LAPORAN TUGAS BESAR 1

IF2123 ALJABAR LINEAR GEOMETRI

Kelompok TBA (Tugas Besar Algeo)

|  |
| --- |
|  |

Disusun oleh:

Nadia Mareta Putri Leiden 13520007

Hansel Valentino Tanoto 13520046

Adelline Kania Setiyawan 13520084

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

# **DAFTAR ISI**

[**DAFTAR ISI** 2](#_Toc83941718)

[**BAB 1 DESKRIPSI MASALAH** 3](#_Toc83941719)

[**BAB 2 TEORI SINGKAT** 4](#_Toc83941720)

[**BAB 3 IMPLEMENTASI PROGRAM** 10](#_Toc83941721)

[**BAB 4 EKSPERIMEN** 39](#_Toc83941722)

[**BAB 5 KESIMPULAN DAN SARAN** 65](#_Toc83941723)

[**DAFTAR REFERENSI** 66](#_Toc83941724)

# **BAB 1 DESKRIPSI MASALAH**

Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. Anda sudah mempelajari berbagai metode untuk menyelesaikan SPL, termasuk menghitung determinan matriks. Sembarang SPL dapat diselesaikan dengan beberapa metode, yaitu metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan , dan kaidah *Cramer* (khusus untuk SPL dengan peubah dan 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 peubah dan persamaan). Selanjutnya, gunakan *library* tersebut di dalam program Java untuk menyelesaikan berbagai persoalan yang dimodelkan dalam bentuk SPL, menyelesaikan persoalan interpolasi, dan persoalan regresi. Penjelasan tentang interpolasi dan regresi adalah seperti di bawah ini.

Beberapa tulisan cara membuat *library* di Java:

1. [https://www.programcreek.com/2011/07/build-a-java-*library*-for-yourself/](https://www.programcreek.com/2011/07/build-a-java-library-for-yourself/)
2. [https://developer.ibm.com/tutorials/j-java*library*/](https://developer.ibm.com/tutorials/j-javalibrary/)
3. [https://stackoverflow.com/questions/3612567/how-to-create-my-own-java-*library*api](https://stackoverflow.com/questions/3612567/how-to-create-my-own-java-libraryapi)

# **BAB 2 TEORI SINGKAT**

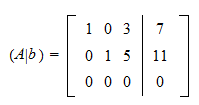
1. **Gauss dan Gauss-Jordan**

Eliminasi Gauss merupakan sebuah metode untuk menyelesaikan persamaan melalui matriks. Eliminasi Gauss menggunakan prinsip matriks eselon, yang memiliki bentuk sebagai berikut:



Metode ini bertumpu pada satu prinsip, yakni bilangan bukan nol harus dimulai dengan angka 1, lalu boleh diikuti oleh angka lainnya dalam satu baris selain angka 1.

Sedangkan metode Eliminasi Gauss-Jordan adalah metode eliminasi yang merupakan kelanjutan dari Eliminasi Gauss. Apabila prinsip Eliminasi Gauss disebut dengan matriks eselon, Eliminasi Gauss-Jordan disebut dengan matriks eselon tereduksi yang memiliki bentuk sebagai berikut:



Angka 1 harus diapit angka nol, sehingga nanti hasil persamaannya langsung berupa variabel persamaan yang ingin dicari oleh pengguna.

1. **Determinan**

Determinan adalah nilai yang dapat dihitung dari elemen-elemen suatu matriks persegi yaitu matriks yang memiliki jumlah baris dan kolom yang sama . Notasi untuk determinan dapat dituliskan sebagai atau . Misalnya, untuk matriks berukuran , determinannya dapat dihitung dengan cara:

Sedangkan untuk matriks , determinannya dapat dihitung dengan cara:

Terdapat sifat khusus saat menghitung nilai determinan dari matriks segitiga atas maupun matriks segitiga bawah. Matriks segitiga atas (*upper triangular*) adalah matriks yang semua elemen di bawah diagonal utamanya adalah nol, sedangkan matriks segitiga bawah (*lower triangular*) adalah matriks yang semua elemen di atas diagonal utamanya adalah nol. Determinan dari matriks segitiga dapat diperoleh hanya dengan mengalikan semua elemen pada diagonal utamanya. Jadi untuk suatu matriks segitiga A berukuran ,

Salah satu metode untuk menghitung determinan adalah metode reduksi baris yaitu dengan menerapkan OBE (Operasi Baris Elementer) sampai didapatkan bentuk matriks segitiga, lalu nilai determinannya dapat dihitung menggunakan rumus determinan untuk matriks segitiga. Beberapa sifat determinan ketika diterapkan OBE adalah sebagai berikut:

1. Mengalikan suatu baris pada matriks dengan konstanta k, maka determinannya menjadi
2. Menukarkan dua buah baris pada matriks , maka determinannya menjadi
3. Menambahkan sebuah baris dengan kelipatan baris yang lain, maka determinannya tetap

Jadi, secara umum, menghitung determinan dengan metode reduksi baris dapat diperoleh dengan cara:

,

dengan menyatakan banyaknya operasi pertukaran baris selama penerapan OBE.

Selain metode reduksi baris, ada metode lain untuk menghitung determinan yaitu metode ekspansi kofaktor. Untuk matriks berukuran , dapat didefinisikan:

1. (Minor entri ) adalah determinan submatriks yang elemen-elemennya tidak berada pada baris i dan kolom j, dan
2. (Kofaktor entri )

Jadi determinan matriks A berukuran dapat dihitung dengan metode ekspansi kofaktor menggunakan salah satu rumus berikut ini:

1. **Matriks Kofaktor dan Adjoin**

Matriks kofaktor dari suatu matriks adalah matriks persegi yang ukurannya sesuai dengan ukuran matriks dan setiap elemennya merupakan kofaktor dari entri pada matriks A. Elemen dapat dihitung menggunakan rumus seperti yang tertera pada subbab II Bab 2. Jadi matriks kofaktor dapat dituliskan sebagai berikut:

Adjoin dari matriks adalah *transpose* dari matriks kofaktor untuk matriks . Matriks adjoin berguna untuk mencari *invers* (matriks balikan) yang penjelasan lebih lanjutnya akan dijelaskan pada subbab berikutnya (subbab 3). Notasi dari adjoin suatu matriks dapat dituliskan sebagai .

1. **Matriks Balikan**

Sebuah matriks bisa diinverskan apabila memiliki sehingga memiliki ukuran yang sama dengan . Sehingga bisa dikatakan, apapun yang terjadi pada A, maka pada akan berlaku sebaliknya. Hasil perkalian dari keduanya adalah berupa matriks identitas sebagai berikut:

Ada 2 cara dalam mencari matriks balikan yaitu dengan matriks kofaktor dan dengan metode reduksi baris. Pada metode matriks kofaktor, matriks balikan dapat dihitung menggunakan rumus:

,

dimana adjoin didapatkan dengan melakukan *transpose* terhadap matriks kofaktor. Sedangkan untuk metode reduksi baris, matriks balikan diperoleh dengan cara membuat matriks *augmented* berukuran yang tersusun atas matriks dan matriks identitas lalu diterapkan OBE hingga bagian kiri matriks augmented (matriks ) berbentuk matriks identitas dan matriks balikannnya adalah matriks pada sisi kanan matriks *augmented*,

1. **Kaidah *Cramer***

Kaidah *Cramer* adalah rumus atau metode yang dapat digunakan untuk menyelesaikan system persamaan linear dengan jumlah variabel sama dengan jumlah persamannya sehingga kaidah *Cramer* hanya berlaku pada sistem persamaan linear yang memiliki solusi tunggal / unik. Untuk mendapatkan solusi dari suatu SPL dengan buah persanaan dan buah variabel, metode ini membutuhkan nilai determinan dari matriks koefisien dan determinan dari buah buah matriks lain yang masing-masing diperoleh dengan mengganti salah satu kolom matriks koefisien dengan matriks konstanta (ruas kanan pada SPL). Jika merupakan SPL yang terdiri dari persamaan dengan buah variabel, maka solusi dari SPL tersebut adalah:

, , ,

dengan,

,  , … ,

1. **Interpolasi Polinom**

Interpolasi polinom adalah salah satu metode untuk memprediksi nilai dari sebuah titik dengan memperkirakan nilai tersebut pada sembarang titik pada selang . Bentuk polinom interpolasi derajat yang menginterpolasikan titik-titik adalah . Dengan menyulihkan semua nilai untuk , akan didapatkan buah sistem persamaan lanjar

Solusi sistem persamaan lanjar ini, yaitu nilai , diperoleh dengan menggunakan metode eliminasi Gauss. Sebagai contoh, misalkan diberikan tiga buah titik yaitu , , dan . Tentukan polinom interpolasi kuadratik lalu estimasi nilai fungsi pada . Polinom kuadratik berbentuk . Dengan menyulihkan ketiga buah titik data ke dalam polinom tersebut, diperoleh sisten persamaan lanjar yang terbentuk adalah

Penyelesaian sistem persamaan dengan metode eliminasi Gauss menghasilkan , , dan . Polinom interpolasi yang melalui ketiga buah titik tersebut adalah . Dengan menggunakan polinom ini, maka nilai fungsi pada dapat ditaksir sebagai berikut: .

1. **Regresi Linier Berganda**

Regresi Linear 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.

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

Text

Description automatically generated

Sistem persamaan linier tersebut diselesaikan dengan menggunakan metode eliminasi Gauss.

# **BAB 3 IMPLEMENTASI PROGRAM**

1. **Main *Class***

*Main Class* adalah *class* program utama dari tugas besar 1 ini. Pada *Main Class*, hanya terdapat satu prosedur yaitu *main*. Pada prosedur ini, pertama-tama pengguna akan diberikan informasi ringkas tentang apa yang bisa dilakukan program ini kemudian akan ditampilkan menu yang bisa pengguna pilih sesuai kebutuhan. Terdapat enam menu, yaitu Sistem Persamaan Linier, Determinan, Matriks Balikasn, Interpolasi Polinom, dan Regresi Linier Berganda. Setelah memilih menu, implementasi algoritma untuk masing-masing menu akan dijelaskan lebih lanjut pada *class* lainnya.

Setiap kali pengguna selesai melakukan perhitungan pada suatu menu tertentu, pengguna akan diberikan opsi untuk kembali melanjutkan perhitungan dengan program atau keluar. Apabila pengguna ingin melanjutkan perhitungan, pengguna akan kembali ke halaman menu dan bisa kembali memilih menu yang diinginkan. Sedangkan, apabila pengguna ingin keluar, pengguna akan langsung keluar dari program.

1. **Gauss\_gauss\_jordan *Class***

Fungsi yang digunakan sebagai fungsi utama operator adalah sebagai berikut secara terurut: elimination\_before(), gauss(), dan gauss\_jordan(). Prosedur utama yang digunakan pada kelas ini adalah gauss\_jordan\_main() yang akan dipanggil pada *Main Class*. Sedangkan, fungsi yang lainnya adalah fungsi yang dibuat untuk mendukung ketiga fungsi di atas dan digunakan untuk memenuhi kondisi-kondisi tertentu.

Seluruh fungsi di bawah ini disimpan di dalam satu *class* yaitu gauss\_gauss\_jordan.*class* dan *file* gauss\_gauss\_jordan.java.

1. public static void swap(float m[][], int i, int j, int neffrow, int neffcols)

**procedure** swap(input/output m: array of array of real, input i: integer, input j: integer,  input neffrow: integer, input neffcols: integer)

{ I.S. terdapat matriks yang sedang dilakukan pengecekan baris

F.S. terjadi penukaran baris }

**KAMUS LOKAL**

temp: real

k: integer

**ALGORITMA**

i traversal [0..neffcols-1]

temp ß m[j][k]

m[j][k] ß m[i][k]

m[i][k] ß temp

1. public static void check\_gauss(float m[][], int neffrow, int neffcols, int swap\_counter[])

**procedure** check\_gauss(input/output m: array of array of real, input neffrow: integer, input neffcols: integer, input/output swap\_counter: array of integer)

{ I.S. matriks belum rapi dan terstruktur untuk dilakukan eliminasi gauss

F.S. matriks sudah terurut dan siap untuk diubah ke bentuk matriks

eselon }

**KAMUS LOKAL**

h, i, j, n: integer

**ALGORITMA**

h traversal [0..neffrow-1]

i traversal [1..neffrow-1]

j traversal [0..neffcols-1]

if (m[i][j] ≠ 0 and m[i-1][j] = 0) then

n ß i – 1

swap(m, i, n , neffrow, neffcols)

swap\_counter[0] ß swap\_counter[0] + 1

break

1. public static void gauss(float m[][], int neffrow, int neffcols)

**procedure** gauss(input/output m: array of array of real, input neffrow: integer, input neffcols: integer, input/output swap\_counter: array of integer)

{ I.S. sudah dieliminasi, terurut, dan membentuk matriks segitiga bawah

F.S. akan terbentuk matriks eselon }

**KAMUS LOKAL**

i, j, k: integer

divider: real

**ALGORITMA**

i traversal [0..neffrow-1]

j traversal [0..neffcols-1]

if (m[i][j] ≠ 0) then

divider ß m[i][j]

k traversal [j..neffcols]

m[i][k] ß m[i][k] `div` divider

break

1. public static void gauss\_jordan(float m[][], int neffrow, int neffcols)

**procedure** gauss\_jordan (input/output m: array of array of real, input neffrow: integer, input neffcols: integer)

{ I.S. Sudah di lakukan eliminasi gauss

F.S. Terbentuk matriks eselon baris tereduksi }

**KAMUS LOKAL**

index\_row, i, j, k, z: integer

arr: array of integer

count: integer

divider, idx: real

**ALGORITMA**

index\_row ß -999

count ß 0

i traversal [0..neffcols-1]

arr[i] ß -9999

i traversal [0..neffrow-2]

j traversal [0..neffcols-1]

index\_row ß search\_row(m, j, neffrow, i)

if (index ≠ -999) then

if (check\_availability(arr, index\_row, j)) then

arr[j] ß index\_row

divider ß m[i][j] / m[index\_row][j]

k traversal [0..neffcols-1]

idx ß m[index\_row][k]

m[i][k] ß m[i][k] – divider\*idx

else

count ß index\_row

while (check\_availability (arr, index\_row, j) = false and count < neffrow) do

index\_row ß search\_row(m, j, neffrow, index\_row)

count ß count + 1

z traversal [0..neffcols–1]

arr[z] ß -9999

1. public static boolean check\_availability(int a[], int val, int j)

**function** check\_availability (a: array of integer, val: integer, j: integer) à boolean

{ Mengecek apakah baris ini sudah digunakan untuk mengeliminasi atau belum oleh elemen yang lainnya, menghindari hasil yang negatif }

**KAMUS LOKAL**

flag: boolean

k: integer

**ALGORITMA**

flag ß true

k traversal [0..j-1]

if (a[k] = val) then

flag ß false

break

à flag

1. public static int search\_row(float m[][], int j, int neffrow, int i)

**function** search\_row(m: array of array of real, j: integer, neffrow: integer, i: integer) à integer

{ Mencari row untuk mengeliminasi baris yang dituju }

**KAMUS LOKAL**

index: integer

k: integer

**ALGORITMA**

index ß -999

k traversal [i+1..neffrow-1]

if (m[k][j] = 1) then

index ß k

break

à index

1. public static void elimination\_before(float m[][], int neffrow, int neffcols, int swap\_counter[])

**procedure** elimination\_before (input/output m: array of array of real, input neffrow: integer, input neffcols: integer, input swap\_counter: array of integer)

{I.S. matriks sudah di swap terlebih dahulu

F.S. terbentuk matriks dengan karakteristik segitiga bawah}

**KAMUS LOKAL**

idx1, idx2: real

i, j, k: integer

count: integer

M: array [0...neffcols-1] of array [0...neffrows-1] of real

**ALGORITMA**

check\_gauss(m, neffrow, neffcols, swap\_counter)

if (neffrow > neffcols) then

i traversal [1..neffrow-1]

if (i neffcols) then

count ß neffcols

else

count ß i

j traversal [0..i-1]

if (m[j][j] = 0) then

if (is\_singular(m, j, i, neffcols)) then

{ do nothing }

else

check\_gauss(m, neffrow, neffcols, swap\_counter)

i ß 1

break

else

idx1 ß m[i][j]

idx2 ß m[j][j]

k traversal [0..neffcols-1]

m[i][k] ß m[i][k] – (idx1\*m[j][k]) `div` 2

m[i][j] ß 0

else

i traversal [1..neffrow-1]

j traversal [0..i-1]

if (m[j][j] = 0) then

if (is\_singular(m, j, i, neffcols)) then

{do nothing}

else

check\_gauss(m, neffrow, neffcols, swap\_counter)

i ß 1

break

else

idx1 ß m[i][j]

idx2 ß m[j][j]

k traversal [0..neffcols-1]

m[i][k] ß m[i][k] – (idx1\*m[j][k]) `div` 2

m[i][j] ß 0

1. public static boolean is\_singular(float m[][], int j, int i, int neffcols)

**function** is\_singular(m: array of array of real, j: integer, i: integer, neffcols: integer) à boolean

{ini adalah fungsi yang digunakan di dalam proses eliminasi untuk melihat apakah kedua baris sama2 matriks yang seluruh elemennya nol, menentukan apakah perlu di swap atau tidak}

**KAMUS LOKAL**

flag: boolean

k: integer

**ALGORITMA**

flag ß true

k traversal [0..neffcols-1]

if (m[i][k] ≠ m[j][k]) then

if (m[i][k] = 0 && m[j][k] ≠ 0) then

flag ß true

break

else if (m[i][k] ≠ 0 && m[j][k] = 0) then

flag ß false

break

à flag

1. public static boolean swap\_singular(float m[][], int neffrow, int neffcols, int [] swap\_counter)

**procedure** swap\_confirm (input/output m: array of array of real, input neffrow: integer, input neffcols: integer, input swap\_counter: array [0] of integer)

{I.S. Sudah melewati fase eliminasi segitiga bawah untuk dianalisis apakah ada yang harus di eliminasi kembali

F.S. Fungsi berhasil dieliminasi}

**KAMUS LOKAL**

idx, idx2, i, k: integer

divider: float

**ALGORITMA**

i…traversal [0…neffrow-2]

k…traversal [0…neffcols-1]

if (m[i][k] != 0) then

if (m[i+1][k] != 0) then

idx2 <- k

idx <- i

divider <- m[idx+1][idx2] / m[idx][idx2]

z…traversal [idx2…neffcols-1]

m[idx+1][z] <- m[idx][z]\*divider

m[idx+1][idx2] <- 0

i <- 0

elimination\_before(m\_ neffrow, neffcols, swap\_counter)

break

else

break

1. **Determinan *Class***

*Determinan Class* adalah kelas yang berisi fungsi-fungsi dan prosedur untuk menghitung nilai determinan suatu matriks dengan 2 metode yaitu metode reduksi baris dan metode ekspansi kofaktor. Pada kelas ini, terdapat 7 *method* yang bersifat *public*. *Method*-*method* tersebut adalah:

1. public static void determinan()

*Method* ini berbentuk prosedur dan bersifat *public* sehingga bisa digunakan pada kelas-kelas lainnya, terutama pada *Main Class*. *Method* ini merupakan *method* utama dari kelas ini yang akan dipanggil di *Main Class* pada bagian menu determinan. *Method* ini akan membaca masukan matriks dari pengguna berupa *input* *keyboard* atau *file* dengan memanggil *method* public static float[][] readInput(boolean isSquare) dari *ReadDisplayArray Class* dengan parameter *input*-nya bernilai *true* karena nilai determinan hanya terdefinisi apabila matriksnya berbentuk persegi . Setelah itu akan dilakukan validasi juga untuk masukan dari *file* apakah berbentuk matriks persegi atau tidak. Kemudian, akan ditampilkan submenu berupa pilihan metode yang ingin digunakan untuk menghitung determinan yaitu metode reduksi baris dan metode ekspansi kofaktor. Apabila submenu yang dipilih adalah metode reduksi baris, maka akan dipanggil *method* public static float detKofaktor(float[][] matriks), sedangkan apabila submenu yang dipilih adalah metode ekspansi kofaktor, maka akan dipanggil *method* public static float detKofaktor(float[][] matriks). Pada akhir *method* ini, program akan memberi opsi jenis *output* apa yang diinginkan (*keyboard* atau *file*).

**procedure** determinan()

**KAMUS LOKAL**

matriks: array of array of real

resultString: string

row: integer

choiceMenu: integer

chooseInput: integer

**ALGORITMA**

matriks ß ReadDisplayArray.readInput(true)

resultString ß ""

row traversal [0..matriks.length-1]

resultString ß resultString + Arrays.toString(matriks[row]) + "\n"

if (matriks.length ≠ matriks[0].length) then

resultString ß resultString + "Matriks di atas tidak memiliki determinan karena bukan matriks persegi"

output(resultString)

else

output("\nSUB-MENU DETERMINAN")

output("1. Metode Reduksi Baris\n2. Metode Ekspansi Kofaktor\n")

choiceMenu ß Utils.chooseOptionValidation(1, 2)

if (choiceMenu = 1) then

resultString ß resultString + "\nHasil determinan untuk matriks di atas dengan metode reduksi baris adalah " + Determinan.detReduksiBaris(matriks)

else {(choiceMenu = 2)}

resultString ß resultString + "\nHasil determinan dari matriks di atas dengan metode ekspansi kofaktor adalah " + Determinan.detKofaktor(matriks)

output("Jenis output apa yang ingin diberikan: ")

output("1. Keyboard\n2. File\n")

chooseInput ß Utils.chooseOptionValidation(1, 2)

if (chooseInput = 2) then

Random randomNum ß new Random()

FileWriter DetFile ß new FileWriter("../test/Determinan" + randomNum.nextInt(100) + ".txt")

DetFile.write(resultString)

DetFile.close()

else {(chooseInput = 2)}

output(resultString)

1. public static float[][] minor(float[][] matriks, int i, int j)

*Method* ini berbentuk fungsi dan bersifat *public* sehingga dapat digunakan pada kelas-kelas yang lain. *Method* ini berfungsi untuk menghasilkan minor () dari suatu matriks yaitu dengan cara menghilangkan / menghapus baris dan kolom pada matriks yang di-*input* melalui iterasi (*traversal*) pada setiap elemen matriks. Parameter dari fungsi ini berupa matriks yang ingin dicari minornya serta *integer* dan yang berasosiasi dengan . *Method* ini akan mengembalikan nilai bertipe float[][].

**function** minor(matriks: array of array of real, i: integer, j: integer) à array of array of real

**KAMUS LOKAL**

size: integer

result: array of array of real

a, b, c, d: integer

**ALGORITMA**

size ß matriks.length – 1

result ß array[0..size-1] of array[0..size-1] of real

c ß 0

a traversal [0..matriks.length-1]

if (a = i) then {Skip baris i}

continue

d ß 0

b traversal [0..matriks.length-1]

if (b = j) then {Skip kolom j}

continue

result[c][d] ß matriks[a][b]

d ß d + 1

c ß c + 1

à result

1. public static float nilaiKofaktor(float[][] matriks, int i, int j)

*Method* ini berbentuk fungsi juga dan bersifat *public* sehingga dapat diakses atau digunakan pada kelas-kelas lainnya. Parameter fungsi ini berupa matriks yang ingin dicari nilai kofaktornya, serta *integer* dan yang berasosiasi dengan . Fungsi ini berfungsi untuk menghitung nilai kofaktor dengan memanggil *method* public static float detKofaktor(float[][] matriks) yang parameter matriksnya diisi dengan memanggil *method* public static float[][] minor(float[][] matriks, int i, int j). *Method* ini akan mengembalikan sebuah nilai bertipe *float*.

**function** nilaiKofaktor(matriks: array of array of real, i: integer, j: integer) à real

**KAMUS LOKAL**

**ALGORITMA**

à detKofaktor(minor(matriks, i, j)) \* Math.pow(-1, i + j)

1. public static float[][] matriksKofaktor(float[][] matriks)

*Method* ini juga berbentuk fungsi dan bersifat *public* sehingga dapat digunakan pada kelas-kelas lainnya. *Method* ini berfungsi untuk menghasilkan matriks kofaktor yaitu matriks yang elemennya adalah kofaktor-kofaktor dari entri matriks yang di-*input*. *Method* ini akan memanggil *method* public static float nilaiKofaktor(float[][] matriks, int i, int j) untuk mengisi elemen-elemen dari matriks kofaktor. *Method* ini akan mengembalikan nilai bertipe float[][].

**function** matriksKofaktor(matriks: array of array of real) à array of array of real

**KAMUS LOKAL**

size: integer

result: array of array of real

**ALGORITMA**

size ß matriks.length

result ß array[0..size-1] of array[0..size-1] of real

a traversal [0..size-1]

b traversal [0..size-1]

result[a][b] ß nilaiKofaktor(matriks, a, b)

à result

1. public static float[][] adjoin(float[][] matriks)

*Method* ini juga berbentuk fungsi dan bersifat *public* sehingga dapat digunakan pada kelas-kelas lainnya, terutama *Invers Class*. *Method* ini berfungsi untuk menghasilkan matriks adjoin yaitu matriks *transpose* dari matriks kofaktor. *Method* ini akan memanggil *method* public static float[][] matriksKofaktor(float[][] matriks) yang digunakan sebagai parameter *input* fungsi ini. *Method* ini akan mengembalikan nilai bertipe float[][].

**function** adjoin(matriks: array of array of real) à array of array of real

**KAMUS LOKAL**

size: integer

result: array of array of real

kofaktor: array of array of real

**ALGORITMA**

size ß matriks.length

result ß array[0..size-1] of array[0..size-1] of real

kofaktor ß matriksKofaktor(matriks)

a traversal [0..size-1]

b traversal [0..size-1]

result[a][b] ß kofaktor[b][a]

à result

1. public static float detKofaktor(float[][] matriks)

*Method* ini juga berbentuk fungsi dan bersifat *public* sehingga dapat digunakan pada kelas-kelas lainnya, terutama *Invers Class* dan *Cramer Class*. *Method* ini digunakan untuk menghitung determinan matriks dengan metode ekspansi kofaktor. Fungsi ini bekerja secara rekursif dengan terus menerus menghitung determinan dari minornya hingga matriks tidak memiliki minor lagi. Jadi dalam penerapnnya, fungsi ini akan memanggil *method* public static float[][] minor(float[][] matriks, int i, int j). Fungsi ini akan mengembalikan sebuah nilai bertipe data *float*.

**function** detKofaktor(matriks: array of array of real) à real

**KAMUS LOKAL**

size: integer

result: integer

j: integer

**ALGORITMA**

size ß matriks[0].length

result ß 0

if (size = 1) then

à matriks[0][0]

else if (size = 2) then

à (matriks[0][0] \* matriks[1][1]) - (matriks[1][0] \* matriks[0][1])

else

j ß 0

i traversal [0..size-1]

result ß result + matriks[i][j] \* detKofaktor(minor(matriks, i, j)) \* Math.pow(-1, i + j)

à result

1. public static float detReduksiBaris(float[][] matriks)

*Method* ini juga berbentuk fungsi dan bersifat *public* sehingga dapat digunakan pada kelas-kelas lainnya, terutama *Invers Class* dan *Cramer Class*. *Method* ini digunakan untuk menghitung determinan matriks dengan metode reduksi baris. Jadi dalam penerapnnya, fungsi ini akan memanggil *method* public static void elimination\_before(float m[][], int neffrow, int neffcols, int swap\_counter[]) untuk melakukan OBE. Fungsi ini akan mengembalikan sebuah nilai bertipe *float*.

**function** detReduksiBaris(matriks: array of array of real) à real

**KAMUS LOKAL**

size: integer

swap: array of integer

det: real

**ALGORITMA**

swap ß array[1] of integer

size ß matriks.length

if (size = 1) then

à matriks[0][0]

else

det ß 1

gauss\_gauss\_jordan.elimination\_before(matriks, size, size, swap)

i ß 0

i traversal [0..size-1]

det ß det \* matriks[i][i]

à det \* Math.pow(-1, swap[0])

1. **Invers *Class***

Penjelasan:

1. Terdapat 1 prosedur utama yaitu InversMain() yang akan digunakan pada *Main Class*. Fungsi utama pada kelas ini adalah invers\_mat\_reduc(), invers\_mat\_kofaktor(), dan SPLInvers() yang parameternya berupa array dua dimensi, fungsi yang lainnya adalah fungsi pendukung yakni multiply\_matrix()
2. Disimpan dalam *class* sendiri invers.*class* dan *file* invers.java

Beberapa fungsi yang terdapat pada kelas ini adalah sebagai berikut:

1. public static float[][] invers\_mat\_kofaktor(float a[][])

**function** invers\_mat\_kofaktor (a: array of array of real) à array of array of real

{ Fungsi utama untuk melakukan invers }

**KAMUS LOKAL**

determinan: real

size: integer

adjoint: array of array of real

array: array [0..size-1] of array [0..size-1] of real

**ALGORITMA**

size ß a.length

determinan ß Determinan.detKofaktor(a)

array ß multiply\_matrix(adjoint, 1/determinan)

à array

1. public static float[][] multiply\_matrix\_kofaktor(float a[][], float var)

**function** multiply\_matrix\_kofaktor (a: array of array of real, var: real) à array of array of real

{ Mengalikan matrix dengan suatu konstanta }

**KAMUS LOKAL**

size: integer

array: array [0..size-1] of array [0..size-1] of real

i, j: integer

**ALGORITMA**

size ß a.length

i traversal [0..size-1]

j traversal [0..size-1]

array[i][j] ß a[i][j] \* var

à array

1. public static float[][] multiply\_matrix\_reduc(float a[][], float var)

**function** multiply\_matrix\_reduc (a: array of array of real, var: real) à array of array of real

{ Mengalikan matrix dengan suatu konstanta untuk reduksi }

**KAMUS LOKAL**

size: integer

array: array [0..size-1] of array [0..size-1] of real

i, j: integer

**ALGORITMA**

size ß a.length

i traversal [0..size-1]

j traversal [0..size-1]

array[i][j] ß a[i][j] \* var

à array

1. public static float[][] multiply\_matrix\_reduc(float a[][], float var)

**function** invers\_mat\_reduc (a: array of array of real) à array of array of real

{ Fungsi utama untuk melakukan invers menggunakan reduksi}

**KAMUS LOKAL**

determinan: real

size: integer

adjoint: array of array of real

array: array [0..size-1] of array [0..size-1] of real

**ALGORITMA**

size ß a.length

determinan ß Determinan.detKofaktor(a)

array ß multiply\_matrix(adjoint, 1/determinan)

à array

1. ***Cramer* *Class***

*Cramer Class* adalah kelas yang berisi fungsi untuk menyelesaikan SPL (Sistem Persamaan Linier) dengan persamaan dan variabel menggunakan Kaidah *Cramer*. Pada kelas ini, terdapat 3 *method* yang terdiri dari 2 *method* *public* dan 1 *method private*. Kedua *method* tersebut adalah:

1. public static String CramerMain(float[][] matriks, int rows, int cols, String resultString)

*Method* ini berbentuk fungsi dan bersifat *public* sehingga dapat dipanggil oleh kelas lain, terutama *Main Class*. *Method* ini merupakan *method* utama pada kelas ini yang akan dipanggil pada *Main Class*. Nilai (*value*) yang dikembalikan fungsi ini bertipe *string*.

1. private static float[][] insertConst(float[][] matriks, int j, float[] constant)

*Method* ini berbentuk fungsi dan bersifat *private* sehingga tidak akan bisa diakses atau dipanggil oleh kelas lain. *Method* ini berfungsi untuk membentuk matriks yang salah satu kolomnya sudah diganti dengan matriks konstanta yaitu matriks yang elemennya adalah konstanta / bagian ruas kanan pada SPL. Nilai (*value*) yang dikembalikan fungsi ini bertipe float[][].

**function** insertConst(matriks: array of array of real, j: integer, constant: array of real) à array of array of real

**KAMUS LOKAL**

rows, cols: integer

result: array of array of real

**ALGORITMA**

rows ß matriks.length

cols ß matriks[0].length

result ß array[0..rows-1] of array[0..cols-1] of real

l traversal [0..rows-1]

k traversal [0..cols-1]

result[l][k] ß matriks[l][k]

i traversal [0..rows-1]

result[i][j] ß constant[i]

à result

1. public static float[] cramerSol(float[][] mAugmented)

*Method* ini berbentuk fungsi dan bersifat *public* sehingga dapat diakses atau dipanggil pada kelas-kelas yang lain karena *method* ini merupakan *method* utama dari kelas ini. *Method* ini berfungsi untuk menghitung solusi dari suatu SPL. *Method* ini akan memanggil *method* public static float detKofaktor(float[][] matriks) dan *method* private static float[][] insertConst(float[][] matriks, int j, float[] constant) untuk menghitung nilai determinan sesuai dengan rumus yang sudah dipaparkan pada subbab V Bab 2. Nilai yang dikembalikan fungsi ini memiliki tipe data float[] yang menampung solusi dari semua variabel pada SPL.

**function** cramerSol(mAugmented: array of array of real) à array of real

**KAMUS LOKAL**

rows, cols: integer

mKoef: array of array of real

temp: array of array of real

detKoef: real

constant: array of real

solution: array of real

**ALGORITMA**

rows ß mAugmented.length

cols ß mAugmented[0].length

mKoef ß array[0..rows-1] of array[0..cols-2] of real

i traversal [0..rows-1]

j traversal [0..cols-2]

mKoef[i][j] ß mAugmented[i][j]

detKoef ß Determinan.detKofaktor(mKoef)

constant ß array[0..rows-1] of real

i traversal [0..rows-1]

constant[i] ß mAugmented[i][cols-1]

solution ß array[0..rows-1] of real

j traversal [0..cols-1]

temp ß mKoef

solution[j] ß (Determinan.detKofaktor(insertConst(temp, j, constant))) / detKoef

à solution

1. **ReadDisplayArray *Class***

Fungsi utama dari *ReadDisplayArray Class* adalah untuk menangani pembacaan *input* dari pengguna berupa *input* *keyboard* atau *file*, baik dalam bentuk matriks maupun titik untuk polinom dan regresi, serta menangani *output* yang akan diberikan, *output* *keyboard* atau *file*, sesuai dengan keinginan pengguna. Terdapat 12 *method* pada *class* ini, namun hanya 6 *method* yang bersifat *public*. *Method*-*method* *public* tersebut adalah:

1. public static float[][] readInput(boolean isSquare)

*Method* berbentuk fungsi ini bersifat *public* dan dapat digunakan di kelas-kelas lainnya untuk membaca input dari pengguna, terkhusus untuk pembacaan matriks. *Method* ini memiliki parameter bertipe *boolean*, yaitu isSquare. Parameter ini untuk menentukan apakah input matriks yang nanti akan dibaca merupakan matriks persegi atau bukan.

*Method* ini, membaca *input* berupa *input* *keyboard* dan *file*. Apabila inputnya *file*, maka akan dipanggil *method* khusus yang membaca *file* dan mengembalikan sebuah matriks, yaitu *method* private static float[][] read*File*toMatrix(). Apabila *input* (berupa *input* *keyboard*) matriks berupa matriks persegi, maka *method* yang akan dipanggil oleh *method* ini adalah *method* private static float[][] readSquareMatrix() pada poin g, sedangkan, jika matriks bukan persegi, *method* ini akan memanggil *method* private static float[][] readMatrix(). *Method* ini, pada akhirnya, akan mengembalikan sebuah matriks bertipe float[][].

**function** readInput(isSquare: boolean) à array of array of real

**KAMUS LOKAL**

chooseInput: integer

matrix: array of array of real

**ALGORITMA**

output(“Jenis input apa yang ingin diberikan:”)

output(“1. Keyboard\n2.File\n”)

chooseInput ß Utils.chooseOptionValidation(1,2)

if chooseInput = 2 then

matrix ß readFiletoMatrix()

else

if isSquare then

matrix ß readSquareMatrix()

else

matrix ß readMatrix()

à matrix

1. public static float[][] readInputPoint()

*Method* ini juga berbentuk fungsi dan bersifat *public* sehingga dapat digunakan pada kelas-kelas lainnya untuk membaca *input* dari pengguna. *Method* fungsi ini dikhususkan untuk membaca buah poin/titik. *Method* ini tidak memiliki parameter apapun.

*Method* ini dapat membaca *input* berupa *input* *keyboard* dan *file*. Apabila inputnya *file*, maka akan dipanggil *method* khusus yang membaca *file* dan mengembalikan sebuah matriks yang berisi poin/titik, yaitu *method* private static float[][] read*File*toMatrix(). Apabila *input* berupa *keyboard*, akan dipanggil *method* private static float[][] readPointMatrix(). *Method* ini, pada akhirnya, akan mengembalikan sebuah matriks bertipe float[][].

**function** readInputPoint() à array of array of real

**KAMUS LOKAL**

chooseInput: integer

matrix: array of array of real

**ALGORITMA**

output(“Jenis input apa yang ingin diberikan:”)

output(“1. Keyboard\n2.File\n”)

chooseInput ß Utils.chooseOptionValidation(1,2)

if chooseInput = 2 then

matrix ß readFiletoMatrix()

else

matrix ß readPointMatrix()

à matrix

1. private static float[][] readRegressionPoints()

*Method* fungsi ini dikhususkan untuk membaca buah poin/titik sebanyak data. *Method* ini tidak memiliki parameter apapun. *Method* ini hanya dapat membaca *input* dari *keyboard*. *Method* ini, pada akhirnya, akan mengembalikan sebuah matriks bertipe float[][].

**function** readRegressionPoints() à array of array of real

**KAMUS LOKAL**

n, k, i, j: integer

matrix: array of array of real

**ALGORITMA**

output(“Masukkan jumlah titix x (n)\n n = ”)

input(n)

output(“Masukkan jumlah data (k)\n k = ”)

input(k)

output(“Masukkan input dengan format x1 x2 .. xn y sebanyak k data ”)

i traversal [0..k+1]

j traversal [0..n+2]

output(“”)

input(matrix[i][j])

à matrix

1. public static float[][] readRointMatrix()

*Method* fungsi ini dikhususkan untuk membaca buah poin/titik. *Method* ini tidak memiliki parameter apapun. *Method* ini hanya dapat membaca *input* dari *keyboard*. *Method* ini, pada akhirnya, akan mengembalikan sebuah matriks bertipe float[][].

**function** readPointMatrix() à array of array of real

**KAMUS LOKAL**

n, k, i, j: integer

matrix: array of array of real

**ALGORITMA**

output(“Masukkan jumlah titix x (n)\n n = ”)

input(n)

output(“Masukkan input dengan format x y”)

i traversal [0..n+1]

j traversal [0..2]

output(“”)

input(matrix[i][j])

à matrix

1. public static void displayOutput(float[][] matrix)

*Method* displayOutput berfungsi untuk menampilkan *output* kepada pengguna, baik melalui *text* pada layar *command* maupun *file*. Apabila *output* yang ditampilkan berupa matriks, maka akan dipanggil *method* private static void displayMatrix(float[][] matrix), sedangkan, apabila ingin menyimpannya dalam sebuah *file*, akan dipanggil *method* private static void create*File*FromMatrix(float[][] matrix). *Method* yang berfungsi untuk membaca nilai-nilai *float* pada sebuah *file* txt kemudian mengubahnya menjadi format matriks *float*.

**procedure** displayOutput(input matrix: array of array of real)

**KAMUS LOKAL**

chooseOutput: integer

**ALGORITMA**

output(“Jenis input apa yang ingin diberikan:”)

output(“1. Keyboard\n2.File\n”)

chooseOutput ß Utils.chooseOptionValidation(1,2)

if chooseOutput = 1 then

displayMatrix(matrix)

else

createFileFromMatrix(matrix)

output(“File matriks sudah berhasil tersimpan”)

1. public static void displayOutputSPL(float[][] matrix)

*Method* *displayOutputSPL* berfungsi untuk menampilkan *output* berupa persamaan SPL beserta taksiran nilainya kepada pengguna, baik melalui *text* pada layar *command* maupun *file*.

**procedure** displayOutputSPL(input resultString: string)

**KAMUS LOKAL**

chooseOutput: integer

SPLFile: File

**ALGORITMA**

output(“Jenis input apa yang ingin diberikan:”)

output(“1. Keyboard\n2.File\n”)

chooseOutput ß Utils.chooseOptionValidation(1,2)

if chooseOutput = 1 then

output(resultString)

else

SPLFile.write(resultString)

SPLFil*e*.close()

1. **Regresi Linier *Class***

Regresi Linier *Class* berfungsi untuk menyelesaikan permasalahan interpolasi polinom dan regresi linier. Terdapat dua *method* public pada *class* ini, yaitu:

1. public static void interpolasiPolinom()

*Method* ini berfungsi untuk menentukan nilai taksiran dengan menggunakan interpolasi polinom. Langkah-langkah algortimanya adalah sebagai berikut:

1. Membaca input berupa poin/titik dari *input* *keyboard* ataupun *file* dan menyimpan *input* dalam matriks
2. Matriks akan diubah dari bentuk poin/titik menjadi bentuk persamaan SPL sesuai dengan aturan polinom kemudian menyimpan matriks baru tersebut pada variabel matriks\_final
3. Menyelesaikan persamaan SPL interpolasi polinom dengan metode Gauss Jordan yang terdapat pada *Gauss\_gauss\_jordan* *Class*
4. Menaksir nilai dengan mensubstitusikan pada persamaan interpolasi polinom yang terbentuk
5. Menampilkan persamaan interpolasi polinom dan taksiran nilai , baik lewat *output* pada layar atau dalam bentuk *file*.

procedure interpolasiPolinom()

**KAMUS LOKAL**

matriks, matrix\_final, tempMatrix: array of array of real

swap\_counter: array[1] of integer

x, result: real

rows, cols, i, j, p, k, m, n: integer

**ALGORITMA**

output(“Jenis input apa yang ingin diberikan: \n1. Keyboard\n2.File\n”)

chooseInput ß Utils.chooseOptionValidation(1,2)

if chooseInput = 1 then

matriks ß ReadDisplayArray.readPointMatrix()

output(“Nilai x yang akan ditaksir = ”)

input(x)

else

tempMatrix ß ReadDisplayArray.readFiletoMatrix()

m ß 0

i traversal [0..tempMatrix.length-1]

n ß 0

j traversal [0..tempMatrix[0].length]

matriks[i][j] <- tempMatrix[m][n]

n ß n + 1

m ß m + 1

x ß tempMatrix[tempMatrix.length – 1][0]

rows ß matriks.length

cols ß rows + 1

k ß 0

i traversal [0..rows]

p ß 0

j traversal [0..cols]

if j = rows then

matrix\_final[i][j] ß matriks[k][1]

else

matrix\_final[i][j] ß Utils.power(matriks[k][0], p)

p ß p + 1

k ß k + 1

swap\_counter ß [1]

rows ß matrix\_final.length

cols ß matrik\_final[0].length

gauss\_gauss\_jordan.elimination\_before(matriks\_final, rows, cols, swap\_count er)

gauss\_gauss\_jordan.gauss(matriks\_final, rows, cols)

gauss\_gauss\_jordan.gauss\_jordan(matriks\_final, rows, cols)

SPL ß SPLInterpolasiRegresi(matrix\_final, true)

result ß 0

i traversal [0..rows]

result ß result + (matrix\_final[i][cols-1] \* Utils.power(x, i))

resultString ß SPL + “\nHasil taksiran nilai untuk x = " + x + " adalah " + result

ReadDisplayArray.displayOutputSPL(resultString)

1. public static void regresiLinier()

*Method* ini berfungsi untuk menentukan nilai taksiran dengan menggunakan metode regresi linier. Langkah-langkah algortimanya adalah sebagai berikut:

1. Membaca *input* berbentuk titik dan nilai sebanyak data baik dalam bentuk input *keyboard* maupun *file* dan menyimpan *input* dalam matriks
2. Matriks akan diubah dari bentuk titik dan nilai sebanyak data menjadi bentuk persamaan SPL sesuai dengan aturan regresi linier kemudian menyimpan matriks baru tersebut pada variabel matriks\_final
3. Menyelesaikan persamaan SPL regresi linier dengan metode Gauss Jordan yang terdapat pada *Gauss\_gauss\_jordan* *Class*
4. Menaksir nilai dengan mensubstitusikan pada persamaan regresi linier yang terbentuk
5. Menampilkan persamaan regresi linier dan taksiran nilai , baik lewat *output* pada layar atau dalam bentuk *file*.

procedure regresiLinier()

**KAMUS LOKAL**

main\_matriks, main\_matrix, matrix2, matrix\_final: array of array of real

n, k, i, j, row\_matrix2, col\_matrix2, b: integer

xs, array\_sum\_xy, array\_sum: array of real

swap: array of integer

SPL: string

result: real

**ALGORITMA**

output(“Jenis input apa yang ingin diberikan: \n1. Keyboard\n2.File\n”)

chooseInput ß Utils.chooseOptionValidation(1,2)

if chooseInput = 1 then

main\_matriks ß ReadDisplayArray.readRegressionPoints()

n ß main\_matriks[0].length – 1

k ß main\_matrix.length

else

main\_matriks ß ReadDisplayArray.read*File*toMatrix()

n ß main\_matriks[0].length – 1

k ß main\_matrix.length – 1

i traversal [0..k+1]

j traversal [0..n+2]

main\_matrix[i][j] ß main\_matriks[i][j]

if chooseInput = 1 then

output(“Nilai x yang akan ditaksir (masukkan sebanyak k data)”)

input(xs)

else

i traversal [0..n+1]

xs[i] ß main\_matriks[main\_matriks.length – 1][i]

array\_sum\_xy ß [n+1]

j traversal [0..n+2)

array\_sum\_xy[j] ß 0

i traversal [0..k+1]

array\_sum\_xy ß array\_sum\_xy + main\_matrix[i][j]

row\_matrix2 ß k

col\_matrix2 ß n \* (n + 1)

matrix2 ß [row\_matrix2][col\_matrix2]

i traversal [0..row\_matrix2+1]

j traversal [0..col\_matrix2+1]

if j < n then

matrix2[i][j] ß main\_matrix[i][n] \* main\_matrix[i][j]

else

matrix2[i][j] ß main\_matrix[i][(j/n)–1] \* main\_matrix[i][j%n]

array\_sum ß [col\_matrix2]

j traversal [0..col\_matrix2+1]

array\_sum[j] ß 0

i traversal [0..row\_matrix2+1]

array\_sum[j] ß array\_sum[j] + matrix2[i][j]

matrix\_final ß [n+1][n+2]

matrix\_final[0][0] ß k

matrix\_final[0][n+1] ß array\_sum\_xy[n]

j traversal [1..n+2]

matrix\_final[0][j] ß array\_sum\_xy[j-1]

i traversal [1..n+2]

matrix\_final[i][0] ß array\_sum\_xy[i-1]

matrix\_final[i][n+1] ß array\_sum[i-1]

b ß n

i traversal [1..n+2]

j traversal [1..n+2]

matrix\_final[i][j] ß array\_sum[b]

b ß b + 1

gauss\_gauss\_jordan.elimination\_before(matriks\_final, rows, cols, swap\_count er)

gauss\_gauss\_jordan.gauss(matriks\_final, rows, cols)

gauss\_gauss\_jordan.gauss\_jordan(matriks\_final, rows, cols)

SPL = SPLInterpolasiRegresi(matrix\_final, false)

result ß 0

i traversal [0..rows]

result ß result + (matrix\_final[i][n+1] \* xs[i-1])

resultString ß SPL + “\nNilai taksiran untuk x ” + xs + “ adalah ” + result

ReadDisplayArray.displayOutputSPL(resultString)

1. **Utils *Class***

Utils *Class* berisi *method*-*method* yang membantu dalam penyelesaian masalah pada program secara umum. Terdapat dua *method* public pada *class* ini yang dapat digunakan pada semua *class* di program ini, yaitu:

1. public static int chooseOptionValidation(int numStart, int numEnd)

*Method* yang berfungsi untuk memvalidasi *input* opsi dari pengguna agar selalu berada pada *range* [numStart…numEnd] sesuai dengan parameter yang diberikan.

1. public static void displayMenu()

*Method* untuk menampilkan menu pada *Main class.*

1. public static boolean goToMenu()

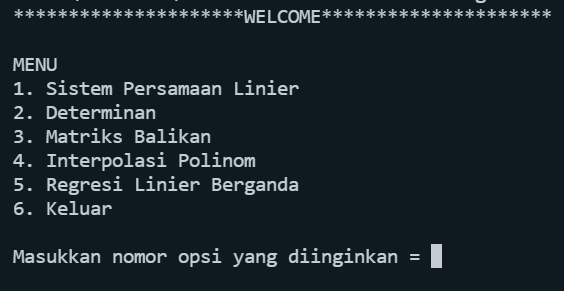
*Method* untuk menanyakan pengguna apakah masih ingin berada pada program (kembali ke menu) atau sudah selesai dan ingin keluar dari program. *Method* ini digunakan pada *Main class*.

1. public static float power(float x, int p)

Fungsi untuk mempangkatkan bilangan *float* sebanyak .

# **BAB 4 EKSPERIMEN**

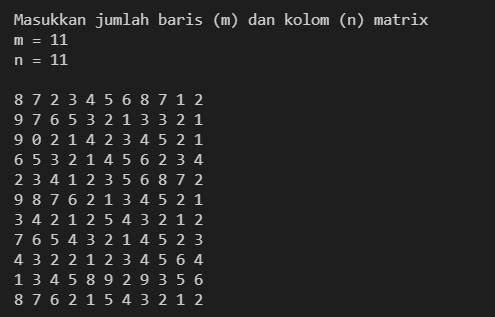
1. **Tampilan Awal dan Akhir**

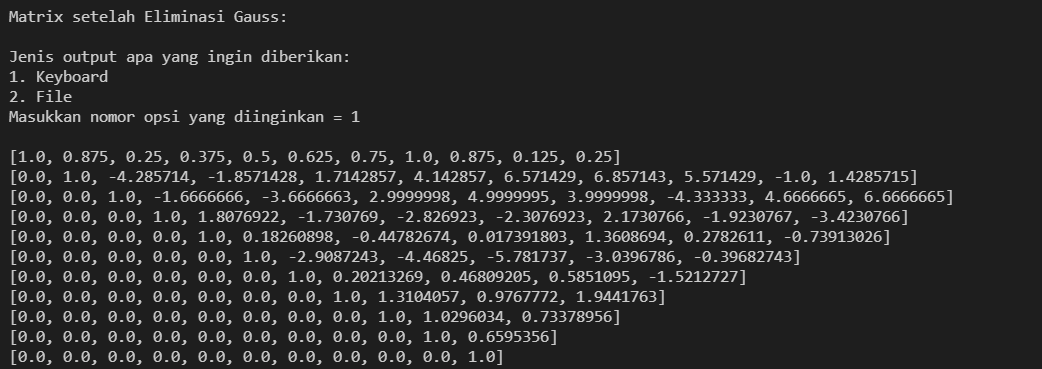
****

****

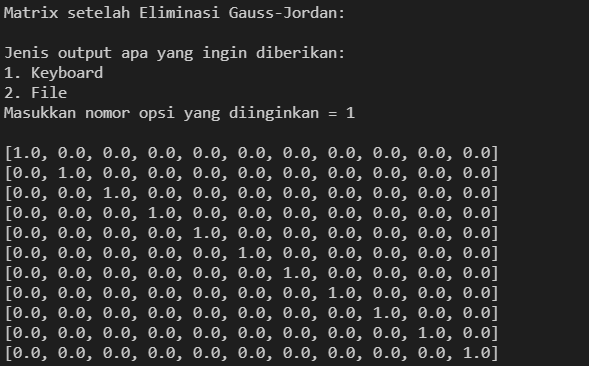
1. **Sistem Persamaan Liniear**

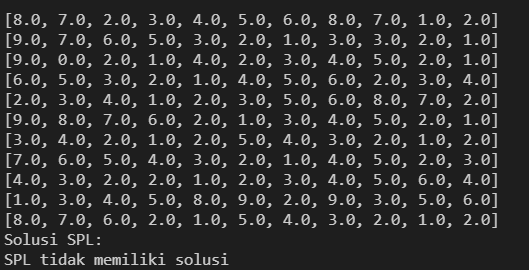
***Eksperimen 1* - *Eliminasi Gauss (Keyboard) Matrix 11 x 11***



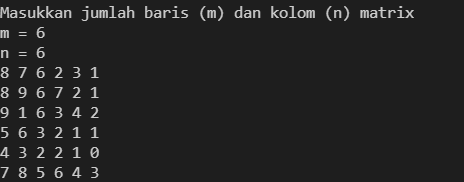


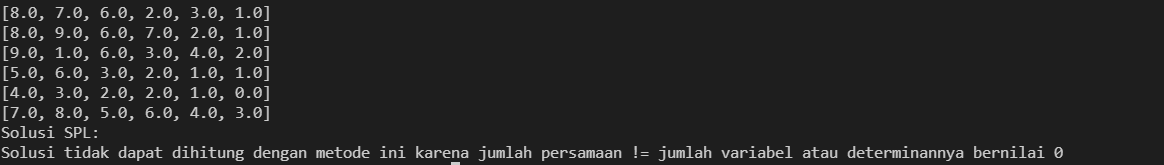
***Eksperimen 2* - *Eliminasi Gauss-Jordan (Keyboard) Matrix 11 x 11***



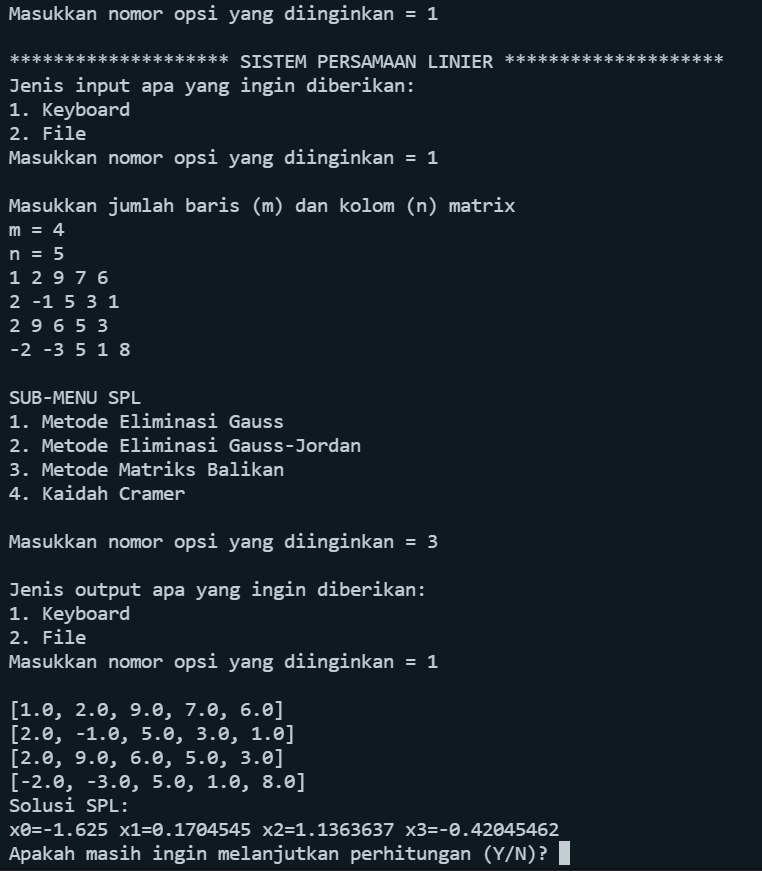


***Eksperimen 3* - *SPL dengan kaidah Cramer Matrix 6 x 6***



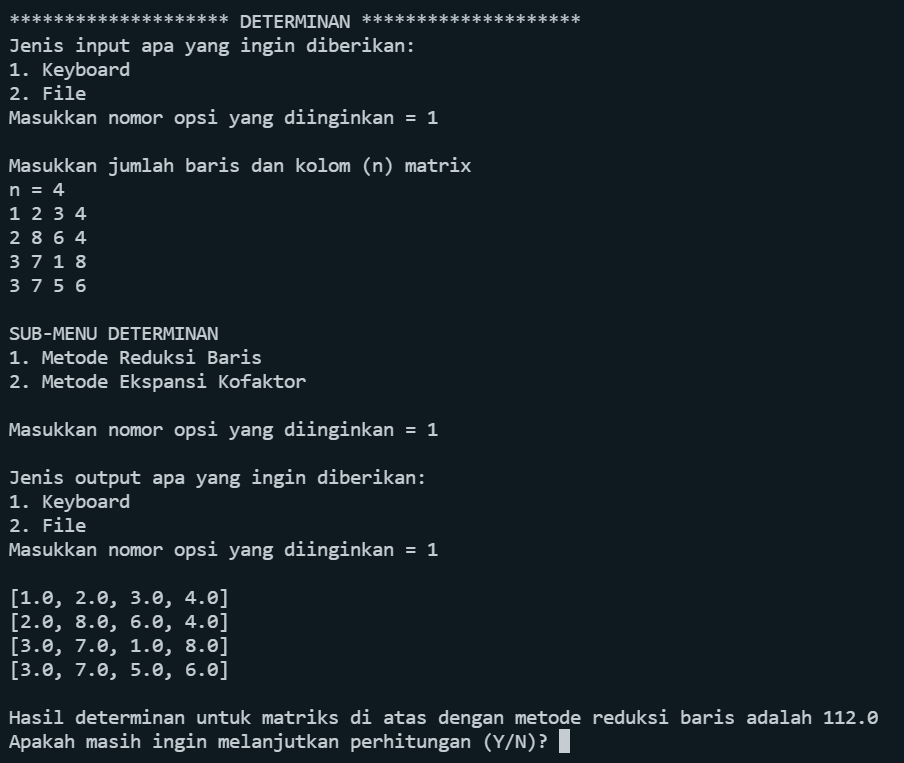


***Eksperimen 4 – SPL dengan metode matriks balikan***

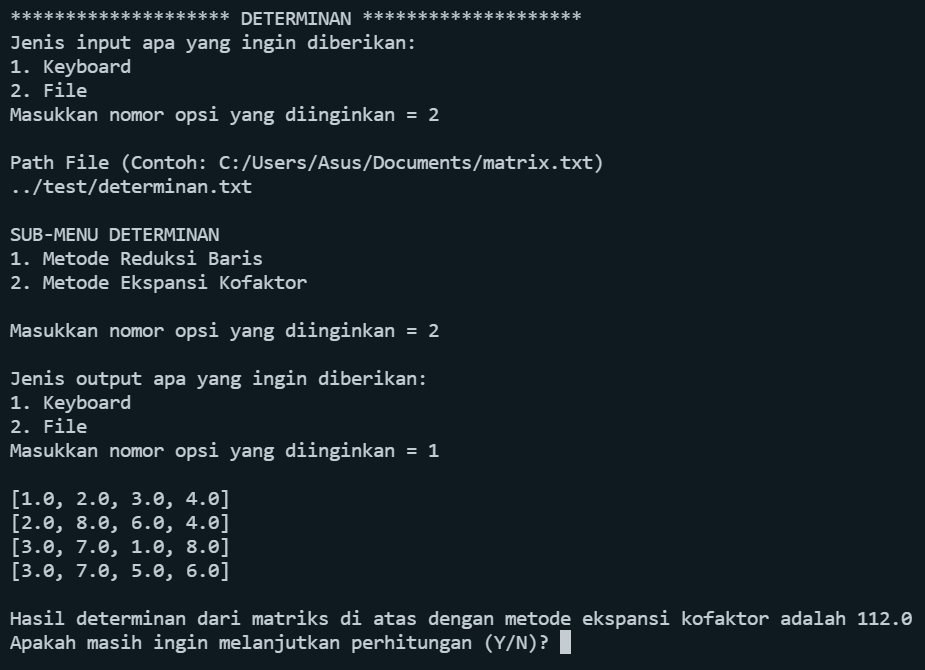


1. **Determinan**

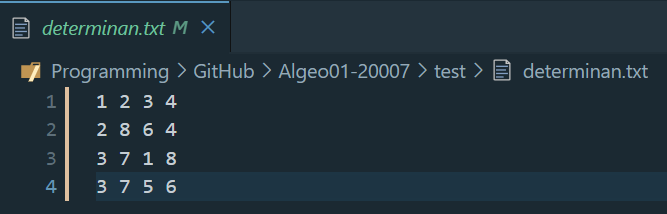
***Eksperimen 1 – Input dan Output Keyboard – Metode Reduksi Baris***



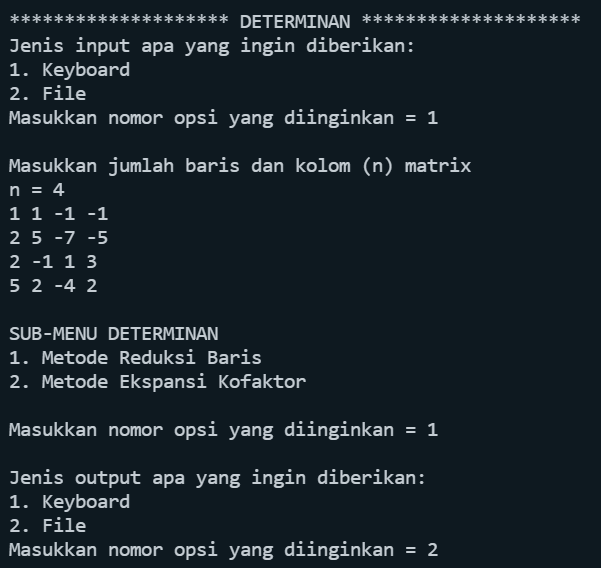
***Eksperimen 2 – Input File dan Output Keyboard – Metode Ekspansi Kofaktor***



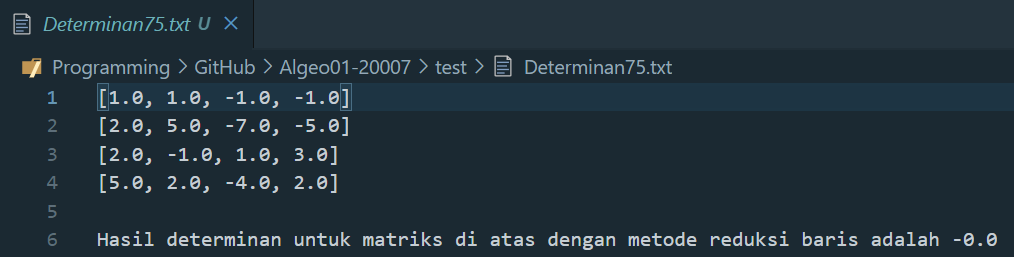
*Input file*: determinan.txt



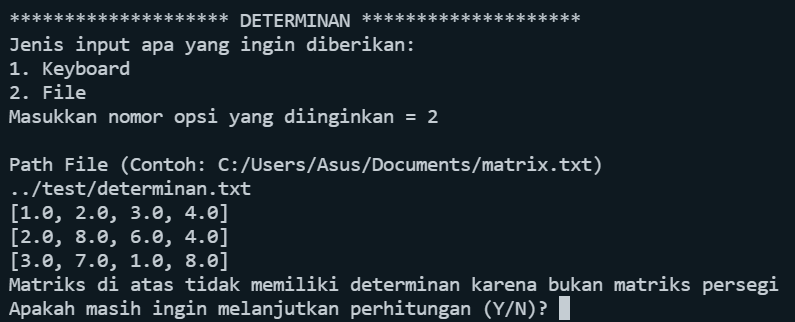
***Eksperimen 3 – Input Keyboard dan Output File – Metode Reduksi Baris***



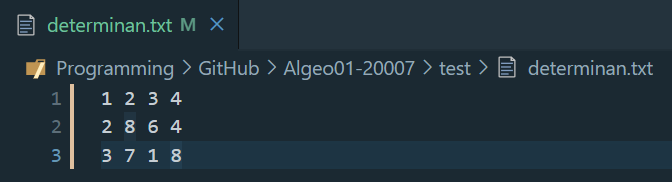
*Output* *file*: Determinan75.txt



***Eksperimen 4 – Input bukan matriks persegi***

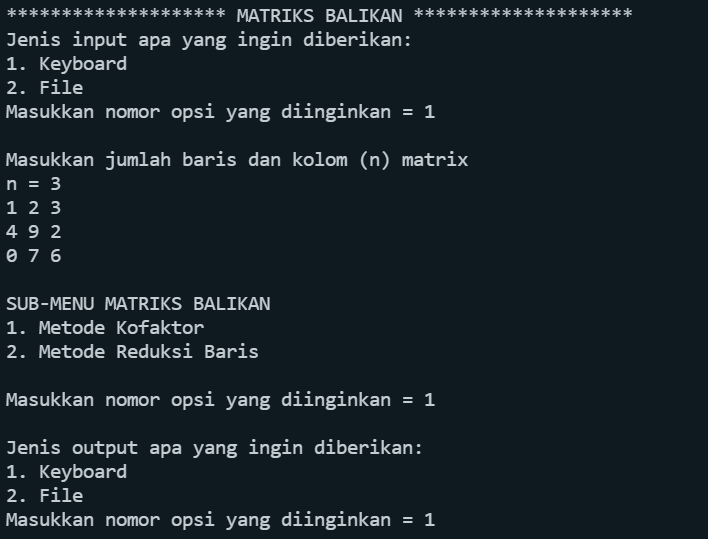


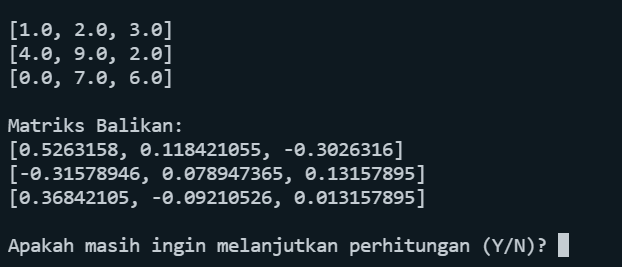
*Input file: determinan.txt*



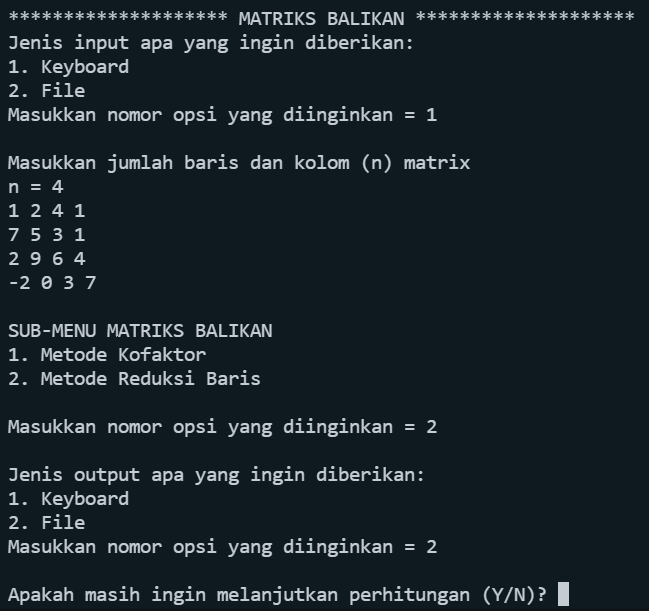
1. **Matriks Balikan**

***Eksperimen 1 – Input dan Output Keyboard – Metode Kofaktor***

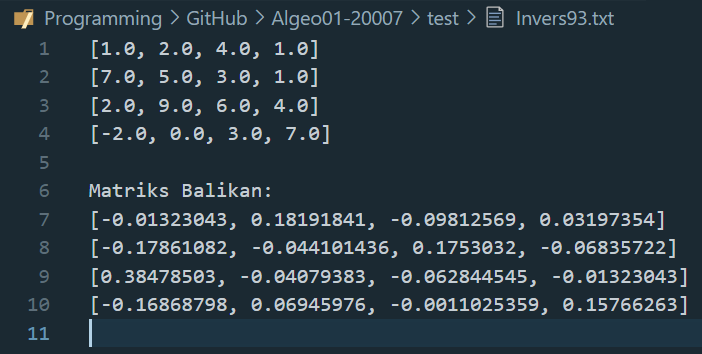




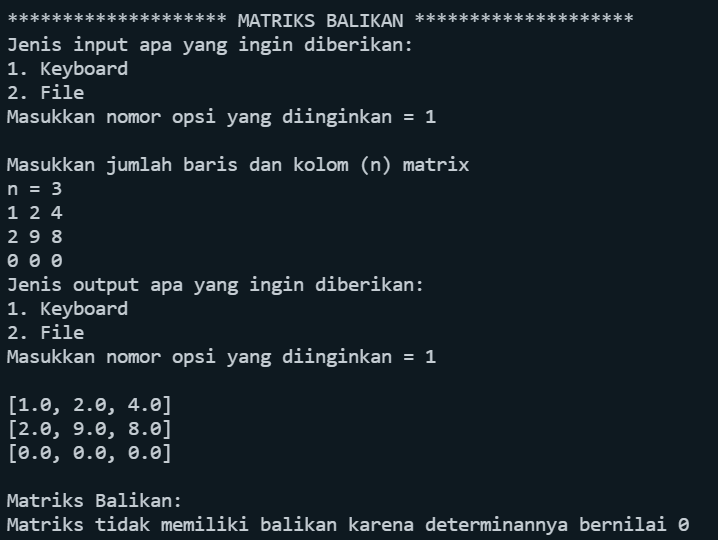
***Eksperimen 2 – Input Keyboard dan Output File – Metode Reduksi Baris***



*Output file*: Invers93.txt



***Eksperimen 3 – Determinan = 0***



1. **Interpolasi Polinom**

***Eksperimen 1***

Text

Description automatically generated

Output *file* yang diberikan:

Graphical user interface, text

Description automatically generated

***Eksperimen 2***

***Text

Description automatically generated***

*File* data yang diberikan:

Text

Description automatically generated with medium confidence

1. **Regresi Linier Berganda**

***Eksperimen 1***

***Text

Description automatically generated***

*File* data yang diberikan (sumber: <https://www.youtube.com/watch?v=4oSfPMec0BY>):

Calendar

Description automatically generated with medium confidence

***Eksperimen 2***

Text

Description automatically generated

*File* data yang diberikan (sumber: <https://www.youtube.com/watch?v=aQexxrQBS00>):

Calendar

Description automatically generated

***Eksperimen 3***

Text

Description automatically generated

Sumber data: <https://www.youtube.com/watch?v=mtm4WMkK27I>

Output *file* yang diberikan:

Graphical user interface, text

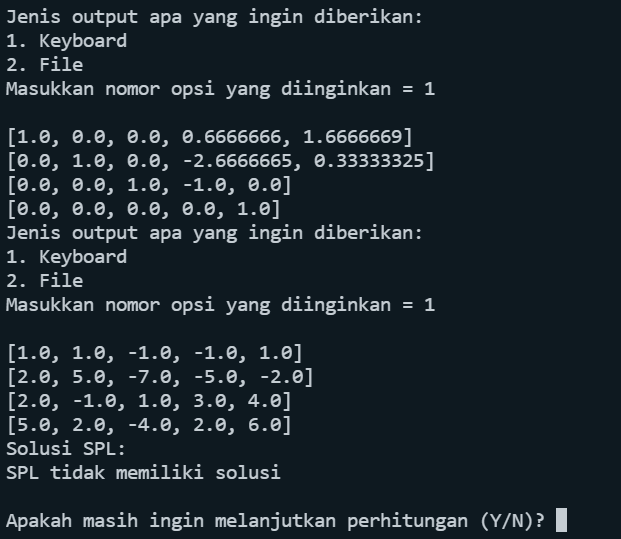
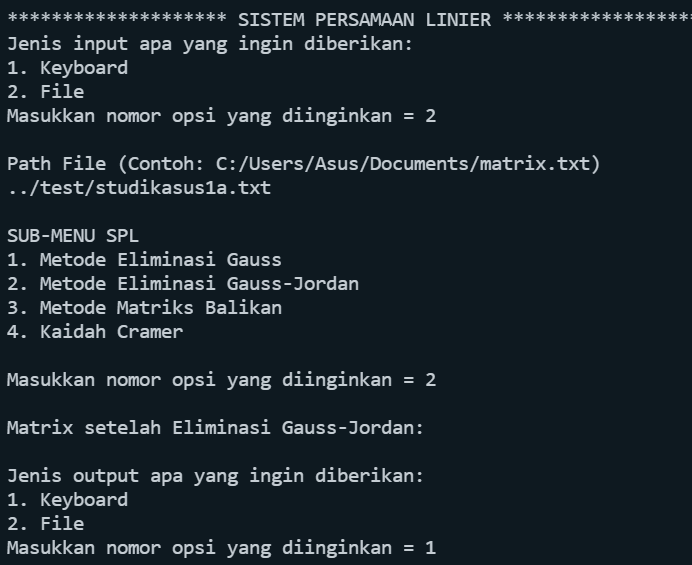
Description automatically generated

1. **Studi Kasus**
2. Temukan solusi SPL ,berikut:

,

Matriks Augmented

Solusi:

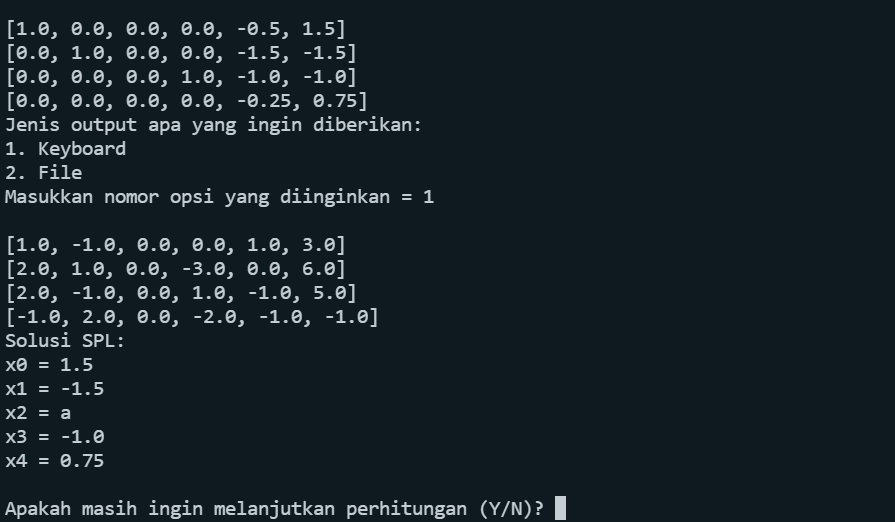
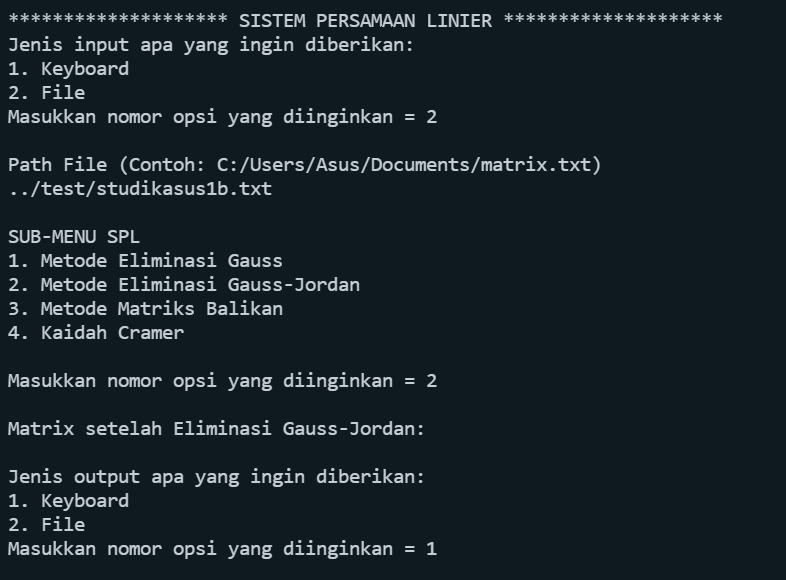




,

Matriks Augmented

Solusi:

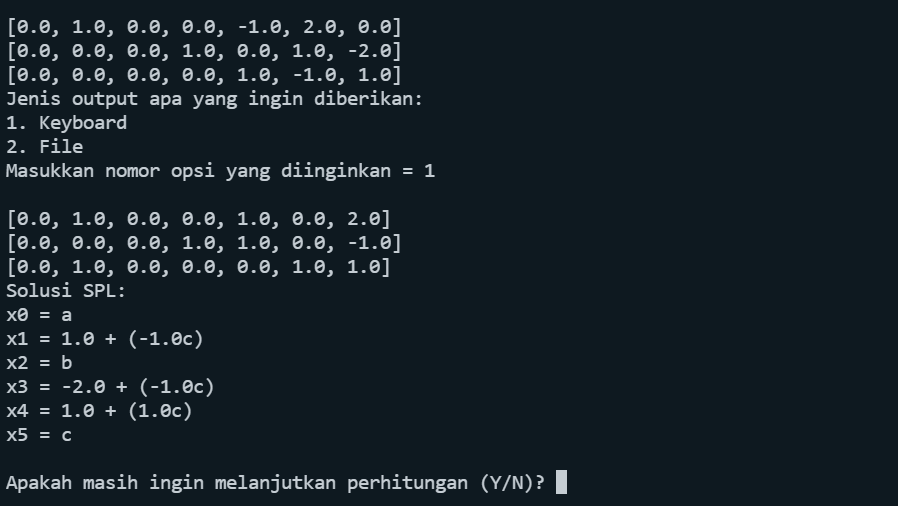
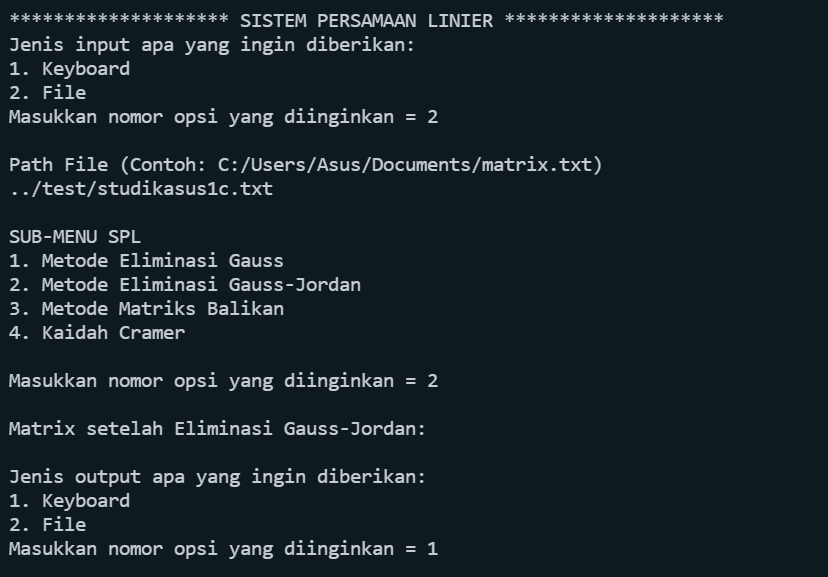




,

Matriks Augmented

Solusi:



* 1. *H adalah matriks Hilbert. Cobakan untuk dan .*

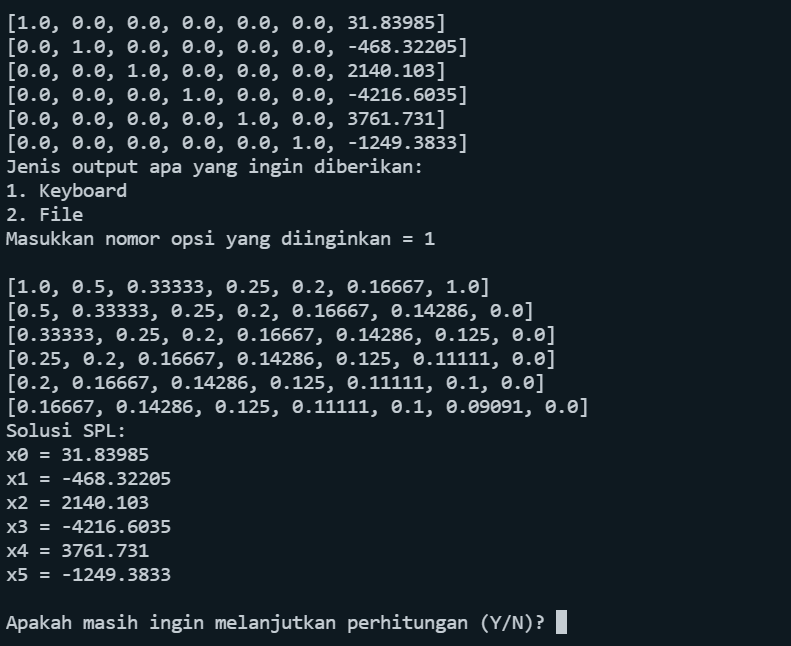
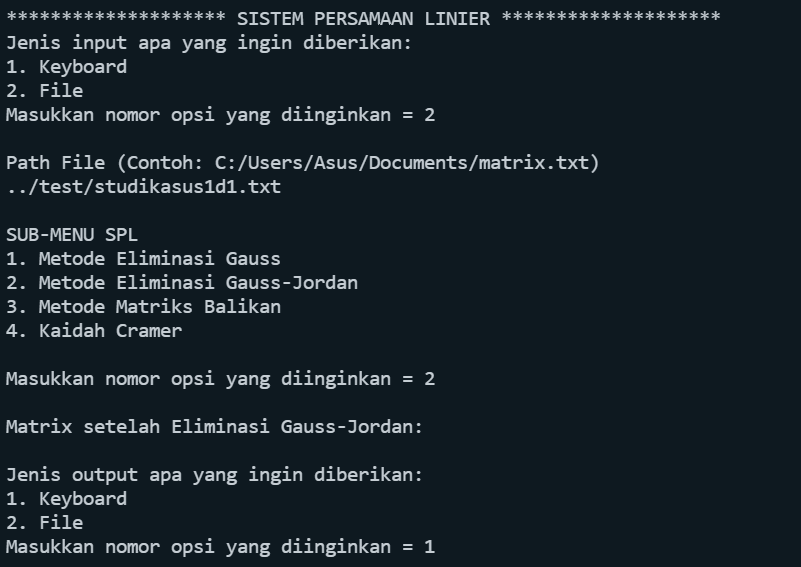
,

* Untuk ,

,

Matriks Augmented

Solusi:

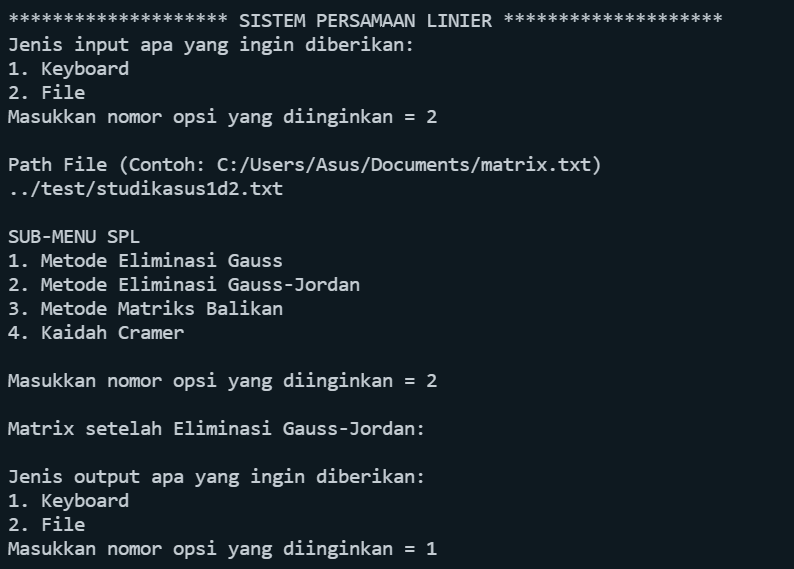


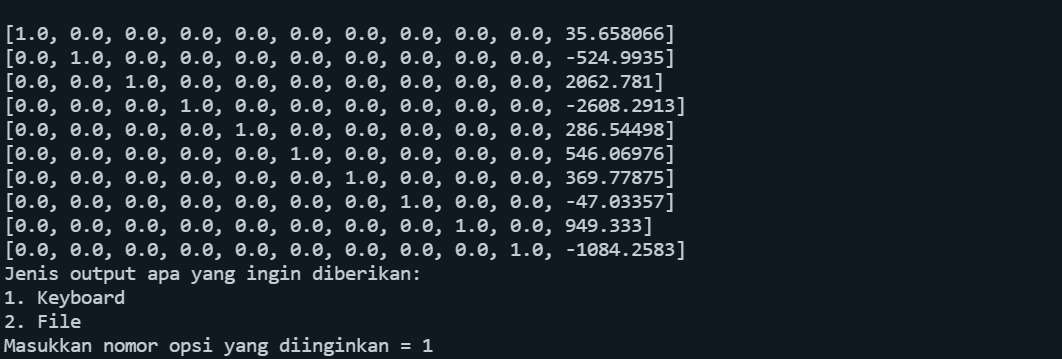
* Untuk ,

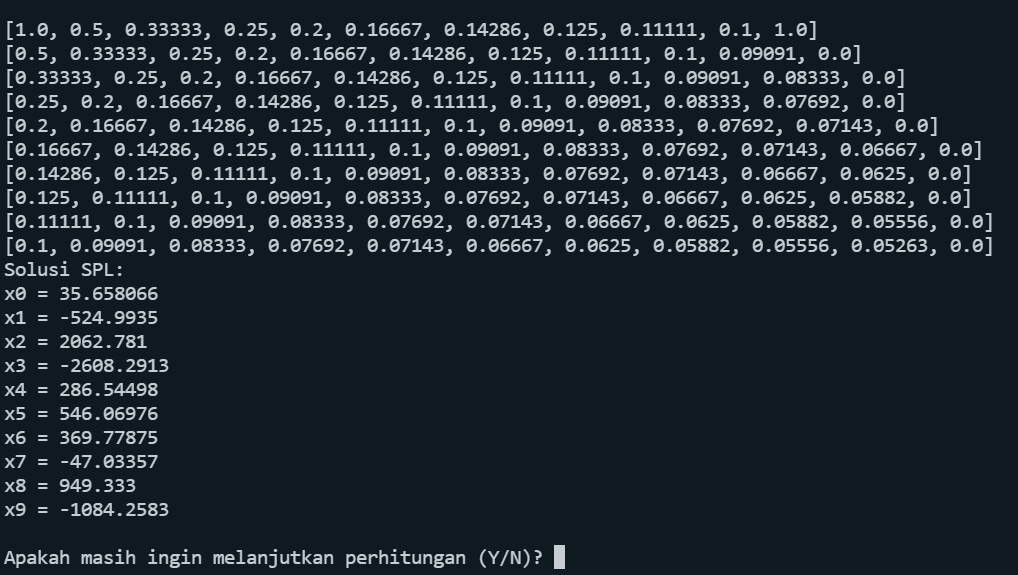
,

Matriks Augmented

Solusi:

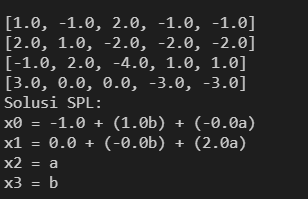






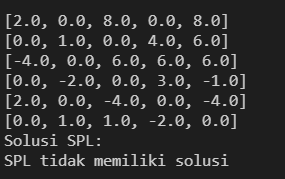
1. SPL berbentuk matriks *augmented*

Solusi:





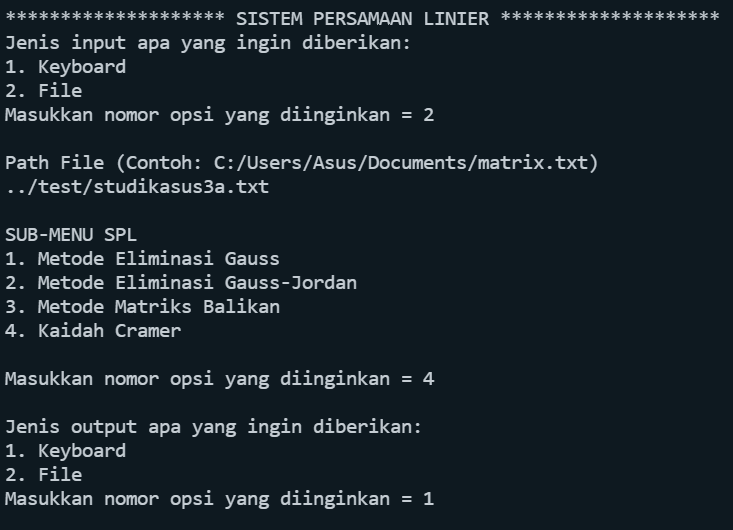
Solusi:

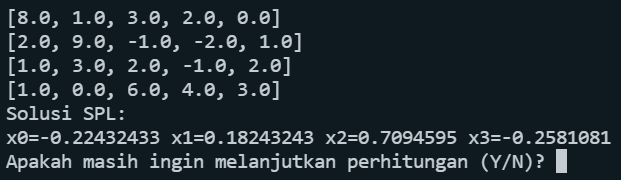


1. SPL berbentuk

Matriks Augmented

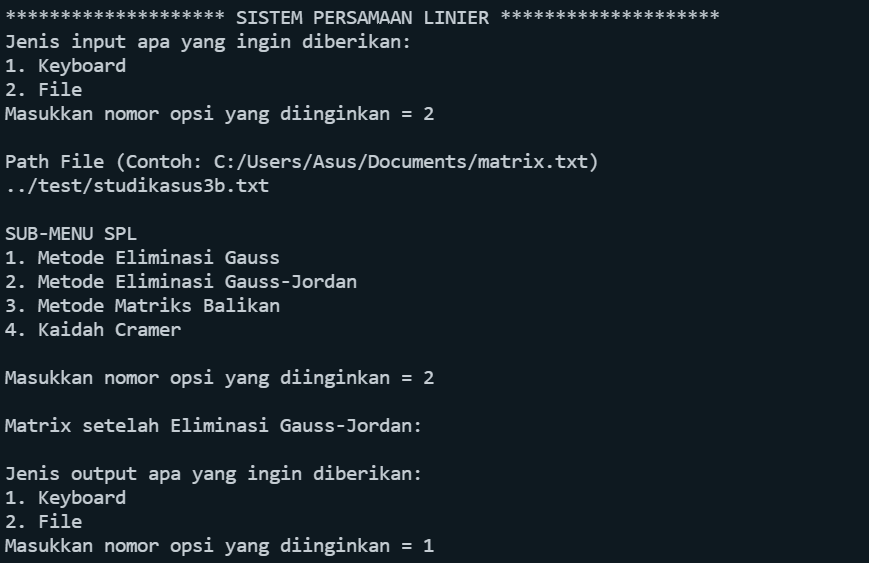
Solusi:

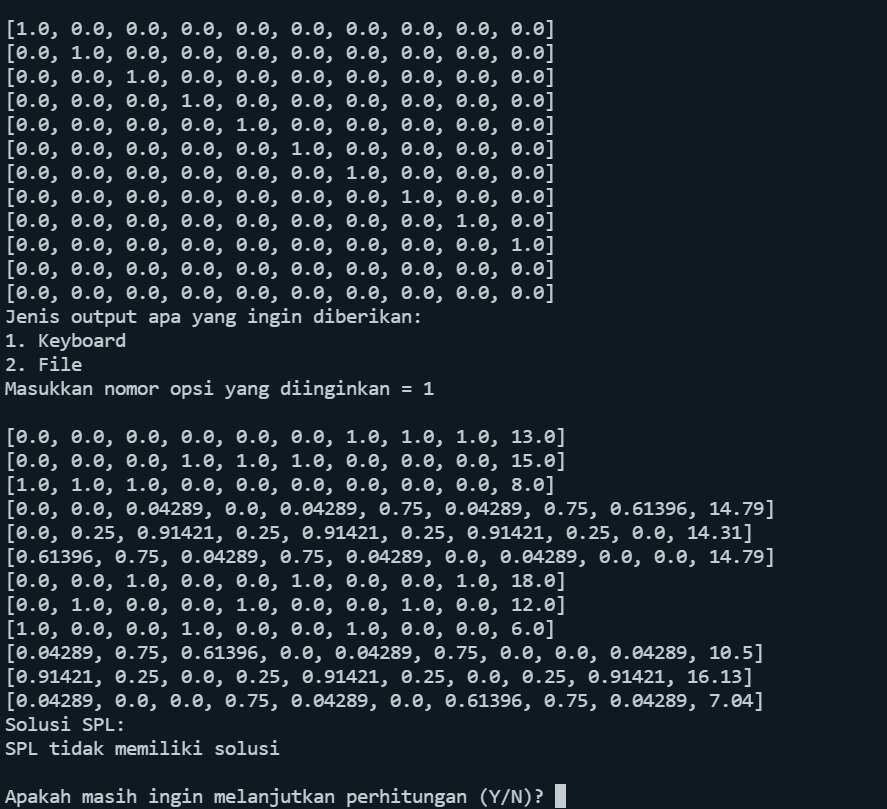




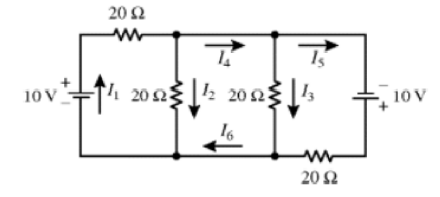
Matriks Augmented

Solusi:

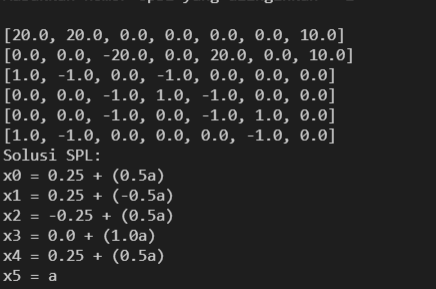




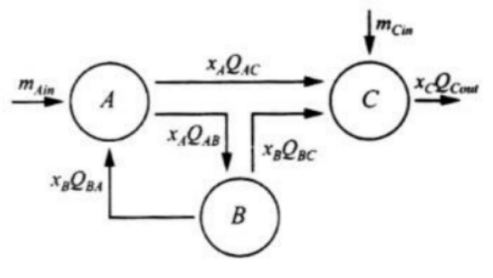
1. Tentukan arus yang mengalir pada rangkaian listrik di bawah ini:



Dengan KVL, dapat diperoleh persamaan-persamaan berikut dalam matrix augmented, solusi:



1. Lihatlah sistem reaktor pada gambar berikut



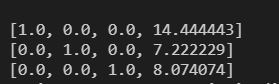
Dengan laju volume Q dalam dan input massa dalam . Konservasi massa pada tiap inti reaktor adalah sebagai berikut:

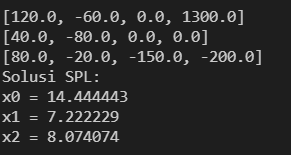
Tentukan solusi dengan menggunakan parameter berikut: dan dan dan .

Solusi:

Setelah dilakukan substitusi, persamaannya menjadi:

Matriks Augmented





1. **Studi Kasus Interpolasi**
   1. Gunakan tabel di bawah ini untuk mencari polinom interpolasi dari pasangan titik-titik yang terdapat dalam tabel. Program menerima masukan nilai yang akan dicari nilai fungsi .



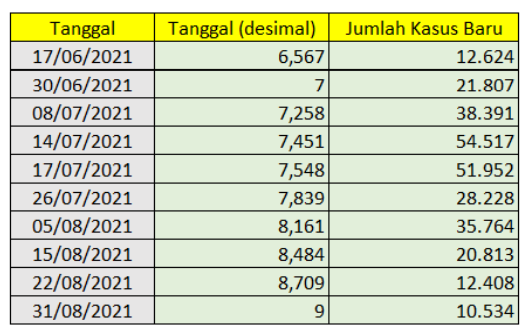
Lakukan pengujian pada nilai-nilai default berikut:

Solusi:

Solusi persamaan interpolasi polinom yang didapat adalah

Sehingga didapatkan

* 1. Jumlah kasus positif baru Covid-19 di Indonesia semakin fluktuatif dari hari ke hari. Di bawah ini diperlihatkan jumlah kasus baru Covid-19 di Indonesia mulai dari tanggal 17 Juni 2021 hingga 31 Agustus 2021:



Tanggal (desimal) adalah tanggal yang sudah diolah ke dalam bentuk desimal 3 angka di belakang koma dengan memanfaatkan perhitungan sebagai berikut:

|  |
| --- |
| tanggal(desimal) = bulan + (tanggal / jumlah hari pada bulan tersebut) |

Sebagai **contoh**, untuk tanggal 17/06/2021 (dibaca: 17 Juni 2021) diperoleh tanggal(desimal) sebagai berikut:

Tanggal(desimal) = 6 + (17/30) = 6,567

Gunakanlah data di atas dengan memanfaatkan **polinom interpolasi** untuk melakukan prediksi jumlah kasus baru Covid-19 pada tanggal-tanggal berikut:

1. 16/07/2021
2. 10/08/2021
3. 05/09/2021
4. beserta masukan user lainnya berupa tanggal (desimal) yang sudah diolah dengan asumsi prediksi selalu dilakukan untuk tahun 2021.

Solusi:

Dengan menggunakan semua data di atas dan dilakukan interpolasi polinom pada program yang telah dibuat, didapatkan persamaan

Namun, apabila disubstitusikan dengan persoalan tanggal-tanggal di atas, didapatkan angka negatif

Text

Description automatically generated

Dari sini, kami berasumsi bahwa interpolasi polinom pada program kami memiliki limit, apalagi jika jumlah data/titik yang dimasukkan cukup banyak. Sehingga untuk persoalan-persoalan tersebut, akan diambil 5 data yang paling dekat dengan nilai yang dicari.

1. Tanggal (desimal) = 7 + (16/31) = 7.516

Range data yang diambil: 30/06/2021 sampai 26/07/2021

Persamaan yang didapat adalah

Sehingga perkiraan jumlah covid pada tanggal 16/07/2021 adalah 64000

Text

Description automatically generated

1. Tanggal (desimal) = 8 + (10/31) = 8.323

Range data yang diambil: 26/07/2021 sampai 30/08/2021

Persamaan yang didapat adalah

Sehingga perkiraan jumlah covid pada tanggal 10/08/2021 adalah 28960

Text

Description automatically generated

1. Tanggal (desimal) = 9 + (5/30) = 9.167

Range data yang diambil: 26/07/2021 sampai 30/08/2021

Persamaan yang didapat adalah

Sehingga perkiraan jumlah covid pada tanggal 5/09/2021 adalah 28960

Text

Description automatically generated

1. Tanggal 07/08/2021

Desimal = 8 + (7/31) = 8.226

Range data yang diambil: 26/07/2021 sampai 30/08/2021

Persamaan yang didapat adalah

Sehingga perkiraan jumlah covid pada tanggal 07/08/2021 adalah 33536

Text

Description automatically generated

* 1. Sederhanakan fungsi

dengan polinom interpolasi derajat di dalam selang . Sebagai contoh, jika , maka titik-titik yang diambil di dalam selang berjarak .

Solusi:

Dengan mengambil , didapat titik-titik:

(0.4, 0.41888), (0.8, 0.50715), (1.2, 0.56092), (1.6, 0.58368), (2, 0.57665)

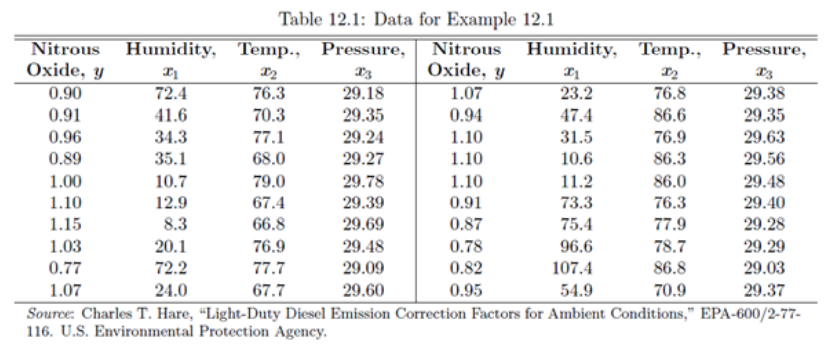
Dengan interpolasi polinom derajat 5 didapatkan persamaan

Text

Description automatically generated

1. **Studi Kasus Regresi Linier Berganda**

Diberikan sekumpulan data sesuai pada tabel berikut ini.



Gunakan *Normal Estimation Equation for Multiple Linear Regression* untuk mendapatkan regresi linear berganda dari data pada tabel di atas, kemudian estimasi nilai Nitrous Oxide apabila Humidity bernilai , temperatur , dan tekanan udara sebesar .

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

Solusi:

Berdasarkan program yang telah dibuat, solusi persamaan linear dari persoalan regresi linear adalah:

Dari persamaan linear tersebut, didapatkan estimasi nilai Nitrous Oxide dengan mensubstitusikan nilai (humidity ), (temperatur ), dan (tekanan udara ), yaitu sebesar 0.9384268

Text

Description automatically generated

# **BAB 5 KESIMPULAN DAN SARAN**

Dari pengerjaan Tugas Besar 1 Aljabar Linear dan Geometri ini, telah berhasil dibuat suatu program yang dapat menyelesaikan permasalahan sistem persamaan linear, determinan matriks, matriks balikan, interpolasi polinom, dan regresi lineaar berganda dengan menggunakan metode eliminasi Gauss, Gauss Jordan, matriks inverse, dan kaidah Cramer.

Akan tetapi, masih diperlukan pengembangan lebih lanjut lagi agar program ini bisa dapat digunakan dengan efektif pada semua kasus masalah. Program ini belum bisa menyelesaikan beberapa kasus, terutama pada interpolasi polinom berderajat lebih dari 10 dengan integer yang terlalu besar, seperti pada studi kasus 6 bagian b. Atau mungkin, apabila bertemu dengan kasus menaksir sebuah nilai yang memiliki jumlah data yang cukup banyak (lebih dari 10) akan lebih baik dan efektif menggunakan metode regresi linear, dibandingkan dengan interpolasi polinom. Selain itu, secara keseluruhan, untuk metode Gauss, Gaus Jordan, determinan, dan kaidah Cramer sudah bisa berjalan dengan baik pada semua test case yang telah dicoba.

# **DAFTAR REFERENSI**

Anton, Howard dan Chris Rorres. 2014. *Elementary Linear Algebra 11th Edition*. Canada: Anton Textbooks, Inc.

*Determinan*. <https://id.wikipedia.org/wiki/Determinan> (dakses tanggal 27 September 2021)