TUGAS BESAR 1 IF 2123 ALJABAR LINIER DAN GEOMETRI

SISTEM PERSAMAAN LINIER, DETERMINAN, DAN APLIKASINYA

LAPORAN

Disusun untuk memenuhi tugas Aljabar Linier dan Aljabar Geometri

DISUSUN OLEH:

NAUFAL ALEXANDER SURYASUMIRAT 13519135

I GEDE GOVINDABHAKTA 13519139

STEFANUS JEREMY ASLAN 13519175

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2020

BAB I

DESKRIPSI MASALAH

I. Abstraksi

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

$$a_{11}\,x_1 + a_{12}\,x_2 + ... + a_{1n}\,x_n = b_1$$
$$a_{21}\,x_1 + a_{22}\,x_2 + ... + a_{2n}\,x_n = b_2$$

$$\vdots \quad \vdots$$
$$\vdots \quad \vdots$$

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

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

Sebuah matriks M berukuran $n \times n$

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

determinannya adalah

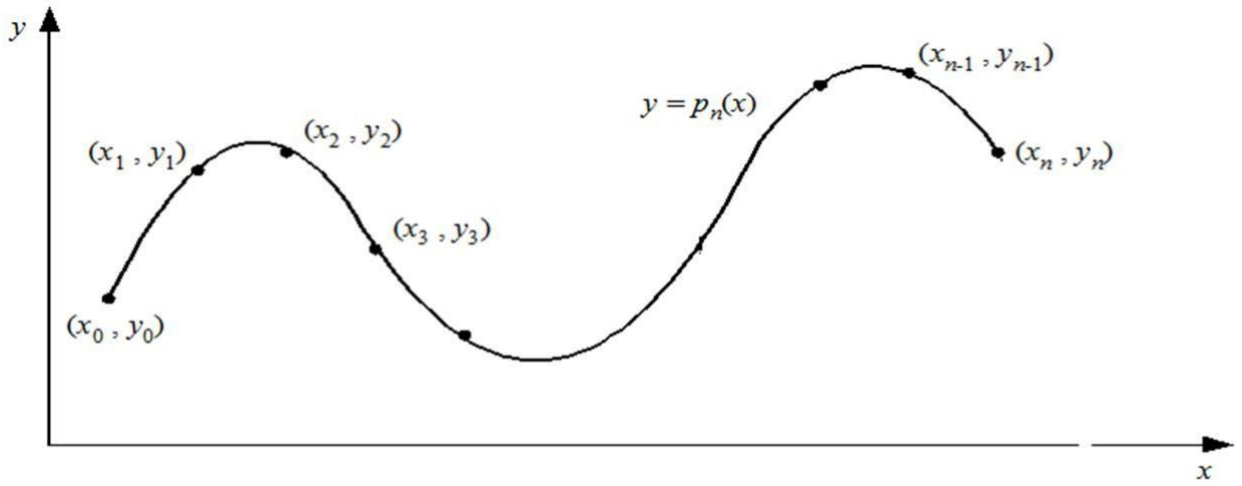
$$\det(M) = \begin{vmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{vmatrix}$$

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

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

II. Interpolasi Polinom

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



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

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

$$
\begin{aligned}
a_0 \quad & + a_1x_0 + a_2x_{02} + \ldots + a_n x_{0n} = y_0 \\
a_0 \quad & + a_1x_1 + a_2x_{12} + \ldots + a_n x_{1n} = y_1 \\
& \ldots \qquad\qquad \ldots \\
a_0 + a_1x_n + a_2x_n^2 & + \ldots + a_n x_n^n = y_n
\end{aligned}
$$

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

$$a_0 + 8.0a_1 + 64.00a_2 = 2.0794$$
$$a_0 + 9.0a_1 + 81.00a_2 = 2.1972$$
$$a_0 + 9.5a_1 + 90.25a_2 = 2.2513$$

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

III. Regresi Linier Berganda

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

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

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

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

Sistem persamaan linier tersebut diselesaikan dengan menggunakan metode eliminasi Gauss.

SPESIFIKASI TUGAS

Buatlah program dalam **Bahasa Java** untuk

1. Menghitung solusi SPL dengan metode eliminasi Gauss, metode Eliminasi Gauss-Jordan, metode matriks balikan, dan kaidah Cramer (kaidah Cramer khusus untuk SPL dengan *n* peubah dan *n* persamaan).
2. Menyelesaikan persoalan interpolasi dan regresi linier.
3. Menghitung matriks balikan
4. Menghitung determinan matriks dengan berbagai metode (reduksi baris dan ekspansi kofaktor).

Spesifikasi program adalah sebagai berikut:

1. Program dapat menerima masukan (input) baik dari *keyboard* maupun membaca masukan dari file text. Untuk SPL, masukan dari *keyboard* adalah *m*, *n*, koefisien a_{ij} , dan b_i. Masukan dari *file* berbentuk matriks *augmented* tanpa tanda kurung, setiap elemen matriks dipisah oleh spasi. Misalnya,

```
 3 4.5 2.8 10 12
-3 7 8.3 11 -4
0.5 -10 -9 12 0
```

2. Untuk persoalan menghitung determinan dan matriks balikan, masukan dari *keyboard* adalah *n* dan koefisien a_{ij}. Masukan dari *file* berbentuk matriks, setiap elemen matriks dipisah oleh spasi. Misalnya,

```
 3 4.5 2.8 10
-3 7 8.3 11
 0.5 -10 -9 12
```

3. Untuk persoalan interpolasi, masukannya jika dari *keyboard* adalah *n*, (x_0, y_0), (x_1, y_1), ..., (x_n, y_n), dan nilai *x* yang akan ditaksir nilai fungsinya. Jika masukannya dari file, maka titik-titik dinyatakan pada setiap baris tanpa koma dan tanda kurung. Misalnya jika titik-titik datanya adalah (8.0, 2.0794), (9.0,
   2.1972), dan (9.5, 2.2513), maka di dalam file text ditulis sebagai berikut:

```
8.0 2.0794
9.0 2.1972
9.5 2.2513
```

4. Untuk persoalan regresi, masukannya jika dari *keyboard* adalah *n* (jumlah peubah *x*), semua nilai-nilai x_{1i}, x_{2i}, ..., x_{ni}, nilai y_i, dan nilai-nilai x_k yang akan ditaksir nilai fungsinya. Jika masukannya dari file, maka titik-titik dinyatakan pada setiap baris tanpa koma dan tanda kurung.

5. Untup persoalan SPL, luaran (*output*) program adalah solusi SPL. Jika solusinya tunggal, tuliskan nilainya. Jika solusinya tidak ada, tuliskan solusi tidak ada, jika solusinya banyak, maka tuliskan solusinya dalam bentuk parametrik (misalnya $x_4 = -2$, $x_3 = 2s - t$, $x_2 = s$, dan $x_1 = t$.)

4. Untuk persoalan determinan dan matriks balikan, maka luarannya sesuai dengan persoalan masing-masing

5. Untuk persoalan polinom interpolasi dan regresi, luarannya adalah persamaan polinom/regresi dan taksiran nilai fungsi pada *x* yang diberikan.

6. Luaran program harus dapat ditampilkan **pada layar komputer dan dapat disimpan ke dalam file**.

7. Bahasa program yang digunakan adalah Java.

8. Program **tidak harus** berbasis GUI, cukup text-based saja, namun boleh menggunakan GUI (memakai kakas *Eclipse* misalnya).
9. Program dapat dibuat dengan pilihan menu. Urutan menu dan isinya dipersilakan dirancang masing-masing. Misalnya, menu:

```
MENU
1. Sistem Persamaaan Linier
2. Determinan
3. Matriks balikan
4. Interpolasi Polinom
5. Regresi linier berganda
6. Keluar
```

Untuk pilihan menu nomor 1 ada sub-menu lagi yaitu pilihan metode:

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
```

Begitu juga untuk pilihan menu nomor 2 dan 3.

BAB II

TEORI SINGKAT

I. Metode Eliminasi Gauss

Metode Eliminasi Gauss adalah algoritma yang digunakan untuk menyelesaikan Sistem Persamaan Linear (SPL). Terdapat tiga jenis operasi yang dapat dilakukan dalam metode ini:

1. Mengganti urutan dua baris
2. Mengalikan baris dengan angka yang bukan nol
3. Menambah suatu baris dengan baris lainnya

Dengan cara ini, matriks dapat diubah menjadi matriks segitiga atas dengan leading one yang terdapat dalam tiap barisnya. Leading one merupakan elemen *non-zero* pertama dalam tiap baris yang dalam Metode Eliminasi Gauss tiap leading one dalam tiap baris tidak memiliki elemen selain 0 di bawah barisnya dan dalam kolom yang sama. Eliminasi gauss mengoperasikan nilai-nilai di dalam matriks menjadi matriks yang lebih sederhana lagi menjadi bentuk *Row Echelon Form*.

Ciri-ciri Metode Eliminasi Gauss adalah:

1. Jika suatu baris tidak semua nol, maka bilangan pertama yang tidak nol adalah 1 (Leading One / 1 Utama).
2. Baris dengan semua elemen nol (jika ada) terletak di paling bawah.
3. Leading One baris berikutnya berada di kanan Leading One di atasnya.
4. Di bawah Leading One harus nol.

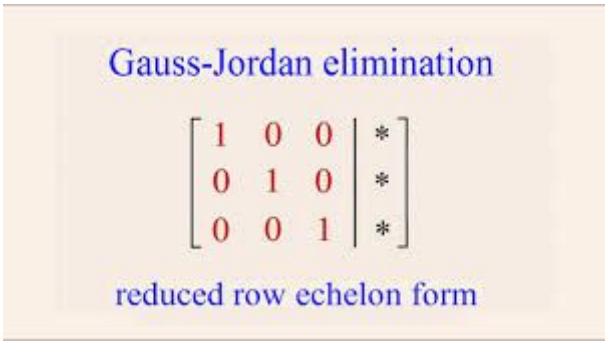
Contoh hasil penggunaan Metode Eliminasi Gauss:

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

II. Metode Eliminasi Gauss-Jordan

Eliminasi Gauss-Jordan adalah algoritma versi dari eliminasi Gauss. Pada metode eliminasi Gauss-Jordan dibuat nol elemen-elemen di bawah maupun atas dari Leading One tiap baris. Hasilnya adalah matriks *Reduced Row Echelon Form*. Metode ini dapat digunakan untuk mencari *Invers*/Balikan dari suatu matriks.

Contoh bentuk Matriks setelah digunakan Metode Eliminasi Gauss-Jordan:



Metode Ini juga dapat digunakan sebagai salah satu metode penyelesaian persamaan linear dengan menggunakan matriks. Caranya dengan mengubah persamaan linear tersebut ke dalam matriks teraugmentasi dan mengoperasikannya. Setelah menjadi matriks *Reduced Row Echelon Form*, maka langsung dapat ditentukan nilai dari variabel variabelnya tanpa substitusi balik.

III. Determinan

Determinan adalah nilai skalar yang dapat dihitung dari elemen matriks berbentuk persegi (*n* x *n*) dan merepresentasikan sifat tertentu dari transformasi linier yang dijelaskan oleh matriks. Determinan dari A dilambangkan dengan det(A), det A, atau |A|. Secara geometris, ini dapat dilihat sebagai faktor skala volume dari transformasi linier yang dijelaskan oleh matriks.

Dalam kasus matriks 2 x 2, determinan dapat didefinisikan sebagai:

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Untuk matriks 3 x 3, determinannya adalah:

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$
$$= aei + bfg + cdh - ceg - bdi - afh.$$

Setiap determinan dari matriks 2 x 2 dalam persamaan ini disebut minor dari matriks A. Prosedur ini dapat diperpanjang untuk memberikan definisi rekursif untuk determinan matriks *n* x *n*, yang dikenal sebagai ekspansi Laplace atau juga bisa disebut Ekspansi Kofaktor.

IV. Matriks Balikan

Suatu matriks dapat dibalik jika dan hanya jika matriks tersebut adalah matriks persegi (matriks yang berukuran n x n) dan matriks tersebut *non-singular* (determinan ≠ 0). Tidak semua matriks memiliki *invers*. *Invers* matriks dapat didefinisikan sebagai berikut.

Jika A adalah suatu matriks kuadrat, dan jika kita dapat mencari matriks B sehingga AB = BA = I, maka A dikatakan dapat dibalik (*invertible*) dan B dinamakan invers dari A. I merupakan matriks Identitas (matriks nxn dengan diagonal utamanya bernilai 1 dan elemen lainnya bernilai nol).

Adjoin dari matriks A dapat digunakan untuk mencari *invers*/balikan dari A sebagai berikut: Jika A adalah matriks n x n, maka

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

V. Matriks Kofaktor

Kofaktor adalah hasil perkalian minor dengan suatu angka yang besarnya menuruti suatu aturan yaitu $(-1)^{i+j}$ dimana i adalah baris dan j adalah kolom. Kofaktor suatu elemen baris ke-*i* dan kolom ke-*j* dari matriks A dilambangkan dengan C*ij*.

$$Cij = (-1)^{i+j} Mij$$

Jumlah kofaktor suatu matriks mengikuti jumlah elemen matriks tersebut.

Lambang M merepresentasikan minor dari matriks tersebut, minor adalah determinan matriks bagian dari matriks *A* yang diperoleh dengan cara menghilangkan elemen – elemennya pada baris ke-*i* dan elemen elemen pada kolom ke-*j*. Dengan demikian untuk matriks 1 x 1, kita tidak bisa mendapatkan minornya. Minor kita bisa dapatkan pada matriks persegi 2 x 2, 3 x 3, dan seterusnya.

Matriks kofaktor merupakan matriks yang terdiri dari kofaktor-kofaktor matriks itu sendiri. Jadi, misalkan terdapat suatu matriks katakanlah matriks A, maka matriks kofaktor A merupakan matriks yang terdiri dari kofaktor-kofaktor dari matriks A. Susunan elemen matriks kofaktor juga mengikuti susunan (letak) kofaktor-kofaktornya.

$$\text{Contoh: } A = \begin{bmatrix} 3 & 1 & -4 \\ 2 & 5 & 6 \\ 1 & 4 & 8 \end{bmatrix}$$

$$M_{11} = \begin{vmatrix} 5 & 6 \\ 4 & 8 \end{vmatrix} = 16,\ C_{11} = (-1)^{1+1}.M_{11} = 16$$

$$M_{32} = \begin{vmatrix} 3 & -4 \\ 2 & 6 \end{vmatrix} = 26,\ C_{32} = (-1)^{3+2}.M_{32} = -26$$

VI. Matriks Adjoin

Adjoin matriks merupakan *transpose* dari matriks kofaktor. Adjoin sering disingkat dengan Adj. Misalkan matriks A, maka adjoin A ditulis Adj (A). *Transpose* sendiri maksudnya adalah pertukaran elemen pada baris menjadi kolom atau kolom menjadi baris. Adjoin matriks digunakan dalam menentukan *invers* matriks atau matriks balikan.

VII. Kaidah Cramer

Kaidah Cramer adalah rumus yang digunakan untuk menyelesaikan Sistem Persamaan Linear (SPL). Metode ini menggunakan determinan suatu matriks dan matriks lain yang diperoleh dengan mengganti salah satu kolom dengan vektor yang terdiri dari angka di sebelah kanan persamaannya.

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

$$x = \frac{\begin{vmatrix} d_1 & b_1 & c_1 \\ d_2 & b_2 & c_2 \\ d_3 & b_3 & c_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}, \quad y = \frac{\begin{vmatrix} a_1 & d_1 & c_1 \\ a_2 & d_2 & c_2 \\ a_3 & d_3 & c_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}, \text{ and } z = \frac{\begin{vmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \\ a_3 & b_3 & d_3 \end{vmatrix}}{\begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}}.$$

VIII. Interpolasi Polinom

Dalam analisis numerik, interpolasi polinomial adalah interpolasi dari kumpulan data tertentu oleh polinomial dengan derajat serendah mungkin yang melewati titik-titik kumpulan data.

Untuk himpunan n + 1 titik data (x_i, y_i) yang tidak ada dua x_i yang sama, polinomial $p: \mathbb{R} \to \mathbb{R}$ dikatakan menginterpolasi data jika $p(x_j) = y_j$ untuk setiap $j \in \{0, 1, \ldots, n\}$.

Misal interpolasi polinomial berbentuk:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0. \qquad (1)$$

Pernyataan bahwa p menginterpolasi titik data berarti:

$$p(x_i) = y_i \qquad \text{for all } i \in \{0, 1, \ldots, n\}.$$

Jika kita mengganti persamaan (1) di sini, kita mendapatkan sistem persamaan linier di koefisien ak. Sistem dalam bentuk matriks-vektor membaca perkalian berikut:

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

Kemudian dapat dibuat matriks augmentednya untuk mencari a_0 sampai dengan a_n untuk mendapatkan persamaan polinomial berderajat n ($P_n(x)$) kemudian persamaan tersebut dapat digunakan untuk memprediksi titik-titik yang berada dalam selang titik-titik yang digunakan tersebut.

$$J = \left[\begin{array}{ccccc|c} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} & y_1 \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} & y_2 \\ 1 & x_3 & x_3^2 & \dots & x_3^{n-1} & v_3 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} & y_n \end{array}\right]$$

IX. Regresi Linier Berganda

Pada Regresi Linear Berganda, input yang digunakan lebih dari satu dimensi maka variabel bebas yang terlibat tidak hanya satu. Pada dasarnya regresi linear berganda adalah model prediksi atau peramalan dengan menggunakan data berskala interval atau rasio serta terdapat lebih dari satu *predictor*.

Model regresi linear berganda dilukiskan dengan persamaan sebagai berikut:

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

Untuk mendapatkan nilai dari setiap β_i dapat digunakan Normal Estimation Equation for Multiple Linear Regression seperti ini:

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

Atau dapat juga dimodelkan sebagai berikut:

$$h_w(x) = w_0 + \sum_{i=0}^{m} w_i x_i$$

Dimana *w* juga merupakan nilai yang akan dicari sedemikian sehingga nilai *w* menjadi optimal dan *x* merupakan variabel bebas atau input. Proses pencarian nilai *w* juga masih dapat dilakukan dengan menggunakan cara yang sama dengan regresi linear univariate, yaitu dengan menggunakan pendekatan least square, *maximum likehood*, atau algoritma gradient descent. Pada dasarnya, pencarian nilai *w* dilakukan hingga nilai error yang didapatkan dari fungsi error merupakan nilai yang paling minimal.

BAB III

IMPLEMENTASI PROGRAM

Garis Besar Kerja Program

Program pertama-tama menerima input dari user untuk memilih metode/apa yang akan dilakukan oleh program.

Contoh: Jika user menginput 1 maka akan melakukan proses SPL dengan user input ataupun dengan file. Setelah memilih metode SPL tersebut akan dipilih metode inputnya (dari file atau dari keyboard), kemudian akan dihitung dan ditampilkan. Setelah ditampilkan akan ada prompt untuk pilihan write file atau tidak. Jika dipilih write file maka akan menuliskan file ke dalam file txt baru dengan nama yang diinginkan oleh user.

Kerja program untuk cara-cara lainnya secara garis besar sama dengan yang sudah dicontohkan.

FileHandler

class **FileHandler**

Attribute dalam **FileHandler**:

integer fileSuffix

BDMatrix data

String outputString

String fileAddress

Method dalam **FileHandler**:

Konstruktor

FileHandler(String fileAddress)

Konstruktor dari FileHandler

FileHandler(String fileAddress, String output)

Konstruktor set String output dan FileHandler

Getter and Setters

void BDMatrix getData()

Mengembalikan data dalam bentuk BDMatrix

void readFile()

Membaca file

void writeFile(String fileAddress)

Menulis/save hasil ke file

void determineSize(String fileAddress)

Menentukan ukuran matrix yang dibaca

MainApp

class **MainApp**

Merupakan program utama yang menggunakan class-class lainnya beserta method-methodnya sekaligus menampilkan teks untuk digunakan oleh user untuk menginput nilai-nilai dan menggunakan class-class yang tersedia.

Contoh penggunaan:

```
MENU PROGRAM
1. Sistem Persamaan Linier
2. Determinan
3. Matriks Balikan
4. Interpolasi Polinom
5. Regresi Linier Berganda
6. Keluar
4
1. Dari File
2. Dari Keyboard
2
6
1
2
3
4
5
6
7
8
9
10
11
12
13
14
P6(x) = 1.0 + 1.0x^1 + (0.0)x^2 + (0.0)x^3 + (0.0)x^4 + (0.0)x^5 + (0.0)x^6
2
3.0
Apakah akan di save ke file? 0 untuk Tidak, 1 untuk Ya
0
1. Dari File
2. Dari Keyboard
0
```

Matrix

class **BDMatrix**

Attribute dalam **BDMatrix**:

BigDecimal[][] element

integer rows

integer columns

BigDecimal zero (berisi 0 dalam BigDecimal)

MathContext mc (berisi metode rounding MathContext.DECIMAL32)

Method dalam **BDMatrix**:

Konstruktor

BDMatrix()

BDMatrix(int row, int column)

Membuat matriks dengan ukuran baris row dan kolom column yang berisi BigDecimal.ZERO

BDMatrix(int row, int column, BigDecimal initialValue)

Membuat matriks dengan ukuran baris row dan kolom column yang berisi BigDecimal initialValue

BDMatrix(int row, int column, BigDecimal[][] data)

Mengubah bentuk matriks BigDecimal menjadi BDMatrix

BDMatrix(int size)

Membuat BDMatrix dengan ukuran baris dan kolom size

Getters dan Setters

BigDecimal getElmt(int row, int column)

Mengembalikan nilai pada baris row dan kolom column

void setElmt(int row, int column, BigDecimal val)

Mengganti nilai elemen pada baris row dan kolom column menjadi val

BigDecimal[] getRow(int row)

Mengembalikan element-elemen pada row

void setRow(int row, BigDecimal[] data)

Me-set element[row] menjadi data (tiap elemennya)

integer getRows()

Mengembalikan ukuran baris BDMatrix tersebut

integer getColumns()

Mengembalikan ukuran kolom BDMatrix tersebut

void setMathContext(int val)

Menggantikan attribute mc tergantung val

BigDecimal getLeadingElmt(int row)

Mengembalikan leading one pada baris tersebut

int getLeadingIndex

Mengembalikan index leading one pada baris tersebut

Method Lainnya

void readUserMatrix()

Menerima input dari pengguna untuk menginisialisasi BDMatrix

void makeMatrix()

Untuk testing membuat Matrix

void bacaMatrix()

Untuk testing membaca Matrix

void printMatrix()

Untuk meng-output/memperlihatkan bentuk matriks

void printRow(int row)

Untuk meng-output/memperlihatkan elemen-elemen pada baris row

String convertToString()

Mengubah bentuk matriks dari BigDecimal ke string untuk diperlihatkan

void upperTri()

Mengubah bentuk matriks menjadi matriks segitiga atas

void echelon()

Mengubah bentuk matriks menjadi Row Echelon Form

void reducedEchelon()

Mengubah bentuk matriks menjadi Reduced Echelon Form

void addHorizontal(BDMatrix newData)

Menambahkan kolom di bagian kanan BDMatrix

void removeHorizontal(int left, int right)

Menghilangkan kolom pada BDMatrix

void replaceColumn(BDMatrix mtrx, int columns)

Menggantikan elemen pada kolom columns

BDMatrix transpose()

Mengembalikan bentuk BDMatrix yang elemen baris dan kolomnya telah ditukar

void orderRows()

Mengurutkan baris berdasarkan leading one

BDMatrix crossProductWith(BDMatrix operand)

Mengembalikan hasil perkalian cross product matrix

void addRows(int row1, int row2)

Menjumlahkan dua row pada BDMatrix

BigDecimal[][] dotMatrix(BDMatrix operand)

Mengembalikan hasil perkalian dot Matrix

void subtractRows(int row1, int row2)

Mengurangkan elemen-elemen pada row1 dengan row2

void multiplyRow(int row, BigDecimal C)

Mengalikan elemen-elemen pada baris row dengan BigDecimal C

void divideRow (int row, BigDecimal C)

Membagi elemen-elemen pada baris row dengan BigDecimal C

void switchRows(int row1, int row2)

Menukarkan elemen-elemen pada row1 dengan row2

int isAleadingB(int row1, int row2)

Mengembalikan 0 jika elemen-elemen pada row1 dan row2 sama, 1 jika row1 leads row2, -1 jika row2 leads row1

BigDecimal rowCrossSum(BigDecimal[] row1, BigDecimal[] row2, int length)

Melakukan operasi cross-sum baris pada matriks

SPL

class **BDSPL**

Attribute dalam **BDSPL**:

BDMatrix original

BDMatrix data

String[] solution (Menyimpan solusi dalam bentuk String)

BDMatrix solutionMatrix (Menyimpan solusi dalam bentuk BDMatrix)

boolean solutionFound (Menyimpan data apakah ditemukan solusi)

Method dalam **BDSPL**:

Konstruktor

BDSPL(BDMatrix input)

Konstruktor untuk BDSPL

void hitungGauss()

Menghitung solusi SPL dengan menggunakan metode Gauss

void hitungGaussJordan()

Menghitung solusi SPL dengan menggunakan metode Gauss-Jordan

void hitungKramer()

Menghitung solusi SPL dengan menggunakan metode Kramer

void hitungInvers()

Menghitung solusi SPL dengan menggunakan metode Matriks Balikan

void calcSPL(BDMatrix input)

Mengubah bentuk Matrix dengan awalnya substitusi balik dan menjadikannya bentuk Matrix solusi

Bentuk hasilnya adalah {C1 x1 x2 x3}, {C2 x1 x2 x3}, solusinya adalah
x1 = C1 + x1.x + x2.x + x3.x

void parseSolutionMatrix()

Mengubah bentuk Matrix Solusi menjadi bentuk Stringnya

String parseRow(BigDecimal[] row, int length)

Dipakai dalam parseSolutionMatrix untuk mempersingkat algoritma

Determinan

class **BDDeterminan**

Attribute dalam **BDDeterminan**:

BigDecimal attributeDeterminan (Menyimpan nilai determinan)

BDMatrix attributeMatriks (Menyimpan matriks yang akan dihitung determinannya)

integer N (Menyimpan ukuran Matriks yang akan dihitung determinannya)

Method dalam **BDDeterminan**:

void readData()

Bertugas untuk menerima input dari User yang berupa BDMatrix beserta ukurannya yang disimpan dalam attribute integer N

BigDecimal getAttributeDeterminanEK()

Bertugas untuk mengembalikan nilai determinan dengan metode Ekspansi Kofaktor setelah input dari keyboard

BigDecimal getAttributeDeterminanOBE()

Bertugas untuk mengembalikan nilai determinan dengan metode OBE setelah input dari keyboard

BigDecimal hitungDeterminanEK(BDMatrix Matriks, int ukuran)

Bertugas untuk menghitung Determinan Matriks dengan metode Ekspansi Kofaktor sekaligus mengembalikan nilainya, digunakan juga untuk menghitung Determinan dari file

BigDecimal hitungDeterminanOBE(BDMatrix data)

Bertugas untuk menghitung Determinan Matriks dengan metode OBE sekaligus mengembalikan nilainya digunakan juga untuk menghitung Determinan dari file

Interpolasi

class **Interpolasi**

Attribute dalam Interpolasi:

double[][] ArrayHasil (Menyimpan

BDMatrix ArrayHasilBD

String Persamaan (Menyimpan persamaan hasil dalam bentuk string)

double[][] ArrayTitik (Menyimpan titik-titik inputan maupun dari keyboard atau file ke dalam Matriks berdimensi [derajat+1][1])

integer derajat (Menyimpan derajat dari interpolasi polinomial)

Method dalam **Interpolasi**

Konstruktor

Interpolasi()

Interpolasi(BDMatrix Titik)

Mengubah dari inputan berbentuk BDMatrix ke double[][] dan menyimpannya dalam atribut

Lainnya

double getTitikInterpolasi()

Menerima input titik x dari keyboard yang kemudian diproses untuk mencari f(x) pada titik tersebut, f(x) sudah didapat dan disimpan dalam ArrayHasil, dan mengembalikan hasilnya.

BigDecimal getTitikInterpolasi(BigDecimal x)

Menerima input x (dari GUI) untuk kemudian diproses dan mengembalikan BigDecimal yang berupa titik y saat x dari persamaan yang telah ditemukan.

BigDecimal getTitikInterpolasiBD()

Menerima input x dari keyboard dan mengembalikan y pada persamaan dalam bentuk BigDecimal (digunakan jika diperlukan)

void setArrayHasil(double[][] Hasil) (Menentukan nilai attribute ArrayHasil)

getPersamaan()

Mengembalikan nilai attribute Persamaan dalam String

BDMatrix getArrayHasilBD()

Mengembalikan Attribute ArrayHasilBD jika akan digunakan

String convertingToString(double[][] hasilInterpolasi)

Mengkonversikan Persamaan yang didapat dari bentuk matriks satu dimensi ke bentuk String dan mengembalikannya

void readData()

Menerima input dari user berupa derajat persamaan beserta titik-titik yang akan diinterpolasi

makeMatriksTitik(int n)

Membuat bentuk matriksTitik dengan input n untuk digunakan dalam readData()

void interpolasi()

Menginterpolasikan dan menentukan nilai attribute Persamaan dan ArrayHasil

Inverse

class **Inverse**

Attribute dalam **Inverse**:

BDMatrix data

Method dalam **Inverse**:

Konstruktor

Inverse()

Inverse(BDMatrix input) (Menentukan nilai attribute data)

Lainnya

double[][] getInvers(double[][] matrixA)

Menggunakan sifat matrix [A | I] = [I | A^{-1}] untuk menghitung dan mengembalikan bentuk matrix menjadi balikannya dalam bentuk double[][]

BDMatrix getInverse()

Mengembalikan bentuk matrix menjadi bentuk balikannya

Regresi Linier

class **BDRegresiLinier**

Attribute dalam **BDRegresiLinier**:

BDMatrix data

BDMatrix B

BDMatrix X

BDMatrix Y

BDMatrix equationData (menyimpan hasil solusi dalam bentuk BDMatrix)

Method dalam **BDRegresiLinier**:

Konstruktor

BDRegresiLinier()

Konstruktor untuk BDRegresiLinier

BDRegresiLinier(BDMatrix inputData)

Konstruktor untuk BDRegresiLinier dengan input BDMatrix

Lainnya

BigDecimal assertY(BDMatrix inputData)

Memasukkan persamaan

BigDecmal readAssertY()

Membaca AssertY

void solve()

Memberikan solusi untuk persoalan regresi

createLeastSquareNormalEquations()

Menghasilkan persamaan dengan square normal terkecil

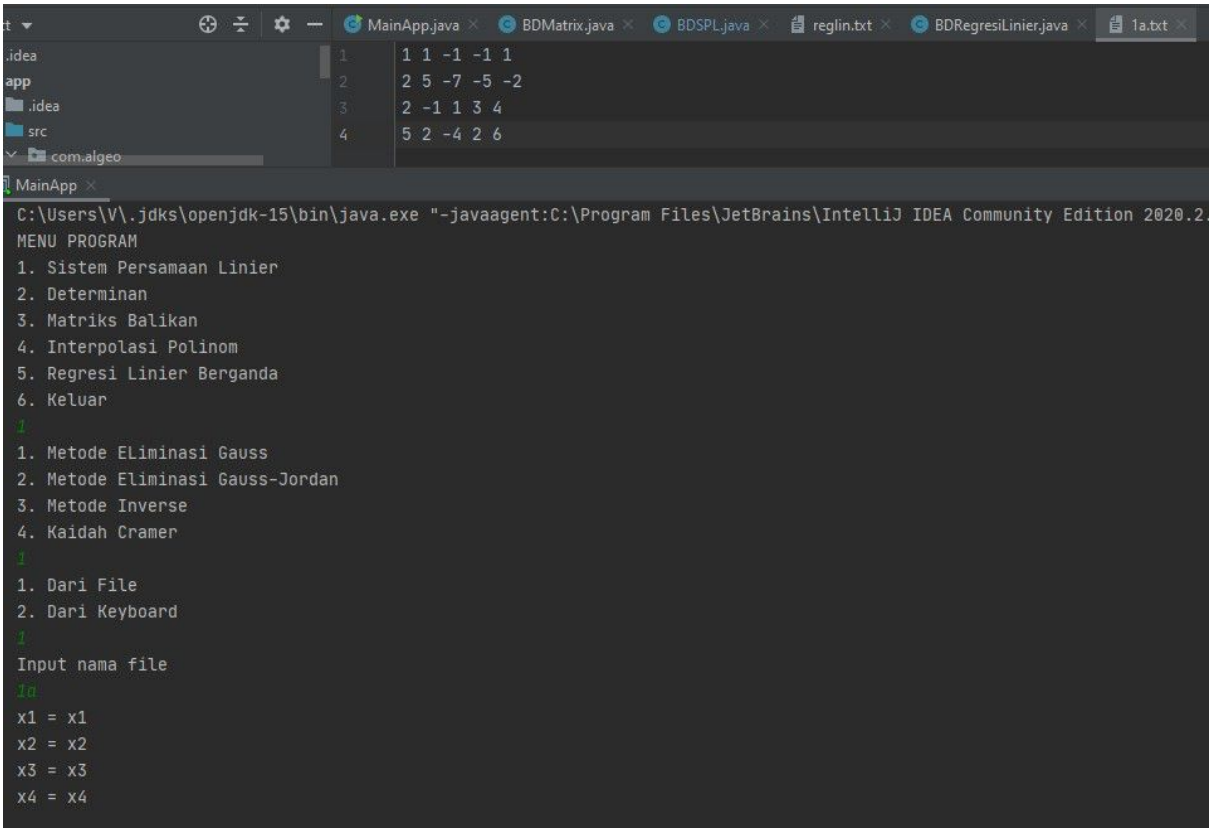
BAB IV

EKSPERIMEN

1. A.

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

Hasil yang didapat:

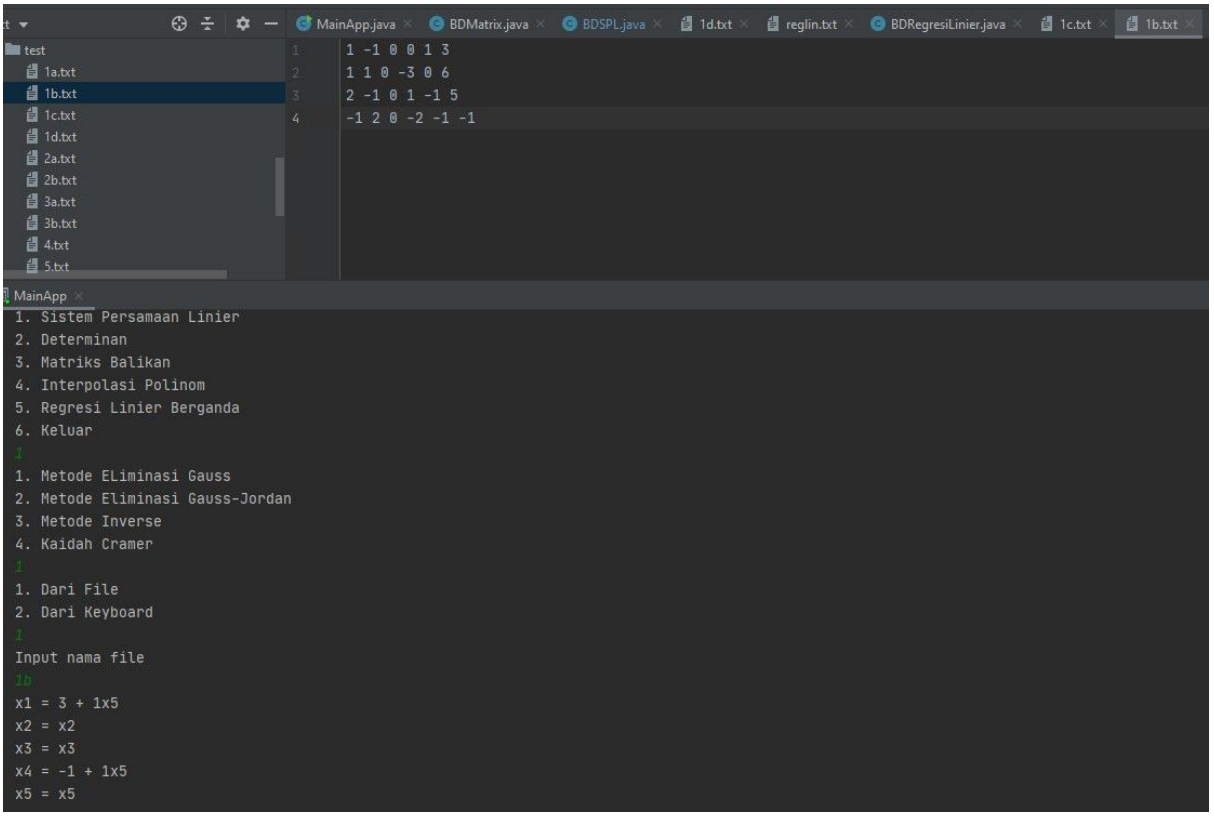


Didapat x1 = x1, x2 = 2, x3 = x3, dan x4 = x4, melainkan tidak ada solusi untuk test case 1a.

1. B.

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

Hasil yang didapat:

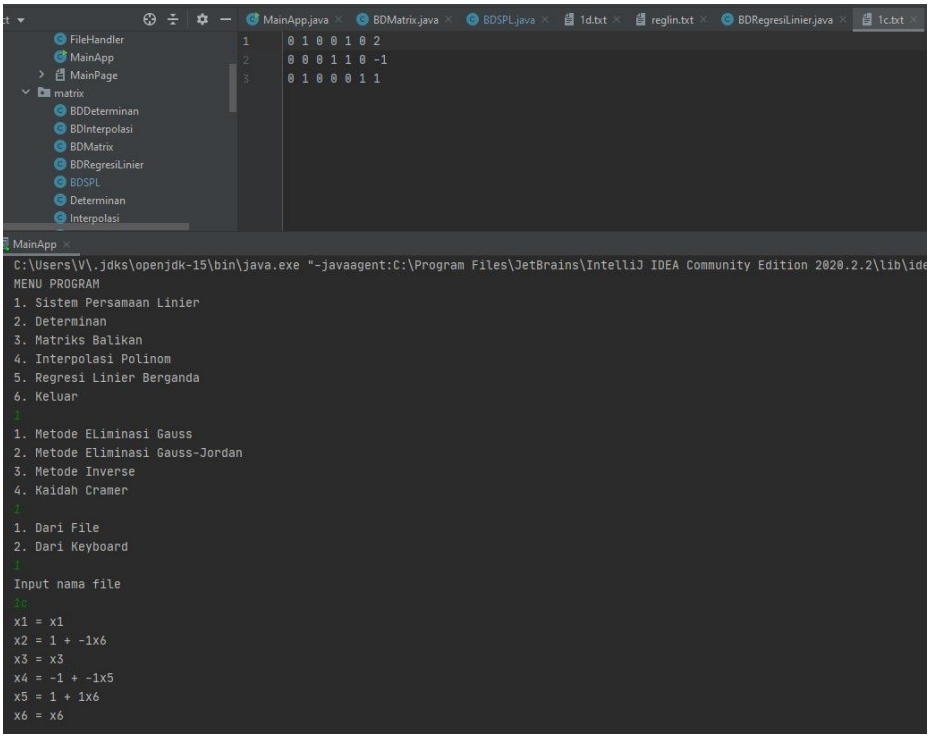


Hasil yang didapat adalah x1 = 3 + x5, x2 = x2, x3 = x3, x4 = -1 + x5 menggunakan inverse

1. C.

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

Hasil yang didapat:

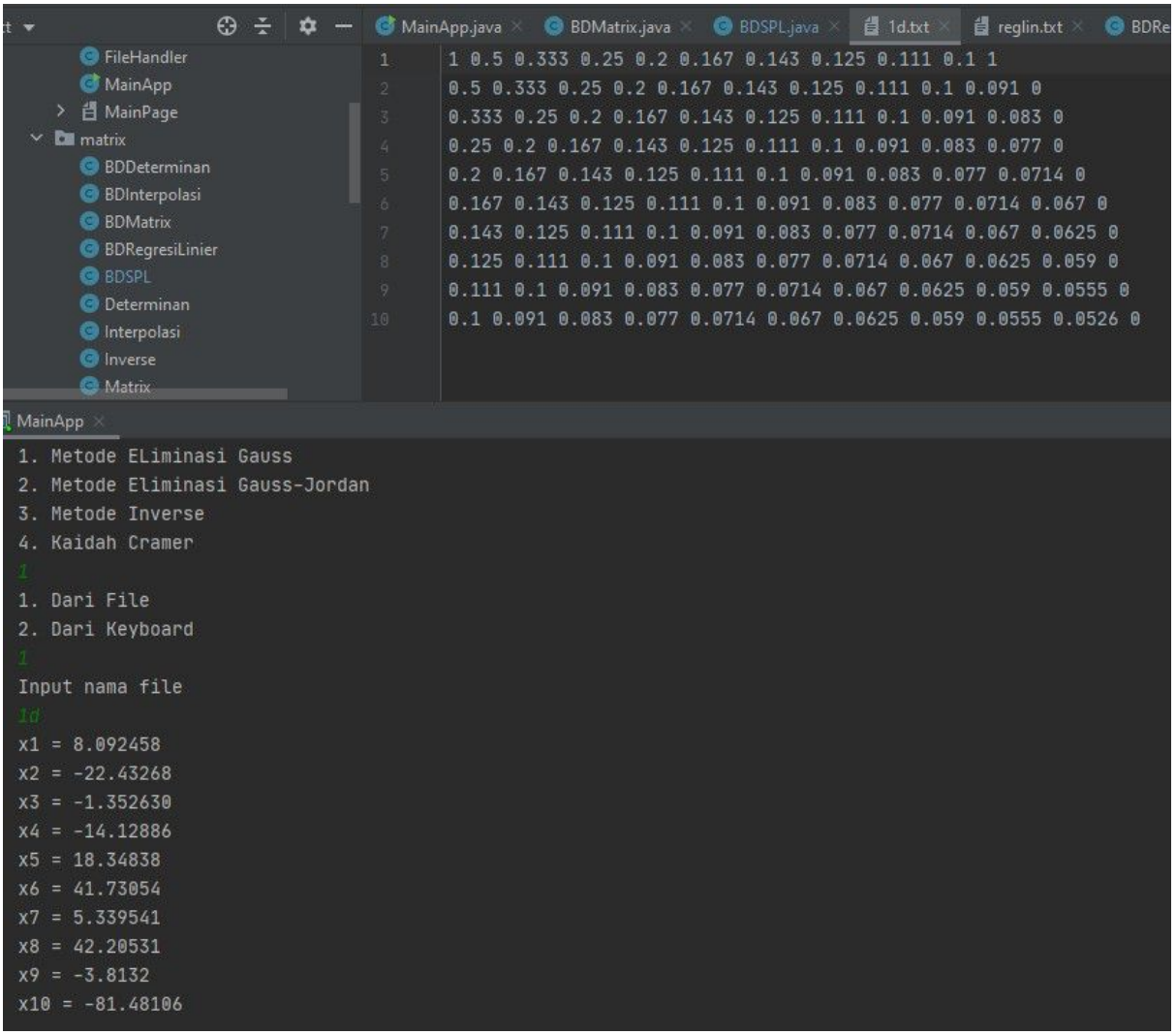


Hasil akhir yang didapat adalah x1 = x1, x2 = 1 - x6, x3 = x3, x4 = -1 -x6, x5 = 1 + x6, x6 = x6,

1. D.

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

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

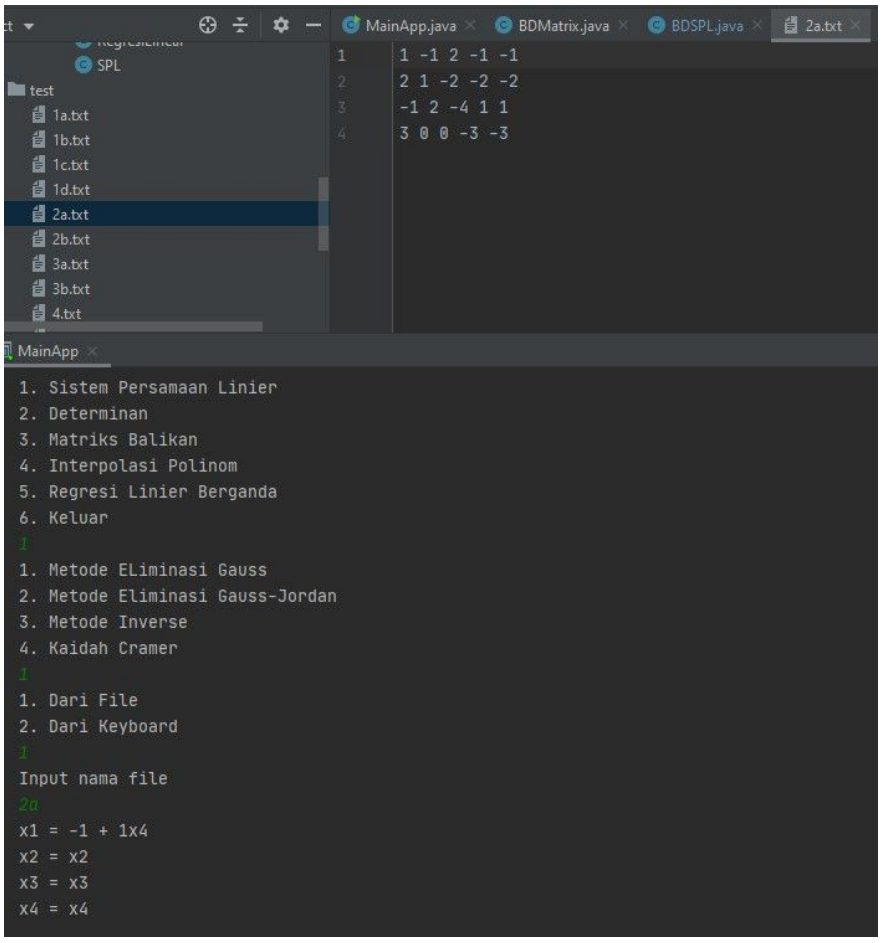


Solusi yang didapat seperti di atas

2. A. SPL berbentuk matriks augmented

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

Hasil yang didapat:



Hasil yang didapat adalah x1 = -2 + x4, x2 = x2, x3 = x3, dan x4 = x4

2. B.

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

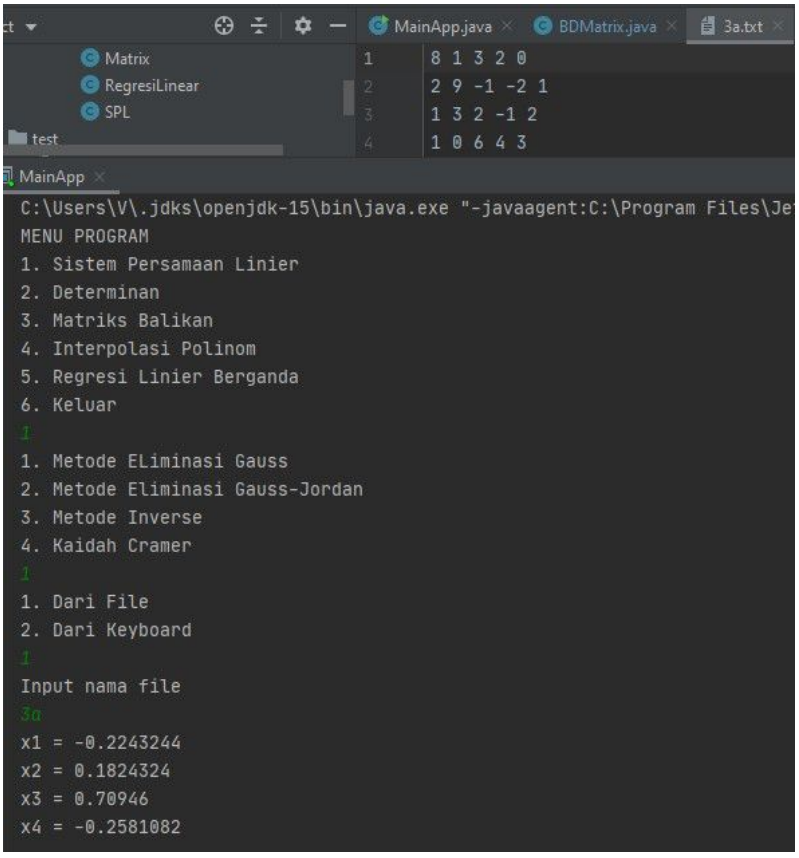
Hasil yang didapat:

Tidak terdapat solusi untuk matriks augmented tersebut

3. A.

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

Hasil yang didapat:

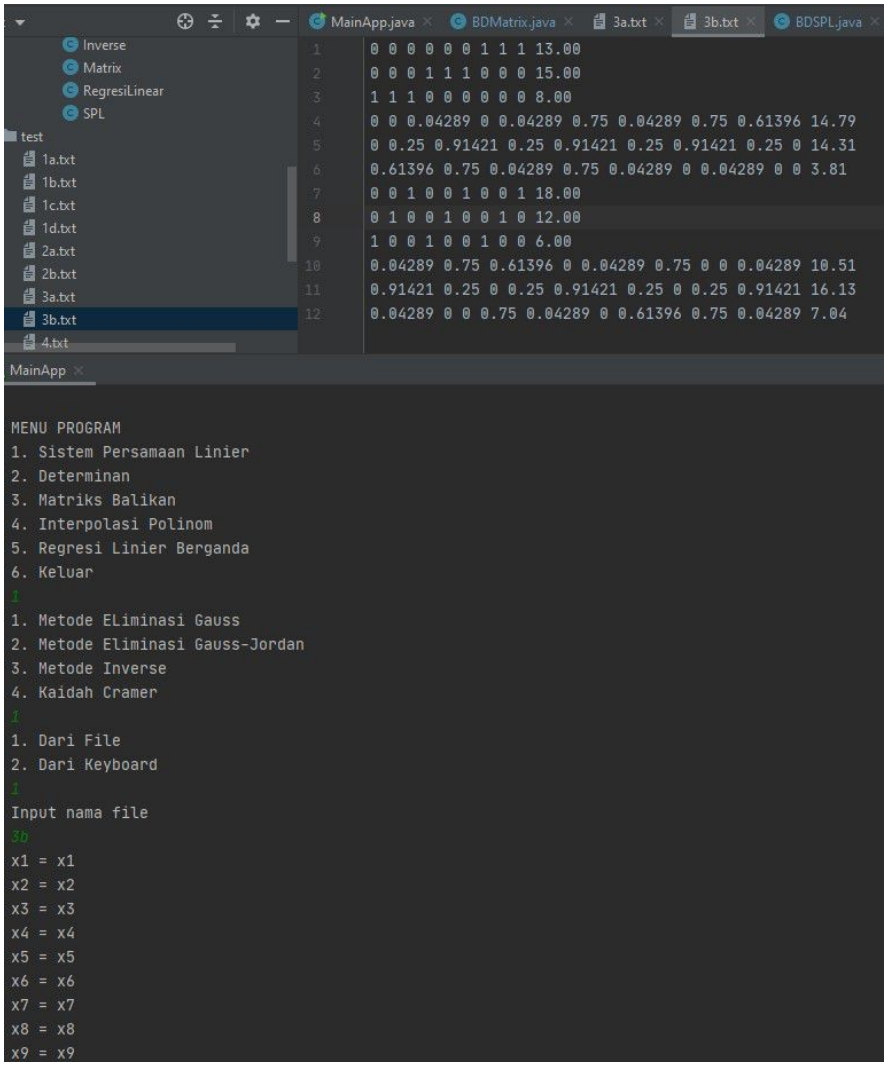


x1 = -0.224, x2 = 0.1024, x3 = 0.71, x4 = -0.25

3. B.

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

Hasil yang didapat:



4.

$$
\begin{array}{cccccccccc}
i_{12} & +i_{52} & +i_{32} & & & & & & & = 0 \\
 & -i_{52} & & +i_{65} & -i_{54} & & & & & = 0 \\
 & & -i_{32} & & & +i_{43} & & & & = 0 \\
 & & & & i_{54} & -i_{43} & & & & = 0 \\
 & & i_{32}R_{32} & & & & V_2 & -V_3 & & = 0 \\
 & & & & & i_{43}R_{43} & & +V_3 & -V_4 & = 0 \\
 & & & i_{65}R_{65} & & & & & +V_5 & = V_6 \\
i_{12}R_{12} & & & & & & +V_2 & & & = V_1 \\
 & & & & i_{54}R_{54} & & & & +V_4 & -V_5 \quad = 0 \\
 & i_{52}R_{52} & & & & & +V_2 & & & -V_5 \quad = 0
\end{array}
$$

Tentukan nilai dari:

$$i_{12}\,,\quad i_{52}\,,\quad i_{32}\,,\quad i_{65}\,,\quad i_{54}\,,\quad i_{13}\,,\quad V_2\,,\quad V_3\,,\quad V_4\,,\quad V_5$$

bila diketahui

$$R_{12} = 5 \text{ ohm}\,,\quad R_{52} = 10 \text{ ohm}\,,\quad R_{32} = 10 \text{ ohm}$$
$$R_{65} = 20 \text{ ohm}\,,\quad R_{54} = 15 \text{ ohm}\,,\quad R_{14} = 5 \text{ ohm}.$$
$$V_1 = 200 \text{ volt}\,,\quad V_6 = 0 \text{ volt}.$$

Hasil yang didapat:

Tidak terdapat solusi untuk test case 4

5. Gunakan tabel di bawah ini untuk mencari polinom interpolasi dari pasangan titk-titik yang terdapat dalam tabel. Program menerima masukan nilai x yang akan dicari nilai fungsi *f(x)*.

x	0.1	0.3	0.5	0.7	0.9	1.1	1.3
$f(x)$	0.003	0.067	0. 148	0.248	0.370	0.518	0.697

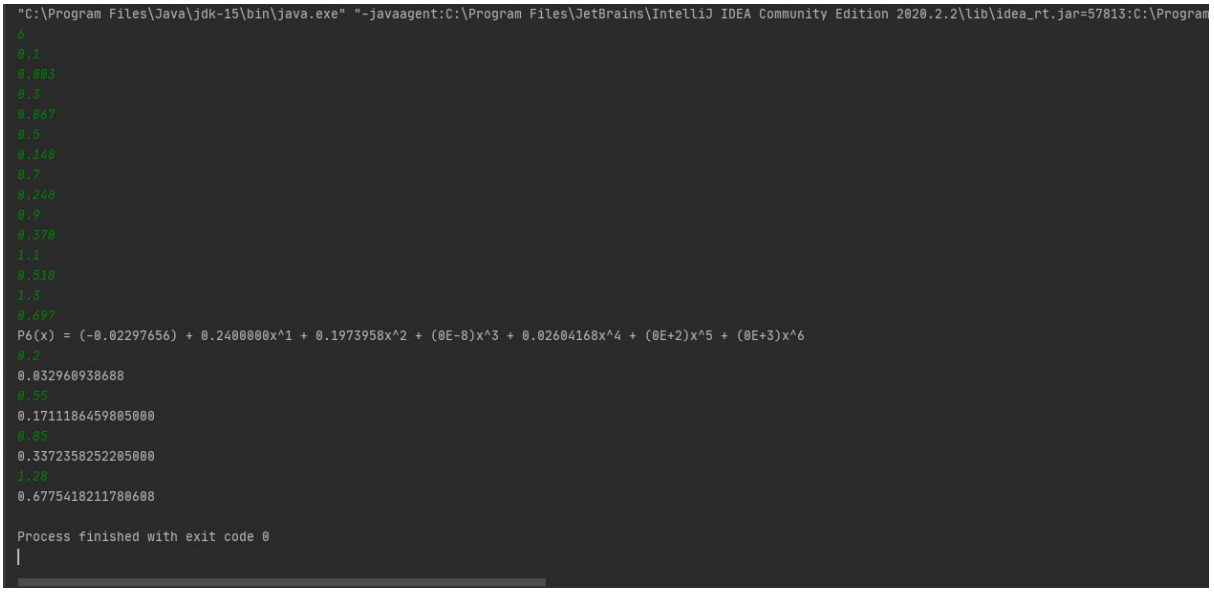
Pengujian yang dilakukan:

$x = 0.2$ f(x) = 0.032960938688

$x = 0.55$ f(x) = 0.1711186459805000

$x = 0.85$ f(x) = 0.3372358252205000

$x = 1.28$ f(x) = 0.6775418211780608



6. Jumlah kasus positif Covid-19 di Indonesia semakin bertambah dari hari ke hari. Di bawah ini diperlihatkan jumlah kasus Covid-19 di Indonesia mulai dari tanggal 24 April 2020 hingga 10 September 2020:

Tanggal	Tanggal (desimal)	Jumlah Kasus
24/04/20	4,800	8.211
30/04/20	5,000	10.118
16/05/20	5,516	17.025
22/05/20	5,710	20.796
15/06/20	6,500	39.294
06/07/20	7,194	64.958
03/08/20	8,097	113.134
08/08/20	8,258	123.503
01/09/20	9,033	177.571
10/09/20	9,333	145.510

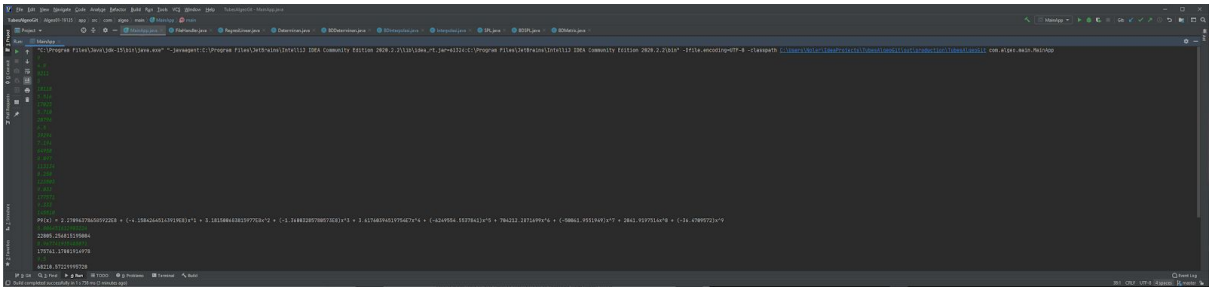
Tanggal (desimal) adalah tanggal yang sudah diolah ke dalam bentuk desimal 3 angka di belakang koma dengan memanfaatkan perhitungan sebagai berikut:

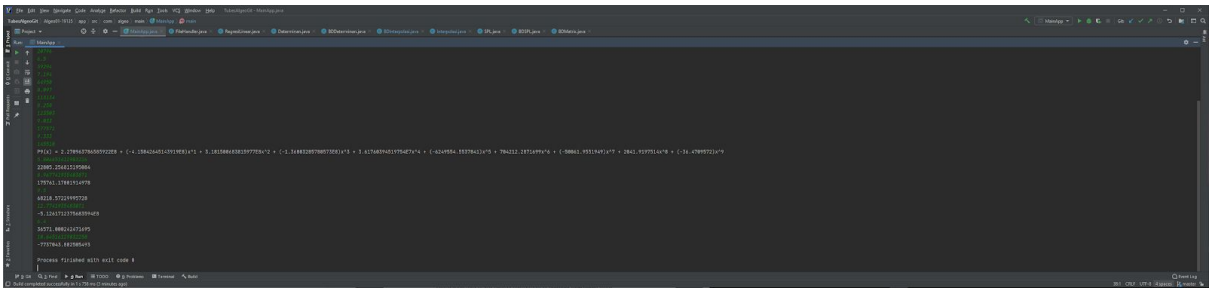
tanggal(desimal) = bulan + (tanggal / jumlah hari pada bulan tersebut)

Sebagai contoh, untuk tanggal 24/04/20 (dibaca: 24 April 2020) diperoleh tanggal(desimal) sebagai berikut: Tanggal(desimal) = 4 + (24/30) = 4,800

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

a. 25/05/20 = 5 + (25/31) = 5,806451612903226
b. 30/08/20 = 8 + (30/31) = 8,967741935483871
c. 15/09/20 = 9 + (15/30) = 9,5
d. Beserta masukan user lainnya berupa tanggal(desimal) yang sudah diolah dengan asumsi prediksi selalu digunakan untuk tahun 2020
e. Tanggal inputan yang digunakan: 24/12/20, 12/06/20 (12,7741935483871 dan 6,4)





Di-input 5 data untuk eksperimen ini, 3 dari soal, dan 2 dari user input. Data dengan nilai desimal dibulatkan.

a. Untuk 25/05/20 didapat 22.805
b. Untuk 30/08/20 didapat 175.761
c. Untuk 15/09/20 didapat 68.210
d. Untuk 24/12/20 didapat -5
e. Untuk 12/06/20 didapat 36.571

Data yang didapat untuk bulan lebih dari data yang tersedia tidak akurat (hingga sampai negatif) dikarenakan seharusnya interpolasi hanya digunakan untuk memprediksi nilai yang berada diantara titik-titik yang disediakan. Jika menggunakan diluar dari *scope* yang disediakan maka sudah menjadi ekstrapolasi, dan tingkat *error*-nya sangatlah tinggi.

7. Sederhanakan fungsi $f(x) = \frac{x^2 + \sqrt{x}}{e^x + x}$ dengan polinom interpolasi derajat n di dalam selang [0,2]. Sebagai contoh, jika n = 5, maka titik-titik x yang diambil dalam selang [0,2] berjarak h = (2-0)/5 = 0.4

n = 5

x = 0 f(x) = 0

x = 0.4 f(x) = 0.418884

x = 0.8 f(x) = 0.507158

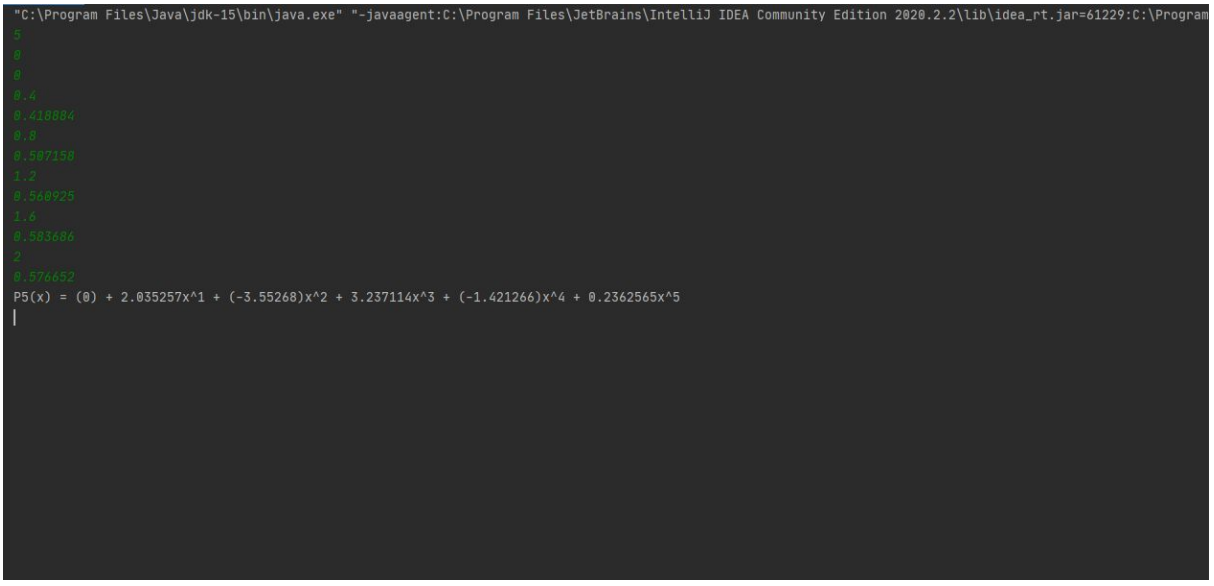
x = 1.2 f(x) = 0.560925

x = 1.6 f(x) = 0.583686

x = 2 f(x) = 0.576652

$P_5(x) = 0 + 2.035257x - 3.55268x^2 + 3.237114x^3 - 1.421266x^4 + 0.2362565x^5$

$= 2.035257x - 3.55268x^2 + 3.237114x^3 - 1.421266x^4 + 0.2362565x^5$



8. Diberikan sekumpulan data sesuai pada tabel berikut ini

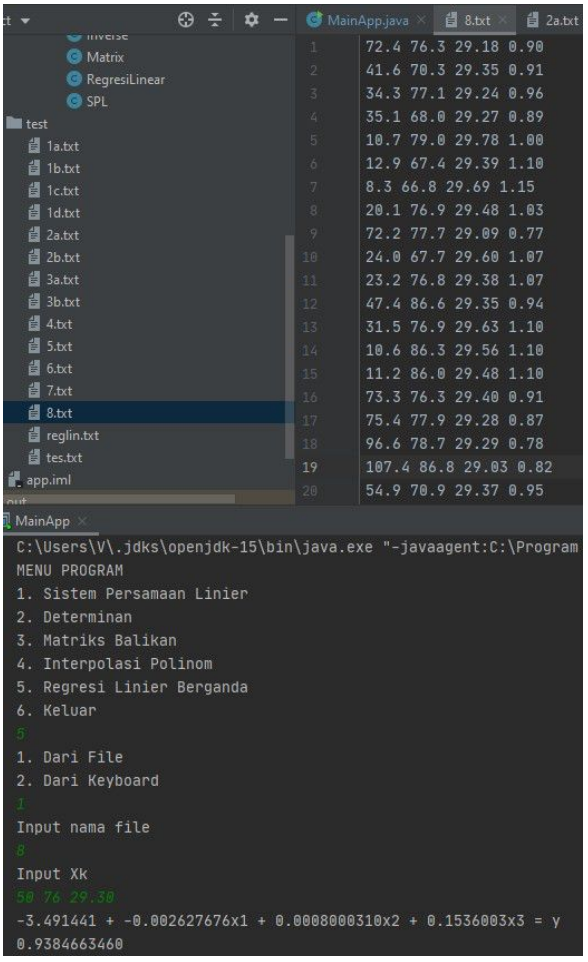
Table 12.1 Data for Example 12.1

Nitrous Oxide, y	Humidity, x_1	Temp., x_2	Pressure, x_3	Nitrous Oxide, y	Humidity, x_1	Temp., x_2	Pressure, x_3
0.90	72.4	76.3	29.18	1.07	23.2	76.8	29.38
0.91	41.6	70.3	29.35	0.94	47.4	86.6	29.35
0.96	34.3	77.1	29.24	1.10	31.5	76.9	29.63
0.89	35.1	68.0	29.27	1.10	10.6	86.3	29.56
1.00	10.7	79.0	29.78	1.10	11.2	86.0	29.48
1.10	12.9	67.4	29.39	0.91	73.3	76.3	29.40
1.15	8.3	66.8	29.69	0.87	75.4	77.9	29.28
1.03	20.1	76.9	29.48	0.78	96.6	78.7	29.29
0.77	72.2	77.7	29.09	0.82	107.4	86.8	29.03
1.07	24.0	67.7	29.60	0.95	54.9	70.9	29.37

Source: Charles T. Hare, "Light-Duty Diesel Emission Correction Factors for Ambient Conditions," EPA-600/2-77-116. U.S. Environmental Protection Agency.

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 50%, temperatur 76F, dan tekanan udara sebesar 29.30

Diperoleh data:



Didapat 0.9384663460 jadi, hasil yang didapat merupakan benar

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

Kesimpulan yang didapat dari Tugas Besar 1 Mata Kuliah Aljabar Linear dan Geometri adalah bahwa hasil yang dicapai untuk beberapa *test case* tidak merepresentasikan nilai sebenarnya dengan baik, maka digunakan BigDecimal untuk perhitungan-perhitungan mendapatkan hasil yang lebih akurat. Pengaplikasian materi-materi yang telah diajarkan ternyata digunakan dalam berbagai hal. Contohnya untuk Gauss dapat digunakan untuk menentukan Interpolasi sekaligus juga Regresi Linear Berganda. Masih lebih banyak lagi pengaplikasian dari teori-teori yang telah diajarkan namun dengan Tugas Besar ini menjadi lebih mengerti dan lebih paham mengenai materi yang sudah diajarkan karena langsung dipakai untuk persoalan sehari-hari seperti dalam test-case regresi linear, ataupun menginterpolasi / memprediksi kasus COVID-19 pada tahun 2020.

Untuk menyimpulkan Tugas Besar ini, tujuan dari Tugas Besar ini adalah untuk memperdalam pengertian mengenai materi-materi yang telah dipelajari dan mengaplikasikan teori-teori tersebut dengan algoritma yang dibuat untuk dapat digunakan dalam test case-test case yang tersedia.

Saran untuk pengembangan adalah untuk lebih terstruktur dalam membuat source code-source codenya agar lebih efisien dan lebih efektif mengerjakannya dan untuk memulai lebih awal proses pengerjaannya, dan meningkatkan komunikasi antar anggota kelompok agar lebih efektif menuliskan source code nya, dan juga lebih terkoordinasi jika akan bekerja kelompok.

Refleksi yang didapat dari Tugas Besar 1 ini adalah bahwa pengimplementasian teori-teori yang telah diajarkan selama mata kuliah Aljabar Linier dan Geometri sedikit berbeda dengan mengerjakan soal-soal sehari-harinya agar dapat diautomasikan.

REFERENSI

https://id.wikipedia.org/wiki/Eliminasi_Gauss

https://adhityafaika.wordpress.com/2018/10/24/metode-eliminasi-gauss-dan-gauss-jordan/

https://id.wikipedia.org/wiki/Eliminasi_Gauss-Jordan

http://yuliantidewi2018.blogspot.com/2018/04/eliminasi-gauss.html

https://en.wikipedia.org/wiki/Determinant

https://en.wikipedia.org/wiki/Invertible_matrix#The_invertible_matrix_theorem

https://aimprof08.wordpress.com/2012/09/26/invers-matriks/

https://www.madematika.net/2017/08/pengertian-minor-kofaktor-matriks.html

https://id.wikipedia.org/wiki/Kaidah_Cramer

https://en.wikipedia.org/wiki/Polynomial_interpolation

https://id.wikipedia.org/wiki/Regresi_linear

https://www.dosenpendidikan.co.id/rumus-interpolasi/