Tugas Besar 1 IF2123 Aljabar Linear dan Geometri

Sistem Persamaan Linear, Determinan, dan Aplikasinya

Semester I Tahun 2022/2023

13521055	Muhammad Bangkit Dwi Cahyono
13521081	Bagas Aryo Seto
13521148	Johanes Lee

Kelompok “aezakmi”

Bab I
Deskripsi Masalah

Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. Anda sudah mempelajari berbagai metode untuk menyelesaikan SPL, termasuk menghitung determinan matriks. Sembarang SPL dapat diselesaikan dengan beberapa metode, yaitu metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan ($x = A^{-1}b$), 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).

Sistem Persamaan Linier memiliki dua bentuk umum berupa persamaan vektor dan persamaan matriks. Untuk Sistem Persamaan Linier m dan n dalam bentuk vektor dapat ditulis seperti:

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

Dalam bentuk matriks dapat ditulis sebagai berikut.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Beberapa metode dapat digunakan untuk menyelesaikan sistem persamaan linier, antara lain seperti metode eliminasi variabel, eliminasi Gauss, eliminasi Gauss-Jordan, dan kaidah Cramer. Metode-metode tersebut dapat digunakan untuk menyelesaikan SPL dalam bentuk matriks. Dengan menentukan determinan, matriks kofaktor, adjoin, dan matriks balikan, solusi SPL dapat ditentukan.

Pengaplikasian metode-metode tersebut memiliki langkah-langkah penyelesaian yang berulang dan berpola. Oleh karena itu, dapat dibuat suatu program untuk menyelesaikan SPL berbentuk matriks dengan berbagai metode dan prosedur. Dengan menggunakan bahasa pemrograman Java, program tersebut dapat direalisasikan.

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

Bab II
Teori Singkat

2.1 Metode Eliminasi Gauss

Sistem persamaan linear (SPL) adalah sekumpulan persamaan linear yang terdiri dari beberapa variabel. SPL dengan m buah persamaan dan n variabel x_1, x_2, …, x_n memiliki bentuk umum:

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

Terdapat 3 macam langkah yang dapat dilakukan untuk mencari penyelesaian suatu sistem persamaan linear dengan metode eliminasi dan substitusi. Langkah-langkah tersebut yaitu:

a. Menukarkan letak dua buah persamaan
b. Mengalikan suatu persamaan dengan suatu konstanta tidak nol
c. Menambahkan suatu persamaan dengan kelipatan persamaan lainnya

SPL juga dapat dinyatakan secara ringkas dalam matriks yang diperluas atau matriks augmented yang berbentuk:

$$[A \mid b] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & \mid b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & \mid b_2 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & \mid b_m \end{bmatrix}$$

Dalam bentuk matriks augmented, langkah-langkah untuk penyelesaian sistem persamaan linear disebut dengan operasi baris elementer (OBE), yaitu:

a. Menukarkan letak dua buah baris

$$\begin{bmatrix} 0 & 0 & -2 & 0 & 7 & 12 \\ 2 & 4 & -10 & 6 & 12 & 28 \\ 2 & 4 & -5 & 6 & -5 & -1 \end{bmatrix} \overset{R1 \leftrightarrow R2}{\sim} \begin{bmatrix} 2 & 4 & -10 & 6 & 12 & 28 \\ 0 & 0 & -2 & 0 & 7 & 12 \\ 2 & 4 & -5 & 6 & -5 & -1 \end{bmatrix}$$

Contoh: baris R1 ditukarkan dengan baris R2.

b. Mengalikan suatu baris dengan suatu konstanta tidak nol

$$\begin{bmatrix} 2 & 4 & -10 & 6 & 12 & 28 \\ 0 & 0 & -2 & 0 & 7 & 12 \\ 2 & 4 & -5 & 6 & -5 & -1 \end{bmatrix} \overset{R1/2}{\sim} \begin{bmatrix} 1 & 2 & -5 & 3 & 6 & 14 \\ 0 & 0 & -2 & 0 & 7 & 12 \\ 2 & 4 & -5 & 6 & -5 & -1 \end{bmatrix}$$

Contoh: baris R1 dikalikan dengan konstanta ½.

c. Menambahkan suatu baris dengan kelipatan baris lainnya

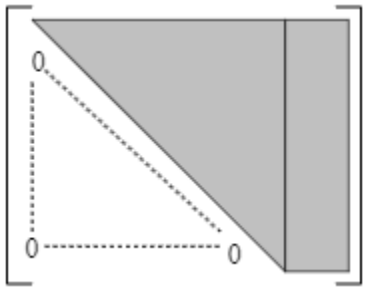
$$\begin{bmatrix} 1 & 2 & -5 & 3 & 6 & 14 \\ 0 & 0 & -2 & 0 & 7 & 12 \\ 2 & 4 & -5 & 6 & -5 & -1 \end{bmatrix} \overset{R3 - 2R1}{\sim} \begin{bmatrix} 1 & 2 & -5 & 3 & 6 & 14 \\ 0 & 0 & -2 & 0 & 7 & 12 \\ 0 & 0 & 5 & 0 & -17 & -29 \end{bmatrix}$$

Contoh: baris R3 ditambahkan dengan kelipatan negatif dua dari baris R1.

Operasi baris elementer dilakukan berulang-ulang hingga matriks augmented berubah menjadi matriks dengan bentuk yang lebih sederhana, yaitu matriks eselon baris. Proses menghasilkan matriks eselon baris ini disebut metode eliminasi Gauss. Metode ini dinamai dari matematikawan kelahiran Jerman bernama Carl Friedrich Gauss (1777 - 1855) yang menyempurnakan metode ini dari metode yang sudah dikenal sejak tahun 179 M oleh matematikawan Tionghoa.

Matriks augmented yang sudah menjadi matriks eselon baris kemudian akan diproses lagi menggunakan substitusi secara mundur untuk memperoleh nilai-nilai variabelnya. Terdapat tiga kemungkinan solusi setelah dilakukan metode eliminasi Gauss terhadap matriks augmented, yaitu:

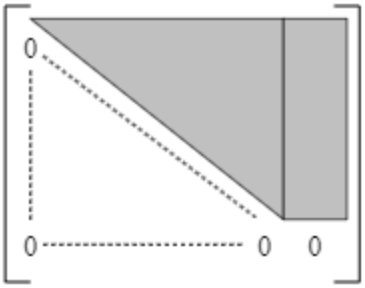
a. Solusi unik



(Bentuk matriks eselon dengan solusi unik)

Setiap variabel memiliki solusi unik jika terbentuk matriks segitiga atas dengan semua elemen diagonal utamanya bernilai 1.

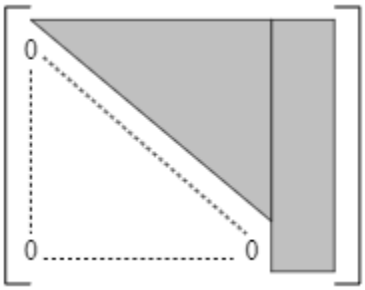
b. Solusi banyak (parametrik)



(Bentuk matriks eselon dengan solusi banyak)

Terdapat lebih dari satu penyelesaian untuk matriks augmented jika terdapat satu atau lebih baris yang seluruh elemennya bernilai 0.

c. Tidak ada solusi



(Bentuk matriks eselon tanpa solusi)

Matriks tidak memiliki penyelesaian jika terdapat baris yang seluruh elemennya bernilai 0 kecuali elemen terakhir baris tersebut.

2.2 Metode Eliminasi Gauss-Jordan

Metode eliminasi Gauss-Jordan merupakan pengembangan dari metode eliminasi Gauss yang hasilnya disederhanakan lagi menjadi matriks eselon baris tereduksi. Metode ini merupakan hasil modifikasi seorang insinyur Jerman bernama Wilhelm Jordan pada tahun 1887. Pada metode eliminasi Gauss-Jordan, tidak diperlukan lagi substitusi mundur untuk memperoleh nilai-nilai variabelnya. Nilai-nilai variabel bisa langsung didapatkan dari matriks augmented terakhirnya.

Metode eliminasi Gauss-Jordan terdiri dari dua fase, yaitu:

a. Fase maju (*forward phase*) atau fase eliminasi Gauss

$$\begin{bmatrix} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{bmatrix} \sim \overset{\text{OBE}}{\cdots} \sim \begin{bmatrix} 1 & 3/2 & -1/2 & 5/2 \\ 0 & 1 & 1/2 & 7/2 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

Fase maju menghasilkan matriks eselon baris.

b. Fase mundur (*backward phase*)

$$\begin{bmatrix} 1 & 0 & -5/4 & -11/4 \\ 0 & 1 & 1/2 & 7/2 \\ 0 & 0 & 1 & 3 \end{bmatrix} \underset{\text{R2 - (1/2)R3}}{\overset{\text{R1 +(5/4)R3}}{\sim}} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

Fase mundur menghasilkan matriks eselon baris tereduksi.

2.3 Determinan Matriks

Determinan adalah nilai yang dapat dihitung dari unsur-unsur suatu matriks persegi. Matriks persegi itu sendiri adalah matriks yang memiliki jumlah baris dan kolom yang sama. Dalam menghitung determinan sebuah matriks, ada dua metode utama yang biasanya digunakan. Metode yang pertama adalah metode ekspansi kofaktor. Metode ini memiliki definisi yaitu:

Jika A adalah sebuah matriks persegi, maka minor entri *aij* dinyatakan oleh *Mij* dan didefinisikan menjadi determinan submatriks yang tetap setelah baris ke *i* dan kolom ke *j* dicoret dari A. Bilangan (-1) *i*+*jMij* dinyatakan sebagai *Cij* dan dinamakan kofaktor entri *aij*. Contoh dari minor entri dan kofaktor adalah sebagai berikut.

Misal diketahui matriks dibawah ini. Tentukan minor entri dan kofaktor dari a_{11}.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 2 \\ 3 & 2 & 0 \end{bmatrix}$$

$$M_{11} = \begin{vmatrix} 5 & 2 \\ 2 & 0 \end{vmatrix} = -4$$

$$C_{11} = (-1)^{(1+1)} M_{11} = -4$$

Itu adalah kofaktor dari suatu matriks.

Untuk mempermudah mengingat tanda positif negatif pada kofaktor, cukup melihat tabel dibawah ini.

$$\begin{bmatrix} + & - & + & - & + & \cdots \\ - & + & - & + & - & \cdots \\ + & - & + & - & + & \cdots \\ - & + & - & + & - & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix}$$

Salah satu contoh perhitungan determinan dengan ekspansi kofaktor adalah sebagai berikut.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 2 \\ 3 & 2 & 0 \end{bmatrix}$$

$$\det(A) = 1\begin{vmatrix} 5 & 2 \\ 2 & 0 \end{vmatrix} - 2\begin{vmatrix} 4 & 2 \\ 3 & 0 \end{vmatrix} + 3\begin{vmatrix} 4 & 5 \\ 3 & 2 \end{vmatrix}$$

$$\det(A) = -4 + 12 - 21 = -13$$

Metode kedua adalah metode reduksi baris. Determinan dicari dengan menggunakan Operasi Baris Elementer untuk mendapatkan matriks segitiga atas ataupun matriks segitiga bawah.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \implies \det(A) = a_{11} \,.\, a_{22} \,.\, a_{33} \,.\, a_{44}$$

Fokus utama dari metode ini adalah menggunakan OBE untuk membuat sebuah matriks menjadi matriks segitiga atas atau matriks segitiga bawah.

$$[A] \sim [segitiga\ atas\ atau\ bawah]$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1k} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2k} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & a_{k3} & \cdots & a_{kk} \end{bmatrix} \sim \begin{bmatrix} a'_{11} & a'_{12} & a'_{13} & \cdots & a'_{1k} \\ 0 & a'_{22} & a'_{23} & \cdots & a'_{2k} \\ 0 & 0 & a_{33} & \cdots & a'_{3k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a'_{kk} \end{bmatrix}$$

$$\det(A) = (-1)^p a'_{11} a'_{22} a'_{33} \ldots a'_{kk}$$

2.4 Matriks Balikan

Balikan suatu matriks A dapat didefinisikan sebagai berikut.

Jika *A* merupakan matriks persegi dan dapat ditemukan matriks *B* dengan ukuran yang sama sehingga *AB* = *BA* = *I*, maka A dikatakan *invertible* atau *nonsingular* dan matriks *B* disebut matriks balikan dari *A*. Jika tidak dapat ditemukan matriks *B* tersebut, maka matriks *A* dikatakan *singular* (Howard Anton, 2010:43).

Suatu matriks hanya memiliki satu buah matriks balikan. Jika matriks *B* dan *C* keduanya merupakan matriks balikan dari suatu matriks *A*, maka *B* = *C*. Dengan demikian, matriks balikan dari suatu matriks *A* dapat ditulis *A^(-1)*. Dengan demikian, *AA^(-1) = A^(-1)A = I*.

Suatu matriks *E* dikatakan matriks elementer jika matriks tersebut dapat diperoleh dari matriks identitas dengan melakukan satu buah operasi baris elementer. Jika matriks elementer *E* diperoleh dari melakukan satu buah operasi baris elementer pada *Im* dan *A* adalah matriks berukuran *m x n*, maka hasil perkalian dot *EA* adalah matriks yang sama yang dihasilkan jika operasi baris elementer tersebut dilakukan ke matriks *A*. Suatu matriks *A* memiliki matriks balikan jika dan hanya jika matriks tersebut dapat dinyatakan sebagai produk matriks-matriks elementer. (Howard Anton, 2010:52).

$$E_k \cdots E_2 E_1 A = I_n$$

$$A = E_1^{-1} E_2^{-1} \cdots E_k^{-1} I_n = E_1^{-1} E_2^{-1} \cdots E_k^{-1}$$

$$A^{-1} = E_k \cdots E_2 E_1 I_n$$

Gambar [No Gambar]. Matriks A dalam Bentuk Produk Matriks-Matriks Elementer
(Howard Anton, 2010:52)

Berdasarkan hal-hal di atas, matriks balikan suatu matriks *A* dapat diperoleh dengan pertama melakukan deretan operasi-operasi baris elementer pada matriks tersebut hingga diperoleh matriks identitas. Hal ini dapat dilakukan dengan metode Gauss-Jordan. Setelah memperoleh matriks identitas, matriks balikan *A* diperoleh dengan melakukan

deretan operasi yang sama pada matriks identitas tersebut. Jika metode Gauss-Jordan gagal membentuk matriks identitas, maka bentuk matriks baris echelon tereduksi dari matriks *A* bukan matriks identitas sehingga matriks *A* tidak memiliki matriks balikan.

2.5 Matriks Kofaktor

Matriks kofaktor adalah matriks yang terbentuk dari kofaktor kofaktor matriks. Susunan elemen pada matriks kofaktor juga sesuai dengan susunan pada matriksnya. Secara umum matriks kofaktor dapat ditulis sebagai berikut.

$$\begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{n1} & C_{n2} & \dots & C_{nn} \end{bmatrix}$$

2.6 Matriks Adjoint

Jika terdapat suatu matriks persegi yang memiliki suatu matriks kofaktor, maka transpos dari matriks kofaktor tersebut adalah matriks adjoint dari A.

2.7 Kaidah Cramer

Kaidah Cramer adalah rumus yang dapat digunakan untuk menyelesaikan sistem persamaan linear dengan banyak persamaan sama dengan banyak variabel, dan berlaku ketika sistem tersebut memiliki solusi yang tunggal. Rumus ini menyatakan solusi dengan menggunakan determinan matriks koefisien (dari sistem persamaan) dan determinan matriks lain yang diperoleh dengan mengganti salah satu kolom matriks koefisien dengan vektor yang berada sebelah kanan persamaan. Misal $Ax = b$ adalah SPL yang terdiri dari i persamaan linear dengan i variabel. Jika $det(A) \neq 0$, maka setiap variabel memiliki solusi unik yaitu.

$$x_1 = \frac{\det(A_1)}{\det(A)},\ x2 = \frac{\det(A_2)}{\det(A)},\ x_3 = \frac{\det(A_3)}{\det(A)},\ x_i = \frac{\det(A_i)}{\det(A)}$$

Matriks A_i didapatkan dengan mengubah entri pada kolom ke-*i* dengan matriks *b*.

2.8 Interpolasi Polinomial

Suatu polinomial interpolasi adalah polinomial yang grafiknya melewati himpunan titik yang spesifik pada suatu bidang. Misalnya terdapat *n* buah titik

$$(x_1, y_1), \quad (x_2, y_2), \quad (x_3, y_3), \ldots, \quad (x_n, y_n)$$

Dari titik-titik tersebut, dapat diperoleh polinomial dalam bentuk

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

karena polinomial tersebut memiliki *n* buah koefisien untuk memenuhi kondisi *n* buah titik tersebut. Namun, polinomial yang diperoleh dapat memiliki derajat yang lebih rendah akibat beberapa koefisien dapat bernilai nol. Berikut adalah teorema interpolasi polinomial.

Given any n points in the xy-plane that have distinct x-coordinates, there is a unique polynomial of degree $n - 1$ or less whose graph passes through those points.

Gambar [No Gambar]. Teorema Interpolasi Polinomial (Howard Anton, 2010:92)

2.9 Interpolasi Bikubik

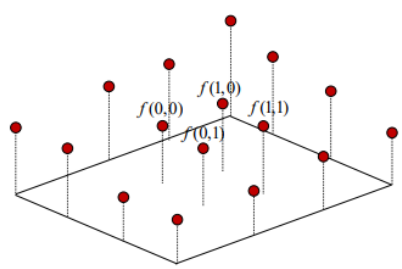
Bicubic interpolation merupakan teknik interpolasi pada data 2D umumnya digunakan dalam pembesaran citra yang merupakan pengembangan dari interpolasi linear dan cubic yang telah dipelajari pada kuliah metode numerik di aljabar geometri.

Diberikan sebuah matrix awal, misal M, kita akan mencari persamaan interpolasi f(x,y) dengan pemodelan sebagai berikut:

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

Model: $f(x,y) = \sum_{j=0}^{3} \sum_{i=0}^{3} a_{ij} x^i y^j$
$x = -1, 0, 1, 2$

Solve: a_{ij}



Setelah itu, melakukan substitusi nilai-nilai diketahui pada matriks 4 x 4 tersebut ke persamaan f(x,y) akan menghasilkan sebuah matriks persamaan:

Values from polynomial.

$$y = Xa$$

$$\begin{bmatrix} f(-1,-1) \\ f(0,-1) \\ f(1,-1) \\ f(2,-1) \\ f(-1,0) \\ f(0,0) \\ f(1,0) \\ f(2,0) \\ f(-1,1) \\ f(0,1) \\ f(1,1) \\ f(2,1) \\ f(-1,2) \\ f(0,2) \\ f(1,2) \\ f(2,2) \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 2 & 4 & 8 & -1 & -2 & -4 & -8 & 1 & 2 & 4 & 8 & -1 & -2 & -4 & -8 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 4 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 & 1 & 2 & 4 & 8 \\ 1 & -1 & 1 & -1 & 2 & -2 & 2 & -2 & 4 & -4 & 4 & -4 & 8 & -8 & 8 & -8 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 8 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 4 & 4 & 4 & 4 & 8 & 8 & 8 & 8 \\ 1 & 2 & 4 & 8 & 2 & 4 & 8 & 16 & 4 & 8 & 16 & 32 & 8 & 16 & 32 & 64 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \\ a_{01} \\ a_{11} \\ a_{21} \\ a_{31} \\ a_{02} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{03} \\ a_{13} \\ a_{23} \\ a_{33} \end{bmatrix}$$

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

x,y=-1,0,1,2

Setelah mendapatkan 16 persamaan, dicari nilai-nilai dari *a*, bisa dengan memanfaatkan inverse matrix. Interpolasi dilakukan untuk mendapatkan nilai f(a, b) untuk a dan b dalam rentang [0, 1] (Daerah di dalam kotak dengan titik sudut berwarna merah pada gambar di bawah ini).

System of Equations: 16 equations, 16 unknowns

System of Equations $y = Xa$

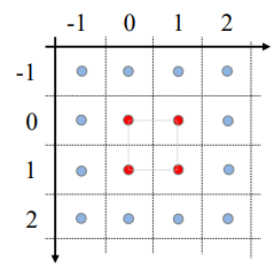
16 equations from $f(x,y)$

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

x,y=-1,0,1,2

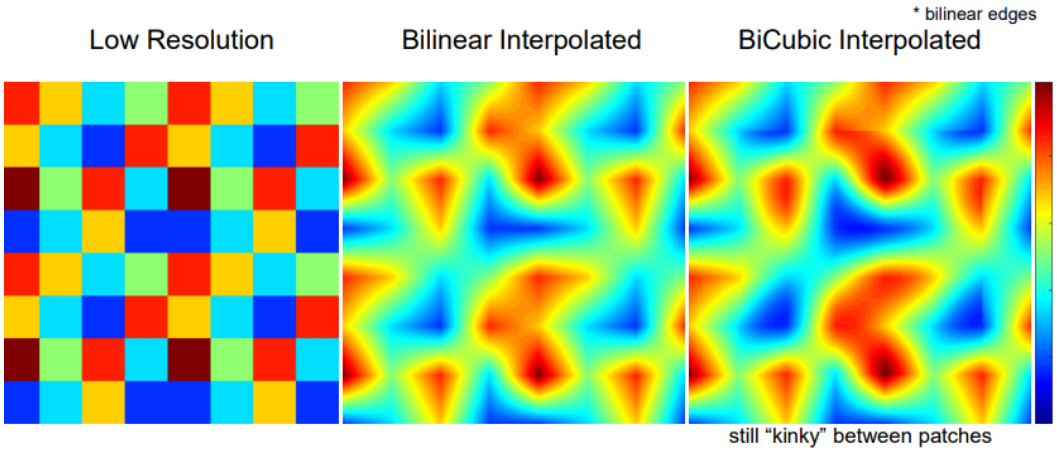
Simply insert all x,y combinations to get 16 equations.

Image $I(x,y)$



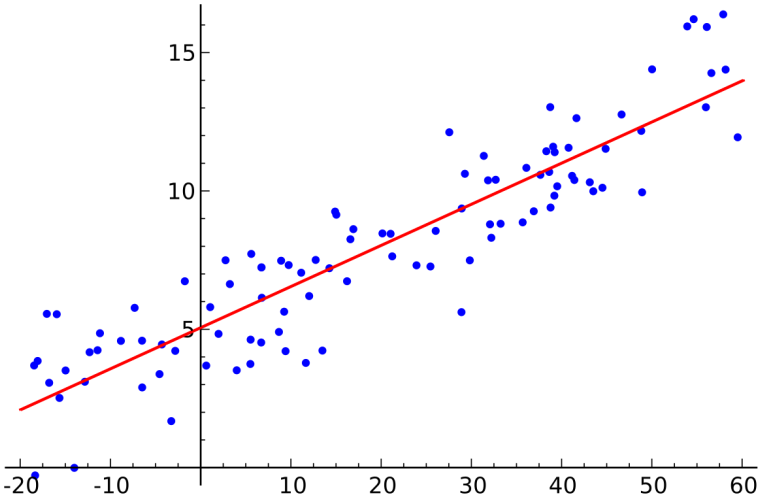
Interpolasi bikubik ini dapat dimanfaatkan untuk *scaling image* (memperbesar gambar) dengan memanfaatkan pixel pada gambar, kemudan memetakan pada setiap pixel berukuran 4 x 4 untuk menyisipkan pixel baru di antara pixel-pixel pada gambar dengan memanfaatkan f(x, y) yang sebelumnya sudah dihitung pada persamaan bikubik. Sebagai contoh, menyisipkan pixel f(0, 0.5) di antara pixel (0, 0) dan (0, 1), dan seterusnya.

Berikut adalah contoh perbesaran 8 kali pada suatu gambar.



2.10 Regresi Linear Berganda

Analisis regresi digunakan untuk meneliti hubungan antar dua atau lebih variabel, dengan salah satu variabelnya merupakan variabel respon (dependen) dan variabel lainnya merupakan variabel prediktor (independen). Regresi linear berganda adalah model regresi linear dengan melibatkan lebih dari satu variabel prediktor. Sama dengan interpolasi polinom, regresi linear berganda juga bertujuan untuk memprediksi nilai suatu variabel. Dibandingkan dengan fungsi polinom yang mengandung semua titik sampel, regresi linear berganda lebih melihat tren perubahan nilai titik secara garis lurus atau linear. Hal tersebut menyebabkan galat dari nilai regresi linear berganda akan lebih besar dari interpolasi polinom apabila titik-titik sampel yang dimasukkan berjarak cukup jauh dari tren.



(https://simple.wikipedia.org/wiki/File:Linear_regression.svg Contoh Regresi Linear)

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 Equation for Multiple Linear Regression* sebagai berikut:

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

Sistem persamaan linear tersebut dapat diselesaikan dengan menggunakan metode eliminasi Gauss.

Bab III
Implementasi Program

3.1 Class IO

Methods

```
public static File[] getListDir(String relativePath) {
      // I.S. relativePath merupakan path yang valid
      // F.S. mengembalikan semua list file yang ada
}

public static void outputListDir(String relativePath) {
      // I.S. relativePath merupakan path yang valid
      // F.S. menampilkan list file yang ada
}

public static int readRow(String fileName, String relativePath) {
      // I.S. fileName dan relativePath valid
      // F.S. mengembalikan banyaknya baris matrix di file
}

public static int readCol(String fileName, String relativePath) {
      // I.S. fileName dan relativePath valid
      // F.S. mengembalikan banyaknya kolom matrix di file
}

public static Matrix readMatrix(String fileName,String relativePath) {
      // I.S. fileName dan relativePath valid
      // F.S. mengembalikan matrix keseluruhan yang ada dalam file
}

public static File getFile(String relativePath,String
defaultExtension) {
      // mengembalikan File yang berada pada relativePath berdasarkan
      // input nama file dari konsol
}

public static String getFileName(String relativePath) {
      // mengembalikan nama file (baru ataupun lama) yang valid dan
      // terletak pada relativePath
}

public static String getExtension(String fileName, String
defaultExtension) {
      // mengembalikan nama file baru yang memiliki ekstensi default
      // jika fileName tidak memiliki penamaan ekstensi file
}

public static void writeFileMatrix(Matrix m, String fileName, String
relativePath) {
      // I.S. m, fileNameOutput, dan relativePath terdefinisi
      // F.S. menulis matrix m ke dalam folder fileNameOutput
```

```
}

public static void writeFileString(String s, String fileNameOutput,
String relativePath) {
      // I.S. s, fileNameOutput, dan relativePath terdefinisi
      // F.S. menulis string s ke dalam folder fileNameOutput
}

public static String inputNewFileName(String relativePath, String
defaultExtension) {
      // I.S. relativePath dan defaultExtension terdefinisi
      // F.S. mengembalikan nama file baru yang ingin disimpan
}
```

3.2 Class Matrix

Attribute

Nama	Tipe	Deskripsi
mem	private double[][]	Elemen Matrix
ROW_EFF	private int	Jumlah baris efektif
COL_EFF	private int	Jumlah kolom efektif

Methods

```
public Matrix(int row, int col) {
      // I.S. atribut matriks sembarang
      // F.S. atribut ROW_EFF dan COL_EFF terdefinisi dengan nilai row
      // dan col dan matriks mem kosong dengan ukuran sesuai dengan
      // ROW_EFF dan COL_EFF
}

public static Matrix(Matrix mIn) {
      // I.S. atribut matriks sembarang
      // F.S. semua atribut matriks sama dengan atribut mIn
}

public static Matrix() {
      // I.S. atribut matriks sembarang
      // F.S. ROW_EFF dan COL_EFF bernilai nol serta ukuran mem nol
}

private void InitializeMatrix() {
      // I.S. atribut mem tidak terdefinisi
      // F.S. atribut mem merupakan matriks kosong dengan ukuran
      // ROW_EFF dan COL_EFF
}

public void setMatrix (int row, int col){
      // I.S. atribut matriks sembarang
```

```
      // F.S. atribut ROW_EFF dan COL_EFF terdefinisi dengan nilai row
      // dan col dan matriks mem kosong dengan ukuran sesuai dengan
      // ROW_EFF dan COL_EFF

}

public double getELMT(int row, int col) {
      // mengembalikan elemen mem pada indeks baris row dan indeks
      // kolom col
}

public void setELMT(int row, int col, double d) {
      // I.S. elemen mem[row][col] sembarang
      // F.S. mem[row][col] bernilai d
}

public int getRow() {
      // mengembalikan ROW_EFF
}

public int getCol() {
      // mengembalikan COL_EFF
}

public int getLastRow() {
      // mengembalikan indeks baris terakhir mem
}

public int getLastCol() {
      // mengembalikan indeks kolom terakhir mem
}

public boolean isSquare() {
      // mengembalikan true jika mem matriks persegi
}

public void inputMatrix() {
      // I.S. matriks terdefinisi (belum memiliki elemen)
      // F.S. membaca input matriks berukuran bebas tergantung input
      pengguna
}

public void inputMatrix(int m, int n) {
      // I.S. matriks terdefinisi (belum memiliki elemen), m dan n
      terdefinisi
      // F.S. membaca input matriks m x n
}

public void inputSquaredMatrix() {
      // I.S. matriks terdefinisi (belum memiliki elemen)
      // F.S. membaca input matriks berukuran n x n, dengan n input
      bebas pengguna
}

public void input4x4Matrix() {
      // I.S. matriks terdefinisi (belum memiliki elemen)
```

```
      // F.S. membaca input matriks berukuran 4 x 4
}

public void inputFileMatrix(String relativePath) {
      // I.S. relativePath terdefinisi
      // F.S. membaca input matriks dari file pada path 'relativePath'
}

public static void displayMatrix(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. output/ display matriks m ke layar
}

public static void displayMatrixAugmented(Matrix m, int col) {
      // I.S. matriks m dan col terdefinisi
      // F.S. output/ display matriks augmented yang dibatasi pada
      kolom col ke layar
}

public void copyELMT(Matrix mIn) {
      // prekondisi : ukuran mIn dan mem sama
      // I.S. elemen-elemen mem sembarang
      // F.S. elemen-elemen mem sama dengan elemen mIn
}

public void copyELMT(int[][] mIn) {
      // prekondisi : ukuran mIn dan mem sama
      // I.S. elemen-elemen mem sembarang
      // F.S. elemen-elemen mem sama dengan elemen mIn
}

public int[][] getIntMatrix(int min, int max) {
      // mengembalikan matriks mem dalam bentuk integer dengan
      // elemen baru berada pada rentang [min..max]
}

public void copyToSubMatrix(Matrix mIn, int InitRow, int LastRow, int
InitCol, int LastCol) {
      // I.S. semua parameter terdefinisi
      // F.S. matriks diubah menjadi matriks bagian yang mencakup
      daerah di dalam InitRow, LastRow, InitCol, LastCol
}

public void toIdentity() {
      // I.S. matriks terdefinisi dan berbentuk persegi
      // F.S. mengubah matriks menjadi matriks identitas
}

public static Matrix getIdentity(int size) {
      // I.S. size terdefinisi
      // F.S. mengembalikan matriks identitas size x size
}


public Matrix getSubMatrix(int InitRow, int LastRow, int InitCol, int
LastCol) {
```

```
      // I.S. semua parameter terdefinisi
      // F.S. mengembalikan matriks bagian yang mencakup daerah di
      dalam InitRow, LastRow, InitCol, LastCol
}

Public Matrix multiply(Matrix m) {
      // I.S. kedua matriks terdefinisi
      // F.S. mengalikan matriks dengan matriks m
}

public Matrix multiply(double k) {
      // I.S. k terdefinisi
      // F.S. mengembalikan matriks yang semua elemennya sudah
      dikalikan dengan k
}

public void multiplyByConst(double k) {
      // I.S. matriks dan k terdefinisi
      // F.S. mengalikan semua elemen matriks dengan k
}

public void createMatrix() {
      // I.S. matriks terdefinisi
      // F.S. membaca matriks dari input console/ input file pengguna
}

public void swapRow(int a, int b) {
      // I.S. matriks awal, a, dan b terdefinisi
      // F.S. menukar baris a dan b pada matriks
}

public Matrix transpose() {
      // I.S. matriks awal terdefinisi
      // F.S. mengembalikan matriks yang sudah ditranspose
}
```

3.3 Class BicubicInterpolation

Methods

```
public static int function(int x, int y, int i, int j) {
      // I.S. x, y, i, j terdefinisi untuk f(x, y) dan (i, j) untuk
      definisi a_ij
      // F.S. mengembalikan koefisien dari masing-masing nilai a
}

public static Matrix Bicubic(Matrix m4) {
      // I.S. m4 terdefinisi yaitu matriks 4x4
      // F.S. mengembalikan matriks hasil bicubic interpolation dari
      matriks m4
}

public static double getBicubicInterpolation(Matrix n, double x,
double y) {
      // I.S. matriks n terdefinisi, x dan y terdefinisi
```

```
      // F.S. mengembalikan nilai dari f(x, y) hasil bicubic
      interpolation
}

public static String fileHandle(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. mengembalikan string formatted (sudah dihandle) yang
      akan ditulis ke dalam output file
}

public static String consoleHandle(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. mengembalikan string formatted (sudah dihandle) yang
      akan ditampilkan ke console (layar)
}

public static void displayBicubic(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. menampilkan hasil dari bicubic interpolation ke console
      (layar)(handle dari file dan dari console)
}

public static void fileBicubic(Matrix m, String fileName, String
relativePath) {
      // I.S. matriks m dan fileName terdefinisi
      // F.S. menuliskan hasil dari bicubic interpolation ke file
      dengan nama 'fileName' (handle dari file dan dari console) pada
      path 'relativePath'
}

public static void bicubicConsole(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. menampilkan hasil dari bicubic interpolation ke console
      (layar) (fungsi ini yang dipanggil dari main)
}

public static void bicubicFile(Matrix m, String relativePath) {
      // I.S. matriks m terdefinisi
      // F.S. menuliskan hasil dari bicubic interpolation ke file
      (fungsi ini yang dipanggil dari main)
}

public static String resultBicubic(double d) {
      // I.S. d terdefinisi
      // F.S. menuliskan hasil dari bicubic interpolation (formatted)
}
```

3.4 Class Cramer

Methods

```
public static String cramer(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. mengembalikan String berisi solusi cramer (bisa tidak
      memiliki solusi)
```

```
}

public static void displayCramer(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. output matriks awal dan hasil dari solusi cramer ke
      console (layar)
}

public static void fileCramer(Matrix m, String fileName, String
relativePath) {
      // I.S. matriks m dan string fileName terdefinisi
      // F.S. menuliskan hasil dari solusi cramer ke file dengan nama
      'fileName' pada path 'relativePath'
}
```

3.5 Class Determinan

Attribute

Nama	Tipe	Deskripsi
MatrixNDet	public static class	Class di dalam class untuk menyimpan matriks, determinan, dan k (banyaknya swapRow)

Methods

```
public static double cofactorDet(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. mengembalikan hasil dari determinan dari suatu matriks
      dengan metode ekspansi kofaktor (memanfaatkan rekursif)
}

public static Matrix smallerMatrix(Matrix m, int row, int col) {
      // I.S. matriks m terdefinisi(nxn), row dan col terdefinisi
      // F.S. mengembalikan submatriks kofaktor dengan mencoret baris
      row dan kolom col dari matriks m
}

public static void displayCofactorDet(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. output matriks awal dan hasil dari determinan dengan
      metode ekspansi kofaktor ke console (layar)
}

public static void fileCofactorDet(Matrix m, String fileName, String
relativePath) {
      // I.S. matriks m dan string fileName terdefinisi
      // F.S. menuliskan hasil dari determinan dengan metode ekspansi
      kofaktor ke file dengan nama 'fileName' pada path 'relativePath'
}
```

```
public static MatrixNDet rowReductionDet(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. mengembalikan MatrixNDet dari matriks m (berisi matriks
      hasil, determinan, dan k swapRow) dengan metode reduksi baris
}

public static void fileRowReductionDet(Matrix m, String fileName,
String relativePath) {
      // I.S. matriks m dan string fileName terdefinisi
      // F.S. menuliskan hasil dari determinan dengan metode reduksi
      baris ke file dengan nama 'fileName' pada path 'relativePath'
}

public static void detConsole(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. handle output dari class determinan ke console (layar)
}

public static void detFile(Matrix m, String relativePath) {
      // I.S. matriks m terdefinisi
      // F.S. handle output dari class determinan ke file
}
```

3.6 Class GaussJordan

Attribute

Nama	Tipe	Deskripsi
KDet	public static class	Class di dalam class untuk menyimpan matriks dan k (banyaknya swapRow)

Methods

```
public static int firstNotZero(Matrix m, int idx) {
      // I.S. matriks m terdefinisi dan idx merupakan indeks suatu
      //      kolom di matriks m
      // F.S. mengembalikan int indeks baris pertama di kolom idx yang
      //      tidak nol
}

public static void sortMatrix(Matix m) {
      // I.S. matrix m terdefinisi
      // F.S. matriks diurutkan berdasar banyaknya angka nol di awal
}

public static makeOne(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. matriks m yang sudah diubah menjadi matriks eselon baris
}

public static OBE(Matrix m, int idx) {
```

```
      // I.S. matriks m terdefinisi dan idx merupakan indeks suatu
      //      baris di matriks m
      // F.S. mengurangkan baris di bawah baris idx dengan
      //      kelipatannya agar didapat angka nol di bawah leading one
      //      baris idx
}

public static KDet sortMatrixK(Matrix m) {
      // I.S. matrix m terdefinisi
      // F.S. matriks diurutkan berdasar banyaknya angka nol di awal
      //      serta int banyaknya penukaran baris dilakukan
}

public static KDet determinantOBE(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. matriks m dieliminasi OBE tanpa dibuat menjadi matriks
      //      eselon baris serta int banyaknya penukaran baris
      //      dilakukan
}
```

3.7 Class ImageResize

Attribute

Nama	Tipe	Deskripsi
oneThird	final static double	Konstanta yang digunakan dalam algoritma pembesaran ukuran citra.

Methods

```
public static Matrix DoubleScale(Matrix m) {
      // PREKONDISI : baris dan kolom matriks m >= 4
      // mengembalikan matriks baru dengan ukuran dua kali matriks m
      // serta elemen-elemen hasil interpolasi bikubik
}

public static BufferedImage scale(BufferedImage src, int w, int h) {
      // menerima BufferedImage src dengan ukuran sembarang dan
      // mengembalikan BufferedImage baru dengan lebar w dan tinggi h.
}

public static processImage(File file, String outputName, String
    relativePath){
      // I.S. file, outName, dan relativePath terdefinisi
      // F.S. Dibentuk file baru bernama outputName berdasarkan gambar
      // pada file awal dengan ukuran diperbesar menjadi dua kali
      // lipat serta file baru tersebut disimpan pada folder yang
      // berada pada relativePath

}
```

```
public static inputImage(String relativePath){
      // I.S. relativePath terdefinisi
      // F.S. nama file yang akan diproses sudah diinput dan
      // dijalankan prosesImage
}
```

3.8 Class Interpolation

Attribute

Nama	Tipe	Deskripsi
Problem	public static Class	Tipe data bentukan untuk static Instance dalam menyelesaikan masalah interpolasi pada script lain.
problem	public static Problem	Instance static yang digunakan untuk masalah interpolasi.

3.9 Subclass Interpolation

Attribute

Nama	Tipe	Deskripsi
n	private int	Menyimpan derajat polinomial interpolasi saat ini.
points	private Matrix	Menyimpan input titik-titik interpolasi saat ini.
m	private Matrix	Menyimpan augmented matrix dari input interpolasi polinomial saat ini.
result	private Matrix	Menyimpan matriks kolom yang merupakan solusi interpolasi polinomial saat ini.
minBound	private Double	Menyimpan batas bawah nilai x pada interpolasi saat ini.
maxBound	private Double	Menyimpan batas atas nilai x pada interpolasi saat ini.

Methods

```
Private void inputPoints() {
      // I.S. points menyimpan input lama atau sembarang
      // F.S. points terdefinisi dengan input baru
}

private void createAugmented() {
      // I.S. m menyimpan augmented matrix yang lama atau sembarang
      // F.S. m menyimpan augmented matrix dari masalah baru
}

private void getBound() {
     // I.S. minBound dan maxBound menyimpan batas atas dan bawah yang
     // lama atau keduanya sembarang
     // F.S. minBound dan maxBound menyimpan batas yang baru
}

private void solve() {
     // I.S. atribut selain points belum diperbarui
     // F.S. seluruh atribut termasuk result diperbarui berdasarkan
     // input data masalah interpolasi polinomial terakhir
}

private void getPointAbsis (int row) {
     // mengembalikan absis suatu titik yang disimpinan pada suatu
     // matriks dengan letak baris row
}

private void getPointOrdinat (int row) {
     // mengembalikan ordinat suatu titik yang disimpinan pada suatu
     // matriks dengan letak baris row
}

public void inputNewProblem(){
     // I.S. seluruh atribut belum diperbarui ke masalah baru
     // F.S. seluruh atribut diperbarui berdasarkan input masalah baru
     // Proses : input masalah dilakukan melalui konsol
}

public void inputProblemFile(String relativePath){
    // I.S. seluruh atribut belum diperbarui ke masalah baru
    // F.S. seluruh atribut diperbarui berdasarkan input masalah baru
    // Proses : input masalah dibaca dari file yang terletak pada
    // relativePath
}

public Matrix getResult() {
    // mengembalikan matriks baru dengan elemen-elemen result
}

Public double interpolate(double x) {
    // mengembalikan nilai fungsi polinomial hasil interpolasi pada x
}
```

```
Public void displayInterpolation(){
     // I.S. seluruh atribut sesuai dengan data saat ini
     // F.S. ditampilkan fungsi polinomial hasil interpolasi serta
     // hasil prediksi nilai fungsi berdasarkan input nilai x melalui
     // konsol
}

public void createOutputFile(String relativePath){
     // I.S. seluruh atribut sesuai dengan data saat ini
     // F.S. dibuat file berisi fungsi polinomial hasil interpolasi
     // serta hasil prediksi nilai fungsi berdasarkan input nilai x
     // melalui konsol

}

public String getPolinom (Matrix result){
     // prekondisi : result tidak kosong
     // mengembalikan string fungsi polinomial hasil interpolasi
}
```

3.10 Class Inverse

Methods

```
public static Matrix getInverse (Matrix m) {
      // mengembalikan inverse dari m dengan metode OBE
}

public static Matrix getInverse (Matrix m, int method) {
      // mengembalikan inverse dari m dengan metode OBE jika method
      // bernilai 1 sedangkan menggunakan metode matriks adjoin jika
      // method bernilai 2
}

public static void outputInverse(Matriks m){
     // I.S. matriks m terdefinisi
     // F.S. Menampilkan hasil inverse ke konsol
}

Public static outputInverse (Matriks m, String relativePath ){
     // I.S. matriks m dan relativePath terdefinisi
     // F.S. Dibuat file yang berisi hasil inverse matriks m dan
     // terletak pada relativePath
}
```

3.11 Class Regression

Methods

```
public static Matrix regressionInput() {
      // I.S. -
      // F.S. matriks berisi input untuk regresi
}

public static Matrix regressionMatrix(Matrix m) {
```

```
      // I.S. matriks m adalah matriks input regresi
      // F.S. matriks representasi SPL regresi linear berganda yang
      //      merupakan hasil pemrosesan dari matriks m
}

public static String equationOutput(Matrix m) {
      // I.S. matriks m adalah matriks SPL yang sudah dieliminasi
      // F.S. string solusi persamaan regresi linear berganda
}

public static String RegressionFunction(Matrix m, Matrix x) {
      // I.S. matriks m adalah matriks SPL yang sudah dieliminasi
      //      matriks x adalah matriks nilai x yang akan ditaksir
      //      nilai fungsinya
      // F.S. string hasil fungsi yang sudah ditaksir nilainya
}

public static void regConsole(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. melakukan regresi pada matriks m dan menampilkannya ke
      //      layar
}

public static void regFile(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. melakukan regresi pada matriks m dan menyimpannya ke
      //      dalam file
}
```

3.12 Class SolveSPL

Methods

```
public static void gauss (Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. matriks eselon baris yang sudah dieliminasi Gauss
}

public static void gaussJordan(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. matriks eselon baris yang sudah dieliminasi Gauss-Jordan
}

public static Matrix getGauss(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. mengembalikan bentuk Gauss dari m
}

public static Matrix getGaussJordan(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. mengembalikan bentuk Gauss-Jordan dari m
}

public static Matrix inverseSolution(Matrix m) {
      // I.S. matriks m terdefinisi
```

```
      // F.S. mengembalikan solusi SPL matriks dengan metode invers
}

public static Matrix inverseSolution(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. mengembalikan solusi SPL matriks dengan metode invers
}

public static String gaussJordanOutputSolution(Matrix m) {
      // I.S. matriks m dalam bentuk Gauss-Jordan
      // F.S. mengembalikan string solusi dari m
}

public static String gaussOutputSolution(Matrix m) {
      // I.S. matriks m dalam bentuk Gauss
      // F.S. mengembalikan string solusi dari m
}

public static String inverseOutputSolution(Matrix m) {
      // I.S. matriks m sudah diproses fungsi inverseSolution()
      // F.S. mengembalikan string solusi dari m
}

public static void splConsole(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. menampilkan solusi SPL ke layar sesuai input pengguna
}

public static void splFile(Matrix m) {
      // I.S. matriks m terdefinisi
      // F.S. menyimpan solusi SPL ke file sesuai input pengguna
}
```

3.13 Class Menu

Methods

```
public static void menuLoop() {
      // I.S. menuLoop dipanggil dari class Main, digunakan untuk
      looping aplikasi
      // F.S. aplikasi berjalan hingga pengguna selesai menggunakan
}

public static void mainMenu() {
      // I.S. mainMenu dipanggil dari menuLoop
      // F.S. aplikasi berjalan dengan alur yang sudah ada
}

public static void inputFileConsole() {
      // I.S. inputFileConsole dipanggil saat user diminta untuk
      memilih tipe input
      // F.S. output pilihan jenis input ke console (layar)
}
```

```
public static void outputFileConsole() {
      // I.S. outputFileConsole dipanggil saat user diminta untuk
      memilih tipe output
      // F.S. output pilihan jenis output ke console (layar)
}

public static void showChoice(String title, String[] methods, int
length) {
      // I.S. showChoice dipanggil saat user diminta untuk memilih
      jenis pilihan berdasarkan String
      // F.S. output pilihan jenis pilihan ke console (layar)
}

public static exit() {
      // I.S. exit dipanggil saat pengguna sudah selesai menggunakan
      // F.S. keluar dari aplikasi
}
```

3.14 Class Utils

Methods

```
public static void print(Object arg0) {
      // I.S. arg0 terdefinisi (bisa integer, string, dll.)
      // F.S. menampilakn arg0 ke layar
}

public static void println(Object arg0) {
      // I.S. arg0 terdefinisi (bisa integer, string, dll.)
      // F.S. menampilan arg0 ke layar diakhiri '\n'
}

public static void printf(String format, Object arg0) {
      // I.S. arg0 terdefinisi (bisa integer, string, dll.) dan format
      terdefinisi
      // F.S. menampilkan arg0 ke layar dengan format
}

public static boolean isInt(String str) {
      // I.S. str terdefinisi
      // F.S. mengembalikan true jika str adalah integer, false jika
      tidak
}

public static boolean isDouble(String str) {
      // I.S. str terdefinisi
      // F.S. mengembalikan true jika str adalah double, false jika
      tidak
}

public static char inputYN() {
      // I.S. digunakan jika user mau menginput Y\N
      // F.S. check hingga user menginput char Y\N, mengembalikan char
      Y\N
}
```

```
public static int inputInt() {
      // I.S. digunakan jika user mau menginput integer
      // F.S. check hingga user menginput integer, mengembalikan
      integer
}

public static double inputDouble() {
      // I.S. digunakan jika user mau menginput double
      // F.S. check hingga user menginput double, mengembalikan double
}

public static int select(int min, int max) {
      // I.S. min dan max terdefinisi
      // F.S. check hingga user menginput integer antara min dan max,
      mengembalikan integer
}

public static int getLengthELMT(double d) {
      // I.S. d terdefinisi
      // F.S. mengembalikan panjang dari d
}

public static double eval(String s) {
      // I.S. s terdefinisi
      // F.S. mengembalikan double dari input string (string bisa
      pecahan)
}

public static String result(double det) {
      // I.S. det terdefinisi
      // F.S. mengembalikan formatted string
}
```

3.15 Class Main

Methods

```
public static void main(String[] args) {
      // I.S. driver utama aplikasi
      // F.S. menjalankan aplikasi mulai dari menu utama
}
```

Bab IV
Eksperimen

1. Solusi SPL $Ax = b$

a.

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

b.

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

c.

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

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.

Nama	Input	Output

<table>
<tr><td>Test case 1a</td><td>

```
Silahkan masukkan setiap elemen matrix:
1 1 -1 -1
2 5 -7 -5
2 -1 1 3
5 2 -4 2

Masukkan data matriks b :
(Input dalam satu baris. Contoh = 1 2 3 4 )

Silahkan masukkan setiap elemen matrix:
1 -2 4 6
```

</td><td>

```
Matrix awal:
 1  1 -1 -1  |  1
 2  5 -7 -5  | -2
 2 -1  1  3  |  4
 5  2 -4  2  |  6

Matrix hasil eliminasi Gauss:
 1      1     -1     -1      |      1
 0      1 -1.667     -1      | -1.333
 0      0      1     -1      |      1
 0      0      0      0      |      1

SPL tidak memiliki solusi.
```

</td></tr>
<tr><td>Analisis 1a</td><td colspan="2">Digunakan metode eliminasi Gauss. SPL tidak memiliki solusi karena terdapat baris yang semua elemennya angka nol kecuali elemen terakhirnya.</td></tr>
<tr><td>Test case 1b</td><td>

```
Silahkan masukkan setiap elemen matrix:
1 -1 0 0 1
1 1 0 -3 0
2 -1 0 1 -1
-1 2 0 -2 -1

Masukkan data matriks b :
(Input dalam satu baris. Contoh = 1 2 3 4 )

Silahkan masukkan setiap elemen matrix:
3 6 5 1
```

</td><td>

```
Matrix awal:
  1 -1  0  0  1  |  3
  1  1  0 -3  0  |  6
  2 -1  0  1 -1  |  5
 -1  2  0 -2 -1  |  1

SPL tidak memiliki solusi atau tidak bisa
diselesaikan dengan Metode Matrix Inverse.
```

```
Matrix awal:
  1 -1  0  0  1  |  3
  1  1  0 -3  0  |  6
  2 -1  0  1 -1  |  5
 -1  2  0 -2 -1  | -1

Matrix hasil eliminasi Gauss-Jordan:
 1  0  0  0 -1  |  3
 0  1  0  0 -2  |  0
 0  0  0  1 -1  | -1
 0  0  0  0  0  |  0

Solusi SPL:
x1 = 3 + t2
x2 = 2t2
x3 = t1
x4 = -1 + t2
x5 = t2
Dengan t1 dan t2 merupakan variabel bebas.
```

</td></tr>
<tr><td>Analisis 1b</td><td colspan="2">Digunakan metode matriks balikan. Metode matriks balikan tidak dapat digunakan untuk menyelesaikan SPL tersebut. Digunakan metode eliminasi Gauss-Jordan. Didapatkan solusi parametrik:
x1 = 3 + t2
x2 = 2t2
x3 = t1
x4 = -1 + t2
x5 = t2
Dengan t1 dan t2 merupakan variabel bebas.</td></tr>
</table>

<table>
<tr>
<td>Test case 1c</td>
<td>

```
Silahkan masukkan setiap elemen matrix:
0 1 0 0 1 0
0 0 0 1 1 0
0 1 0 0 0 1

Masukkan data matriks b :
(Input dalam satu baris. Contoh = 1 2 3 4 )

Silahkan masukkan setiap elemen matrix:
2 -1 1
```

</td>
<td>

```
Matrix awal:
0  1  0  0  1  0  2
0  0  0  1  1  0 -1
0  1  0  0  0  1  1

Matrix ini tidak memiliki solusi atau tidak bisa di
selesaikan dengan Cramer. Silahkan coba pakai cara
lain. Rekomendasi: Gauss/Gauss-Jordan
```

```
Matrix awal:
 0  1  0  0  1  0  |  2
 0  0  0  1  1  0  | -1
 0  1  0  0  0  1  |  1

Matrix hasil eliminasi Gauss-Jordan:
 0  1  0  0  0  1  |  1
 0  0  0  1  0  1  | -2
 0  0  0  0  1 -1  |  1

Solusi SPL:
x1 = t1
x2 = 1 - t3
x3 = t2
x4 = -2 - t3
x5 = 1 + t3
x6 = t3
Dengan t1, t2 dan t3 merupakan variabel bebas.
```

</td>
</tr>
<tr>
<td>Analisis 1c</td>
<td colspan="2">Digunakan kaidah Cramer, tetapi karena matriks tidak bisa diselesaikan dengan kaidah Cramer, maka digunakan metode Gauss-Jordan. Didapatkan solusi:
x1 = t1
x2 = 1 - t3
x3 = t2
x4 = -2 - t3
x5 = 1 + t3
x6 = t3
Dengan t1, t2 dan t3 merupakan variabel bebas.</td>
</tr>
<tr>
<td>Test case 1d</td>
<td>

n = 6

```
Silahkan masukkan setiap elemen matrix:
1 1/2 1/3 1/4 1/5 1/6
1/2 1/3 1/4 1/5 1/6 1/7
1/3 1/4 1/5 1/6 1/7 1/8
1/4 1/5 1/6 1/7 1/8 1/9
1/5 1/6 1/7 1/8 1/9 1/10
1/6 1/7 1/8 1/9 1/10 1/11

Masukkan data matriks b :
(Input dalam satu baris. Contoh = 1 2 3 4 )

Silahkan masukkan setiap elemen matrix:
1 0 0 0 0 0
```

n = 10

```
Silahkan masukkan setiap elemen matrix:
1 1/2 1/3 1/4 1/5 1/6 1/7 1/8 1/9 1/10
1/2 1/3 1/4 1/5 1/6 1/7 1/8 1/9 1/10 1/11
1/3 1/4 1/5 1/6 1/7 1/8 1/9 1/10 1/11 1/12
1/4 1/5 1/6 1/7 1/8 1/9 1/10 1/11 1/12 1/13
1/5 1/6 1/7 1/8 1/9 1/10 1/11 1/12 1/13 1/14
1/6 1/7 1/8 1/9 1/10 1/11 1/12 1/13 1/14 1/15
1/7 1/8 1/9 1/10 1/11 1/12 1/13 1/14 1/15 1/16
1/8 1/9 1/10 1/11 1/12 1/13 1/14 1/15 1/16 1/17
1/9 1/10 1/11 1/12 1/13 1/14 1/15 1/16 1/17 1/18
1/10 1/11 1/12 1/13 1/14 1/15 1/16 1/17 1/18 1/19

Masukkan data matriks b :
(Input dalam satu baris. Contoh = 1 2 3 4 )

Silahkan masukkan setiap elemen matrix:
1 0 0 0 0 0 0 0 0 0
```

</td>
<td>

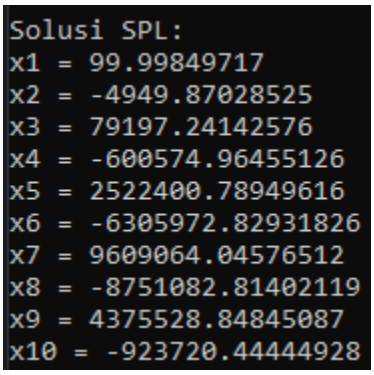
n = 6

```
Matrix awal:
     1   0.5 0.333  0.25   0.2 0.167     |     1
   0.5 0.333  0.25   0.2 0.167 0.143     |     0
 0.333  0.25   0.2 0.167 0.143 0.125     |     0
  0.25   0.2 0.167 0.143 0.125 0.111     |     0
   0.2 0.167 0.143 0.125 0.111   0.1     |     0
 0.167 0.143 0.125 0.111   0.1 0.091     |     0

Matrix hasil eliminasi Gauss-Jordan:
 1     0     0     0     0     0     |    36
 0     1     0     0     0     0     |  -630
 0     0     1     0     0     0     |  3360
 0     0     0     1     0     0     | -7560
 0     0     0     0     1     0     |  7560
 0     0     0     0     0     1     | -2772

Solusi SPL:
x1 = 36
x2 = -630.00000005
x3 = 3360.00000035
x4 = -7560.00000092
x5 = 7560.00000103
x6 = -2772.00000041
```

n = 10

```
Matrix awal:
     1   0.5 0.333  0.25   0.2 0.167 0.143 0.125 0.111   0.1     |     1
   0.5 0.333  0.25   0.2 0.167 0.143 0.125 0.111   0.1 0.091     |     0
 0.333  0.25   0.2 0.167 0.143 0.125 0.111   0.1 0.091 0.083     |     0
  0.25   0.2 0.167 0.143 0.125 0.111   0.1 0.091 0.083 0.077     |     0
   0.2 0.167 0.143 0.125 0.111   0.1 0.091 0.083 0.077 0.071     |     0
 0.167 0.143 0.125 0.111   0.1 0.091 0.083 0.077 0.071 0.067     |     0
 0.143 0.125 0.111   0.1 0.091 0.083 0.077 0.071 0.067 0.063     |     0
 0.125 0.111   0.1 0.091 0.083 0.077 0.071 0.067 0.063 0.059     |     0
 0.111   0.1 0.091 0.083 0.077 0.071 0.067 0.063 0.059 0.056     |     0
   0.1 0.091 0.083 0.077 0.071 0.067 0.063 0.059 0.056 0.053     |     0
```

</td>
</tr>
</table>

		 Matrix hasil eliminasi Gauss-Jordan: 1 0 0 0 0 0 0 0 0 0 | 99.998 0 1 0 0 0 0 0 0 0 0 | -4949.87 0 0 1 0 0 0 0 0 0 0 | 79197.241 0 0 0 1 0 0 0 0 0 0 | -600574.965 0 0 0 0 1 0 0 0 0 0 | 2522400.789 0 0 0 0 0 1 0 0 0 0 | -6305972.829 0 0 0 0 0 0 1 0 0 0 | 9609064.046 0 0 0 0 0 0 0 1 0 0 | -8751082.814 0 0 0 0 0 0 0 0 1 0 | 4375528.848 0 0 0 0 0 0 0 0 0 1 | -923720.444  Solusi SPL: x1 = 99.99849717 x2 = -4949.87028525 x3 = 79197.24142576 x4 = -600574.96455126 x5 = 2522400.78949616 x6 = -6305972.82931826 x7 = 9609064.04576512 x8 = -8751082.81402119 x9 = 4375528.84845087 x10 = -923720.44444928
Analisis 1d	Digunakan metode eliminasi Gauss-Jordan. Untuk matriks Hilbert dengan n = 6 didapat solusi: x1 = 36 x2 = -630.00000005 x3 = 3360.00000035 x4 = -7560.00000092 x5 = 7560.00000103 x6 = -2772.00000041 Untuk matriks Hilbert dengan n = 10 didapat solusi: x1 = 99.99849717 x2 = -4949.87028525 x3 = 79197.24142576 x4 = -600574.96455126 x5 = 2522400.78949616 x6 = -6305972.82931826 x7 = 9609064.04576512 x8 = -8751082.81402119 x9 = 4375528.84845087 x10 = -923720.44444928	

2. SPL Berbentuk Matriks *Augmented*

a.

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

b.

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

Nama	Input (file)	Output

<table>
<tr>
<td>Test case 2a</td>
<td>1 -1 2 -1 -1
2 1 -2 -2 -2
-1 2 -4 1 1
3 0 0 -3 -3</td>
<td>

```
Matrix awal:
 1 -1  2 -1 -1
 2  1 -2 -2 -2
-1  2 -4  1  1
 3  0  0 -3 -3

Matrix ini tidak memiliki solusi
karena determinan matrix adalah 0
```

```
Matrix awal:
  1 -1  2 -1  | -1
  2  1 -2 -2  | -2
 -1  2 -4  1  |  1
  3  0  0 -3  | -3

Matrix hasil eliminasi Gauss-Jordan:
 1  0  0 -1  | -1
 0  1 -2  0  |  0
 0  0  0  0  |  0
 0  0  0  0  |  0

Solusi SPL:
x1 = -1 + t2
x2 = 2t1
x3 = t1
x4 = t2
Dengan t1 dan t2 merupakan variabel bebas.
```

</td>
</tr>
<tr>
<td>Analisis 2a</td>
<td colspan="2">Digunakan kaidah Cramer. SPL tidak memiliki solusi karena determinan matriks bernilai 0. Digunakan metode Gauss-Jordan. Didapatkan solusi parametrik:
x1 = -1 + t2
x2 = 2t1
x3 = t1
x4 = t2
Dengan t1 dan t2 merupakan variabel bebas.</td>
</tr>
<tr>
<td>Test case 2b</td>
<td>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</td>
<td>

```
Matrix awal:
  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

Matrix hasil eliminasi Gauss-Jordan:
 1 0 0 0 | 0
 0 1 0 0 | 2
 0 0 1 0 | 1
 0 0 0 1 | 1
 0 0 0 0 | 0
 0 0 0 0 | 0

Solusi SPL:
x1 = 0
x2 = 2
x3 = 1
x4 = 1
```

</td>
</tr>
</table>

Analisis 2b	Digunakan metode eliminasi Gauss-Jordan. Didapatkan solusi SPL: x1 = 0 x2 = 2 x3 = 1 x4 = 1

3. SPL Bentuk Normal

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

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

Nama	Input (file)	Output
Test case 3a	8 1 3 2 0 2 9 -1 -2 1 1 3 2 -1 2 1 0 6 4 3	Matrix awal: 8 1 3 2 | 0 2 9 -1 -2 | 1 1 3 2 -1 | 2 1 0 6 4 | 3 Solusi: x1 = -0.22432432 x2 = 0.18243243 x3 = 0.70945946 x4 = -0.25810811
Analisis 3a	Digunakan metode matriks balikan. Didapat solusi: x1 = -0.22432432 x2 = 0.18243243 x3 = 0.70945946 x4 = -0.25810811	

Test case 3b	0 0 0 0 0 0 1 1 1 13.00 0 0 0 1 1 1 0 0 0 15.00 1 1 1 0 0 0 0 0 0 8.00 0 0 0.04289 0 0.04289 0.75 0.04289 0.75 0.61396 14.79 0 0.25 0.91421 0.25 0.91421 0.25 0.91421 0.25 0 14.31 0.61396 0.75 0.04289 0.75 0.04289 0 0.04289 0 0 3.81 0 0 1 0 0 1 0 0 1 18.00 0 1 0 0 1 0 0 1 0 12.00 1 0 0 1 0 0 1 0 0 6.00 0.04289 0.75 0.61396 0 0.04289 0.75 0 0 0.04289 10.51 0.91421 0.25 0 0.25 0.91421 0.25 0 0.25 0.91421 16.13 0.04289 0 0 0.75 0.04289 0 0.61396 0.75 0.04289 7.04	Matrix hasil eliminasi Gauss-Jordan 1 0 0 0 0 0 0 0 0 | 0 0 1 0 0 0 0 0 0 0 | 0 0 0 1 0 0 0 0 0 0 | 0 0 0 0 1 0 0 0 0 0 | 0 0 0 0 0 1 0 0 0 0 | 0 0 0 0 0 0 1 0 0 0 | 0 0 0 0 0 0 0 1 0 0 | 0 0 0 0 0 0 0 0 1 0 | 0 0 0 0 0 0 0 0 0 1 | 0 0 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 | 0 SPL tidak memiliki solusi.
Analisis 3b	Digunakan metode eliminasi Gauss-Jordan. SPL tidak memiliki solusi karena terdapat baris yang semua elemennya angka nol kecuali elemen terakhirnya.	

4. Studi Kasus Interpolasi

Nama	Input	Output
Test case 4a	6 0.4 0.043 0.7 0.005 0.11 0.058 0.14 0.072 0.17 0.1 0.2 0.13 0.23 0.147 Masukkan nilai x [0.11, 0.7] (inklusif) untuk interpolasi. (input bukan bilangan akan memberhentikan proses ini) > 0.2 Masukkan nilai x [0.11, 0.7] (inklusif) untuk interpolasi. (input bukan bilangan akan memberhentikan proses ini) > 0.55 Masukkan nilai x [0.11, 0.7] (inklusif) untuk interpolasi. (input bukan bilangan akan memberhentikan proses ini) > 0.85 Nilai x berada di luar rentang interpolasi. Apakah Anda ingin membatalkan input x ini? (0 : batalkan, 1 : lanjutkan) > 1 Masukkan nilai x [0.11, 0.7] (inklusif) untuk interpolasi. (input bukan bilangan akan memberhentikan proses ini) > 1.28 Nilai x berada di luar rentang interpolasi. Apakah Anda ingin membatalkan input x ini? (0 : batalkan, 1 : lanjutkan) > 1	f(x) = - 4212.4345x6 + 7102.3992x5 - 4346.314x4 + 1220.8549x3 - 163.9157x2 + 10.2764x - 0.1846 f(0.2) = 0.13 f(0.55) = 2.13757162 f(0.85) = -66.26963931 f(1.28) = -3485.14490149
Analisis 4a	Pada file keluaran, ditampilkan fungsi polinomial hasil interpolasi dan empat nilai hasil polinomial interpolasi dengan x = 0.2, x = 0.55, x = 0.85, dan x = 1.28. Diberikan peringatan ketika input x berada di luar rentang interpolasi.	

<table>
<tr>
<td>Test case 4b</td>
<td>
9
6.567 12624
7 21807
7.258 38391
7.451 54517
7.548 51952
7.839 28228
8.161 35764
8.484 20813
8.709 12408
9 10534

Masukkan nilai x [6.567, 9] (inklusif) untuk interpolasi.
(input bukan bilangan akan memberhentikan proses ini)
> 7.516

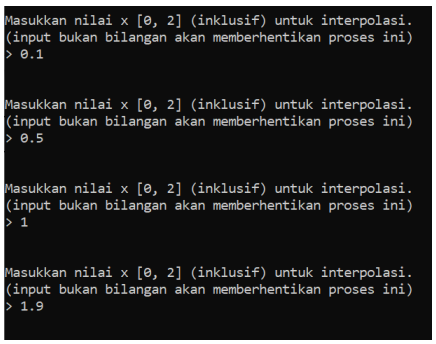
Masukkan nilai x [6.567, 9] (inklusif) untuk interpolasi.
(input bukan bilangan akan memberhentikan proses ini)
> 8.323

Masukkan nilai x [6.567, 9] (inklusif) untuk interpolasi.
(input bukan bilangan akan memberhentikan proses ini)
> 9.167

Nilai x berada di luar rentang interpolasi. Apakah Anda ingin membatalkan input x ini?
(0 : batalkan, 1 : lanjutkan)

> 1

Masukkan nilai x [6.567, 9] (inklusif) untuk interpolasi.
(input bukan bilangan akan memberhentikan proses ini)
> 8
</td>
<td>
f(x) = - 141000.0848x9 + 9373297.9647x8 - 2.754885663224E8x7
+ 4.6960617941394E9x6 - 5.11348645851892E10x5 + 3.685740749582228E11x4
- 1.75693084314076 7E12x3 + 5.334604798379654E12x2 - 9.347772451153225E12x
+ 7.187737250847979E12
f(7.516) = 53467.4921875
f(8.323) = 36174.109375
f(9.167) = -667902.734375
f(8) = 27654.67578125
</td>
</tr>
<tr>
<td>Analisis 4b</td>
<td colspan="2">Karena angka yang terlalu besar, akurasi program mencari berkurang sehingga terdapat galat pada koefisien fungsi polinomial yang dihasilkan.</td>
</tr>
<tr>
<td>Test case 4c</td>
<td>
5
0 0
0.4 0.418884
0.8 0.507158
1.2 0.560925
1.6 0.583686
2 0.576652
</td>
<td>
f(x) = 0.2363x5 - 1.4213x4 + 3.2371x3 - 3.5527x2 + 2.0353x
f(0.1) = 0.17109626
f(0.5) = 0.45265138
f(1) = 0.5346803
f(1.9) = 0.57303737
</td>
</tr>
</table>

		
Analisis 4c	Seluruh input berada pada rentang interpolasi. Fungsi dan seluruh hasil prediksi ditulis di dalam file keluaran.	

5. Studi Kasus Interpolasi Bicubic

Diberikan matriks input:

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

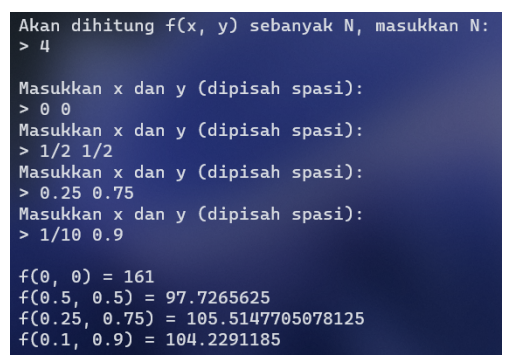
Tentukan nilai:

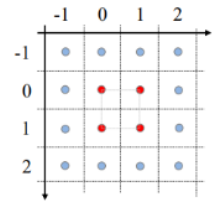
f(0,0) = ?

f(0.5, 0.5) = ?

f(0.25, 0.75) = ?

f(0.1, 0.9) = ?

Nama	Input	Output
Test case	$A = \begin{bmatrix} 153 & 59 & 210 & 96 \\ 125 & 161 & 72 & 81 \\ 98 & 101 & 42 & 12 \\ 21 & 51 & 0 & 16 \end{bmatrix}$	
Analisis	Dari data tersebut diperoleh persamaan f(x, y) dari $f(x,y) = \sum_{j=0}^{3}\sum_{i=0}^{3} a_{ij} x^i y^j$ dengan x = [-1,2] dan y = [-1,2], pada studi kasus diminta untuk menghitung nilai:	

	 1. f(0, 0) dapat dilihat jika x dan y bernilai 0, maka hasil yang akan didapat adalah nilai pada titik (0, 0), jika dilihat pada matriks, maka hasil dari f(0, 0) = 161 2. f(0.5, 0.5) berarti kita menghitung nilai suatu titik di tengah-tengah matriks, didapat hasil f(0.5, 0.5) = 97.7265625 3. f(0.25, 0.75) = 105.5147705078125 4. f(0.1, 0.9) = 104.2291185

6. Studi Kasus Regresi Linier Berganda

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.

ditanyakan estimasi nilai Nitrous Oxide apabila Humidity bernilai 50%, temperatur 76°F, dan tekanan udara sebesar 29.30.

Nama	Input (file)	Output

<table>
<tr>
<td>Test case</td>
<td>72.4 76.3 29.18 0.90
41.6 70.3 29.35 0.91
34.3 77.1 29.24 0.96
35.1 68.0 29.27 0.89
10.7 79.0 29.78 1.00
12.9 67.4 29.39 1.10
08.3 66.8 29.69 1.15
20.1 76.9 29.48 1.03
72.2 77.7 29.09 0.77
24.0 67.7 29.60 1.07
23.2 76.8 29.38 1.07
47.4 86.6 29.35 0.94
31.5 76.9 29.63 1.10
10.6 86.3 29.56 1.10
11.2 86.0 29.48 1.10
73.3 76.3 29.40 0.91
75.4 77.9 29.28 0.87
96.6 78.7 29.29 0.78
107.4 86.8 29.03 0.82
54.9 70.9 29.37 0.95
50.0 76.0 29.30</td>
<td>
<pre>Matriks SPL yang terbentuk:
     20   863.1   1530.4   587.84 | 19.42
  863.1 54876.89 67000.09 25283.395 | 779.477
 1530.4 67000.09 117912.32 44976.867 | 1483.437
 587.84 25283.395 44976.867 17278.509 | 571.122

Matriks SPL yang telah dieliminasi:
 1   0   0   0 | -3.508
 0   1   0   0 | -0.003
 0   0   1   0 | 0.001
 0   0   0   1 | 0.154

Persamaan regresi yang terbentuk:
f(x) = -3.50777814 - 0.00262499x1 + 7.9894E-4x2 + 0.15415503x3

Maka untuk input:
x1 = 50
x2 = 76
x3 = 29.3
Didapatkan f(x) = 0.93843423</pre>
keterangan:
Untuk persoalan regresi, masukannya jika dari keyboard adalah n (jumlah peubah x), m (jumlah sampel), 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.
</td>
</tr>
<tr>
<td>Analisis</td>
<td colspan="2">
Dari data-data tersebut, apabila diterapkan Normal Estimation Equation for Multiple Linear Regression, maka diperoleh sistem persamaan linear sebagai berikut.
$20b_0 + 863.1b_1 + 1530.4b_2 + 587.84b_3 = 19.42$
$863.1b_0 + 54876.89b_1 + 67000.09b_2 + 25283.395b_3 = 779.477$
$1530.4b_0 + 67000.09b_1 + 117912.32b_2 + 44976.867b_3 = 1483.437$
$587.84b_0 + 25283.395b_1 + 44976.867b_2 + 17278.5086b_3 = 571.1219$
Matriks SPL yang terbentuk pada output sudah identik dengan SPL hasil Normal Estimation Equation for Multiple Linear Regression pada referensi soal. Maka dengan mengeliminasi matriks augmented dengan metode Gauss-Jordan, akan terbentuk fungsi regresi linear berganda yang dapat mengestimasi nilai output dari variabel-variabel input yang diberikan. Berdasarkan persamaan yang diperoleh, yaitu :
f(x) = -3.50777814 - 0.00262499x1 + 7.9894E-4x2 + 0.15415503x3
Apabila Humidity bernilai 50%, temperatur 76°F, dan tekanan udara sebesar 29.30 , didapatkan estimasi nilai Nitrous Oxide sebesar 0.93843423.
</td>
</tr>
</table>

7. **Studi Kasus Perbesaran Citra dengan Interpolasi Bikubik**

Nama	Input (file)	Output

Test case 1		
Analisis	Gambar masukan diperbesar dua kali dengan teknik interpolasi bikubik. Perhitungan dilakukan dengan menggunakan seluruh kemungkinan submatriks berukuran 4x4 yang ada pada matriks RGBA gambar masukan lalu menyisipkan hasil nilai-nilai interpolasi bikubik beserta nilai asal ke matriks baru dengan ukuran dua kali lipat. File masukan dan keluaran dapat berbentuk file png, jpg, ataupun jpeg (tentu pengguna umumnya perlu memilih file masukan dan keluaran dalam bentuk png untuk mendukung transparansi gambar).	
Test case 2		
	Gambar masukan pada contoh kasus ini bernilai 0 untuk seluruh nilai RGB pada matriksnya sehingga gambar tersebut hanya bergantung pada nilai alpha. Terlihat bahwa gambar keluaran masih mendukung transparansi gambar tersebut.	

Bab V
Kesimpulan, Saran, dan Refleksi

Sistem Persamaan Linear (SPL) dapat diselesaikan dengan berbagai metode diantaranya dengan menggunakan metode Eliminasi Gauss, Eliminasi Gauss-Jordan, kaidah Cramer, dan Matriks Balikan (Inverse). Masing-masing metode memiliki ketentuannya masing-masing, seperti pada kaidah Cramer dan Matriks Balikan (dengan Adjoint) yang hanya dapat digunakan untuk matriks persegi yang memiliki determinan tak nol.

Sistem Persamaan Linear (SPL) selalu memiliki penyelesaian, ada 3 jenis penyelesaiannya, yakni solusi tunggal, solusi banyak, dan tidak ada solusi.

Untuk solusi tunggal dapat dilihat pada contoh kasus nomor 1d, 2b, dan 3a. Sistem persamaan linear dengan solusi tunggal dapat didapatkan dengan mengeliminasi matriks augmented menggunakan metode Gauss, Gauss-Jordan, matriks balikan, maupun kaidah Cramer.

Untuk solusi banyak (menggunakan parametrik) dapat dilihat pada contoh kasus nomor 1b, 1c, dan 2a. Solusi parametrik hanya dapat didapatkan menggunakan metode eliminasi Gauss atau metode eliminasi Gauss-Jordan. Metode matriks balikan dan kaidah Cramer tidak dapat digunakan untuk mencari penyelesaian parametrik karena metode matriks balikan dan kaidah Cramer memiliki batasan determinan matriks tidak boleh nol.

Untuk contoh kasus SPL yang tidak memiliki solusi dapat dilihat pada contoh kasus nomor 1a dan 3b. Salah satu ciri sistem persamaan linear yang tidak memiliki solusi adalah setelah dieliminasi menggunakan metode Gauss-Jordan, terdapat satu atau lebih baris matriks yang seluruh elemennya berupa angka nol kecuali elemen terakhirnya yang bernilai satu.

Dengan menggunakan Sistem Persamaan Linear (SPL) ini, berbagai macam masalah dapat terselesaikan. SPL ini juga dipakai pada perhitungan Interpolasi Polinom, Regresi Linear, dan Interpolasi Bikubik.

Interpolasi polinom dapat digunakan untuk memprediksi kasus berdasarkan fungsi yang didapat. Pada contoh kasus 4(b), Interpolasi Polinom digunakan untuk memprediksi banyaknya kasus positif Covid-19 pada rentang tertentu. Didapat fungsi polinomial hasil interpolasi sebagai berikut.

f(x) = -141000.0848x9 + 9373297.9647x8 - 2.754885663224E8x7 + 4.6960617941394E9x6 - 5.11348645851892E10x5 + 3.685740749582228E11x4 - 1.756930843140767E12x3 + 5.334604798379654E12x2 - 9.347772451153225E12x + 7.187737250847979E12

Namun, koefisien-koefisien pada fungsi ini memiliki galat yang besar akibat melibatkan angka yang sangat besar. Persamaan tersebut dipakai untuk memprediksi nilai f(7.516), f(8.323), f(9.167), serta f(8) yang masing-masing merupakan nilai prediksi jumlah kasus positif baru Covid-19 pada tanggal 16 Juli 2022, 10 Agustus 2022, 5 September 2022, dan 31 Juli 2022.

Pada Regresi Berganda, perhitungan pada SPL pun diterapkan. Contohnya pada contoh kasus 6, data-data percobaan Nitrous Oxide yang diberikan diubah menjadi sistem persamaan linear, lalu matriks augmentednya dieliminasi menggunakan metode Gauss-Jordan agar didapat persamaan regresi linear berganda yaitu

f(x) = -3.50777814 - 0.00262499x1 + 7.9894E-4x2 + 0.15415503x3

Selanjutnya, persamaan regresi tersebut dipakai untuk mengestimasi Nitrous Oxide ketika Humidity bernilai 50%, temperatur 76°F, dan tekanan udara sebesar 29.30. Didapatkan estimasi nilai Nitrous Oxide sebesar 0.93843423.

Pada Interpolasi Bikubik digunakan juga SPL saat mencari koefisien dari nilai tiap *a*, dimana persamaan berbentuk *y* = *Xa*, dan akan dicari nilai dari setiap *a*. Didapatkan hasil sebagai berikut.

$$f(0,0) = 161$$
$$f(0.5,0.5) = 97.7265625$$
$$f(0.25,0.75) = 105.5147705078125$$
$$f(0.1,0.9) = 104.2291185$$

Interpolasi Bikubik juga dapat juga digunakan untuk memperbesar gambar (bonus). Pada kasus ini kita mencoba memperbesar “.png” file dengan ukuran 256 x 256 pixel sebesar dua kali lipat.

Gambar awal (256 x 256)

Gambar setelah diperbesar (512 x 512)

Kita juga dapat menggandakan ukuran gambar hasil sehingga memperoleh perbesaran 4 kali dan seterusnya. Namun, tentunya proses ini akan semakin mengurangi resolusi dari gambar itu sendiri.

Saran untuk selanjutnya, bisa disertakan solusi untuk setiap studi kasus yang ada. Hal ini agar memudahkan kami untuk pengecekan hasilnya. Selain itu, menambahkan referensi untuk setiap persoalan terutama regresi linear berganda yang penjelasannya sangat berkecukupan.

Untuk tugas ini, kami merasa cukup puas dan bekerja secara maksimal karena waktu yang kami taruh untuk tugas ini begitu disiplin dan teratur. Kami juga ingin mengucapkan terima kasih kepada seluruh teman-teman yang sudah meluangkan waktunya untuk berdiskusi mengenai tugas ini. Secara keseluruhan, dapat dikatakan kinerja dan kekompakan tim sudah cukup baik dan mantap.

Demo dan GitHub

Demo

Hari, tanggal : Sabtu, 15 Oktober 2022

Pukul : 19.00 WIB

Asisten : Paranoia

GitHub

https://github.com/bangkitdc/Algeo01-21055

Referensi

- Howard Anton, Elementary Linear Algebra, 11th edition, John Wiley and Sons, 2010.
- Robert G. Keys, Cubic Convolution Interpolation for Digital Image Processing, IEEE Transaction On Acoustics, Speech, and Signal Processing, VOL. ASSP-29, NO. 6, December 1981.
- Daniel B. Rowe, BiLinear, Bicubic, and In Between Spline Interpolation, MU MSCS Spring, 2018.
- Nov 23, 2016. Bicubic Interpolation - Computerphile [Video]. *Youtube*. https://www.youtube.com/watch?v=poY_nGzEEWM
- Mar 29, 2020. Image Interpolation Examples (Introduction) - Riazul Islam [Video]. *Youtube*. https://www.youtube.com/watch?v=8bTDssnJyZc