

Laporan Tugas Besar 1

IF2123 Aljabar Linier dan Geometri

Disusun oleh:



Kelompok Chaewon Noona :

Abrar Abhirama Widyadhana 13523038

Ahsan Malik Al Farisi 13523074

Aramazaya 13523082

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER 1 TAHUN 2023/2024

Daftar Isi

Daftar Isi.....	1
BAB I.....	1
1. Deskripsi Masalah.....	1
1.1. Sistem Persamaan Linear (SPL).....	1
1.2. Interpolasi Polinomial.....	1
1.3. Regresi Berganda.....	2
1.3.1. Regresi Linier Berganda.....	3
1.3.2. Regresi Kuadratik Berganda.....	3
1.4. Bicubic Spline Interpolation.....	4
BAB II.....	7
2. Teori Singkat.....	7
2.1. Sistem Persamaan Linear.....	7
2.2. Operasi Baris Elementer (OBE).....	7
2.3. Metode Eliminasi Gauss.....	8
2.4. Metode Eliminasi Gauss-Jordan.....	8
2.5. Determinan.....	8
2.6. Matriks Balikan.....	9
2.7. Matriks Kofaktor dan Adjoin.....	9
2.8. Kaidah Cramer.....	9
2.9. Interpolasi Polinom.....	10
2.10. Interpolasi Bicubic Spline.....	10
2.11. Regresi Linier dan Kuadratik Berganda.....	10
BAB III.....	12
3. Implementasi.....	12
3.1. Implementasi Program.....	12
3.1.1. Struktur Data.....	12
3.1.1.1. Folder primitive.....	12
3.1.1.2. Folder regression.....	19
3.1.1.3. Folder bicubicspline.....	20
3.1.1.4. Folder main.....	22
3.1.1.5. Folder interpolation.....	22
3.1.1.6. Folder lib.....	22
BAB IV.....	23
4. Hasil dan Pembahasan.....	23
4.1. Hasil Implementasi.....	23
4.1.1. Test Case Studi Kasus.....	23
4.1.1.1. Test Case No 1.....	23
4.1.1.2. Test Case No 2.....	29
4.1.1.3. Test Case No 3.....	32
4.1.1.4. Test Case No 4.....	34

4.1.1.5. Test Case No 5.....	36
4.1.1.6. Test Case No 6.....	40
4.1.1.7. Test Case No 7.....	42
4.1.1.8. Test Case Bonus.....	44
BAB V.....	45
5. Kesimpulan dan Saran.....	45
5.1. Kesimpulan.....	45
5.2. Saran.....	45
5.3. Refleksi.....	45
Daftar Pustaka.....	46
Lampiran.....	47
Hasil Asistensi.....	47

BAB I

Deskripsi Masalah

1.1. Sistem Persamaan Linear (SPL)

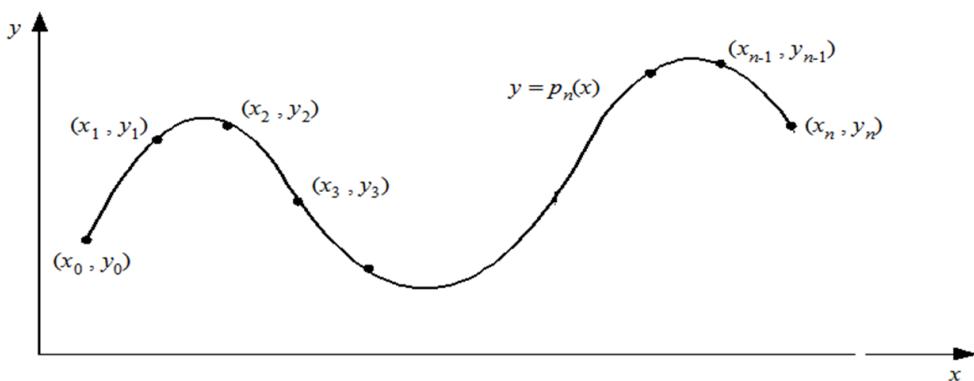
Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. Andstrea sudah mempelajari berbagai metode untuk menyelesaikan SPL, termasuk menghitung determinan matriks. Sembarang SPL dapat diselesaikan dengan beberapa metode, yaitu metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan ($x = A^{-1}b$), dan kaidah *Cramer* (khusus untuk SPL dengan n peubah dan n persamaan). Solusi sebuah SPL mungkin tidak ada, banyak (tidak berhingga), atau hanya satu (unik/tunggal).

$$\left[\begin{array}{cccc} 0 & 2 & 1 & -1 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 \end{array} \right] \cdot \left[\begin{array}{cccc} 0 & 1 & 0 & -\frac{2}{3} \\ 0 & 0 & 1 & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{array} \right].$$

Gambar 1. Eliminasi Gauss dilakukan dengan matriks eselon baris dan eliminasi Gauss-Jordan dengan matriks eselon baris tereduksi.

1.2. Interpolasi Polinomial

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



Gambar 2. Ilustrasi beberapa titik yang diinterpolasi secara polinomial.

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

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

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

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

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

1.3. Regresi Berganda

Regresi (akan dipelajari lebih lanjut di Probabilitas dan Statistika) merupakan salah satu metode untuk memprediksi nilai selain menggunakan Interpolasi Polinom. Pada tugas besar

ini, anda diminta untuk membuat 2 jenis regresi yaitu Regresi Linier Berganda dan Regresi Kuadratik Berganda.

1.3.1. Regresi Linier Berganda

Meskipun sudah ada persamaan jadi untuk menghitung regresi linear sederhana, terdapat persamaan 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:

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

1.3.2. Regresi Kuadratik Berganda

Dalam kasus ini, proses mengubah data-data dalam regresi kuadratik berganda cukup berbeda dengan Regresi Linier Berganda. Bentuk persamaan dari regresi kuadratik ada 3, yaitu:

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

Setiap n-peubah, jumlah variabel linier, kuadrat, dan interaksi akan berbeda-beda. Perhatikan contoh regresi kuadratik 2 variabel peubah sebagai berikut!

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

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

1.4. Bicubic Spline Interpolation

Bicubic spline interpolation adalah metode interpolasi yang digunakan untuk mengaproksimasi fungsi di antara titik-titik data yang diketahui. *Bicubic spline interpolation* melibatkan konsep *spline* dan konstruksi serangkaian polinomial kubik di dalam setiap sel segi empat dari data yang diberikan. Pendekatan ini menciptakan permukaan yang halus dan kontinu, memungkinkan untuk perluasan data secara visual yang lebih akurat daripada metode interpolasi linear.

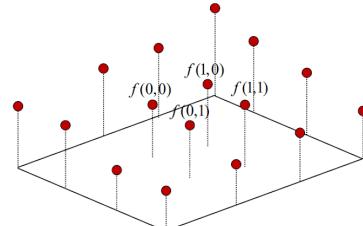
Dalam pemrosesan menggunakan interpolasi *bicubic spline* digunakan 16 buah titik, 4 titik referensi utama di bagian pusat, dan 12 titik di sekitarnya sebagai aproksimasi turunan dari keempat titik referensi untuk membagun permukaan bikubik. Bentuk pemodelannya adalah sebagai berikut.

Normalization: $f(0,0), f(1,0)$

$f(0,1), f(1,1)$

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

Solve: a_{ij}



Gambar 3. Pemodelan interpolasi *bicubic spline*.

Selain melibatkan model dasar, juga digunakan model turunan berarah dari kedua sumbu, baik terhadap sumbu x , sumbu y , maupun keduanya. Persamaan polinomial yang digunakan adalah sebagai berikut.

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

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

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

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

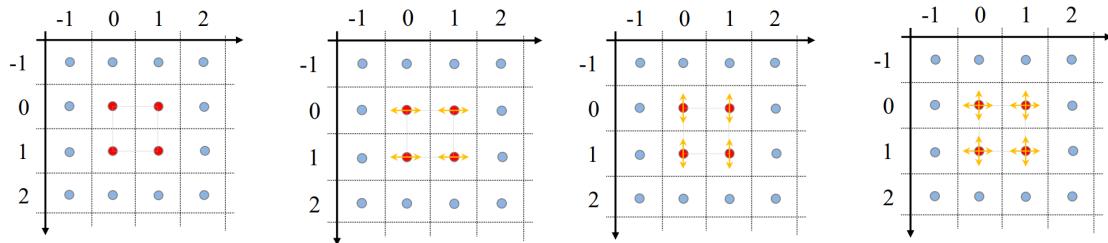
Dengan menggunakan nilai fungsi dan turunan berarah tersebut, dapat terbentuk sebuah matriks solusi X yang membentuk persamaan penyelesaian sebagai berikut.

$$\begin{matrix} y = Xa \\ \left[\begin{array}{c} f(0,0) \\ f(1,0) \\ f(0,1) \\ f(1,1) \\ f_x(0,0) \\ f_x(1,0) \\ f_x(0,1) \\ f_x(1,1) \\ f_y(0,0) \\ f_y(1,0) \\ f_y(0,1) \\ f_y(1,1) \\ f_{xy}(0,0) \\ f_{xy}(1,0) \\ f_{xy}(0,1) \\ f_{xy}(1,1) \end{array} \right] = \left[\begin{array}{cccccccccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 2 & 4 & 6 & 0 & 3 & 6 & 9 & 0 & 0 \end{array} \right] \end{matrix} \begin{matrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \\ a_{01} \\ a_{11} \\ a_{21} \\ a_{31} \\ a_{02} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{03} \\ a_{13} \\ a_{23} \\ a_{33} \end{matrix}$$

Perlu diketahui bahwa elemen pada matriks X adalah nilai dari setiap komponen koefisien a_{ij} yang diperoleh dari persamaan fungsi maupun persamaan turunan yang telah dijelaskan sebelumnya. Sebagai contoh, elemen matriks X pada baris 8 kolom ke 2 adalah koefisien dari a_{10} pada ekspansi sigma untuk $f_x(1, 1)$ sehingga diperoleh nilai konstanta $1 \times 1^{1-1} \times 1^0 = 1$, sesuai dengan isi matriks X .

Nilai dari vektor a dapat dicari dari persamaan $y = Xa$, lalu vektor a tersebut digunakan sebagai nilai variabel dalam $f(x, y)$, sehingga terbentuk fungsi interpolasi bicubic sesuai model. Tugas Anda pada studi kasus ini adalah membangun persamaan $f(x, y)$ yang akan digunakan untuk melakukan interpolasi berdasarkan nilai $f(a, b)$ dari masukan matriks 4×4 .

Nilai masukan a dan b berada dalam rentang $[0, 1]$. Nilai yang akan diinterpolasi dan turunan berarah disekitarnya dapat diilustrasikan pada titik berwarna merah pada gambar di bawah.



Gambar 4. Nilai fungsi yang akan di interpolasi pada titik merah, turunan berarah terhadap sumbu x , terhadap sumbu y , dan keduanya (kiri ke kanan).

BAB II

Teori Singkat

2.1. Sistem Persamaan Linear

Sistem Persamaan Linear adalah sebuah persamaan yang pangkat tertinggi di dalam variabelnya sama dengan satu. Sebuah SPL dengan m buah persamaan dan n Variabel x_1, x_2, \dots, x_n berbentuk :

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

Ada berbagai cara untuk menyelesaikan Sistem Persamaan Linear, salah satunya dengan metode Eliminasi Gauss, metode Eliminasi Gauss-Jordan, dan lainnya. Untuk solusinya sendiri, hanya terdapat 3 jenis solusi, yaitu :

1. Terdapat satu solusi unik,
2. Terdapat solusi tidak terhingga,
3. Tidak ada solusi umum.

2.2. Operasi Baris Elementer (OBE)

OBE adalah suatu operasi penjumlahan dan perkalian yang paling sering digunakan untuk menyelesaikan sebuah SPL. Solusi sebuah SPL diperoleh dengan menerapkan OBE pada matriks *augmented* sampai terbentuk matriks eselon baris atau matriks eselon baris tereduksi.

Tiga OBE terhadap matriks *augmented* :

1. Kalikan sebuah baris dengan konstanta tidak nol,
2. Tukarkan dua buah baris
- 3.Tambahkan sebuah baris dengan kelipatan baris lainnya.

Jika berakhir pada matriks eselon baris, maka kita akan gunakan metode Eliminasi Gauss dan jika berakhir pada matriks eselon baris tereduksi, maka gunakan metode Eliminasi Gauss-Jordan.

2.3. Metode Eliminasi Gauss

Untuk melakukan metode Eliminasi Gauss, Nyatakan SPL dalam bentuk matriks augmented kemudian diubah ke dalam bentuk eselon baris menggunakan OBE. Hal ini dilakukan agar diakhir lebih mudah menggunakan metode substitusi dan mendapatkan solusi persamaan.

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

Lalu selesaikan persamaan yang berkorespondensi pada matriks eselon baris dengan teknik penyulihan mundur (backward substitution).

2.4. Metode Eliminasi Gauss-Jordan

Metode Eliminasi Gauss-Jordan merupakan pengembangan metode eliminasi Gauss, Operasi baris elementer (OBE) diterapkan pada matriks augmented sehingga menghasilkan matriks eselon baris tereduksi

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

Tidak diperlukan lagi substitusi secara mundur untuk memperoleh nilai-nilai variabel langsung diperoleh dari matriks augmented akhir (jika solusinya unik). Metode Eliminasi Gauss-Jordan terdiri dari dua fase:

1. Fase maju (forward phase) atau fase eliminasi Gauss : Menghasilkan nilai-nilai 0 di bawah 1 utama
2. Fase mundur (backward phase) : Menghasilkan nilai-nilai 0 di atas satu utama

2.5. Determinan

Determinan adalah bilangan yang dapat dihitung dari sebuah matriks persegi ($n \times n$). Determinan sangat penting dalam analisis matriks, terutama dalam menyelesaikan Sistem Persamaan Linear, memeriksa apakah sebuah matriks memiliki invers, serta dalam berbagai aplikasi lainnya. Untuk matriks 2×2 , determinan dihitung dengan rumus sederhana:

$$\det(A) = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}$$

Sedangkan untuk matriks berukuran lebih besar ($n > 2$), determinan dapat dihitung menggunakan metode ekspansi kofaktor atau metode eliminasi Gauss

2.6. Matriks Balikan

Matriks balikan atau invers adalah matriks yang jika dikalikan dengan matriks asalnya akan menghasilkan matriks identitas. Sebuah matriks A memiliki balikan (A^{-1}) jika dan hanya jika determinan matriks tersebut tidak sama dengan nol ($\det(A) \neq 0$). Matriks balikan dihitung dengan rumus:

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

Di mana $\text{adj}(A)$ adalah matriks adjoint dari A. Invers matriks sering digunakan dalam penyelesaian SPL menggunakan metode invers matriks.

2.7. Matriks Kofaktor dan Adjoin

Kofaktor dari elemen matriks adalah nilai yang didapat dengan menghapus baris dan kolom dari elemen tersebut, kemudian menghitung determinan dari matriks yang tersisa, dan mengalikan hasilnya dengan $(-1)^{i+j}$, di mana i adalah indeks baris dan j adalah indeks kolom elemen tersebut. Matriks kofaktor adalah matriks yang berisi kofaktor dari setiap elemen dalam matriks asal.

Matriks adjoint ($\text{adj}(A)$) adalah transpose dari matriks kofaktor. Matriks adjoint digunakan untuk menghitung invers suatu matriks.

2.8. Kaidah Cramer

Kaidah Cramer adalah metode untuk menyelesaikan Sistem Persamaan Linear menggunakan determinan. Jika kita memiliki SPL dengan n persamaan dan n variabel, maka solusi SPL tersebut dapat diperoleh dengan:

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

Di mana A adalah matriks koefisien dan A_i adalah matriks yang didapat dengan mengganti kolom ke-i dari A dengan vektor konstanta (b). Kaidah Cramer hanya berlaku jika determinan A tidak nol.

2.9. Interpolasi Polinom

Interpolasi polinom adalah teknik untuk menemukan polinom yang melewati sejumlah titik data tertentu. Jika diberikan $n+1$ titik data, maka interpolasi polinom mencari polinom derajat n yang melalui semua titik tersebut. Polinom interpolasi sering digunakan dalam berbagai aplikasi seperti analisis data, grafik, dan prediksi.

Interpolasi Lagrange dan Newton adalah dua metode yang umum digunakan dalam interpolasi polinom.

2.10. Interpolasi Bicubic Spline

Interpolasi bicubic spline adalah metode interpolasi yang digunakan untuk memperhalus data dua dimensi, seperti gambar atau permukaan. Metode ini melibatkan interpolasi spline kubik pada dua arah (horizontal dan vertikal) secara berturut-turut untuk memberikan hasil interpolasi yang lebih halus daripada metode interpolasi linier atau bilinear.

Interpolasi bicubic spline memberikan hasil yang sangat baik ketika diperlukan interpolasi data yang halus dan kontinu, seperti dalam grafika komputer dan pemrosesan citra.

2.11. Regresi Linier dan Kuadratik Berganda

Regresi linier berganda adalah metode statistik yang digunakan untuk memodelkan hubungan antara variabel dependen dengan dua atau lebih variabel independen. Persamaan regresi linier berganda berbentuk:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon$$

Di mana $\beta_0, \beta_1, \dots, \beta_n$ adalah koefisien yang harus diperkirakan, dan ϵ adalah kesalahan atau residual. Regresi ini digunakan untuk membuat prediksi atau untuk mengetahui pengaruh variabel independen terhadap variabel dependen.

Regresi kuadratik berganda adalah pengembangan dari regresi linier berganda, di mana variabel independen juga dapat berbentuk kuadrat, sehingga model persamaan menjadi:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_{12} + \beta_4 x_{22} + \dots + \epsilon$$

Model ini lebih fleksibel dan dapat menangkap pola yang lebih kompleks dibandingkan regresi linier biasa.

BAB III

Implementasi

3.1. Implementasi Program

3.1.1. Struktur Data

3.1.1.1. Folder primitive

a. BasicFunction

Class BasicFunction memiliki fungsi dan prosedur yang sekiranya akan sering digunakan

- **Fungsi**

Fungsi	Deskripsi
inputMatrixn()	Fungsi untuk mengeluarkan matrix dengan ukuran $n \times n$.
inputMatrix()	Fungsi mengeluarkan matrix double dengan dimensi dan elemen sesuai input user dengan input per line.
inputPolinomial()	Fungsi mengeluarkan matrix berukuran jumlah titik $\times 2$ sesuai dengan format matriks polinomial
getIdentity(int n)	Fungsi mengeluarkan matrix identity dalam format double[][] dengan dimensi n.
rowZero(double[][] matrix, int rowIndex)	Fungsi mengembalikan true jika seluruh elemen matrix pada baris rowIndex merupakan 0.
colzero(double[][] matrix, int colIndex)	Fungsi mengembalikan true jika seluruh elemen pada kolom colIndex merupakan 0.
transpose(double[][] matrix)	Mengeluarkan hasil matrix transpose dari parameter matrix
readInput()	Fungsi mengeluarkan variable string hasil input user. Dapat mengeluarkan IOException.
multiplication(double[][] m1, double[][] m2)	Fungsi mengeluarkan hasil kali matrix m1 dan m2.
stripMatrix(double[][] matrix, int colIndex)	Fungsi mengembalikan array yang berupa kolom index ke colIndex.
stripMatrixPolinomial(double[][] matrix)	Mengembalikan variabel-variabel x,y untuk hasil input dari class interpolasi polinom.

stripMatrixEquation(double[][] matrix)	Fungsi memisahkan persamaan (titik x dan y) dari nilai x yang akan diprediksi.
factorial(int n)	Fungsi rekursif untuk mengambil n!.

- **Prosedur**

Prosedur	Deskripsi
copyMatrix(double[][] matrix, double[][] matrix_out)	Merubah elemen pada matrix_out menjadi deep clone elemen pada matrix.
scaleVector(double[][] matrix, int rowIndex, int scale)	Mengalikan row posisi rowIndex pada matrix dengan bilangan skalar.
printMatrix(double[][] matrix)	Menampilkan tiap elemen matrix pada layar/konsol.
printArray(double[] solution)	Menampilkan tiap elemen solution pada layar/konsol.
swapRow(double[][] matrix, int row1, int row2)	Menukar posisi row1 dan row2 pada matrix.
partialPivot(double[][] matrix, int currRow)	Menukar posisi baris jika terdapat elemen pada kolom tersebut yang lebih elemen dari elemen pada currRow.
setColumnElement(double[][] matrix, int colIndex, double[] values)	Merubah elemen pada sebuah kolom pada matrix menjadi elemen pada array values.
setColumnOneElement(double[][] matrix, int colIndex, double value)	Merubah tiap elemen pada sebuah kolom pada matrix menjadi value.

- b. **CofactorExpansion**

Mengandung fungsi dalam perhitungan determinan menggunakan cofactor expansion.

- **Fungsi**

Fungsi	Deskripsi
takeCofactor(double[][] matrix, int row, int col)	mengembalikan matrix $n-1 \times n-1$ yang merupakan kofaktor C_{ij} dimana $i = \text{row}$ dan $j = \text{col}$.
determinant(double[][] matrix)	mengembalikan determinan dari matrix dengan menggunakan metode kofaktor.

c. Cramer

Mengandung fungsi yang berkaitan dengan pencarian hasil SPL melalui kaidah cramer.

- **Fungsi**

Fungsi	Deskripsi
driverCramerSolver()	Fungsi dipanggil untuk menghasilkan nilai array dengan perhitungan cramer.
CramerSolver(double[][] matrix)	Fungsi mengembalikan hasil SPL yang didapat menggunakan kaidah Cramer. Jika determinan bernilai nol, maka fungsi melempar error.
stripVectorB(double[][] matrix)	fungsi mengembalikan array B dari persamaan $Ax = B$.
stripVectorA(double[][] matrix)	fungsi mengembalikan matrix A dari persamaan $Ax = B$.

d. Determinant

Mengandung fungsi yang berkaitan dengan pencarian determinan menggunakan OBE.

- **Fungsi**

Fungsi	Deskripsi
driverRowReductionDet()	Fungsi dipanggil untuk mengembalikan nilai dari row reduction determinan.
rowReductionDeterminant(double[][] matrix)	Fungsi mengembalikan determinan dari matrix yang didapat melalui OBE.

e. GaussElimination

Mengandung fungsi yang dipakai dalam pencarian hasil SPL menggunakan metode eliminasi Gauss

- **Fungsi**

Fungsi	Deskripsi
driverGaussElimination()	Fungsi dipanggil dalam main untuk mengembalikan hasil dari eliminasi gauss.
gaussElimination(double[][] matrix)	Metode utama kelas GaussElimination dimana fungsi akan mengembalikan string yang berisi hasil SPL atau hasil paramterik

	SPL.
readInput()	Fungsi mengembalikan hasil input user.
isRowZero(double[] row)	Fungsi mengembalikan true jika row hanya berisi 0.
switchRow(double[][] matrix, int i)	Fungsi mengubah baris jika elemen diagonalnya 0. Jika terjadi penukaran baris, fungsi mengembalikan true.
normalBackSubstitution(double[][] matrix)	Mengembalikan nilai tiap variabel jika matrix memiliki hasil unik.
printNormalBackSubstitution(double [] resultArray)	menampilkan hasil SPL pada layar dan mengembalikan hasil tersebut dalam bentuk String.
ParametricBackSubstitution(double[][] matrix)	Mengembalikan hasil parametrik dari SPL jika SPL memiliki banyak hasil.

f. GaussJordanElimination

Class ini mengandung fungsi-fungsi yang digunakan dalam pencarian solusi SPL dengan metode eliminasi Gauss-Jordan.

- **Fungsi**

Fungsi	Deskripsi
driverGaussJordanElimination()	Fungsi yang dipanggil dalam main untuk mengembalikan hasil dari eliminasi gauss jordan.
gaussJordanElimination(double[][] matrix)	metode utama kelas GaussJordanElimination dimana fungsi akan mengembalikan string yang berisi hasil SPL atau hasil paramterik SPL.
readInput()	Fungsi mengembalikan hasil input user.
isRowZero(double[] row)	Fungsi mengembalikan true jika row hanya berisi 0.
switchRow(double[][] matrix, int i)	Fungsi mengubah baris jika elemen diagonalnya 0. Jika terjadi penukaran baris, fungsi mengembalikan true.

- **Prosedur**

Prosedur	Deskripsi
printArrayJordan(double[][] matrix)	Menampilkan hasil SPL di konsol/layar.

g. **InputOutput**

Mengandung semua metode yang berhubungan dengan file handling.

- **Fungsi**

Fungsi	Deskripsi
getValidIntegerInput(Scanner scanner, String prompt)	Fungsi untuk meminta masukan nilai berupa integer dan mengembalikannya.
readInputRegression(String filepath, double[][] matrix, double[] predictors, int n, int m)	Fungsi untuk memanggil dan memakai fungsi regresi dengan input melalui pembacaan file.
readInputPolynomialInterpolation(String filepath, double[][] matrix, int n, double xEstimate)	Fungsi untuk memanggil dan memakai fungsi interpolasi polinomial dengan input melalui pembacaan file.
readInputBicubic(String filepath, double[][] matrix, double[] predictors, int x, int y)	Fungsi untuk memanggil dan memakai fungsi bicubic dengan input melalui pembacaan file.
readMatrixFile(string filepath)	Fungsi mengembalikan matrix yang dibaca dari file .txt pada filepath.
checkFilePath(String outputPath)	Fungsi mengembalikan true jika file pada outputPath sudah ada.

- **Prosedur**

Prosedur	Deskripsi
writeMatrixToFile(double[][] matrix, String outputPath)	Prosedur menulis matrix langsung pada file di outputPath.
writeArrayToFile(double[] matrix, String outputPath)	Prosedur menulis array langsung pada file di outputPath.
writeStringToFile(String content, String outputPath)	Prosedur menulis string langsung pada file di outputPath.
writeDoubleToFile(double[][] matrix, String outputPath)	Prosedur menulis double langsung pada file di outputPath.

writeMatrixFile(double[][] matrix)	Prosedur meminta masukkan outputPath untuk lokasi save dan menulis matrix pada file di outputPath.
writeStringFile(String string)	Prosedur meminta masukkan outputPath untuk lokasi save dan menulis string pada file di outputPath.
writeDoubleFile(String string)	Prosedur meminta masukkan outputPath untuk lokasi save dan menulis double pada file di outputPath.
writeArrayFile(double[] matrix)	Prosedur meminta masukkan outputPath untuk lokasi save dan menulis array pada file di outputPath.

h. Inverse

Class Mengandung semua fungsi yang dipakai untuk mendapatkan nilai Inverse baik melalui OBE maupun Adjoint

- **Fungsi**

Fungsi	Deskripsi
driverInverseCofactor()	Fungsi untuk dipanggil dalam main dan mengembalikan nilai matrix inverse cofactor.
InverseCofactor(double[][] Matrix)	Fungsi mengembalikan matrix invers dari Matrix dengan menggunakan metode Adjoin.
switchRowIden(double[][] matrix, double[][] identity, int i)	Fungsi menukar baris pada matrix dan identity jika elemen diagonalnya 0. Jika setidaknya satu baris ditukar, fungsi mengembalikan true.
isRowZeroIden(double[] row)	Melakukan pengecekan apakah row pada matrix dan identity matrix merupakan full 0.
driverInverseERO()	Fungsi untuk dipanggil dan menghasilkan matriks inverse ERO.
InverseERO(double[][] Matrix)	Mengembalikan matriks balikan dari Matrix dengan menggunakan metode OBE.
getInverseOutput(double[][] matriks, String Function)	Mengembalikan matriks balikan dari matriks dalam bentuk String dan dengan metode sesuai dengan yang

	tertulis di Function.
MatriksIdentity_maker(double[][] matrix)	Mengembalikan matriks identitas yang memiliki dimensi sama dengan matrix.
Add_MatrixIdentity(double[][] matrix)	Mengembalikan matriks augmented yang akan dipakai dalam metode balikan dengan OBE.
zeroCounter(double[][] matrix, int row)	Menghitung jumlah elemen ‘0’ pada baris ke-’row’
Gauss_Operation(double[][] matrix)	Melakukan operasi gauss pada matrix dan mengembalikan matrix eselon
getInverseMatriks(double[][] matrix)	Mengembalikan hasil matriks balikan dengan OBE.
getInverseFromAdjoin	Mengembalikan hasil inverse dengan metode cofactor.
getKofaktor(double[][] matriks)	Mengembalikan matriks kofaktor dari matriks.
getKofaktorValue(double[][] matriks, int x, int y)	Mengembalikan nilai kofaktor C_{xy} .
getTranspose(double[][] matriks)	Mengembalikan transpose dari matriks.
determinan_kofaktor(double[][] m)	Mengembalikan hasil determinan dengan menggunakan metode ekspansi kofaktor.
submatriks(double[][] m, int row, int col)	Mengembalikan submatriks yang tidak ada baris row dan kolom col nya.

- **Prosedur**

Prosedur	Deskripsi
toIdentity(double[][] matrix, double[][] identity)	Merubah matriks matrix menjadi bentuk identity matriks dengan menggunakan OBE dan melakukan operasi OBE yang sama pada matriks identity sehingga terbentuk matriks balikan.
swapping_Operation(double[][] matrix)	Menukar baris pada matriks jika sebuah baris memiliki jumlah nol yang lebih banyak.
matrix_swapping(double[][] matrix, int row_will_be_changed, int	Operasi menukar matriks pad baris row_will_be_changed dan pada baris

row_who_changed)	row_who_changed.
Multipy_Operation(double[][] matrix, int row_will_be_changed, int column_will_be_changed)	Mengalikan seluruh elemen pada baris row_will_be_changed dengan nilai 1/elenen matriks pada (row_will_be_changed, col_will_be_changed)
Reduce_Operation(double[][] matrix, int row_will_be_changed, int column_will_be_changed, int row_who_changed)	Mengurangi nilai elemen pada baris row_will_be_changed dengan elemen tersebut dikalikan rasio antara elemen (row_will_be_changed, column_will_be_changed) dan (row_who_changed, column_will_be_changed).

i. Matriks Balikan

Fungsi	Deskripsi
driverSPLInverse()	Fungsi untuk dipanggil dan menerima masukkan, kemudian mengembalikan hasil dari matrix SPL Invers.
SPLInverse(double[][] matrix)	Fungsi untuk menghitung dan mengembalikan hasil dari matrix SPL Invers.

3.1.1.2. Folder regression

a. MultipleLinearRegression

Fungsi	Deskripsi
inputArrayReg(int m)	Fungsi untuk menerima masukan dari user dan mengembalikan input array regresi.
inputMatrixReg(int n, int m)	Fungsi untuk menerima masukan dari user dan mengembalikan input matrix regresi.
multipleLinearRegression()	Fungsi regresi linear berganda, menghitung dan mencetak koefisien regresi
getCoefficient(double[][] IndependentVar, double[][]	Menghitung koefisien regresi linear berganda menggunakan eliminasi gauss

DependentVar, DoubleWrapper Coeff)	
------------------------------------	--

b. MultipleQuadraticRegression

Fungsi	Deskripsi
multipleQuadRegression()	Menghitung dan mencetak koefisien regresi kuadratik.
getCoefficient(double[][] IndependentVar, double[][] DependentVar, DoubleWrapper Coeff)	Fungsi untuk membantu perhitungan koefisien untuk regresi kuadratik berganda menggunakan eliminasi Gauss. Mengembalikan True jika mode bisa diselesaikan
numOfVar(double[][] IndependentVar)	Menghitung jumlah variabel untuk regresi kuadratik berdasarkan jumlah variabel bebas.

3.1.1.3. Folder bicubicspline

a. InterpolasiBicubicSpline

Fungsi	Deskripsi
driverBicubicSpline()	Fungsi untuk dipanggil dan menerima input, kemudian mengembalikan hasil String.
bicubicSpline()	menginisialisasi matriks input 4x4, X dan Y, dan menjalankan proses interpolasi bicubic spline.
bicubiInterpolation(double x, double y)	menghitung hasil interpolasi untuk koordinat (x, y) dengan menggunakan matriks koefisien yang telah dihitung sebelumnya

Prosedur	Deskripsi
matrixSingular(double[][] input)	mengonversi matriks input 4x4 menjadi vektor kolom 16x1. Setiap elemen matriks input disusun ulang menjadi bentuk vektor
matrixCoefficient()	menghitung matriks koefisien dengan mengalikan invers dari matriks invers

	dengan vektor input
konstanta(int i)	menentukan tipe turunan berdasarkan indeks i, lalu memanggil fungsi derivative untuk menghitung nilai yang sesuai di matriks invers
derivative(int x, int y, int i)	menghitung nilai turunan dari fungsi interpolasi bicubic

b. ImageResizer.java

Fungsi	Deskripsi
bicubicInterpolation(double x, double y, BufferedImage image)	melakukan interpolasi bicubic pada koordinat (x, y) dari gambar. Gambar dibagi menjadi tiga komponen warna (R, G, B), dan interpolasi dilakukan pada masing-masing komponen warna untuk mendapatkan nilai warna baru.
performBicubicInterpolation(double x, double y, int[][] pixelGrid)	melakukan interpolasi bicubic pada grid 4x4 piksel untuk komponen warna tertentu (R, G, atau B).
matrixSingular(double[][] input)	mengonversi grid 4x4 input menjadi vektor 16x1

Prosedur	Fungsi
resizeImage()	Fungsi memperbesar atau memperkecil ukuran gambar berdasarkan skala yang diberikan oleh pengguna.
matrixCoefficient()	Menghitung matriks koefisien dengan mengalikan matriks invers
konstanta(int i)	Fungsi ini menentukan jenis turunan berdasarkan indeks i
derivative(int x, int y, int i)	Fungsi menghitung nilai turunan parsial dari fungsi interpolasi bicubic untuk setiap entri dalam matriks invers

3.1.1.4. Folder main

Folder main mengandung class utama pada program. Class ini menyatukan semua metode metode dari class lain.

3.1.1.5. Folder interpolation

a. PolinomialInterpolation

Fungsi	Deskripsi
polinomialInterpolationSolver()	Fungsi untuk dipanggil dan menerima input kemudian mengembalikan hasil dari interpolasi polinomial.
polinomialInterpolation(double[][] matrix)	Fungsi melakukan interpolasi polinomial berdasarkan data titik-titik yang diberikan.
BasicFunction.copyMatrix(double[][] src, double[][] dest)	Mengopi isi dari matriks sumber ke matriks tujuan.
BasicFunction.stripMatrixPolynomial(double[][] matrix)	Mengambil titik-titik (x, y) dari data yang diberikan untuk digunakan dalam interpolasi.
BasicFunction.setColumnOneElement(double[][] matrix, int col, double value)	Mengisi elemen kolom pertama dengan angka 1.
Cramer.CramerSolver(double[][] matrix)	Menyelesaikan sistem persamaan linier menggunakan aturan Cramer untuk mendapatkan solusi dari koefisien persamaan polinomial.

3.1.1.6. Folder lib

Folder lib mengandung file .jar dari tiap folder agar metode dari tiap folder dapat digunakan pada folder lain.

BAB IV

Hasil dan Pembahasan

4.1. Hasil Implementasi

4.1.1. Test Case Studi Kasus

4.1.1.1. Test Case No 1

Temukan Solusi SPL $Ax = b$, berikut:

a.

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

- Metode Gauss

```
Matrix akhir:  
1,0000 1,0000 -1,0000 -1,0000 1,0000  
0,0000 3,0000 -5,0000 -3,0000 -4,0000  
0,0000 0,0000 -2,0000 2,0000 -2,0000  
0,0000 0,0000 0,0000 0,0000 1,0000  
  
Tidak ditemukan solusi unik
```

- Metode Gauss Jordan

```
1. Metode eliminasi Gauss  
2. Metode eliminasi Gauss-Jordan  
3. Kaidah Cramer  
4. Kembali  
Pilih Metode: 1  
Metode eliminasi Gauss  
Ambil variabel dari file?(Y/n/C) : y  
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\2a.txt  
filename: C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\2a.txt  
1,0000 -1,0000 2,0000 -1,0000 -1,0000  
2,0000 1,0000 -2,0000 -2,0000 -2,0000  
-1,0000 2,0000 -4,0000 1,0000 1,0000  
3,0000 0,0000 0,0000 -3,0000 -3,0000  
  
Matrix akhir:  
1,0000 -1,0000 2,0000 -1,0000 -1,0000  
0,0000 3,0000 -6,0000 0,0000 0,0000  
0,0000 0,0000 0,0000 0,0000 0,0000  
0,0000 0,0000 0,0000 0,0000 0,0000  
  
Ditemukan solusi parametric  
x1 = -1,00 + 2,00x3 + -2,00x3 + 1,00x4  
x2 = + 2,00x3  
x3 = x3  
x4 = x4
```

- Metode Cramer

```
MENU  
1. Sistem Persamaan Linier  
2. Determinan  
3. Matriks Balikan  
4. Interpolasi Polinom  
5. Regresi Linear dan Kuadratik Berganda  
6. Interpolasi Bicubic Spline  
7. Interpolasi Gambar  
8. Keluar  
Pilih Menu: 1  
1. Metode eliminasi Gauss  
2. Metode eliminasi Gauss-Jordan  
3. Kaidah Cramer  
4. Kembali  
Pilih Metode: 3  
Ambil variabel dari file?(Y/n/C) : y  
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\1a.txt  
Exception in thread "main" java.lang.IllegalStateException: Determinan bernilai nol sehingga tidak ada solusi
```

- Metode Matriks Balikan

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Metode matriks balikan
5. Kembali
Pilih Metode: 4
Aambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): D:\java testcase\1b.txt
filename: D:\java testcase\1b.txt
Matrix is singular.

```

b.

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

- Metode Gauss\

```

Pilih Menu: 1
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 1
Metode eliminasi Gauss
Aambil variabel dari file?(Y/n/C) : Y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\1b.txt
filename: C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\1b.txt
1,0000 -1,0000 0,0000 0,0000 1,0000 3,0000
1,0000 1,0000 0,0000 -3,0000 0,0000 0,0000
2,0000 -1,0000 0,0000 1,0000 -1,0000 5,0000
-1,0000 2,0000 0,0000 -2,0000 -1,0000 -1,0000

Matrix akhir:
1,0000 -1,0000 0,0000 0,0000 1,0000 3,0000
0,0000 2,0000 0,0000 -3,0000 -1,0000 3,0000
0,0000 0,0000 0,0000 2,5000 -2,5000 -2,5000
0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

Ditemukan solusi parametric
x1 = 3,00 + 1,50 * (-1,00 + 1,00x5) + 0,50x5 + -1,00x5
x2 = 1,50 + 1,50 * (-1,00 + 1,00x5) + 0,50x5
x3 = x3
x4 = -1,00 + 1,00x5
x5 = x5

```

- Metode Gauss Jordan

```

Pilih Menu: 1
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 2
Aambil variabel dari file?(Y/n/C) : Y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\1b.txt
filename: C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\1b.txt
Matrix akhir:
1,0000 0,0000 0,0000 0,0000 -1,0000 3,0000
0,0000 1,0000 0,0000 0,0000 -2,0000 0,0000
0,0000 0,0000 0,0000 1,0000 -1,0000 -1,0000
0,0000 0,0000 0,0000 0,0000 0,0000 0,0000

Ditemukan solusi parametric
x1 = 3,00 + 1,00x5
x2 = + 2,00x5
x3 = x3
x4 = -1,00 + 1,00x5
x5 = x5

```

- Metode Cramer

```

MENU
1. Sistem Persamaan Linier
2. Determinan
3. Matriks Balikan
4. Interpolasi Polinom
5. Regresi Linear dan Kuadratik Berganda
6. Interpolasi Bicubic Spline
7. Interpolasi Gambar
8. Keluar
Pilih Menu: 1
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 3
Ambil variabel dari file?(Y/n/C) : Y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shifts\Java Shifts\Algeo01-23038\test\Input\1c.txt
Matriks harus berukuran n x (n+1)

```

- Metode Matriks Balikan

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Metode matriks balikan
5. Kembali
Pilih Metode: 4
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): D:\java testcase\1d1.txt
filename: D:\java testcase\1d1.txt
x0 = -5.94860851084731
x1 = 31.601757670692926
x2 = -0.24582408427836996
x3 = -0.594123045688349
x4 = -2.2614389775268524
x5 = -49.01696489459181

```

C.

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

- Metode Gauss

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 1
Metode eliminasi Gauss
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shifts\Java Shifts\Algeo01-23038\test\Input\1c.txt
filename: C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shifts\Java Shifts\Algeo01-23038\test\Input\1c.txt
0,0000 1,0000 0,0000 0,0000 1,0000 0,0000 2,0000
0,0000 0,0000 0,0000 1,0000 1,0000 0,0000 -1,0000
0,0000 0,0000 0,0000 0,0000 0,0000 1,0000 1,0000
0,0000 1,0000 0,0000 0,0000 1,0000 0,0000 2,0000
0,0000 0,0000 0,0000 1,0000 1,0000 0,0000 -1,0000
0,0000 0,0000 0,0000 0,0000 -1,0000 1,0000 -1,0000

Matrix akhir:
0,0000 1,0000 0,0000 0,0000 1,0000 0,0000 2,0000
0,0000 0,0000 0,0000 1,0000 1,0000 0,0000 -1,0000
0,0000 0,0000 0,0000 0,0000 -1,0000 1,0000 -1,0000

Ditemukan solusi parametric
x1 = x1
x2 = 2,00 + -1,00 * (1,00 + 1,00x6)
x3 = x3
x4 = -1,00 + -1,00 * (1,00 + 1,00x6)
x5 = 1,00 + 1,00x6
x6 = x6

```

- Metode Gauss Jordan

```

Pilih Menu: 1
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 2
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\1c.txt

Matrix akhir:
 0,0000  1,0000  0,0000  0,0000  0,0000  1,0000  1,0000
 0,0000  0,0000  0,0000  1,0000  0,0000  1,0000 -2,0000
 0,0000  0,0000  0,0000  0,0000  1,0000 -1,0000  1,0000

Ditemukan solusi parametric
x1 = x1
x2 = 1,00 + -1,00x6
x3 = x3
x4 = -2,00 + -1,00x6
x5 = 1,00 + 1,00x6
x6 = x6

```

- Metode Cramer

```

MENU
1. Sistem Persamaan Linier
2. Determinan
3. Matriks Balikan
4. Interpolasi Polinom
5. Regresi Linear dan Kuadratik Berganda
6. Interpolasi Bicubic Spline
7. Interpolasi Gambar
8. Keluar
Pilih Menu: 1
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 3
Ambil variabel dari file?(Y/n/C) : Y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\1c.txt
Matriks harus berukuran n x (n+1)

```

- Metode Matriks Balikan

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Metode matriks balikan
5. Kembali
Pilih Metode: 4
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): D:\java testcase\1c.txt
filename: D:\java testcase\1c.txt
Matrix is singular.

```

d.

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

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

1d.1.

- Metode Gauss

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 1
Metode eliminasi Gauss
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\id1.txt
filename: C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\id1.txt
 1,0000  0,5000  0,3333  0,2500  0,2000  0,1667  1,0000
 0,5000  0,3333  0,2500  0,2000  0,1667  0,1429  0,0000
 0,3333  0,2500  0,2000  0,1667  0,1429  0,1111  0,0000
 0,2500  0,2000  0,1667  0,1429  0,1111  0,0909  0,0000
 0,2000  0,1667  0,1429  0,1111  0,0909  0,0769  0,0000
 0,1667  0,1429  0,1111  0,0909  0,0769  0,0667  0,0000

Matrix akhir:
 1,0000  0,5000  0,3333  0,2500  0,2000  0,1667  1,0000
 0,0000  0,0833  0,0833  0,0750  0,0667  0,0595  -0,5000
 0,0000  0,0000  0,0056  0,0083  0,0095  -0,0040  0,1667
 0,0000  0,0000  0,0000  0,0004  -0,0132  0,0016  -0,0500
 0,0000  0,0000  0,0000  0,0000  -0,5066  0,0629  -1,9379
 0,0000  0,0000  0,0000  -0,0000  0,0000  -0,0061  0,2972

x6 = -49,02
x5 = -2,26
x4 = -0,59
x3 = -0,25
x2 = 31,60
x1 = -5,95
```

- Metode Gauss Jordan

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 2
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\id1.txt
Matrix akhir:
 1,0000  0,0000  0,0000  0,0000  0,0000  -0,2972  8,6196
 0,0000  1,0000  0,0000  0,0000  0,0000  1,2949 -31,8686
 0,0000  0,0000  1,0000  0,0000  0,0000  -0,4508  21,8525
 0,0000  0,0000  0,0000  1,0000  0,0000  -0,0338  1,0617
 0,0000  0,0000  0,0000  0,0000  1,0000  -0,1242  3,8254
 0,0000  0,0000  0,0000  0,0000  0,0000  -0,0061  0,2972

x6 = -49,02
x5 = -2,26
x4 = -0,59
x3 = -0,25
x2 = 31,60
x1 = -5,95
```

- Metode Cramer

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 3
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\id1.txt
Soluksi dari persamaan adalah :
x1 = -5,9486
x2 = 31,6018
x3 = -0,2458
x4 = -0,5941
x5 = -2,2614
x6 = -49,0170
```

- Metode Matriks Balikan

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Metode matriks balikan
5. Kembali
Pilih Metode: 4
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): D:\java testcase\1d1.txt
filename: D:\java testcase\1d1.txt
x0 = -5.94860851084731
x1 = 31.601757670692926
x2 = -0.24582408427836996
x3 = -0.594123045688349
x4 = -2.2614389775268524
x5 = -49.01696489459181

```

1.d.2

- Metode Gauss

```

Matrix akhir:
 1,0000  0,5000  0,3333  0,2500  0,2000  0,1667  0,1429  0,1250  0,1111  0,1000  1,0000
 0,0000  0,0833  0,0833  0,0750  0,0667  0,0595  0,0536  0,0486  0,0444  0,0409  -0,5000
 0,0000  0,0000  0,0056  0,0083  0,0095  0,0099  0,0099  0,0097  0,0094  0,0091  0,1667
 0,0000  0,0000  0,0000  0,0004  0,0007  0,0010  0,0012  0,0013  0,0014  0,0015  -0,0500
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0001  0,0001  0,0001  0,0001  0,0002  0,0146
 0,0000  0,0000  0,0000  0,0000  0,0000  -0,0000  -0,0000  -0,0000  -0,0000  -0,0000  -0,0071
 0,0000  0,0000  0,0000  0,0000  -0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  -0,0064
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  -0,0029
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  -0,0000  -0,0000  0,0153
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  -0,0000  0,0000  0,0000  -0,0019

x10 = -456,36
x9 = -623,84
x8 = 3257,96
x7 = -4622,22
x6 = 5660,15
x5 = -5084,98
x4 = 1983,94
x3 = 20,76
x2 = -159,55
x1 = 19,53

```

- Metode Gauss Jordan

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 2
Ambil variabel dari file?(Y/n/c) : y
Masukkan path ke file (:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-230308\test\Input\1d2.txt

Matrix akhir:
 1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  18,6620
 0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,0315 -145,1822
 0,0000  0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,1276 -37,4920
 0,0000  0,0000  0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,3862 2160,1660
 0,0000  0,0000  0,0000  0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  1,5754 -5803,9415
 0,0000  0,0000  0,0000  0,0000  0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -4,0125 7491,2831
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  5,3896 -7081,8876
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -4,5448 5331,9968
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  2,8769 -1936,7144
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,0019

x10 = -456,36
x9 = -623,84
x8 = 3257,96
x7 = -4622,22
x6 = 5660,15
x5 = -5884,98
x4 = 1983,94
x3 = 20,76
x2 = -159,55
x1 = 19,53

```

- Metode Cramer

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 3
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:\Documents\var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\ld2.txt
Soluksi dari persamaan adalah :
x1 = 19,5256
x2 = -159,5457
x3 = 20,7591
x4 = 1983,9408
x5 = -5084,9767
x6 = 5660,1473
x7 = -4622,2295
x8 = 3257,9602
x9 = -623,8360
x10 = -456,3553
```

- Metode Matriks Balikan

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Metode matriks balikan
5. Kembali
Pilih Metode: 4
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): D:\java testcase\1d2.txt
filename: D:\java testcase\1d2.txt
x0 = 19.52254749029149
x1 = -159.05022053981156
x2 = 31.586082013097606
x3 = 1880.9239613369068
x4 = -4834.984865827541
x5 = 5418.421646335902
x6 = -4416.69884431262
x7 = 3064.034969169111
x8 = -652.815882810619
x9 = -353.4521718384177
```

4.1.1.2. Test Case No 2

SPL berbentuk matriks augmented

a.

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

- Metode Gauss

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 1
Metode eliminasi Gauss
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\2b.txt
filename: C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\2b.txt
 2,0000  0,0000  8,0000  0,0000  8,0000
 0,0000  1,0000  0,0000  4,0000  6,0000
-4,0000  0,0000  6,0000  0,0000  6,0000
 0,0000 -2,0000  0,0000  3,0000 -1,0000
 2,0000  0,0000 -4,0000  0,0000 -4,0000
 0,0000  1,0000  0,0000 -2,0000  0,0000

Matrix akhir:
 2,0000  0,0000  8,0000  0,0000  8,0000
 0,0000  1,0000  0,0000  4,0000  6,0000
 0,0000  0,0000  22,0000  0,0000  22,0000
 0,0000  0,0000  0,0000  11,0000  11,0000
 0,0000  0,0000  0,0000  0,0000  0,0000
 0,0000  0,0000  0,0000 -6,0000 -6,0000

Ditemukan solusi parametric
x1 = 4,00 + -4,00 * (1,00)
x2 = 6,00 + -4,00 * (1,00)
x3 = 1,00
x4 = 1,00
```

- Metode Gauss Jordan

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 2
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\2a.txt
Matrix akhir:
 1,0000  0,0000  0,0000 -1,0000 -1,0000
 0,0000  1,0000 -2,0000  0,0000  0,0000
 0,0000  0,0000  0,0000  0,0000  0,0000
 0,0000  0,0000  0,0000  0,0000  0,0000

Ditemukan solusi parametric
x1 = -1,00 + 1,00x4
x2 = + 2,00x3
x3 = x3
x4 = x4
```

- Metode Cramer

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 3
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\2a.txt
Determinan bernilai nol sehingga tidak ada solusi
```

- Metode Matriks Balikan

```
Pilih Menu: 1
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Metode matriks balikan
5. Kembali
Pilih Metode: 4
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): D:\java testcase\2a.txt
filename: D:\java testcase\2a.txt
Matrix is singular.
```

b.

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

- Metode Gauss

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 1
Metode eliminasi Gauss
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/Var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\2b.txt
filename: C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\2b.txt
 2,0000 0,0000 8,0000 0,0000 8,0000
 0,0000 1,0000 0,0000 4,0000 6,0000
-4,0000 0,0000 6,0000 0,0000 6,0000
 0,0000 -2,0000 0,0000 3,0000 -1,0000
 2,0000 0,0000 -4,0000 0,0000 -4,0000
 0,0000 1,0000 0,0000 -2,0000 0,0000

Matrix akhir:
 2,0000 0,0000 8,0000 0,0000 8,0000
 0,0000 1,0000 0,0000 4,0000 6,0000
 0,0000 0,0000 22,0000 0,0000 22,0000
 0,0000 0,0000 0,0000 11,0000 11,0000
 0,0000 0,0000 0,0000 0,0000 0,0000
 0,0000 0,0000 0,0000 -6,0000 -6,0000

Ditemukan solusi parametric
x1 = 4,00 + -4,00 * (1,00)
x2 = 6,00 + -4,00 * (1,00)
x3 = 1,00
x4 = 1,00

```

- Metode Gauss Jordan

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 2
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/Var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\2b.txt
Matrix akhir:
 1,0000 0,0000 0,0000 0,0000 0,0000
 0,0000 1,0000 0,0000 4,0000 6,0000
 0,0000 0,0000 1,0000 0,0000 1,0000
 0,0000 0,0000 0,0000 11,0000 11,0000
 0,0000 0,0000 0,0000 0,0000 0,0000
 0,0000 0,0000 0,0000 -6,0000 -6,0000

Ditemukan solusi parametric
x1 = 0,00
x2 = 6,00 + -4,00 * (1,00)
x3 = 1,00
x4 = 1,00

```

- Metode Cramer

```

Pilih Menu: 3
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 3
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/Var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\2b.txt
Matriks harus berukuran n x (n+1)

```

- Metode Matriks Balikan

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Metode matriks balikan
5. Kembali
Pilih Metode: 4
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): D:\java testcase\2b.txt
filename: D:\java testcase\2b.txt
Index 4 out of bounds for length 4

```

4.1.1.3. Test Case No 3

SPL Berbentuk

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

- Metode Gauss

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 1
Metode eliminasi Gauss
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\3a.txt
filename: C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\3a.txt
 8,0000  1,0000  3,0000  2,0000  0,0000
 2,0000  9,0000 -1,0000 -2,0000  1,0000
 1,0000  3,0000  2,0000 -1,0000  2,0000
 1,0000  0,0000  6,0000  4,0000  3,0000

Matrix akhir:
 8,0000  1,0000  3,0000  2,0000  0,0000
 0,0000  8,7500 -1,7500 -2,5000  1,0000
 0,0000  0,0000  2,2000 -0,4286  1,6714
 0,0000  0,0000  0,0000  4,8052 -1,2403

x4 = -0,26
x3 = 0,71
x2 = 0,18
x1 = -0,22

```

- Metode Gauss Jordan

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 2
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\3a.txt
Matrix akhir:
 1,0000  0,0000  0,0000  0,3636 -0,3182
 0,0000  1,0000  0,0000 -0,3247  0,2662
 0,0000  0,0000  1,0000 -0,1948  0,7597
 0,0000  0,0000  0,0000  4,8052 -1,2403

x4 = -0,26
x3 = 0,71
x2 = 0,18
x1 = -0,22

```

- Metode Cramer

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 3
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\3a.txt
Soluksi dari persamaan adalah :
X1 = -0,2248
X2 = 0,1824
X3 = 0,7095
X4 = -0,2581

```

- Metode Matriks Balikan

b.

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

- Metode Gauss

```

Matrix akhir:
 1,0000  1,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  8,0000
 0,0000  0,2500  0,9142  0,2500  0,9142  0,2500  0,9142  0,2500  0,0000  14,3100
 0,0000  0,0000  0,0429  0,0000  0,0429  0,7500  0,0429  0,7500  0,6140  14,7900
 0,0000  0,0000  0,0000  1,0000  1,0000  1,0000  0,0000  0,0000  0,0000  15,0000
 0,0000  0,0000  0,0000  0,0000 -1,0000 -16,4866 -1,0000 -17,4866 -13,3148 -326,8356
 0,0000  0,0000  0,0000  0,0000 -0,0000 17,9352  0,6140  18,5492  15,2960  350,3749
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  1,0000  1,0000  1,0000  13,0000
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,8860  0,2114  -4,2578
 0,0000  0,0000  0,0000  0,0000  0,0000 -0,0000  0,0000  0,0000  2,3628  11,8159
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,0000 -2,3628 -11,8159
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  1,0800  5,3903
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,6240 -3,1127

x12 = 0,00
x11 = 0,00
x10 = 0,00
x9 = 5,00
x8 = 6,00
x7 = 2,00
x6 = 9,00
x5 = 5,01
x4 = 1,00
x3 = 4,00
x2 = 1,00
x1 = 3,00

```

- Metode Gauss Jordan

```

Matrix akhir:
 1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,0000  1,0000  8,0041
 0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,9428 -3,7190
 0,0000  0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,0572  3,7149
 0,0000  0,0000  0,0000  1,0000  0,0000  0,0000  0,0000  0,0000  0,1242  1,6173
 0,0000  0,0000  0,0000  0,0000  1,0000  0,0000  0,0000 -0,0000 -1,1814 -0,9024
 0,0000  0,0000  0,0000  0,0000 -0,0000  1,0000  0,0000  0,0000  1,0572  14,2851
 0,0000  0,0000  0,0000  0,0000  0,0000  1,0000  0,0000  0,0000  1,2386  8,1946
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  1,0000  0,0000 -0,2386  4,8054
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  2,3628  11,8159
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,0000 -0,0000 -2,3628 -11,8159
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,0000  1,0800  5,3903
 0,0000  0,0000  0,0000  0,0000  0,0000  0,0000 -0,6240 -3,1127

x12 = 0,00
x11 = 0,00
x10 = 0,00
x9 = 5,00
x8 = 6,00
x7 = 2,00
x6 = 9,00
x5 = 5,01
x4 = 1,00
x3 = 4,00
x2 = 1,00
x1 = 3,00

```

- Metode Cramer

```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Kembali
Pilih Metode: 3
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/var.txt): C:\Users\Lenovo\OneDrive - Institut Teknologi Bandung\College Shits\Java Shits\Algeo01-23038\test\Input\3b.txt
Matriks harus berukuran n x (n+1)

```

- Metode Matriks Balikan

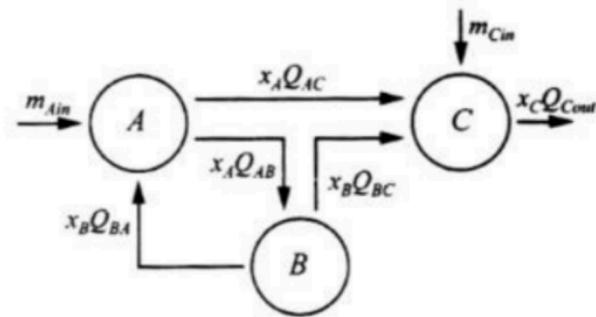
```

1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Kaidah Cramer
4. Metode matriks balikan
5. Kembali
Pilih Metode: 4
Ambil variabel dari file?(Y/n/C) : Y
Masukan path ke file (D:/Documents/var.txt): D:\java testcase\3b.txt
filename: D:\java testcase\3b.txt
Matriks tidak memiliki invers

```

4.1.1.4. Test Case No 4

Lihatlah sistem reaktor pada gambar berikut.



Dengan laju volume Q dalam m^3/s dan input massa min dalam mg/s . Konservasi massa pada tiap inti reaktor adalah sebagai berikut:

$$A: \quad m_{A_{in}} + Q_{BA}x_B - Q_{AB}x_A - Q_{AC}x_A = 0$$

$$B: \quad Q_{AB}x_A - Q_{BA}x_B - Q_{BC}x_B = 0$$

$$C: \quad m_{C_{in}} + Q_{AC}x_A + Q_{BC}x_B - Q_{C_{out}}x_C = 0$$

Tentukan solusi x_A , x_B , x_C dengan menggunakan parameter berikut : $Q_{AB} = 40$, $Q_{AC} = 80$, $Q_{BA} = 60$, $Q_{BC} = 20$ dan $Q_{C_{out}} = 150 \text{ } m^3/s$ dan $m_{A_{in}} = 1300$ dan $m_{C_{in}} = 200 \text{ mg/s}$.

- Metode Gauss

```

Matrix akhir:
-120,0000  60,0000   0,0000 -1300,0000
 0,0000 -60,0000   0,0000 -433,3333
 0,0000   0,0000 -150,0000 -1500,0000

x3 = 10,00
x2 = 7,22
x1 = 14,44
  
```

- Metode Gauss Jordan

```

Matrix akhir:
 1,0000   0,0000  -0,0000  14,4444
 0,0000   1,0000  -0,0000   7,2222
 0,0000   0,0000 -150,0000 -1500,0000

x3 = 10,00
x2 = 7,22
x1 = 14,44
  
```

- Metode Cramer

```
Solusi dari persamaan adalah :  
x1 = 14,4444  
x2 = 7,2222  
x3 = 10,0000
```

- Metode Matriks Balikan

```
1. Metode eliminasi Gauss  
2. Metode eliminasi Gauss-Jordan  
3. Kaidah Cramer  
4. Metode matriks balikan  
5. Kembali  
Pilih Metode: 4  
Ambil variabel dari file?(Y/n/C) : y  
Masukan path ke file (D:/Documents/var.txt): D:\java testcase\4.txt  
filename: D:\java testcase\4.txt  
x0 = 14.444444444444445  
x1 = 7.22222222222222  
x2 = 10.000000000000002
```

4.1.1.5. Test Case No 5

- a. Gunakan tabel di bawah ini untuk mencari polinom interpolasi dari pasangan titik-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

Lakukan pengujian pada nilai-nilai berikut:

$$\begin{array}{ll} x = 0.2 & f(x) = ? \\ x = 0.55 & f(x) = ? \\ x = 0.85 & f(x) = ? \\ x = 1.28 & f(x) = ? \end{array}$$

```
Apakah anda ingin membaca dari file (y/N/c):  
y  
Masukkan jumlah titik:  
7  
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\5a1.txt  
 $f(x) = -0,0230 + 0,2400x^1 + 0,1974x^2 + 0,0000x^3 + 0,0260x^4 + 0,0000x^5 -0,0000x^6$   
 $f(0,2000) = 0.03296093749997216$ 
```

```
Apakah anda ingin membaca dari file (y/N/c):  
y  
Masukkan jumlah titik:  
7  
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\5a2.txt  
 $f(x) = -0,0230 + 0,2400x^1 + 0,1974x^2 + 0,0000x^3 + 0,0260x^4 + 0,0000x^5 -0,0000x^6$   
 $f(0,5500) = 0.1711186523436975$ 
```

```

Pilih Menu: 4
Apakah anda ingin membaca dari file (y/N/c):
y
Masukkan jumlah titik:
7
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\5a3.txt
f(x) = -0,0230 + 0,2400x^1 + 0,1974x^2 + 0,0000x^3 + 0,0260x^4 + 0,0000x^5 -0,0000x^6
f(0,8500) = 0.33723583984367567

```

```

Apakah anda ingin membaca dari file (y/N/c):
y
Masukkan jumlah titik:
7
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\5a4.txt
f(x) = -0,0230 + 0,2400x^1 + 0,1974x^2 + 0,0000x^3 + 0,0260x^4 + 0,0000x^5 -0,0000x^6
f(1,2800) = 0.6775418374998929

```

- b. 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 2022 hingga 31 Agustus 2022:

Tanggal	Tanggal (desimal)	Jumlah Kasus Baru
17/06/2022	6,567	12.624
30/06/2022	7	21.807
08/07/2022	7,258	38.391
14/07/2022	7,451	54.517
17/07/2022	7,548	51.952
26/07/2022	7,839	28.228
05/08/2022	8,161	35.764
15/08/2022	8,484	20.813
22/08/2022	8,709	12.408
31/08/2022	9	10.534

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

$$\text{Tanggal (desimal)} = \text{bulan} + (\text{tanggal} / \text{jumlah hari pada bulan tersebut})$$

Sebagai contoh, untuk tanggal 17/06/2022 (dibaca: 17 Juni 2022) diperoleh tanggal(desimal) sebagai berikut:

$$\text{Tanggal (desimal)} = 6 + (17/30) = 6,567$$

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

- a. 16/07/2022
- b. 10/08/2022
- c. 05/09/2022
- d. Masukan user lainnya berupa **tanggal (desimal) yang sudah diolah**
dengan asumsi prediksi selalu dilakukan untuk tahun 2022.

```
Apakah anda ingin membaca dari file (y/N/c):
Masukkan jumlah titik:
10
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\sb1.txt
F(x) = 7182524565995,5228 -9350472247994,2030x^1 + 5335741780993,0410x^2 -1757283710674,5496x^3 + 368640760119,9879x^4 -51143429348,5744x^5 + 4696794214,5339x^6 -275528781,1378x^7 + 9374584,4194x^8 -141018,3527x^9
F(7,5160) = -1.446900465138086E10

Apakah anda ingin membaca dari file (y/N/c):
Masukkan jumlah titik:
10
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\sb2.txt
F(x) = 7182524565995,5228 -9350472247994,2030x^1 + 5335741780993,0410x^2 -1757283710674,5496x^3 + 368640760119,9879x^4 -51143429348,5744x^5 + 4696794214,5339x^6 -275528781,1378x^7 + 9374584,4194x^8 -141018,3527x^9
F(8,3228) = -1.619903893917969E10

Apakah anda ingin membaca dari file (y/N/c):
Masukkan jumlah titik:
10
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\sb3.txt
F(x) = 7182524565995,5228 -9350472247994,2030x^1 + 5335741780993,0410x^2 -1757283710674,5496x^3 + 368640760119,9879x^4 -51143429348,5744x^5 + 4696794214,5339x^6 -275528781,1378x^7 + 9374584,4194x^8 -141018,3527x^9
F(9,1678) = -1.80990697212015E10

Apakah anda ingin membaca dari file (y/N/c):
Masukkan jumlah titik:
10
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\sb4.txt
F(x) = 7182524565995,5228 -9350472247994,2030x^1 + 5335741780993,0410x^2 -1757283710674,5496x^3 + 368640760119,9879x^4 -51143429348,5744x^5 + 4696794214,5339x^6 -275528781,1378x^7 + 9374584,4194x^8 -141018,3527x^9
F(10,6120) = -2.4518416974375E10
```

- c. Sederhanakan fungsi $f(x)$ yang memenuhi kondisi

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

dengan polinom interpolasi derajat n di dalam selang $[0, 2]$.

Sebagai contoh, jika $n = 5$, maka titik-titik x yang diambil di dalam selang $[0, 2]$ berjarak $h = (2 - 0)/5 = 0.4$.

untuk test case 1 kami menggunakan $n = 5$ dan mendapatkan poin point $(x, f(x))$ berikut didapatkan fungsi berikut:

5c1.txt

File Edit View

```
0 0
0.4 0.418884230
0.8 0.507157968
1.2 0.560924674
1.6 0.583685661
2 0.576651529
1|
```

```
Pilih Menu: 4
Apakah anda ingin membaca dari file (y/N/c):
y
Masukkan jumlah titik:
6
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\5c1.txt
f(x) = + 2,0353x^1 -3,5527x^2 + 3,2371x^3 -1,4213x^4 + 0,2363x^5
```

untuk test case 2 kami menggunakan $n = 10$ dan mendapatkan poin point $(x,f(x))$ berikut didapatkan fungsi berikut:

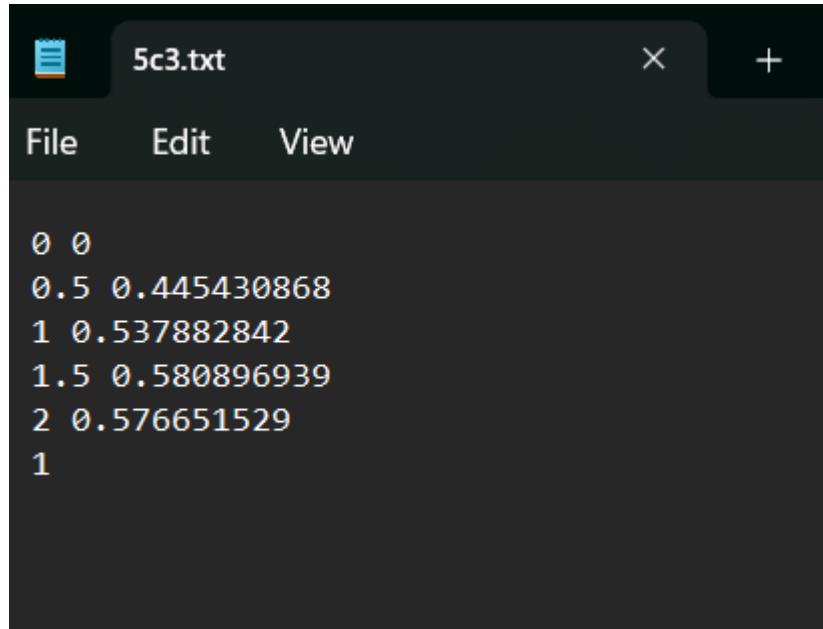
5c2.txt

File Edit View

```
0 0
0.2 0.342769558
0.4 0.418884230
0.6 0.468431469
0.8 0.507157968
1 0.537882842
1.2 0.560924674
1.4 0.576187119
1.6 0.583685661
1.8 0.583674720
2 0.576651529
1|
```

```
Apakah anda ingin membaca dari file (y/N/c):  
y  
Masukkan jumlah titik:  
11  
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\5c2.txt  
 $f(x) = + 3,8816x^1 -18,8128x^2 + 57,3206x^3 -111,5050x^4 + 143,2764x^5 -123,1121x^6 + 69,9619x^7 -25,2211x^8 + 5,2207x^9 -0,4723x^10$ 
```

untuk test case 3 kami menggunakan $n = 4$ dan mendapatkan poin point $(x, f(x))$ berikut didapatkan fungsi berikut:



x	f(x)
0	0
0.5	0.445430868
1	0.537882842
1.5	0.580896939
2	0.576651529

```
Apakah anda ingin membaca dari file (y/N/c):  
y  
Masukkan jumlah titik:  
11  
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\5c2.txt  
 $f(x) = + 3,8816x^1 -18,8128x^2 + 57,3206x^3 -111,5050x^4 + 143,2764x^5 -123,1121x^6 + 69,9619x^7 -25,2211x^8 + 5,2207x^9 -0,4723x^10$ 
```

```
Apakah anda ingin membaca dari file (y/N/c):  
y  
Masukkan jumlah titik:  
5  
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\5c3.txt  
 $f(x) = + 1,5969x^1 -1,8655x^2 + 1,0074x^3 -0,2009x^4$ 
```

4.1.1.6. Test Case No 6

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 76°F, dan tekanan udara sebesar 29.30.

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

$$\begin{aligned}
 20b_0 + 863.1b_1 + 1530.4b_2 + 587.84b_3 &= 19.42 \\
 863.1b_0 + 54876.89b_1 + 67000.09b_2 + 25283.395b_3 &= \\
 779.477 & \\
 1530.4b_0 + 67000.09b_1 + 117912.32b_2 + 44976.867b_3 &= 1483.437 \\
 587.84b_0 + 25283.395b_1 + 44976.867b_2 + 17278.5086b_3 &= \\
 571.1219 &
 \end{aligned}$$

Silahkan terapkan model-model ini pada *Multiple Quadratic Equation* juga dan bandingkan hasilnya. Sistem persamaan linear tidak akan diberikan untuk kasus ini.

- Metode Multiple Linear Regression

```
1. Kuadratik Berganda
2. Linear Berganda
3. Kembali
Pilih Metode: 1
Masukan jumlah sampel dan variabel dalam satu baris dengan spasi:
20 4
Ambil variabel dari file?(Y/n/C) : y
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\6.txt

Matrix akhir:
 20,0000  0,0000  0,0000  0,0000  0,0000
  0,0000  0,0000  0,0000  0,0000  0,0000
  0,0000  0,0000  0,0000  0,0000  0,0000
  0,0000  0,0000  0,0000  0,0000  0,0000

Ditemukan solusi parametric
x1 = NaN
x2 = NaN
x3 = NaN
x4 = NaN
Hasil Prediksi Model : NaN
Save Hasil ke file?(Y/n)
n
File tidak disimpan.
```

- Metode Multiple Quadratic Equation

```
1. Kuadratik Berganda
2. Linear Berganda
3. Kembali
Pilih Metode: 2
Masukan jumlah sampel dan variabel dalam satu baris dengan spasi:
20 4
Ambil variabel dari file?(Y/n) : y
Masukan path ke file (D:/Documents/regresi.txt): D:\java testcase\6.txt

Matrix akhir:
 20,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000
   0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000
   0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000
   0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000

Ditemukan solusi parametric
 20,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000
   0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000
   0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000
   0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000  0,0000
```

4.1.1.7. Test Case No 7

Diberikan matriks input dengan bentuk sebagai berikut. Format matriks masukan bukan mewakili nilai matriks, tetapi mengikuti format masukan pada bagian "Spesifikasi Tugas" nomor 7.

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

Tentukan nilai:

$$f(0, 0) = ?$$

$$f(0.5, 0.5) = ?$$

$$f(0.25, 0.75) = ?$$

$$f(0.1, 0.9) = ?$$

```
Masukan Jumlah Baris : 4
Masukan Jumlah Kolom : 4
Masukan Matrix sesuai format
21 98 125 153
51 101 161 59
0 42 72 210
16 12 81 96
0 0
Hasil Interpolasi:
(0.00, 0.00) is 21.000000
```

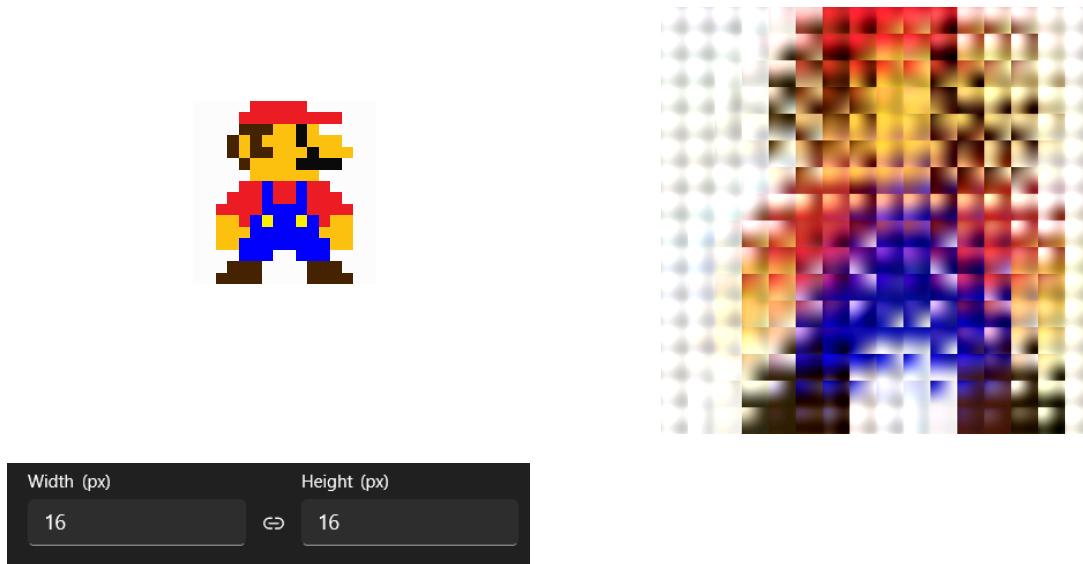
```
Masukan Jumlah Baris : 4
Masukan Jumlah Kolom : 4
Masukan Matrix sesuai format
21 98 125 153
51 101 161 59
0 42 72 210
16 12 81 96
0.5 0.5
Hasil Interpolasi:
(0.50, 0.50) is 87.796875
```

```
Masukan Jumlah Baris : 4
Masukan Jumlah Kolom : 4
Masukan Matrix sesuai format
21 98 125 153
51 101 161 59
0 42 72 210
16 12 81 96
0.25 0.75
Hasil Interpolasi:
(0.25, 0.75) is 117.732178
```

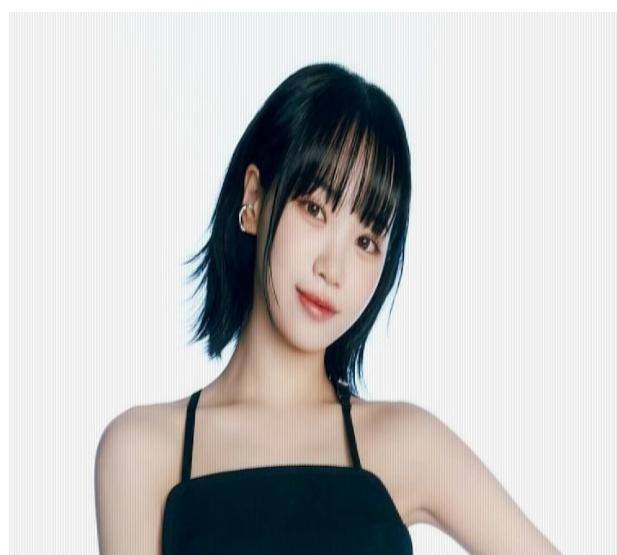
```
Masukan Jumlah Baris : 4
Masukan Jumlah Kolom : 4
Masukan Matrix sesuai format
21 98 125 153
51 101 161 59
0 42 72 210
16 12 81 96
0.1 0.9
Hasil Interpolasi:
(0.10, 0.90) is 128.575187
```

4.1.1.8. Test Case Bonus

Pada eksperimen ini sebuah gambar dengan nama mariotes.jpeg akan dibesarkan dengan skala 100. Dimensi awal marioot.png adalah 16x16 (lebar 16 px dan tinggi 16px). Dengan perbesaran 100. kali hasil akhir adalah 1600x1600. Berikut ini adalah proses perbesaran gambar.



Lalu kamu melakukan Stretching terhadap gambar yang di beri nama chaewon.jpg dengan skala 1.5 pada width dan skala 2 pada height.



BAB V

Kesimpulan dan Saran

5.1. Kesimpulan

Telah dibuat program yang dapat menyelesaikan Sistem persamaan linear dengan menggunakan metode Gauss, Gauss-Jordan, hingga menggunakan kaidah Cramer. Selain itu, program juga dapat melakukan operasi matriks seperti mencari determinan dan balikan dengan menggunakan berbagai metode. Program juga dapat menggunakan library yang berisi metode-metode dalam penyelesaian masalah SPL dan menggunakannya untuk menyelesaikan masalah Interpolasi Polinomial, Regresi ganda, dan interpolasi bikubik. Selain itu, program juga dapat memperbesar skala sebuah gambar dengan menggunakan interpolasi bikubik.

5.2. Saran

1. Komunikasi antar anggota tentang fungsi fungsi harus diperbaiki agar tidak perlu membuat dua fungsi atau lebih yang sama.
2. Pengerajan harus dilakukan secara bersih dan rapi agar mudah dipahami, jika perlu bisa ditambahkan comment.
3. Nama-nama fungsi harus sesuai dengan maksud tujuan dari fungsi tersebut

5.3. Refleksi

Dari tugas ini, kami menyadari bahwa kerja sama tim yang baik dan komunikasi yang efektif merupakan kunci utama dalam pengerajan proyek pemrograman. Koordinasi yang kurang dapat menyebabkan duplikasi fungsi dan ketidaksesuaian tujuan antar bagian program, sehingga setiap anggota tim harus memahami perannya dengan jelas. Selain itu, penamaan fungsi harus menggambarkan dengan tepat tujuan dari fungsi tersebut agar kode mudah dipahami dan di-maintain. Struktur kode yang rapi, terorganisir, serta adanya dokumentasi atau komentar yang jelas juga sangat penting untuk memudahkan proses debugging, pengembangan, dan kolaborasi antar anggota tim.

Daftar Pustaka

Tim Asisten IF2123 Aljabar Linier dan Geometri (2 Oktober 2024) Tugas Besar IF2123 Aljabar Linier dan Geometri Sistem Persamaan Linier, Determinan, dan Aplikasinya Semester I Tahun 2024/2025. Diakses pada 3 Oktober 2024, dari https://docs.google.com/document/d/1_VVhIat1qDTyzMbPL2if3Rct3LGRefEYWxBFMCOeeUM/edit?tab=t.0

Mssc.mu.edu (15 Februari 2018). BiLinear, Bicubic, and In Between Spline Interpolation. Diakses pada 10 Oktober 2024, dari https://www.mssc.mu.edu/~daniel/pubs/RoweTalkMSCS_BiCubic.pdf

Lampiran

Hasil Asistensi

8 Oktober 2024, Asistensi dilakukan Via Zoom pada malam hari jam 8. Berikut hasil yang didapat :

1. Pembagian Tugas: Pastikan pembagian tugas merata, dan setiap anggota tim mendapatkan tanggung jawab lebih dari sekadar pembuatan dokumen.
2. Laporan Fungsi: Setiap anggota tim harus menyertakan laporan mengenai fungsi yang mereka kerjakan di dalam laporan akhir.
3. Pertimbangan Beban Kerja: Perhitungan determinan memerlukan usaha yang cukup signifikan, sehingga penting untuk mempertimbangkan beban kerja tiap anggota sesuai dengan kompleksitas fungsi yang diberikan.
4. Prioritas Fungsi: Beberapa fungsi akan digunakan kembali (reused), oleh karena itu, tentukan prioritas fungsi mana yang harus diselesaikan terlebih dahulu.
5. Pengerjaan Bonus: Jika berencana mengerjakan bonus, sebaiknya tidak menunggu sampai akhir. Lebih baik mulai mencicilnya dari sekarang.
6. Keseriusan dalam Pengerjaan: Terdapat kecenderungan beberapa anggota menyepelekan tugasnya. Pastikan setiap tugas dikerjakan dengan serius dan penuh tanggung jawab.
7. Kesesuaian README dengan Program: README yang tidak sesuai dengan implementasi program dapat mengurangi nilai. Pastikan README selalu selaras dengan program yang dibuat.
8. Persiapan untuk Demo: Saat demo, pertanyaan akan berfokus pada detail program. Sebaiknya, setiap anggota tim memahami secara menyeluruh program yang telah dikerjakan sebelum demo berlangsung.
9. Kesiapan Program saat Demo: Sebelum demo, pastikan program sudah berjalan dengan baik. Hindari melakukan debugging selama demo berlangsung.
10. Clean Code: Tulislah kode yang bersih (clean code)

<https://github.com/Aramazaya/Algeo01-23038>