LAPORAN TUGAS BESAR

IF-2123 ALJABAR LINIER DAN GEOMETRI

SEMESTER 1 2022/23

Sistem Persamaan Linier, Determinan, dan Aplikasinya

Kelompok 23 – Oracle :

Christian Albert Hasiholan (13521078)

Fakih Anugerah Pratama (13521091)

Zidane Firzatullah (13521163)

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

1. DESKRIPSI MASALAH

Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. Ada 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, mahasiswa diminta untuk membuat satu atau lebih *library* aljabar linier dalam Bahasa Java. Library tersebut berisi fungsi-fungsi seperti eliminasi Gauss, eliminasi Gauss-Jordan, menentukan balikan matriks, menghitung determinan, kaidah Cramer (kaidah Cramer khusus untuk SPL dengan *n* peubah dan *n* persamaan). Selanjutnya, *library* tersebut digunakan di dalam program Java untuk menyelesaikan berbagai persoalan yang dimodelkan dalam bentuk SPL, menyelesaikan persoalan interpolasi, dan persoalan regresi.

2. TEORI SINGKAT

Sistem persamaan linier (SPL) adalah kumpulan persamaan yang memiliki beberapa variabel. Dalam menentukan solusi variabel dari sebuah SPL dapat digunakan sebuah matriks.

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

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

Matriks eselon adalah matriks yang memiliki *leading one* pada setiap barisnya, kecuali pada baris yang seluruhnya 0. Baris yang seluruhnya tidak 0, maka bilangan tidak 0 pertamanya bernilai 1 dan disebut sebagai leading one. Kemudian baris pada matriks diurutkan berdasarkan letak leading zeronya, dimana baris dengan leading zero lebih kiri akan berada di atas baris dengan leading zero lebih kanan. Bentuk lain dari matriks eselon adalah eselon baris tereduksi, bedanya adalah pada eselon baris tereduksi kolom dengan leading one berisi 0 kecuali pada leading one.

• Berbentuk:

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

Keterangan: * adalah sembarang nilai

Matriks Eselon

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

Matriks Eselon Tereduksi

Pada matriks SPL kemudian dilakukan Operasi Baris Elementer (OBE), yaitu dengan mengalikan sebuah baris dengan konstanta tidak 0, menukar 2 baris, dan menambahkan sebuah baris dengan kelipatan baris lainnya. Tujuan akhir dari OBE adalah mengubah matriks SPL menjadi bentuk eselon atau eselon baris tereduksi. Bila bentuk akhirnya matriksnya dalam eselon, maka gunakan metode eliminasi Gauss untuk menentukan solusinya. Sedangkan bila bentuk akhirnya eselon baris tereduksi, maka gunakan metode eliminasi Gauss-Jordan. Namun setelah melakukan OBE terdapat 3 kemungkinan solusi, yaitu solusi unik/tunggal, solusi banyak/infinite, dan tidak ada solusi.

Selain mengubah matriks SPL menjadi matriks eselon, masih ada lagi metode untuk menentukan solusi SPL, yaitu dengan metode inverse dan kaidah Cramer. Untuk mengubah matriks SPL menjadi inversenya maka matriks tersebut bisa digabungkan dengan matriks identitas, lalu dilakukan OBE, sampai terbentuk matriks identitas di tempat awal matriks SPL. Kemudian matriks inverse tersebut dikalikan dengan kolom hasil dari matriks SPL awal, sehingga didapat solusinya. Sedangkan pada kaidah Cramer dilakukan pertukaran

kolom dari kolom konstanta dengan kolom hasil. Kemudian determinan matriks yang telah ditukar kolomnya dibagi dengan determinan matriks awal, sehingga didapat solusinya.

Selain menggunakan beberapa metode di atas, penentuan solusi matriks dapat juga dilakukan dengan memanfaatkan matriks kofaktor dan ajoin dari matriks awal. Matriks kofaktor adalah matriks yang dibentuk dari nilai-nilai hasil kali (-1^{i+j}) dengan determinan yang dihasilkan oleh matriks semu yang dibentuk dengan mengabaikan nilai pada kolom-j dan baris-i matriks awal untuk kolom-j dan baris-i matriks kofaktor. Sedangkan, matrik ajoin sederhananya adalah transpos dari matriks kofaktor yang sudah dihasilkan sebelumnya.

Sebelum menentukan solusi matriks, kita perlu menentukan determinan matriks awal dengan melakukan penjumlahan hasil perkalian nilai matriks dan nilai kofaktor setiap kolom untuk baris sembarang i atau setiap baris untuk kolom sembarang j.

Solusi matriks dapat ditentukan dengan mencari invers matriks awal yang dapat ditentukan dengan melakukan perkalian matriks ajoin yang telah dibentuk dengan 1/determinan yang telah dihitung sebelumnya.

Setelah didapatkan invers, nilai solusi x dapat ditentukan dengan melakukan perkalian matriks invers koefisien yang didapat dengan matriks nilai konstanta. Sehingga didapatkan sebuah matriks yang berisi nilai-nilai untuk x.

Interpolasi polinom adalah teknik interpolasi dengan menentukan fungsi polynomial yang berlaku untuk set tertentu berdasarkan 2 atau lebih titik yang dimasukkan dalam perhitungan. Banyak titik yang diperhitungkan menentukan derajat fungsi polynomial yang dihasilkan. Solusi persamaan polinom tersebut dapat dihitung dengan memanfaatkan teori-teori pencarian solusi yang sudah dijelaskan sebelumnya.

Interpolasi bikubik adalah Teknik interpolasi 2 Dimensi dengan memanfaatkan 4 x 4 titik yang memiliki indeks -1, 0, 1, 2. Tujuan dari interpolasi bikubik adalah menentukan fungsi bikubik untuk menghitung estimasi *value* suatu titik pada area antara titik (0,0) sampai dengan (1,1) berdasarkan masukan 16 titik di sekitarnya. Interpolasi bikubik dilakukan dengan menentukan koefisien fungsi hasil menggunakan teori penentuan solusi persamaan dalam bentuk matriks. Matriks koefisien yang terbentuk dapat digunakan untuk melakukan interpolasi titik (x,y) pada area dengan range x,y = [0,1].

Regresi linier berganda mirip dengan regresi linear, tujuannya untuk mendapatkan fungsi regresi serta memprediksi suatu nilai, namun bedanya adalah pada regresi linear berganda terdapat lebih dari 1 variabel.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

where, for $i = n$ observations:

y_i = dependent variable

x_i = expanatory variables

β_0 = y-intercept (constant term)

β_p = slope coefficients for each explanatory variable

ϵ = the model's error term (also known as the residuals)

Untuk mendapat konstanta dan koefisien dari tiap variabel maka beberapa persamaan di atas akan diubah bentuknya menjadi sebuah matriks.

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

Sehingga selanjutnya bisa dicari solusinya dengan metode eliminasi Gauss.

3. IMPLEMENTASI DALAM JAVA

Class Matrix		
State		
Nama - tipe	Modifier	Deskripsi
matrix - double	private	matrix utama
horizontalSize, verticalSize - int	private	ukuran matrix
matrixType - int	private	tipe matrix
Fungsi		

Nama – Return Type	Modifier	Params	Deskripsi
-	Public	matrix: double[][]	Matrix constructor
loadMatrix – void	Public	filePath: String	Load matrix dari sebuah file dengan extensi .txt
getProductMatrix - Matrix	Public	matrix1, matrix2: Matrix	Melakukan perkalian matrix1 . matrix 2
solveMatrixCramer – double[][]	Public	-	Memberikan solusi spl dengan metode cramer
solveDeterminantByCofactor – double	Public	-	Memberikan nilai determinan matriks menggunakan approach kofaktor
solveDeterminantByRowReduction - double	Public	-	Memberikan nilai determinan matriks menggunakan approach reduksi baris (obe)
solveInverseByAdjoin - double[][]	Public	-	Memberikan inverse matriks menggunakan adjoint dan determinant

solve - double[][]	Public	-	Memberikan 2d double array berisi solusi dari spl
getEchelonMatrix - Matrix	Public	-	Memberikan object Matrix baru dengan state matrix dalam bentuk echelon matrix
getReducedEchelonMatrix - Matrix	Public	-	Memberikan object Matrix baru dengan state matrix dalam bentuk reduced echelon matrix
getInverseMatrix - Matrix	Public	-	Memberikan object Matrix baru dengan state matrix dalam bentuk matrix inverse
copyMatrix - Matrix	Public	-	Memberikan copy dari object Matrix
mergeAndCopyMatrix - Matrix	Public	mergingAgent: double[][]	Memberikan object Matrix baru dengan state matrix merged dengan mergingAgent
saveMatrixToFile - Matrix	Public	folderPath, fileName: String matrix: double[][]	Menyimpan state matrix ke file txt
Inner class MatrixHelper			

printMatrix - void	private	matrix: double[][]	print matrix
switchRows - void	private	row1, row2: int matrix: double[][]	menukar row1 dengan row2 dalam matrix
multiplyRow - void	private	row: int multiplier: double matrix: double[][]	mengalikan row matrix dengan multiplier
rowAdditionRow - void	private	targetRow, baseRow: int rowMultiplication: double matrix: double[][]	melakukan operasi r1 + x(r2) pada matrix
getEchelonMatrix - double[][]	private	matrix: double[][]	melakukan operasi obe pada matrix untuk mendapatkan matrix dalam bentuk echelon
getReducedEchelonMatrix - double[][]	private	matrix: double[][]	melakukan operasi obe pada matrix untuk mendapatkan matrix dalam bentuk reduced echelon
getInverseMatrixSpl - double[][]	private	matrix: double[][]	mendapatkan inverse matrix

getInverseMatrix - double[][]	private	matrix: double[][]	mendapatkan inverse matrix
getInverseMatrixMaster - double[][]	private	matrix: double[][] boolean: spl	mendapatkan inverse matrix
mergeMatrices - double[][]	private	matrix, mergingAgent: double[][]	melakukan merging pada matrix dengan mergingAgent
cutMatrixColumns - double[][]	private	startColumn, endColumn: int matrix: double[][]	mendapatkan matrix baru dengan isi kolom startClumn sampai endColumn
multiplyMatrix - double[][]	private	matrix1, matrix2: double[][]	mengalikan matrix1 dengan matrix2
isIdentityMatrix - boolean	private	matrix: double[][]	memeriksa apakah matrix merupakan identitas
generateIdentityMatrix - double[][]	private	dimension: int	membuat matriks identitas dengan ukuran dimensi x dimensi

inferSolutionType - int	private	matrix: double[][]	menentukan apakah matriks memiliki solusi unik, parametrik, tidak memiliki solusi
solveEchelonFormMatrix - double[][]	private	matrix: double[][]	memberikan solusi matriks dalam bentuk echelon
solveInverseMatrix - double[][]	private	matrix: double[][]	memberikan solusi matriks dalam matrix inverse
copyMatrix - double[][]	private	matrix: double[][]	memberikan copy matrix
getMinorMatrix - double[][]	private	matrix: double[][] cofactorRow, cofactorCol: int	membuat minor dari matrix
getDeterminantByCofactor - double	private	matrix: double[][]	memberikan determinan matrix dengan metode ekspansi kofaktor
swapZeroRow - void	private	matrix: double[][] zeroRow: int	menukar row matrix yang elemen diagonalnya 0 agar matrix terurut

getDeterminantByRowReduction - double	private	matrix: double[][]	memberikan determinan matrix dengan metode reduksi baris
changeColumn - double[][]	private	mainMatrix, rightSideMatrix: double[][] colTarget: int	menukar sebuah kolom dengan kolom hasil untuk menyelesaikan Cramer's rule
solveCrammerMatrix - double[][]	private	mainMatrix, rightSideMatrix: double[][]	mendapatkan solusi SPL dengan metode Cramer's rule
transposeMatrix - double[][]	private	matrix: double[][]	mentranspose matrix
getInverseMatrixByAdjoint - double[][]	private	matrix: double[][]	mendapat inverse matrix dengan adjoint dan determinant
getCrammerMatrix - double[][]	private	matrix: double[][]	mengambil matrix dan memisahkannya menjadi matrix koefisien dan matrix hasil

Class BicubicInterpolationSolver		
State		
Nama - tipe	Modifier	Deskripsi

fvalue- Matrix	private	variabel matriks yang berisi nilai fungsi f, berdasarkan definisi persamaan interpolasi bikubik	
aCoeffMatrix - Matrix	private	variabel matriks yang berisi nilai koefisien-koefisien a, berdasarkan definisi persamaan interpolasi bikubik	
Fungsi			
Nama – Return Type	Modifier	Params	Deskripsi
getXMatrix - Matrix	private	-	mendapatkan matriks berisi solusi persamaan bikubik
loadFValue - void	private	-	meminta user untuk memasukkan nilai konstantat fungsi f berupa sebuah matriks
loadVariables - void	public	String absFilePath	prosedur untuk melakukan definisi variabel menggunakan masukan path file
loadVariables - void	public	Matrix fVal	prosedur untuk melakukan definisi variabel menggunakan masukan matriks
loadVariables - void	public	-	prosedur untuk melakukan definisi variabel melalu masukan pengguna

solveF - double	public	Double x, Double y	fungsi yang akan menerima variabel x dan y sebuah titik dan mengembalikan nilai fungsi f(x,y) berdasarkan masukan variabel konstanta dan koefisien yang sudah didefinisikan.
solve - void	public	-	prosedur untuk menyelesaikan persamaan interpolasi bikubik

Class Image			
State			
Nama - tipe	Modifier	Deskripsi	
width, height - int	private	lebar dan tinggi gambar	
imageColor - imageColor[]	private	variabel yang berisi kumpulan nilai RGB titik-titik yang ada pada gambar dalam bentuk integer	
Fungsi			
Nama – Return Type	Modifier	Params	Deskripsi
Build - Matrix	private	image: BufferedImage	membangun sebuah objek gambar berdasarkan masukan BufferedImage
readAsBufferedImage - BufferedImage	public	filename: String	prosedur untuk membaca gambar dari sebuah file dan mengembalikannya sebagai sebuah BufferedImage baru

readImage - void	public	filename: String	prosedur untuk membaca gambar dari sebuah file
setImage - void	public	img: bufferedimage	prosedur untuk melakukan definisi image
getBufferedImage - BufferedImage	public	-	prosedur untuk mendapatkan definisi properti BufferedImage
saveAsFile - void	public	filename: string imgtosave: BufferedImage	prosedur untuk menyimpan sebuah objek BufferedImage menjadi sebuah file baru

Class ImageColor			
State			
Nama - tipe	Modifier	Deskripsi	
r, g, b - int	private	nilai red, green, dan blue warna image	
Fungsi			
Nama – Return Type	Modifier	Params	Deskripsi
- ImageColor	public	r, g, b: int	constructor

- ImageColor	public	rgb: int	constructor
getRGB - int	public	-	mengembalikan nilai RGB untuk sebuah titik

Class ImageInterpolationSolver			
State			
Nama - tipe	Modifier	Deskripsi	
targetHeight, targetWidth - int	-	target lebar dan tinggi gambar hasil interpolasi	
Image - image	-	properti image yang akan diinterpolasi	
Fungsi			
Nama – Return Type	Modifier	Params	Deskripsi
Clamp - int	private	int val, int a, int b	fungsi untuk melakukan *clamping* pada sebuah nilai, diberikan maksimum dan minimumnya
getFValue - Matrix	private	int xStart, int yStart	fungsi untuk mendapatkan nilai matriks konstanta f berdasarkan masukan titik

solve - void	public	-	prosedur untuk melakukan penyelesaian pada interpolasi image
setImage - void	public	Image img	prosedur untuk melakukan definisi properti image

Class MainProgram			
Fungsi			
Nama – Return Type	Modifier	Params	Deskripsi
main - void	public	-	main function

Class Point		
State		
Nama - tipe	Modifier	Deskripsi
x, y: double	-	nilai x dan y sebuah titik
Fungsi		
Nama – Return Type	Modifier	Params	Deskripsi
printPoint - void	public	-	print

Class PolynomInterpolationSolver			
State			
Nama - tipe	Modifier	Deskripsi	
solutions - Matrix	private	matiks solusi	
polynomDegree - int	private	derajat polinom	
points - Point[]	private	titik-titik masukan	
fValueToCalculate - double	private	nilai f yang akan dihitung	
Fungsi			
Nama – Return Type	Modifier	Params	Deskripsi
loadVariables - void	public	-	load definisi variabel berdasarkan masukan pengguna
loadVariables - void	public	absFilePath: String	load definisi variabel berdasarkan berdasarkan masukan file
solveSolutions - Matrix	private	-	prosedur untuk menyelesaikan persamaan interpolasi polinom, dan mengembalikan solusi

solve - void	public	-	prosedur untuk menyelesaikan persamaan interpolasi polinom, dan menyimpan solusi, dan menampilkan solusi
Inner Class SolverHelper			
getXMatrix - double[][]	private	polynomDegree: int points: Point[]	mengembalikan matriks x berdasarkan definisi persamaan interpolasi polinom
getYMatrix - double[][]	private	polynomDegree: int points: Point[]	mengembalikan matriks y berdasarkan definisi persamaan interpolasi polinom
getLinearEquation - String	private	polynomDegree: int solutions: Matrix	mencetak persamaan linier
f : double	private	x: double solutions: Matrix	prosedur untuk menghitung nilai f

Class ProgramHandler			
Fungsi			
Nama – Return Type	Modifier	Params	Deskripsi

readMode- void	private	-	print
mainMenu - int	private	-	get user input
mainMenuSatu - int	private	-	get user input
mainMenuDua - int	private	-	get user input
start - void	public	-	function utama program handler
loadMatrix - double[][]	private	-	load matrix
checkMatrix - double[][]	private	matrix: double[][]	cek tipe dari solusi matrix, apakah unik, infinite, atau tidak ada

createParametrixSolution - void	private	matrix: double[][]	membuat solusi parametrix dalam bentuk fungsi
printMatrix - void	private	matrix: double[][]	print matrix
printEqLeft - void	private	matrix: double[][]	print matrix dalam format x(n) = ….

Class RegresiLinearBerganda			
Fungsi			
Nama – Return Type	Modifier	Params	Deskripsi
printMatrix - void	private	matrix: double[][]	print matrix
loadRegressionData - double[][]	private	matrix, mergingAgent: double[][]	menerima input data yang akan dilakukan regresi
transposeMatrix - double[][]	private	matrix: double[][]	mentranspose matrix

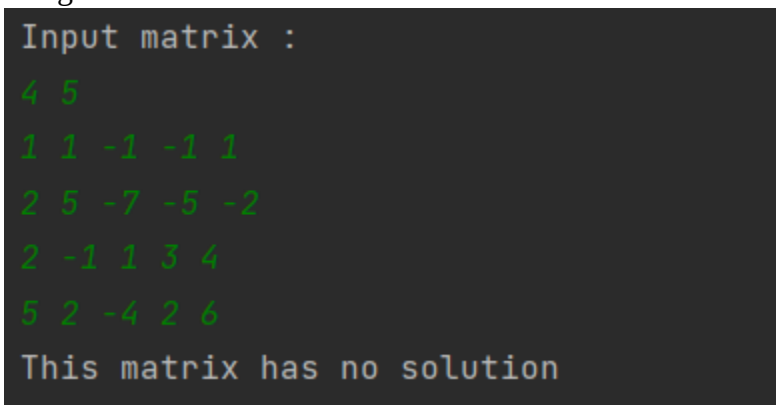
oneMatrix - Matrix	private	row: int	membuat 1 baris matrix berisi hanya elemen 1
createMultiLinearRegressionMatrix - double[][]	private	matrix: double[][]	membuat matrix regresi multi linear
checkEchelonMatrix - int	private	matrix: double[][]	melakukan cek tipe matrix setelah diubah menjadi matrix eselon
createRegressionFunction - void	private	matrix: double[][]	membuat fungsi regresi
solveRegressionFunction - void	private	regression: double[][]	menerima input variabel dan menghitungnya taksiran nilai fungsi regresi
solve - void	public	-	menerima input, menghasilkan fungsi regresi dan taksiran nilai fungsinya

Garis Besar Program
program meminta input dari user dalam bentuk angka untuk pemilihan menu dan sub-menu (jika ada) -> program meminta input matrix (dalam bentuk file atau user input) -> program mengolah data atau melemparkan pesan error jika input matrix tidak sesuai -> program menampilkan hasil pengolahan data ke layar lalu menanyakan apakah user ingin menyimpan matriks ke dalam bentuk file -> tampilkan menu semula.

4. EKSPERIMEN

1. Tentukan solusi SPL Ax = b, berikut

a. Dengan metode eliminasi Gauss



b. Dengan metode eliminasi Gauss-Jordan

```
Input matrix :
4 6
1 -1 0 0 1 3
1 1 0 -3 0 6
2 -1 0 1 -1 5
-1 2 0 -2 -1 -1
x1 = 3.00 + 1.00 x5
x2 = 2.00 x5
x4 = -1.00 + 1.00 x5
```

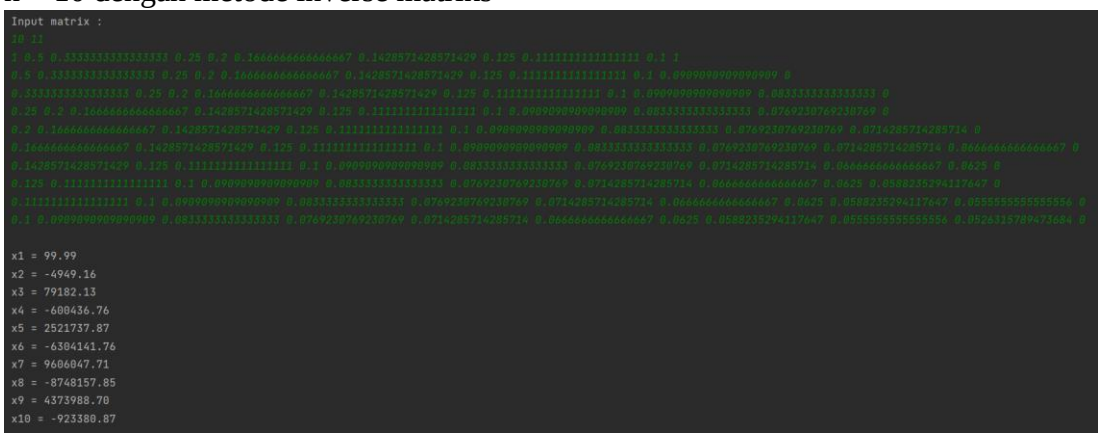
c. Dengan metode eliminasi Gauss

```
Input matrix :
3 7
0 1 0 0 1 0 2
0 0 0 1 1 0 -1
0 1 0 0 0 1 1
x2 = 2.00 - 1.00 x5
x4 = -1.00 - 1.00 x5
x5 = 1.00 + 1.00 x6
```

d. n = 6 dengan kaidah Cramer

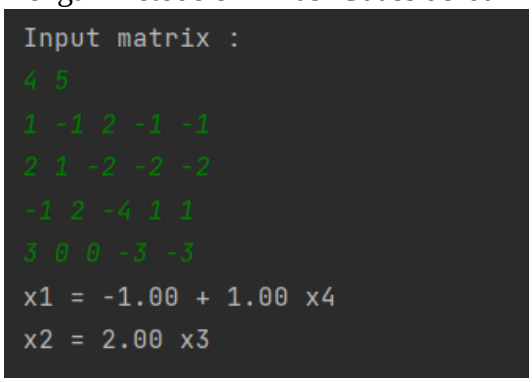
```
Input matrix :
6 7
1.0 0.5 0.3333333333333333 0.25 0.2 0.16666666666666666 1
0.5 0.3333333333333333 0.25 0.2 0.16666666666666666 0.14285714285714285 0
0.3333333333333333 0.25 0.2 0.16666666666666666 0.14285714285714285 0.125 0
0.25 0.2 0.16666666666666666 0.14285714285714285 0.125 0.1111111111111111 0
0.2 0.16666666666666666 0.14285714285714285 0.125 0.1111111111111111 0.1 0
0.16666666666666666 0.14285714285714285 0.125 0.1111111111111111 0.1 0.09090909090909091 0

x1 = 36.00
x2 = -630.00
x3 = 3360.00
x4 = -7560.00
x5 = 7560.00
x6 = -2772.00
```

n = 10 dengan metode inverse matriks

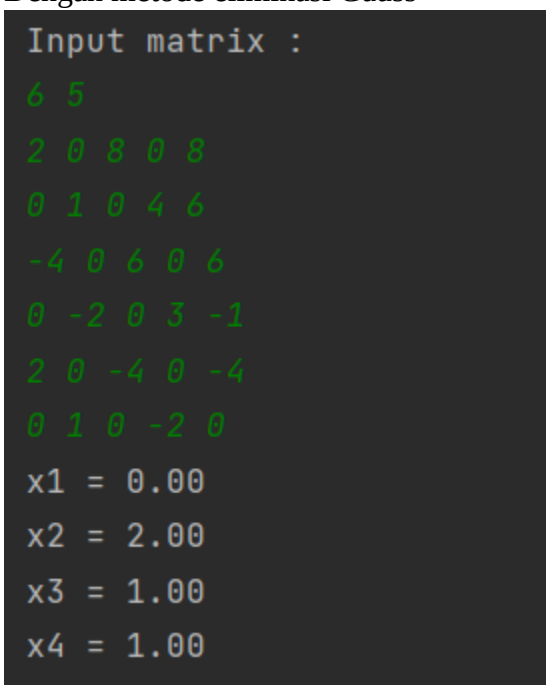


2.

a. Dengan metode eliminasi Gauss-Jordan



b. Dengan metode eliminasi Gauss



3.

a. Dengan metode inverse matriks

```
Input matrix :
4 5
8 1 3 2 0
2 9 -1 -2 1
1 3 2 -1 2
1 0 6 4 3

x1 = -0.22
x2 = 0.18
x3 = 0.71
x4 = -0.26
```

b.

4. Studi Kasus interpolasi bikubik

```
153 59 210 96
125 161 72 81
98 101 42 12
21 51 0 16
0 0
f(0.0000, 0.0000) = 161.0000
```

```
153 59 210 96
125 161 72 81
98 101 42 12
21 51 0 16
0.5 0.5
f(0.5000, 0.5000) = 97.7266
```

```
153 59 210 96
125 161 72 81
98 101 42 12
21 51 0 16
0.25 0.75
f(0.2500, 0.7500) = 105.5148
```

```
153 59 210 96
125 161 72 81
98 101 42 12
21 51 0 16
0.1 0.9
f(0.1000, 0.9000) = 104.2291
```

5.

```
Regresi Linier Berganda
Jumlah peubah x : 3
Jumlah sampel : 20
Input sample (4x20) :
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

f(x1,x2,x3) = -3.51 + -0.00 x2 + 0.00 x3 + 0.15 x4
x1 = 50
x2 = 76
x3 = 29.30
f(50.0,76.0,29.3) = 0.94
```

5. KESIMPULAN, SARAN, DAN REFLEKSI

a. Kesimpulan

Melalui penerapan teori untuk melakukan penyelesaian permasalah matriks, menggunakan bahasa Java, kami berhasil membangun sebuah library yang berisikan fungsi-fungsi untuk menyelesaikan berbagai persoalan mengenai Matriks.

b. Saran

Library yang telah dibuat melalui tugas ini tentunya belum bisa digunakan untuk menjawab **setiap** permasalahan matriks yang ada. Masih banyak teori-teori mengenai matriks dan operasinya yang bisa diterapkan menjadi sebuah fungsi yang memudahkan pembelajaran dan penghitungan operasi matriks.

Library yang telah dibuat juga dapat dimanfaatkan untuk sebuah program yang lebih kompleks seperti kalkulator matriks lengkap yang dapat

c. Refleksi
Refleksi yang dapat diperoleh dari tugas ini adalah dari error handling dimana ketilitian dan kecermatan akan sangat diperlukan untuk mengatasi error yang ada. Terutama dalam mengatasi rounding dan tipe matriks yang dihasilkan. Kemudian kerjasama dan komunikasi juga sangat baik bila dimiliki, terutama di antara tugas dan kuis yang banyak, sehingga dalam pengerjaan tubes menjadi lebih ringan.

LAMPIRAN

Link Repo Github :

https://github.com/zidane-itb/Algeo01-21078.git