEMENTASI PUSTAKA DAN PROGRAM DALAM JAVA

3.1 Folder ADTMatrix

3.1.1 Matrix.java

- Artibut

`matrix.row`	Jumlah baris pada matrix
`matrix.colom`	Jumlah kolom pada matrix

- Konstruktor

`public double MARK = Double.NaN;`	Menginisialisasi nilai MARK sebagai NaN
`public void toMatrix (double a[][], int row, int col)`	Membentuk matriks dari inputan array dua dimensi
`public void createMatrix (int row, int col)`	Membuat matriks baru dengan ukuran *row* kali *column*.

- Metode

Nama Procedure/Function	Deskripsi
`public int getRowLength()`	Fungsi untuk mengembalikan jumlah baris
`public int getColLength()`	Fungsi untuk mengembalikan jumlah kolom
`public int getLastRowIndex()`	Fungsi untuk mengembalikan nilai index baris terakhir
`public int getLastColIndex()`	Fungsi untuk mengembalikan nilai index kolom terakhir
`public boolean isSquare()`	Fungsi untuk mengembalikan true jika matriks persegi, dan false jika tidak

`public double getElement (int row, int col)`	Fungsi untuk mengambil nilai elemen dari baris ke *row* dan kolom ke *col*
`public void setElement (int row, int col, double val)`	Prosedur untuk mengisi elemen matriks pada baris ke *row* dan kolom ke *col* dengan nilai *val*
`public void rowSwap (int row1, int row2)`	Prosedur untuk melakukan swap baris *row1* dengan *row2*
`public static Matrix multiplyMatrix (Matrix m1, Matrix m2)`	Fungsi untuk mengalikan dua buah matriks *m1* dan *m2* dan mengembalikan matriks hasil perkaliannya
`public void multiplyRow (int row, double pengali)`	Prosedur untuk mengalikan baris dengan pengali
`public static double sumRow (Matrix m, int row)`	Fungsi untuk menjumlahkan satu baris matriks dan mengembalikan nilainya
`public static double sumColumn (Matrix m, int col)`	Fungsi untuk menjumlahkan satu kolom matriks dan mengembalikan nilainya
`public static double sumMultiplyCol (Matrix m, int i, int j)`	Fungsi untuk menjumlahkan hasil perkalian dua kolom dan mengembalikan nilainya
`public double sumMultiplyRow(int r1, int r2, double multiplier)`	Prosedur untuk menjumlahkan hasil perkalian dua baris dan mengembalikan nilainya
`public static Matrix addRow (Matrix m, double[] row)`	Fungsi untuk menambahkan sebuah baris baru
`public static Matrix addCol (Matrix m, double[] col)`	Fungsi untuk menambahkan sebuah kolom baru
`public boolean isManySolution()`	Mengembalikan nilai true jika memiliki solusi parametrik

`public boolean isNoSolution()`	Mengembalikan nilai true jika tidak memiliki solusi
`public Matrix subMatrix (Matrix m, int row, int col)`	Fungsi untuk membuat sub matrix dengan menghilangkan satu baris dan satu kolom
`public static double detKofaktorIJ (Matrix m, int row, int col)`	Fungsi untuk menghitung determinan
`public Matrix matrixKofaktor (Matrix m)`	Fungsi untuk menghitung kofaktor dan mengembalikan matriks nya
`public Matrix Adjoin (Matrix m)`	Fungsi untuk mencari matriks Adjoin dan mengembalikan matriks nya
`public void cekMinNol()`	Fungsi untuk menangani kasus min nol
`public String[] solveManySolution()`	Fungsi untuk mencari solusi parametrik
`public Matrix gaussElimination()`	Fungsi mengembalikan hasil reduksi baris sehingga membentuk matriks eselon baris
`public Matrix gaussJordanElimination()`	Fungsi mengembalikan hasil reduksi baris sehingga membentuk matriks eselon baris tereduksi
`public static void backSubstitution (Matrix matrix, double[] X)`	Prosedur untuk melakukan substitusi balik
`public Matrix transpose (Matrix m)`	Fungsi untuk menghasilkan matriks transpose
`public static double[] multiplyVektor (Matrix m, double[] A)`	Fungsi untuk mengalikan sebuah matriks dengan vektor dan mengembalikan hasilnya

3.2 Folder Function

3.2.1 Bicubic

- Artibut

  -

- Konstruktor

  -

- Metode

Nama Procedure/Function	Deskripsi
`public static void f (Matrix m)`	Mengisi 4 baris pertama matriks m dengan nilai dari fungsi f di titik (0,0), (0,1), (1,0) dan (1,1)
`public static void fx (Matrix m)`	Mengisi 4 baris kedua matriks m dengan nilai dari turunan berarah fungsi f di x di titik (0,0), (0,1), (1,0) dan (1,1)
`public static void fy (Matrix m)`	Mengisi 4 baris ketiga matriks m dengan nilai dari turunan berarah fungsi f di y di titik (0,0), (0,1), (1,0) dan (1,1)
`public static void fxy (Matrix m)`	Mengisi 4 baris keempat matriks m dengan nilai dari turunan berarah fungsi f di x dan y di titik (0,0), (0,1), (1,0) dan (1,1)
`public static double BicubicSplineInterpolation (double[][] fInput)`	Mengaproksimasi nilai f di titik (x,y) berdasarkan matriks m yang sudah dibuat.

3.2.2 Determinan

- Artibut

  -

- Konstruktor

  -

- Metode

Nama Procedure/Function	Deskripsi
`public static double detKofaktor(Matrix m)`	Mencari determinan dengan metode kofaktor
`public static double detOBE(Matrix m)`	Mencari determinan dengan metode OBE segitiga atas

3.2.3 Interpolasi

- Artibut

  -

- Konstruktor

  -

- Metode

Nama Procedure/Function	Deskripsi
`public static String[] findFunction (double[][] titik, int banyakTitik)`	Fungsi untuk mencari nilai dari fungsi A dan mengembalikan nya
`public static double findY(String[] function, double x)`	Fungsi untuk menghasilkan nilai Y

3.2.3 Invers

- Artibut

  -

- Konstruktor

  -

- Metode

Nama Procedure/Function	Deskripsi
`public static Matrix invers (Matrix m)`	Fungsi untuk mengembalikan invers dari matriks

3.2.4 Regresi

- Artibut

  -

- Konstruktor

  -

- Metode

Nama Procedure/Function	Deskripsi
`public static void regresiLinearKeyboard (Matrix m)`	Prosedur untuk menghasilkan nilai regresi dari banyak variabel dan melakukan prediksi suatu nilai dari data independen atau variabel bebas yang dimasukkan dari keyboard
`public static void regresiLinearFile (Matrix m)`	Prosedur untuk menghasilkan nilai regresi dari banyak variabel dan melakukan prediksi suatu nilai dari data independen atau variabel bebas yang dimasukkan dari file

3.2.5 SPL

- Artibut

  -

- Konstruktor

  -

- Metode

Nama Procedure/Function	Deskripsi
`public static String[] metode_gauss(Matrix m)`	Fungsi untuk mencari solusi dari SPL dengan metode Gauss
`public static String[] metode_gauss_jordan(Matrix m)`	Fungsi untuk mencari solusi dari SPL dengan metode eliminasi Gauss-Jordan
`public static String[] solve (Matrix m)`	Fungsi untuk menghasilkan solusi SPL

3.3 Folder IO

3.3.1 Input

- Artibut

  -

- Konstruktor

`public static Scanner scanner = new Scanner(System.in)`	Membuat sebuah objek *scanner* untuk membaca input dari pengguna melalui *keyboard*

- Metode

Nama Procedure/Function	Deskripsi
`public static Matrix readMatrix()`	Fungsi untuk membaca matrix
`public static double[][] readInterpolasi()`	Fungsi untuk membaca inputan dari interpolasi
`public static double[][] readBicubic()`	Fungsi untuk membaca inputan dari bicubic
`public static Matrix readMatrixFile()`	Fungsi untuk membaca file dan merubahnya ke bentuk matriks
`public static int caraInput (boolean salahInput)`	Fungsi untuk mengembalikan pilihan *input* yang dipilih

3.3.2 Output

- Artibut

  -

- Konstruktor

`public static Scanner scanner = new Scanner(System.in)`	Membuat sebuah objek *scanner* untuk membaca input dari pengguna melalui *keyboard*

- Metode

Nama Procedure/Function	Deskripsi
`public static void header()`	Prosedur untuk print header
`private static void header_menu_metode()`	Prosedur untuk print menu metode
`private static void header_menu_input()`	Prosedur untuk print menu input
`private static void header_menu_output()`	Prosedur untuk print menu output
`private static void header_menu_regresi()`	Prosedur untuk print menu regresi
`public static void menu_utama()`	Prosedur untuk print menu utama
`public static void menu_spl()`	Prosedur untuk print menu spl
`public static void menu_determinan()`	Prosedur untuk print menu determinan
`public static void menu_invers()`	Prosedur untuk print menu invers
`public static void menu_regresi()`	Prosedur untuk print menu regresi
`public static void menu_input()`	Prosedur untuk print menu input
`public static void menu_output()`	Prosedur untuk print menu output
`public static void pesan_keluar()`	Prosedur untuk print menu keluar
`public static boolean pesan_salah_input(boolean`	Fungsi mengecek apakah input salah atau tidak

`salahInput)`	
`public static void solusi_spl(String[] solusi)`	Fungsi untuk mendapatkan solusi spl
`public static void solusi_interpolasi_polinomial(String[] solusi)`	Fungsi untuk mendapatkan solusi dari interpolasi polinomial

3.3.2 OutputFile

- Artibut

  -

- Konstruktor

  -

- Metode

Nama Procedure/Function	Deskripsi
`public static void printMatrix (Matrix m)`	Prosedur untuk print matrix
`public static void OutputFile (Matrix m, int opsi)`	Prosedur untuk mengeluarkan output matrix berbentuk file
`public static void OutputDetFile ()`	Prosedur untuk mengeluarkan output determinan berbentuk file

BAB IV
EKSPERIMEN

4.1 SPL

4.1.1 Menentukan Solusi SPL

a. Test case 1a

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

```
1,0000 1,0000 -1,0000 -1,0000 1,0000
2,0000 5,0000 -7,0000 -5,0000 -2,0000
2,0000 -1,0000 1,0000 3,0000 4,0000
5,0000 2,0000 -4,0000 2,0000 6,0000

----------------------------------------

          Kalkulator Matriks
             by Kopi Jawa

----------------------------------------
              Menu Output
----------------------------------------
1. Buat Output File
2. Tidak Menghasilkan Output File

> 2
Tidak ada solusi yang memenuhi.
```

b. Tesc case 1b

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

```
Melakukan Eliminasi Gauss untuk mendapatkan matriks eselon:

1,0000 -1,0000 0,0000 0,0000 1,0000 3,0000
0,0000 1,0000 0,0000 -1,5000 -0,5000 1,5000
0,0000 0,0000 0,0000 1,0000 -1,0000 -1,0000
0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

Dilakukan penyelesaian dan didapatkan:
x_1 = 3,0000 + (-1,0000(1,5000 + (-1,5000(-1,0000 + (-1,0000(t_{5})))) + (-0,5000(t_{5})))) + (1,0000(t_{5}))
x_2 = 1,5000 + (-1,5000(-1,0000 + (-1,0000(t_{5})))) + (-0,5000(t_{5}))
x_3 = t_{3}
x_4 = -1,0000 + (-1,0000(t_{5}))
x_5 = t_{5}
```

c. Test case 1c

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

```
Melakukan Eliminasi Gauss untuk mendapatkan matriks eselon:

0,0000 1,0000 0,0000 0,0000 0,0000 1,0000 1,0000
0,0000 0,0000 0,0000 1,0000 1,0000 0,0000 -1,0000
0,0000 0,0000 0,0000 0,0000 1,0000 -1,0000 1,0000

Dilakukan penyelesaian dan didapatkan:
x_1 = t_{1}
x_2 = 1,0000 + (1,0000(t_{6}))
x_3 = t_{3}
x_4 = -1,0000 + (1,0000(1,0000 + (-1,0000(t_{6}))))
x_5 = 1,0000 + (-1,0000(t_{6}))
x_6 = t_{6}
```

d. Test case 1d

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

H adalah matriks *Hilbert*. Cobakan untuk $n = 6$ dan $n = 10$.

```
Melakukan Eliminasi Gauss untuk mendapatkan matriks eselon:

1,0000 0,5000 0,3333 0,2500 0,2000 0,1667 1,0000
0,0000 1,0000 1,0006 0,9004 0,8007 0,7137 -6,0024
0,0000 0,0000 1,0000 1,5115 1,7056 1,8060 30,3026
0,0000 0,0000 0,0000 1,0000 4,0493 4,6879 -286,6483
0,0000 0,0000 0,0000 0,0000 1,0000 1,0374 -95,5795
0,0000 0,0000 0,0000 0,0000 0,0000 1,0000 12,3393

Dilakukan penyelesaian dan didapatkan:
x_1 = 11,9129
x_2 = -63,2613
x_3 = 50,2401
x_4 = 94,3671
x_5 = -108,3801
x_6 = 12,3393
```

```
Melakukan Eliminasi Gauss untuk mendapatkan matriks eselon:

1,0000 0,5000 0,3333 0,2500 0,2000 0,1667 0,1428 0,1250 0,1111 0,1000 1,0000
0,0000 1,0000 1,0006 0,9004 0,8007 0,7137 0,6435 0,5834 0,5336 0,4910 -6,0024
0,0000 0,0000 1,0000 1,5115 1,7056 1,8060 1,7914 1,7616 1,7045 1,6413 30,3026
0,0000 0,0000 0,0000 1,0000 4,0493 4,6879 6,1363 6,6854 7,1632 7,6990 -286,6483
0,0000 0,0000 0,0000 0,0000 1,0000 1,0374 1,4640 1,5855 1,6705 1,8501 -95,5795
0,0000 0,0000 0,0000 0,0000 0,0000 1,0000 1,4216 2,4111 2,8360 3,6152 12,3393
0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 1,0000 0,8086 2,1335 0,1951 226,6733
0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 1,0000 -6,1111 -11,9244 580,4823
0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 1,0000 1,4035 -67,2662
0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 1,0000 -72,9123

Dilakukan penyelesaian dan didapatkan:
x_1 = 12,0218
x_2 = -77,1603
x_3 = 124,5157
x_4 = 57,3643
x_5 = -268,2617
x_6 = 34,5441
x_7 = 226,4496
x_8 = -74,6534
x_9 = 35,0669
x_10 = -72,9123
```

4.1.2 Menentukan solusi SPL matriks *augmented*

a. Test case 2 a

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

```
Melakukan Eliminasi Gauss untuk mendapatkan matriks eselon:

1,0000 -1,0000 2,0000 -1,0000 -1,0000
0,0000 1,0000 -2,0000 0,0000 0,0000
0,0000 0,0000 0,0000 0,0000 0,0000
0,0000 0,0000 0,0000 0,0000 0,0000

Dilakukan penyelesaian dan didapatkan:
x_1 = -1,0000 + (-1,0000(0,0000 + (-2,0000(t_{3})))) + (2,0000(t_{3})) + (-1,0000(t_{4}))
x_2 = 0,0000 + (-2,0000(t_{3}))
x_3 = t_{3}
x_4 = t_{4}
```

b. Test case 2 b

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

```
Melakukan Eliminasi Gauss untuk mendapatkan matriks eselon:

1,0000 0,0000 4,0000 0,0000 4,0000
0,0000 1,0000 0,0000 4,0000 6,0000
0,0000 0,0000 1,0000 0,0000 1,0000
0,0000 0,0000 0,0000 1,0000 1,0000
0,0000 0,0000 0,0000 0,0000 0,0000
0,0000 0,0000 0,0000 0,0000 0,0000

Dilakukan penyelesaian dan didapatkan:
x_1 = 4,0000 + (4,0000(1,0000))
x_2 = 6,0000 + (4,0000(1,0000))
x_3 = 1,0000
x_4 = 1,0000
```

4.1.3 Menentukan solusi SPL dari persamaan

a. Test case 3 a

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

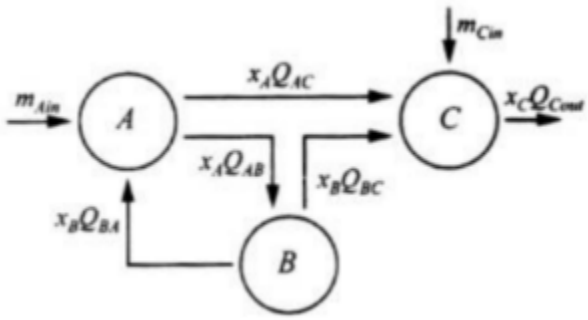
```
Melakukan Eliminasi Gauss untuk mendapatkan matriks eselon:

1,0000 0,1250 0,3750 0,2500 0,0000
0,0000 1,0000 -0,2000 -0,2857 0,1143
0,0000 0,0000 1,0000 -0,1948 0,7597
0,0000 0,0000 0,0000 1,0000 -0,2581

Dilakukan penyelesaian dan didapatkan:
x_1 = -0,2243
x_2 = 0,1824
x_3 = 0,7095
x_4 = -0,2581
```

b. Test case 3 b

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

```
Reading from: Test\Input\tes3b_SPL.txt
0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 1,0000 1,0000 1,0000 13,0000
0,0000 0,0000 0,0000 1,0000 1,0000 1,0000 0,0000 0,0000 0,0000 15,0000
1,0000 1,0000 1,0000 0,0000 0,0000 0,0000 0,0000 0,0000 0,0000 8,0000
0,0000 0,0000 0,0429 0,0000 0,0429 0,7500 0,0429 0,7500 0,6140 14,7900
0,0000 0,2500 0,9142 0,2500 0,9142 0,2500 0,9142 0,2500 0,0000 14,3100
0,6140 0,7500 0,0429 0,7500 0,0429 0,0000 0,0429 0,0000 0,0000 3,8100
0,0000 0,0000 1,0000 0,0000 0,0000 1,0000 0,0000 0,0000 1,0000 18,0000
0,0000 1,0000 0,0000 0,0000 1,0000 0,0000 0,0000 1,0000 0,0000 12,0000
1,0000 0,0000 0,0000 1,0000 0,0000 0,0000 1,0000 0,0000 0,0000 6,0000
0,0429 0,7500 0,6140 0,0000 0,0429 0,7500 0,0000 0,0000 0,0429 10,5100
0,9142 0,2500 0,0000 0,2500 0,9142 0,2500 0,0000 0,2500 0,9142 16,1300
0,0429 0,0000 0,0000 0,7500 0,0429 0,0000 0,6140 0,7500 0,0429 7,0400

---------------------------------------

          Kalkulator Matriks
             by Kopi Jawa

---------------------------------------
             Menu Output
---------------------------------------
1. Buat Output File
2. Tidak Menghasilkan Output File

> 2
Tidak ada solusi yang memenuhi.
```

4.1.4 Test case 4




```
Melakukan Eliminasi Gauss untuk mendapatkan matriks eselon:

1,0000 -0,5000 0,0000 10,8333
0,0000 1,0000 0,0000 7,2222
0,0000 0,0000 1,0000 10,0000

Dilakukan penyelesaian dan didapatkan:
x_1 = 14,4444
x_2 = 7,2222
x_3 = 10,0000
```

4.2 Determinan

4.2.1 Determinan Reduksi Baris

```
Masukkan jumlah baris: 3
Masukkan jumlah kolom: 3
Masukkan elemen matriks:
7 3 4
2 5 7
4 6 1
```

```
Dengan menggunakan metode reduksi baris, diperoleh nilai determinan:
-213,0000
```

4.2.2 Determinan Ekspansi Kofaktor

```
Masukkan jumlah baris: 3
Masukkan jumlah kolom: 3
Masukkan elemen matriks:
7 3 4
2 5 7
4 6 1
```

```
Dengan menggunakan metode ekspansi kofaktor, diperoleh nilai determinan:
-213,0000
```

4.3 Matriks Balikan

4.3.1 Matriks Balikan dengan Matriks Identitas

```
Masukkan jumlah baris: 3
Masukkan jumlah kolom: 3
Masukkan elemen matriks:
6 1 2
1 4 8
3 9 5
```

```
Hasil invers menggunakan metode adjoin:
0.1739 -0.0435 -0.0000
-0.0635 -0.0803 0.1538
0.0100 0.1706 -0.0769
```

4.3.2 Matriks Balikan dengan Matriks Adjoin

```
Masukkan jumlah baris: 3
Masukkan jumlah kolom: 3
Masukkan elemen matriks:
6 1 2
1 4 8
3 9 5
```

```
Hasil invers menggunakan metode matriks identitas:
0.1739 -0.0435 0.0000
-0.0635 -0.0803 0.1538
0.0100 0.1706 -0.0769
```

4.4 Interpolasi Polinomial

```
Masukkan banyak titik yang diketahui> 3

Titik ke-1: 3 4
Titik ke-2: 2 5
Titik ke-3: 9 6

Masukkan absis (x) dari ordinat (y) yang ingin Anda cari> 1 7
```

```
Dilakukan penyelesaian dan diperoleh persamaan dan koordinat sebagai berikut:

y = 8.1429 + -1.9524x^1 + 0.1905x^2

(1.0000,6.3810)
```

4.5 Interpolasi Bicubic Spline

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

Tentukan nilai:

$f(0, 0)$ = ?
$f(0.5, 0.5)$ = ?
$f(0.25, 0.75)$ = ?
$f(0.1, 0.9)$ = ?

4.5.1 Nilai f(0,0)

```
Hasil :
21.0000
```

4.5.2 Nilai f(0.5, 0.5)

```
Hasil :
87.7969
```

4.5.3 Nilai f(0.25, 0.75)

```
Hasil :
82.1482
```

4.5.4 Nilai f(0.1,0.9)

```
Hasil :
91.2713
```

4.6 Regresi Berganda

```
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
50.0 76.0 29.30
```

Input

4.6.1 Regresi Linear Berganda

```
f(x) = - 3,5078 - 0,0026x1 + 0,0008x2 + 0,1542x3, f(xk) = 0,9384
----------------------------------------
```

4.6.2 Regresi Kuadratik Berganda

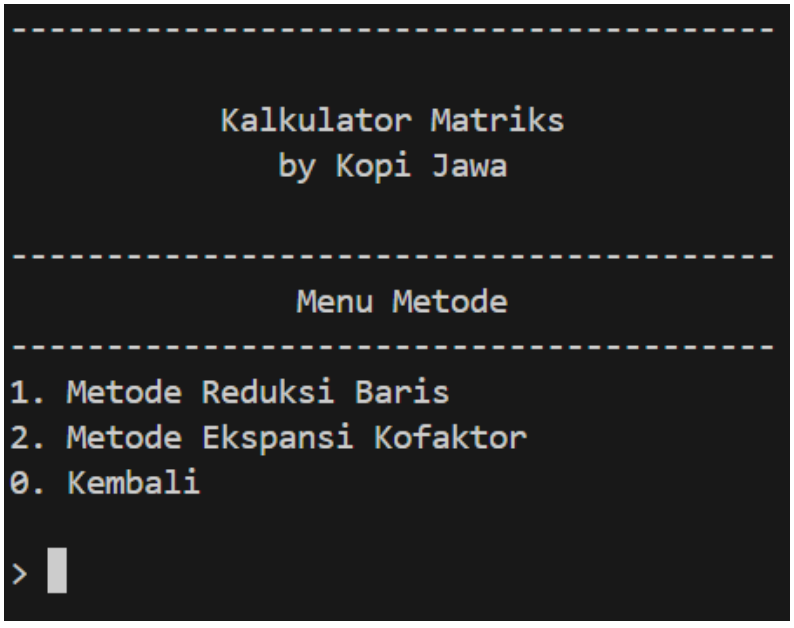
```
f(x) = - 3,5078 - 0,0026x1 + 0,0008x2 + 0,1542x3, f(xk) = 0,9384
----------------------------------------
```

4.7 Main

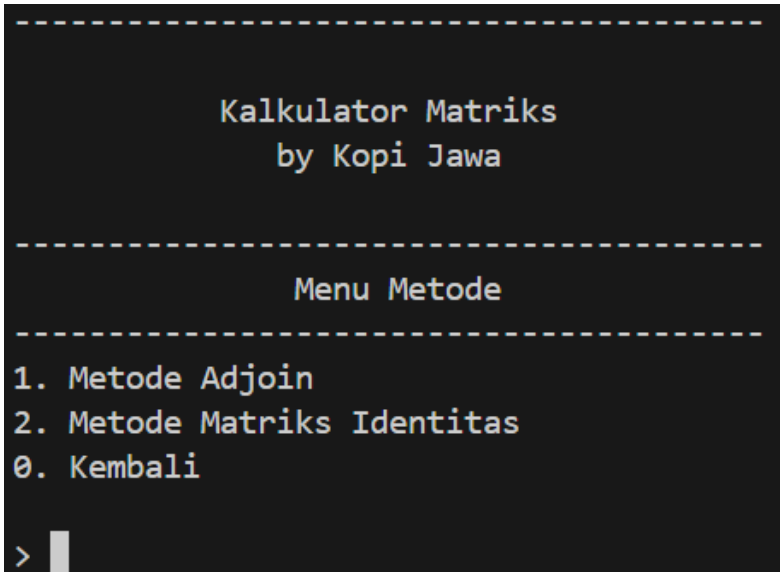
4.7.1 Menu Pilihan

4.7.2 Menu Metode SPL

4.7.3 Menu Metode Determinan



4.7.4 Menu Metode Matriks Balikan



4.7.4 Menu metode Regresi Berganda

BAB V
KESIMPULAN

5.1 Kesimpulan

Berdasarkan materi kuliah Aljabar Linear dan Geometri IF2123 yang telah dipelajari di kelas, kami mengimplementasikan materi terkait matriks dan SPL ke dalam sebuah program bahasa Java untuk membuat kalkulator matriks. Kalkulator yang telah kami buat dapat menyelesaikan beberapa persoalan seperti mencari solusi sistem persamaan linear, menghitung determinan, menghitung matriks balikan, menghitung taksiran nilai dari fungsi interpolasi polinom dan interpolasi bikubik, menghitung hasil regresi linear dan kuadratik berganda.

Melalui tugas besar ini, kami menjadi paham tentang berbagai metode untuk menyelesaikan sistem persamaan linear, menerapkan matriks untuk menaksir suatu nilai menggunakan beberapa metode. Kami juga menjadi paham bagaimana cara mengimplementasikan program ke dalam bahasa Java.

5.2 Saran

Berdasarkan dari pengalaman yang telah kami alami selama mengerjakan tugas besar ini, ada beberapa hal yang dapat diambil sebagai pembelajaran, diantaranya :

- Buat time line yang baik agar tugas berjalan dengan lancar dan tidak deadliner
- Buat pembagian modul dan fungsi secara jelas agar tidak membingungkan saat pengerjaan kedepannya
- Lakukan debugging jauh-jauh hari agar jika ada error dapat ditangani dengan maksimal

5.3 Refleksi

Dalam mengerjakan tugas besar ini, kami merasa telah mempelajari banyak hal. Dari segi teknikal, kami mempelajari bahasa pemrograman baru yaitu bahasa Java dan juga programming secara modular. Kami juga belajar bahwa ketika mengerjakan suatu project kode secara bersama sama, lebih baik menerapkan konvensi syntax (misalnya camel case untuk nama class dan snake case untuk nama fungsi) agar kode yang dihasilkan lebih rapi. Pada tugas besar ini, kami belum menerapkan konvensi tersebut sehingga hasil kodenya tidak serapi yang kami harapkan. Dari segi non teknikal, kami belajar cara bekerja sama, komunikasi yang baik, dan juga

memanajemen waktu dengan efektif. Manajemen waktu adalah hal yang sangat krusial dalam pengerjaan tugas dan kami belum bisa melakukannya dengan baik di tugas besar ini sehingga kami kewalahan ketika tenggat waktu mendekat.

LAMPIRAN

Referensi :
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2022-2023/algeo22-23.htm
https://www.w3schools.com/java/java_try_catch.asp

Tautan repository :
https://github.com/izzatjundy/Algeo01-23050.git

Tautan video :
https://drive.google.com/file/d/1dzoKs9zGscPjtH39LNhHY4fFEHAKy9Oa/view?usp=drivesdk