Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Tugas Besar 1 IF2123 Aljabar Linier dan Geometri
Sistem Persamaan Linier, Determinan, dan Aplikasinya
Semester I Tahun 2023/2024

Kelompok 50 - “cepetBeres”

Kresna Adi Prayogo	13522017
Ahmad Mudabbir Arif	13522072
Berto Richardo Togatorop	13522118

Daftar Isi

1. Deskripsi Masalah

Sistem persamaan linier (SPL) banyak ditemukan di dalam bidang sains dan rekayasa. Anda sudah mempelajari berbagai metode untuk menyelesaikan SPL, termasuk menghitung determinan matriks. Sebuah SPL dengan *m* buah persamaan dan *n* variabel berbentuk Ax = b berikut,

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

- SPL dapat dinyatakan secara ringkas dalam bentuk matriks *augmented*:

$$[A \mid \mathbf{b}] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]$$

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).

$$\begin{bmatrix} 0 & \mathbf{2} & 1 & -1 \\ 0 & 0 & \mathbf{3} & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad \begin{bmatrix} 0 & \mathbf{1} & 0 & -\frac{2}{3} \\ 0 & 0 & \mathbf{1} & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

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

Di dalam Tugas Besar 1 ini, kami membuat satu atau lebih *library* aljabar linier dalam Bahasa Java. Library tersebut berisi fungsi-fungsi seperti eliminasi Gauss, eliminasi Gauss-Jordan, menentukan balikan matriks, menghitung determinan, kaidah Cramer (kaidah Cramer khusus untuk SPL dengan *n* peubah dan *n* persamaan). Selanjutnya, kami menggunakan *library* tersebut di dalam program Java untuk menyelesaikan berbagai persoalan yang dimodelkan dalam bentuk SPL, menyelesaikan persoalan interpolasi, dan persoalan regresi.

2. Teori Singkat

A. Metode Eliminasi Gauss

Eliminasi Gauss merupakan salah satu metode yang digunakan untuk mempermudah mendapatkan penyelesaian Sistem Persamaan Lanjar (SPL). Sesuai dengan namanya, eliminasi ini ditemukan oleh matematikawan asal Jerman, Carl Friedrich Gauss. SPL terlebih dahulu direpresentasikan menjadi matriks *augmented*, yang merupakan hasil gabungan dari matriks koefisien (A) dan konstanta hasil (b) pada persamaan SPL Ax = b. Matriks tersebut akan direduksi menjadi matriks eselon baris. Matriks eselon baris akan memiliki *leading* 1 dengan tiga syarat utama,

1. Jika sebuah baris tidak terdiri dari seluruhnya nol, maka bilangan tidak nol pertama di dalam baris tersebut adalah 1 (disebut 1 utama)
2. Jika ada baris yang seluruhnya nol, maka semua baris itu dikumpulkan pada bagian bawah matriks.
3. Di dalam dua baris berturutan yang tidak seluruhnya nol, maka 1 utama pada baris yang lebih rendah terdapat lebih jauh ke kanan daripada 1 utama pada baris yang lebih tinggi.

- Contoh-contoh matriks eselon baris:

$$\begin{bmatrix} 1 & 2 & 7 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 2 & 6 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- Bukan matriks eselon baris:

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

- Ciri-ciri matriks eselon baris: memiliki semua nilai nol di bawah 1 utama

Metode ini perlu melakukan 3 operasi utama pada SPL, yaitu operasi baris elementer atau akan disingkat menjadi OBE. Tiga OBE tersebut adalah sebagai berikut,

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

Dengan menggunakan 3 operasi tersebut, tujuannya adalah mereduksi matriks augmented SPL menjadi lebih sederhana berbentuk matriks eselon baris supaya bisa lebih mudah melakukan substitusi mundur untuk menentukan setiap nilai x.

B. Metode Eliminasi Gauss-Jordan

Pada dasarnya, eliminasi Gauss-Jordan memiliki metode yang mirip dengan metode Gauss. Akan tetapi, perbedaan yang mendasar terletak pada hasil akhirnya. Metode eliminasi Gauss-Jordan akan lebih mereduksi matriks augmented yang sudah dijalankan fase majunya menjadi matriks yang lebih tereduksi setelah dilakukan OBE pada fase maju, yaitu matriks eselon baris tereduksi. Seperti syarat matriks eselon baris, tetapi dengan tambahan satu syarat bahwa setiap kolom yang memiliki 1 utama memiliki nol di tempat lain. Berikut contohnya,

- Contoh-contoh matriks eselon baris tereduksi:

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} \quad \begin{bmatrix}1&0&0&4\\0&1&0&7\\0&0&1&-1\end{bmatrix} \quad \begin{bmatrix}0&1&-2&0&1\\0&0&0&1&3\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix} \quad \begin{bmatrix}0&0\\0&0\end{bmatrix}$$

- Bukan matriks eselon baris tereduksi:

$$\begin{bmatrix}0&1&0\\1&0&0\\0&0&0\end{bmatrix} \quad \begin{bmatrix}1&0&0&5\\0&0&1&3\\0&1&0&8\end{bmatrix} \quad \begin{bmatrix}0&1&-2&0&1\\0&0&0&1&3\\0&0&0&0&0\\0&0&0&0&1\end{bmatrix}$$

Mengapa?

Untuk melanjutkan solusi metode Gauss dan metode Gauss Jordan dan mengacu pada matriks *augmented* Ax = b, ada tiga kemungkinan solusi yang dapat terjadi,

1. Solusi unik, yang ekuivalen dengan matriks A memiliki determinan tidak sama dengan 0 (nol) atau matriks A memiliki invers/balikan (A matriks singular),
2. Banyak solusi, atau bisa dikatakan tak hingga solusi,
3. Tidak memiliki solusi.

Pada persamaan dua dan tiga variabel, solusi SPL tersebut juga bisa dinyatakan secara geometris dengan beberapa kemungkinan kondisi, misalnya garis berimpit, bidang saling paralel, bidang-bidang persamaan bertemu pada satu titik atau membentuk garis atau bahkan tidak pernah bertemu pada satu titik temu, dan berbagai kondisi geometris lainnya.

Pada program kami, setelah melakukan metode Gauss atau Gauss-Jordan, kami menganalisis untuk menentukan kemungkinan solusi yang mana dari ketiga solusi di atas sesuai dengan kondisi SPL,

1. Jika terdapat baris yang bernilai 0 pada semua kolomnya, kecuali pada ujung kanan berupa konstanta bukan 0, maka SPL tersebut tidak memiliki solusi,
2. Jika kasus 1 tidak terpenuhi dan matriks memiliki *leading* berbentuk diagonal, maka SPL memiliki solusi unik,
3. Jika kedua kasus di atas tidak terpenuhi, maka SPL memiliki tak hingga solusi yang mana diwujudkan dalam persamaan parametrik antar variabel x.

C. Determinan dan Kaidah Cramer

Determinan suatu matriks memiliki berbagai kegunaan, salah satunya adalah pada penentuan suatu sifat matriks serta aplikasi-aplikasi pada perhitungan luas dan volume dua dan tiga dimensi. Jika matriks memiliki nilai determinan 0 (nol), maka matriks tersebut tidak memiliki invers sehingga SPL yang direpresentasikan tidak memiliki solusi unik, kemungkinan tidak memiliki solusi atau memiliki tak hingga solusi. Pada projek kali ini, kami membatasi untuk mendapatkan determinan matriks persegi, matriks yang memiliki ukuran baris dan kolom yang sama. Selain itu, determinan matriks persegi berukuran dua merepresentasikan perhitungan luas suatu *parallelogram* oleh dua nilai vektor berbasis dua, begitu pula mirip dalam perhitungan volume *parallelpide* untuk determinan matriks persegi berukuran tiga.

Determinan matriks persegi berukuran dua sangat mudah dihitung. Untuk ukuran tiga, bisa juga dihitung dengan metode Sarrus dengan mengekstensi matriks, yakni menambah dua kolom dari kolom satu dan dua. Selain itu, untuk ukuran matriks yang lebih besar, akan sulit dihitung nilai determinannya. Maka dari itu, kami membuat *library* yang berisi suatu fungsi untuk mencari suatu masukan matriks menggunakan 2 metode. Metode pertama adalah metode reduksi baris dengan memanfaatkan OBE dan beberapa aturan determinan untuk mereduksi matriks menjadi matriks segitiga atas (atau bawah) sehingga nilai determinannya adalah hasil perkalian nilai-nilai elemen pada diagonal matriks segitiga tersebut.

Table 1

Relationship	Operation
$\begin{vmatrix} ka_{11} & ka_{12} & ka_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = k \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$ $\det(B) = k\det(A)$	The first row of A is multiplied by k.
$\begin{vmatrix} a_{21} & a_{22} & a_{23} \\ a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = - \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$ $\det(B) = -\det(A)$	The first and second rows of A are interchanged.
$\begin{vmatrix} a_{11} + ka_{21} & a_{12} + ka_{22} & a_{13} + ka_{23} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$ $\det(B) = \det(A)$	A multiple of the second row of A is added to the first row.

Perhatikan juga bahwa jika A mengandung baris atau kolom yang seluruh nilainya nol, maka determinannya akan bernilai nol.

Nilai determinan juga digunakan untuk menentukan nilai kofaktor dari suatu matriks. Lalu, dari nilai kofaktor pada baris-baris dan kolom-kolom tersebut, bisa ditentukan matriks Adjoin-nya. Selanjutnya akan dibahas lebih lanjut pada sub bahasan matriks balikan. Selain itu, determinan juga digunakan untuk menyelesaikan solusi SPL menggunakan kaidah Cramer,

- Jika A**x** = **b** adalah SPL yang terdiri dari n persamaan linier dengan n peubah (variable) sedemikian sehingga det(A) ≠ 0, maka SPL tersebut memiliki solusi yang unik yaitu

$$x_1 = \frac{\det(A_1)}{\det(A)}, \quad x_2 = \frac{\det(A_2)}{\det(A)}, \quad \ldots, \quad x_n = \frac{\det(A_n)}{\det(A)}$$

yang dalam hal ini, A_j adalah matriks yang diperoleh dengan mengganti entri pada kolom ke-j dari A dengan entri dari matriks

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Penentuan nilai determinan dengan ekspansi kofaktor akan dijelaskan lebih lanjut pada sub bahasan matriks kofaktor dan adjoin.

D. Matriks Balikan

Matriks balikan sejatinya juga memiliki definisi yang sama pada sistem operasi-operasi matematis lainnya. Berdasarkan definisi, suatu matriks jika dikalikan dengan matriks balikan akan menghasilkan matriks identitas (I), begitu pula jika urutan operannya dibalik. Akan tetapi, perlu diingat bahwa perkalian matriks tidak bersifat komutatif.

Seperti yang sudah dijelaskan di atas, A tidak memiliki balikan atau disebut juga matriks singular, maka SPL Ax = b tidak memiliki solusi unik (bisa jadi tak memiliki solusi atau tak hingga solusi). Namun, jika A memiliki balikan, SPL tersebut memiliki solusi unik. Kami tidak akan membahas terlalu banyak sampai bahasan SPL homogen.

- Misalkan *A* adalah matriks persegi berukuran n x n. Balikan (*inverse*) matriks A adalah A^{-1} sedemikian sehingga $AA^{-1} = A^{-1}A = I$.

Matriks balikan juga berfungsi untuk menemukan solusi dari suatu SPL apabila nilai konstanta b bervariasi, Cukup mencari matriks balikan A, maka berapapun nilai b, x bisa didapatkan.

- Tinjau SPL $A\mathbf{x} = \mathbf{b}$. Kalikan kedua ruas persamaan dengan A^{-1}

$$\begin{aligned} (A^{-1})A\mathbf{x} &= (A^{-1})\,\mathbf{b} \\ I\mathbf{x} &= A^{-1}\,\mathbf{b} \qquad (\text{karena } A^{-1}A = I) \\ \mathbf{x} &= A^{-1}\,\mathbf{b} \qquad (\text{karena } I\mathbf{x} = \mathbf{x}) \end{aligned}$$

- Jadi, solusi SPL $A\mathbf{x} = \mathbf{b}$ adalah $\mathbf{x} = A^{-1}\,\mathbf{b}$

Pada kesempatan kali ini, kami menggunakan dua metode untuk mencari matriks balikan, yaitu dengan eliminasi Gauss-Jordan dan dengan Adjoin. Metode eliminasi Gauss-Jordan diterapkan secara simultan untuk A maupun I seperti matriks *augmented* pada representasi SPL.

- Untuk matriks A yang berukuran n x n, matriks balikannya, yaitu A^{-1}, dicari dengan cara berikut:

$$[A|I] \overset{\text{G-J}}{\sim} [I|A^{-1}]$$

yang dalam hal ini *I* adalah matriks identitas berukuran n x n.

Metode dengan menggunakan Adjoin akan dijelaskan lebih lanjut pada sub bahasan berikutnya.

E. Matriks Kofaktor dan Matriks Adjoin

Sebelum masuk ke definisi matriks kofaktor dan adjoin, perlu dimengerti definisi minor entri suatu elemen pada baris *i* dan kolom *j* (*Mij*). Misalkan suatu matriks persegi A berukuran *nxn* maka pada baris *i* dan kolom *j* elemennya adalah *aij*. Minor entri adalah determinan upa-matriks yang elemen-elemennya tidak berada pada baris *i* dan kolom *j*.

Berdasarkan definisi, kofaktor entri *aij* (*Cij*) berkorespondensi dengan minor entri hanya berbeda tanda tergantung nilai *i* dan *j*, dengan rumus sebagai berikut,

- Misalkan A adalah matriks berukuran n x n

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

- Didefinisikan:
  M_{ij} = minor entri a_{ij}
  = determinan upa-matriks (*submatrix*) yang elemen-elemennya tidak berada pada baris *i* dan kolom *j*
  $C_{ij} = (-1)^{i+j} M_{ij}$ = kofaktor entri a_{ij}

- Cara mengingat tanda positif dan negative untuk C_{ij} adalah dengan memperhatikan pola berikut:

$$\begin{bmatrix} + & - & + & - & \dots \\ - & + & - & + & \dots \\ + & - & + & - & \dots \\ - & + & - & + & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

Penentuan nilai determinan dengan ekspansi kofaktor dapat dilakukan dengan memilih baris atau kolom tertentu dengan tips memilih elemen nol terbanyak untuk mengefisiensi perhitungan. Apabila elemen nol tersebut bisa didapatkan lebih mudah, sangat disarankan untuk menggunakan OBE dengan tetap memperhatikan aturannya terhadap nilai determinan suatu matriks. Contohnya,

- Dengan menggunakan kofaktor, maka determinan matriks

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

dapat dihitung dengan salah satu dari persamaan berikut:

$$\det(A) = a_{11}C_{11} + a_{12}C_{12} + \dots + a_{1n}C_{1n}$$
$$\det(A) = a_{21}C_{21} + a_{22}C_{22} + \dots + a_{2n}C_{2n}$$
$$\vdots$$
$$\det(A) = a_{n1}C_{n1} + a_{n2}C_{n2} + \dots + a_{nn}C_{nn}$$

Secara baris

$$\det(A) = a_{11}C_{11} + a_{21}C_{21} + \dots + a_{n1}C_{n1}$$
$$\det(A) = a_{12}C_{12} + a_{22}C_{22} + \dots + a_{n2}C_{n2}$$
$$\vdots$$
$$\det(A) = a_{1n}C_{1n} + a_{2n}C_{2n} + \dots + a_{nn}C_{nn}$$

Secara kolom

Selanjutnya, matriks kofaktor dan matriks adjoin didefinisikan sebagai berikut,

Matriks Kofaktor

- Misalkan A adalah matriks n x n dan C_{ij} adalah kofaktor entri a_{ij}.
- Maka matriks kofaktor dari A adalah

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

- *Adjoin* dari A adalah transpose matriks kofaktor:

adj(A) = transpose matriks kofaktor

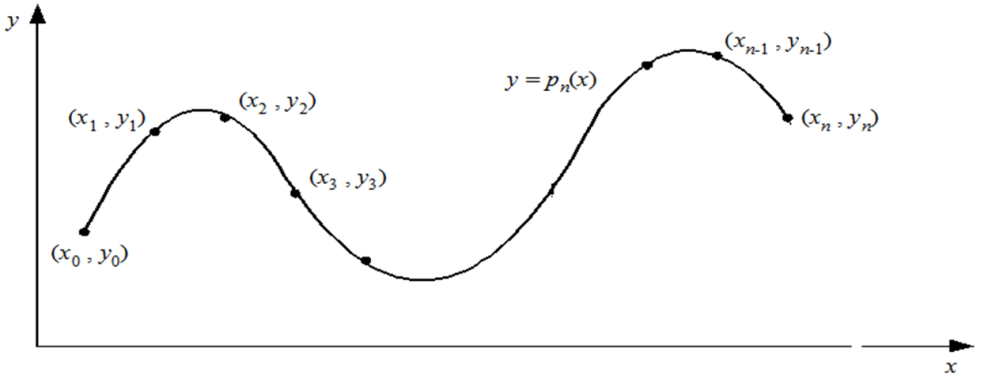
Setelah mendapatkan matriks adjoin tersebut dan nilai determinan matriks tersebut, maka dapat diperoleh matriks balikannya sebagai berikut,

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

Sebenarnya, kami masih belum sepenuhnya memahami *sense* kenapa matriks bisa direpresentasikan demikian dan dapat diterapkan dalam berbagai hal karena bagian matematisnya cukup susah untuk dipahami, sedangkan kami tidak memiliki waktu luang untuk mengeksplor ke bagian yang lebih detail.

F. Interpolasi Polinom

Tersedia n+1 buah titik berbeda, (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) maka dapat ditentukan polinom $p_n(x)$ yang menginterpolasi (melewati) semua titik-titik tersebut sehingga $y_i = p_n(x_i)$ untuk i = 0, 1, 2, …, n. Keperluan dan pemodelan sedemikian rupa agar SPL yang dibentuk memiliki solusi unik membentuk koefisien tiap suku x yang derajatnya bervariasi 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 $[x_0, x_n]$. Apabila menaksir nilai y di luar selang dinamakan **ekstra**polasi.

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

$$\begin{aligned} a_0 + a_1x_0 + a_2x_0^2 + \ldots + a_n x_0^n &= y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + \ldots + a_n x_1^n &= y_1 \\ \ldots \qquad\qquad & \ldots \\ a_0 + a_1x_n + a_2x_n^2 + \ldots + a_n x_n^n &= y_n \end{aligned}$$

G. Interpolasi *Bicubic Spline*

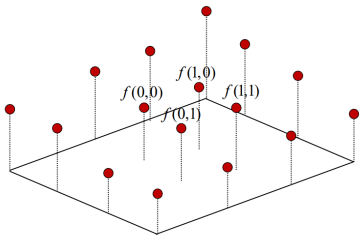
Bicubic spline interpolation adalah metode interpolasi yang digunakan untuk mengaproksimasi fungsi di antara titik-titik data yang diketahui. *Bicubic spline interpolation* melibatkan konsep *spline* dan konstruksi serangkaian polinomial kubik di dalam setiap sel segi empat dari data yang diberikan. 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: $f(0,0), f(1,0)$
$f(0,1), f(1,1)$

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,* mapun keduanya. Persamaan polinomial yang digunakan adalah sebagai berikut.

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

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

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

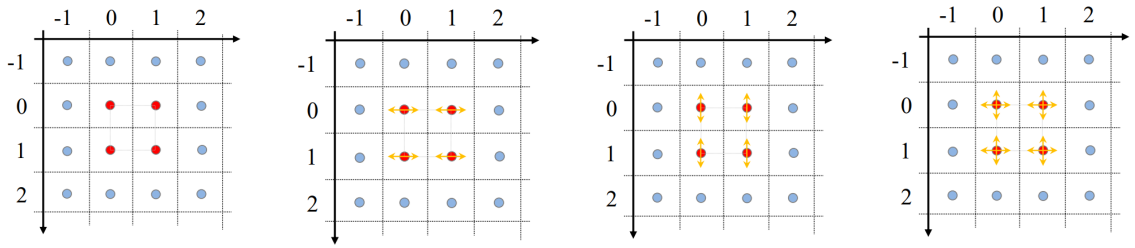
$$f_{xy}(x,y) = \sum_{j=0}^{3}\sum_{i=0}^{3} a_{ij} ij x^{i-1} y^{j-1}$$

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

$$y = Xa$$

$$\begin{bmatrix} f(0,0) \\ f(1,0) \\ f(0,1) \\ f(1,1) \\ f_x(0,0) \\ f_x(1,0) \\ f_x(0,1) \\ f_x(1,1) \\ f_y(0,0) \\ f_y(1,0) \\ f_y(0,1) \\ f_y(1,1) \\ f_{xy}(0,0) \\ f_{xy}(1,0) \\ f_{xy}(0,1) \\ f_{xy}(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 2 & 4 & 6 & 0 & 3 & 6 & 9 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \\ a_{01} \\ a_{11} \\ a_{21} \\ a_{31} \\ a_{02} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{03} \\ a_{13} \\ a_{23} \\ a_{33} \end{bmatrix}$$

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. 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. Kami berhasil membangun persamaan *f*(*x, y*) yang akan digunakan untuk melakukan interpolasi berdasarkan nilai *f*(*a, b*) dari masukan matriks 4 x 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).

H. Regresi Linier Berganda

Regresi Linear merupakan salah satu metode untuk memprediksi nilai selain menggunakan Interpolasi Polinom. Meskipun sudah ada rumus jadi untuk menghitung regresi linear sederhana, terdapat rumus umum dari regresi linear yang bisa digunakan untuk regresi linear berganda sebagai berikut.

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

Untuk mendapatkan nilai dari setiap β_1, dapat digunakan Normal Estimation Equation for Multiple Linear Regression sebagai berikut.

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

Bentuk ini merupakan sistem persamaan linier, Sistem persamaan linear tersebut dapat diselesaikan dengan menggunakan berbagai metode penyelesaian, seperti eliminasi Gauss-Jordan.

3. Implementasi Pustaka dan Program dalam Bahasa Java

Pustaka yang kami buat berisi Tipe Data Abstrak (ADT) Matrix beserta fungsi-fungsi primitifnya, termasuk *constructor, setter,* dan *getter*, serta berbagai fungsi yang digunakan dalam operasi aljabar linear elementer pada matriks, seperti metode eliminasi Gauss, metode eliminasi Gauss-Jordan, metode matriks balikan ($x = A^{-1}b$) dengan beberapa cara menemukan matriks balikan, dan kaidah *Cramer* (khusus untuk SPL dengan n peubah dan n persamaan) dengan beberapa cara menemukan determinan suatu matriks. Kami juga menerapkan beberapa prosedur operasi baris elementer dan beberapa operasi yang sering digunakan. Selain itu, kami memanfaatkan pustaka-pustaka terkait matriks untuk menentukan model penyelesaian matematika metode numerik, seperti Interpolasi Polinomial, Regresi Linear Berganda, dan Interpolasi *Bicubic Spline*, serta memanfaatkan Interpolasi *Bicubic Spline* untuk melakukan *resize* gambar, terutama memperbesar ukuran gambar. Kami juga mencoba menerapkan beberapa prinsip pemrograman berorientasi objek pada penggunaan bahasa Java pada kesempatan ini. Namun, program kami masih membatasi kapasitas matriks sejumlah pada awal didefinisikan, sampai diperlukan untuk melakukan input melalui prosedur readMatrix().

Struktur *class* (atribut dan metode), garis besar program, dll

☐ Folder operators

1. Kelas Matriks

- Atribut
  a. private int rowSize : menyimpan jumlah baris matriks saat ini
  b. private int colSize : menyimpan jumlah kolom matriks saat ini
  c. private double[][] Mat : menyimpan elemen-elemen dalam matriks

- Konstruktor
  a. public Matrix()
     Konstruktor membentuk objek Matrix kosong/belum terdefinisi dengan jumlah baris dan kolom 0.
  b. public Matrix(int row, int col)
     Konstruktor membentuk objek Matrix dengan jumlah baris dan kolom sesuai parameter row dan col dengan semua elemennya 0.

- *Getter*
  a. public int getRow()

Mengembalikan nilai rowSize

b. public int getCol()
   Mengembalikan nilai colSize
c. public double getELMT(int i, int j)
   Mengembalikan nilai elemen pada Mat baris ke-i kolom ke-j
d. public double[] getRowELMT(int i)
   Mengembalikan baris ke-i
e. public Matrix getMatConst(Matrix Mat)
   Mengembalikan konstanta matriks pada matriks augmented

- *Setter*
  a. public void setRow(int row)
     I.S. : Matrix terdefinisi
     F.S. : Ukuran baris matrix menjadi sebesar row
  b. public void setCol()
     I.S. : Matrix terdefinisi
     F.S. : Ukuran kolom matrix menjadi sebesar row
  c. public void setELMT(int i, int j, double val)
     I.S. : Matrix terdefinisi
     F.S. : Elemen pada matrix baris ke-i kolom ke-j menjadi val
  d. public void setRowELMT(int row, double[] rowELMT)
     I.S. : Matrix terdefinisi
     F.S. : Baris ke-i pada matrix menjadi berisi rowELMT
  e. public void setMat(Matrix Mat)
     I.S. : Matrix sembarang
     F.S. : Matrix direalokasi menjadi ukuran baris menjadi row dan kolom menjadi col dengan isi elemen kosong

- Metode Fungsi dan Prosedur Operators
  a. public void transpose()
     I.S. : Matrix terdefinisi
     F.S. : Matrix ditransposisikan dan atribut disesuaikan
  b. public void swapRow(int row1, int row2)
     I.S. : Matrix terdefinisi
     F.S. : Baris pada matrix ke-row1 dan ke-row2 ditukar
  c. public void subtractRow(int row, int subtractorRow, double subtractorMagnitude)
     I.S. : Matrix terdefinisi

F.S. : Baris pada matrix ke-row dikurangi subtractorMagnitude kali baris ke-subtractorRow

d. public void copyMatrix(Matrix Mat)
   I.S. : Matrix terdefinisi
   F.S. : Matrix Mat disalin ke matrix this
e. public static Matrix copyMatrix2(Matrix Mat)
   Mengembalikan salinan matrix Mat
f. public boolean isSquare()
   Mengembalikan kebenaran apakah matrix this adalah matrix persegi (berukuran n x n)
g. public static Matrix getMinorMat(Matrix Mat, int row, int col)
   Prekondisi: matriks persegi dan bukan matriks augmented
   Mengembalikan matrix minor untuk baris ke-row dan kolom ke-col
h. public static double getKofaktor(Matrix Mat, int row, int col)
   Prekondisi: matriks persegi dan bukan matriks augmented
   Mengembalikan matrix kofaktor untuk baris ke-row dan kolom ke-col

- Metode Fungsi dan Prosedur untuk Mencari Determinan
  a. public static double detKofaktor(Matrix Mat)
     Prekondisi: Matrix persegi dan bukan matriks augmented
     Mengembalikan determinan dari matrix mat menggunakan metode ekspansi kofaktor
  b. public static double detMatrixSegitiga(Matrix Mat)
     Prekondisi: Matrix persegi dan bukan matriks augmented
     Mengembalikan determinan dari matrix mat menggunakan OBE dan aturan determinannya supaya menjadi matrix segitiga

- Metode Fungsi dan Prosedur untuk Mencari Invers/Balikan
  a. public Matrix getAdj()
     Prekondisi: matriks persegi dan bukan matriks augmented
     Mengembalikan matrix adjoint
  b. public void scalarMultiply(double scale)
     I.S. : Matrix terdefinisi
     F.S. : Semua elemen matrix dikali dengan konstanta scale
  c. public Matrix inverseEkspansiCofactor()
     Prekondisi: matriks persegi atau matrix augmented yang bisa dibuat persegi
     Mengembalikan matrix balikan menggunakan metode ekspansi kofaktor
  d. public void pMultRow(int row, double scale)

I.S. : Matrix terdefinisi
F.S. : Baris ke-row pada matrix dikali dengan scale

e. public void strictGauss()
I.S. : Matrix terdefinisi, dipastikan matrix harus memiliki invers
F.S. : Matrix eselon baris dengan *leading 1* terdiagonalisasi sempurna

f. public void strictGaussJordan()
I.S. : Matrix terdefinisi, dipastikan matrix harus memiliki invers
F.S. : Matrix eselon baris tereduksi dengan *leading 1* terdiagonalisasi sempurna

g. public Matrix inverseGaussJordan()
Prekondisi: matriks persegi atau matrix augmented yang bisa dibuat persegi
Mengembalikan matrix balikan menggunakan metode eliminasi Gauss-Jordan

- Metode Fungsi dan Prosedur untuk Meminta Masukan dan Keluaran (I/O)
  a. public void readMatrix(int row, int col)
  I.S. : Matrix sembarang
  F.S. : Matrix terisi sesuai masukan pengguna dengan ukuran row x col
  b. public static Matrix readMatNXM()
  Mengembalikan matrix augmented sesuai masukan pengguna (untuk input menu SPL)
  c. public static Matrix readMatSquare()
  Mengembalikan matrix persegi sesuai masukan pengguna (untuk input menu determinan dan matriks invers)
  d. public static void printMatrix()
  Mengembalikan matrix augmented sesuai masukan pengguna (untuk input SPL)

- Beberapa metode yang tidak jadi dipakai tidak didokumentasikan di sini

2. Kelas SPL

- Metode Fungsi dan Prosedur untuk Mencari Solusi Persamaan Linear
  a. public static String metodeGauss(Matrix matrix)
  Prekondisi : Matrix augmented terdefinisi
  Mengembalikan string solusi dari persamaan linear yang diperoleh melalui metode eleminasi Gauss dan substitusi mundur/balik. String $x_i = k + t_1 + t_2 + .. + t_i$ dengan k adalah konstanta dan t_i adalah parameter untuk i solusi, jika SPL tidak memiliki solusi akan mengembalikan string “SPL tidak memiliki solusi“.
  b. public static String metodeGaussJordan(Matrix matrix)
  Prekondisi : Matrix augmented terdefinisi

Mengembalikan string solusi dari persamaan linear yang diperoleh melalui metode eleminasi Gauss-Jordan dan substitusi mundur/balik. String $x_i = k + t_1 + t_2 + .. + t_i$ dengan k adalah konstanta dan t_i adalah parameter untuk i solusi, jika SPL tidak memiliki solusi akan mengembalikan string "SPL tidak memiliki solusi".

c. public static String metodeInverse(Matrix matrix)
Prekondisi : Matrix augmented terdefinisi dan berukuran n x n+1 (Matrix A adalah persegi dan b)
Mengembalikan string solusi dari persamaan linear yang diperoleh melalui metode matriks balikan atau inverse dengan $x = A^{-1}b$, string $x_i = k$ dengan k adalah konstan. Jika matriks A terdefinisi tidak persegi maka mengembalikan string "Matriks yang dimasukkan tidak bisa menggunakan metode Inverse".

d. public static String kaidahCramer(Matrix matrix)
Prekondisi : Matrix augmented terdefinisi dan berukuran n x n+1 (Matrix A adalah persegi dan b)
Mengembalikan string solusi dari persamaan linear yang diperoleh melalui metode kaidah cramer dengan $x_i = det(A_j) / det(A)$, $det(A) \neq 0$ dalam hal ini A_j adalah matriks yang diperoleh dengan mengganti entri pada kolom ke-j dengan entri dari matriks b , string $x_i = k$ dengan k adalah konstan. Jika matriks A terdefinisi tidak persegi maka mengembalikan string "Matriks yang dimasukkan tidak bisa menggunakan metode Inverse".

- Metode Fungsi dan Prosedur Pendukung
  a. public static boolean isNoSolution(Matrix matrix)
  Mengembalikan true jika matrix tidak memiliki solusi.
  b. public static boolean isNonUnique(Matrix matrix)
  Mengembalikan true jika matrix memiliki solusi banyak atau non-unik.
  c. public static String parametric(Matrix matrix)
  Mengembalikan string dalam bentuk parametrik, $x_i = k + t_1 + t_2 + .. + t_i$ dengan k adalah konstanta dan t_i adalah parameter.

3. Kelas InterpolasiPolinom

- Atribut
  a. private static Scanner scanner: atribut untuk menerima masukan dari terminal pengguna
- Metode Fungsi dan Prosedur
  a. public static Matrix pointToMatrix(Matrix matPoint, int n)
  Prekondisi : matPoint dan n terdefinisi dengan matPoint adalah matrix yang berisi point (x, y) dan n adalah banyaknya titik.

Mengembalikan matrix dalam bentuk matriks augmented dari Polinom yang dihasilkan dari input n point.

b. public static double taksirNilai(double[] root, double x)
Prekondisi : root dan x terdefinisi, dengan root adalah array of double yang berisi konstanta dari Polinom yang terbentuk dan x adalah nilai yang ingin ditaksir.
Mengembalikan hasil taksiran dari subtitusi nilai x ke Polinom Interpolasi.

c. public static String isInterpolasiFromFile(boolean fromFile)
Mengembalikan string yang berisi persamaan Polinom Interpolasi dan nilai taksiran. Berdasarkan boolean fromFile, jika true maka Matrix point akan diambil dari file, dan jika false Matrix point akan diambil dari fungsi readFromKeyboard().

d. public static Matrix readFromKeyboard()
Mengembalikan Matrix Point dari masukkan user melalui keyboard.

4. Kelas RegresiLinearBerganda

- Atribut
  a. private static Scanner scanner: atribut untuk menerima masukan dari terminal pengguna
- Metode untuk mencari RegresiLinearBerganda
  a. public static void readRegresiFromKeyboard(Matrix ret, Matrix peubah)
  I.S : Matrix ret dan Matrix peubah bebas
  F.S : Matrix ret berisi sample data yang terdiri dari peubah dan hasilnya, matrix peubah berisi nilai yang ingin ditaksir
  b. public static Matrix getNormalEst(Matrix Maug)
  Mengembalikan Persamaan Normal Estimasi dari Matrix Maug
  c. public static double taksir(Matrix param, Matrix input){
  Mengembalikan hasil taksiran dari Matrix param berdasarkan data yang ingin di taksir pada Matrix input
  d. public static String isRegresiFromFile(boolean fromFile)
  Mengembalikan output berupa string yang berisi persamaan dan hasil taksiran dari data peubah yang di-*input* oleh user.

5. Kelas Bicubic

- Atribut
  a. private static Scanner scanner: atribut untuk menerima masukan dari terminal pengguna

- Metode fungsi untuk mendapatkan beberapa matrix dan nilai taksiran sesuai model interpolasi *bicubic spline*
    a. public static Matrix getBicubicPolynomialMatrix()
    Mengembalikan matriks bikubik polinomial sesuai untuk model *bicubic spline interpolation*
    b. public static Matrix getInverseBicubicPolynomialMatrix()
    Mengembalikan balikan matriks model pada poin a
    c. public static Matrix getImageMatrix()
    Mengembalikan representasi matriks D pada persamaan y = DI yang mana y adalah data titik dan turunan berarah di sekitar dan I adalah data titik
    d. public static Matrix getImageValue(Matrix I)
    Mengembalikan nilai koefisien a suatu gambar yang sudah dimodelkan dengan interpolasi
    e. public static double getBicubicFunctionValue(Matrix fMat, double x, double y)
    Mengembalikan hasil taksiran pada titik (x, y) model interpolasi berdasarkan titik-titik data dan nilai turunan berarah di sekitarnya yang direpresentasi oleh matrix fMat

☐ Folder myUtils

1. Kelas myUtils

- Atribut
    a. private static Scanner scanner: atribut untuk menerima masukan dari terminal pengguna
- Metode
    a. public static boolean inputSrcValidation()
    Prekondisi: berada pada pilihan submenu tertentu
    Mengembalikan apakah pengguna ingin melakukan input melalui file
    b. public static double setPrec(double num, int decPlaces)
    Mengembalikan angka num dengan presisi sebanyak decPlaces angka belakang koma
    c. public static double strToDouble(String str)
    Mengembalikan nilai double dari suatu string str yang sudah divalidasi saat konversi
    d. public static Matrix readMatrixFromFile()
    Mengembalikan matrix dari nama file yang dimaksud pengguna
    e. public static void matrixToFile(Matrix mSimpan)
    I.S. : Matrix mSimpan terdefinisi

F.S. : Matrix mSimpan disimpan dalam sebuah file jika pengguna ingin menyimpannya

f. public static void strToFile(String s)
I.S. : String s terdefinisi
F.S. : String s disimpan dalam sebuah file jika pengguna ingin menyimpannya

☐ Folder menu (atribut scanner dan metode menu)

1. Kelas SPLMenu

- private static Scanner scanner: untuk meminta masukan pengguna melalui terminal
- public static void menu()
  I.S. : Berada pada menu utama
  F.S. : Berada pada menu untuk mencari solusi SPL yang direpresentasikan sebagai matriks augmented berdasarkan masukan pengguna

2. Kelas DeterminantMenu

- private static Scanner scanner: untuk meminta masukan pengguna melalui terminal
- public static void menu()
  I.S. : Berada pada menu utama
  F.S. : Berada pada menu untuk menghitung determinan suatu matriks berdasarkan masukan pengguna

3. Kelas InverseMenu

- private static Scanner scanner: untuk meminta masukan pengguna melalui terminal
- public static void menu()
  I.S. : Berada pada menu utama
  F.S. : Berada pada menu untuk mencari matriks balikan berdasarkan masukan pengguna

4. Kelas InterpolasiMenu

- private static Scanner scanner: untuk meminta masukan pengguna melalui terminal
- public static void menu()
  I.S. : Berada pada menu utama
  F.S. : Berada pada menu untuk menghitung taksiran berdasarkan model interpolasi polinom dari data masukan pengguna

5. Kelas RegresiMenu

- private static Scanner scanner: untuk meminta masukan pengguna melalui terminal

- public static void menu()
  I.S. : Berada pada menu utama
  F.S. : Output persamaan regresi dan hasil taksiran ditampilkan, kembali ke menu utama setelah sebelumnya berada pada menu untuk menghitung taksiran berdasarkan model regresi dari data masukan pengguna

6. Kelas BicubicMenu

- private static Scanner scanner: untuk meminta masukan pengguna melalui terminal
- public static void menu()
  I.S. : Berada pada menu utama
  F.S. : Berada pada menu untuk menghitung taksiran sekitar data pada titik dan turunan berarah sekitar dari masukan pengguna dengan model *bicubic spline interpolation*

4. Eksperimen

Hasil eksekusi pakai contoh-contoh kasus, serta analisis2

A. Solusi SPL $Ax = b$, Beberapa Determinan dan Invers Matriks Persegi

a.

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

Sesuai pada tes solusi SPL tak berhingga (parametrik), matriks tersebut tidak memiliki matriks balikan (dihitung dengan metode matriks adjoin) dan setelah menginput data matriks A nilai determinannya 0 (dihitung dengan metode ekspansi kofaktor).

Solusi SPL $Ax = b$

Determinan

```
1. Matrix Segitiga
2. Determinan Kofaktor
Pilih Metode penyelesaian :2
Anda akan menginput matriks segi empat dengan ukuran n x n.
Masukkan n: 4
Masukkan matriks dengan ukuran 4 x 4:
1 1 -1 -1
2 5 -7 -5
2 -1 1 3
5 2 -4 2
Determinan: 0.00000
```

Inverse

```
1. Ekspansi Kofaktor
2. Gauss-Jordan
Pilih Metode penyelesaian : 1
Masukkan nama file: 1_a.txt
Matriks persegi di atas tidak memiliki invers
```

b.

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

Solusi SPL $Ax = b$

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
Pilih Metode penyelesaian :2
Masukkan nama file: 1_b.txt
SPL memiliki solusi banyak
x1 = 3.0 + t5
x2 = -2.0t5
x3 = t3 dengan t3 adalah bilangan Real
x4 = -1.0 + t5
x5 = t5 dengan t5 adalah bilangan Real
```

Determinan

Inverse

c.

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

Solusi SPL $Ax = b$

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
Pilih Metode penyelesaian :2
Masukkan nama file: 1_c.txt
SPL memiliki solusi banyak
x1 = t1 dengan t1 adalah bilangan Real
x2 = 1.0
x3 = t3 dengan t3 adalah bilangan Real
x4 = t4 dengan t4 adalah bilangan Real
x5 = t5 dengan t5 adalah bilangan Real
x6 = t6 dengan t6 adalah bilangan Real
```

Determinan
Inverse

d.

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

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

Solusi SPL $Ax = b$

Determinan
Inverse

e. Solusi SPL dari masukan matriks augmented dan nilai determinan (dengan metode matriks segitiga) serta matriks balikan (dengan metode Gauss-Jordan) konstanta A-nya oleh pengguna melalui terminal,

Matriks *augmented*

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

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

Anda akan menginput matriks augme
Masukkan jumlah n: 3
Masukkan jumlah m: 4
3 4.5 2.8 10 12
-3 7 8.3 11 -4
0.5 -10 -9 12 0
Solusi dari SPL tersebut adalah:
x1 = -0.6880532212885146
x2 = 0.9546498599439779
x3 = 2.7743697478991587
```

Solusi SPL $Ax = b$
Determinan
Inverse

Matriks A dalam $Ax = b$

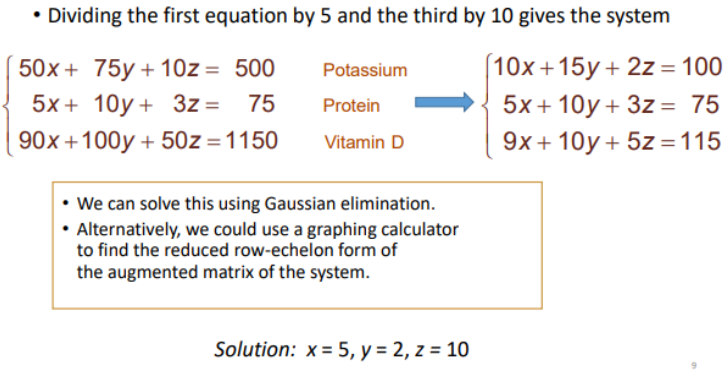
```
3 4.5 2.8
-3 7 8.3
```

```
0.5 -10 -9
```

```
1. Ekspansi Kofaktor
2. Gauss-Jordan
Pilih Metode penyelesaian : 2
Anda akan menginput matriks segi empat dengan ukuran n x n.
Masukkan n: 3
Masukkan matriks dengan ukuran 3 x 3:
3 4.5 2.8
-3 7 8.3
0.5 -10 -9
Matriks Balikan dari matriks persegi di atas adalah:
0.6375 0.3984 0.5657
-0.7283 -0.9052 -1.0614
0.8446 1.0279 1.0996
```

```
1. Matrix Segitiga
2. Determinan Kofaktor
Pilih Metode penyelesaian :1
Anda akan menginput matriks segi empat dengan ukuran n x n.
Masukkan n: 3
Masukkan matriks dengan ukuran 3 x 3:
3 4.5 2.8
-3 7 8.3
0.5 -10 -9
Determinan: 31.37500
```

f. Aplikasi analisis nutrisi pada bidang ilmu gizi dan kesehatan,



Karena solusi unik, dapat dibuktikan juga bahwa matriks tersebut memiliki invers dan determinannya tidak 0,

Solusi SPL $Ax = b$

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

Anda akan menginput matriks augmented dengan ukuran n x m.
Masukkan jumlah n: 3
Masukkan jumlah m: 4
10 15 2 100
5 10 3 75
9 10 5 115
Solusi dari SPL tersebut adalah:
x1 = 5.0
x2 = 2.0
x3 = 10.0
```

Determinan
Inverse

```
1. Matrix Segitiga
2. Determinan Kofaktor
Pilih Metode penyelesaian :2
Anda akan menginput matriks segi empat dengan ukuran n x n.
Masukkan n: 3
Masukkan matriks dengan ukuran 3 x 3:
10 15 2
5 10 3
9 10 5
Determinan: 150.00000
```

```
1. Ekspansi Kofaktor
2. Gauss-Jordan
Pilih Metode penyelesaian : 1
Anda akan menginput matriks segi empat dengan ukuran n x n.
Masukkan n: 3
Masukkan matriks dengan ukuran 3 x 3:
10 15 2
5 10 3
9 10 5
adjoinnya (sblm dibagi det 150.000000):
20.0000 -55.0000 25.0000
2.0000 32.0000 -20.0000
-40.0000 35.0000 25.0000
Matriks Balikan dari matriks persegi di atas adalah:
0.1333 -0.3667 0.1667
0.0133 0.2133 -0.1333
-0.2667 0.2333 0.1667
```

B. SPL Matriks Augmented,

a.

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

Karena matriks tersebut memiliki solusi parametrik, maka tidak ada invers dan determinannya 0,

Solusi SPL $Ax = b$

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
Pilih Metode penyelesaian :2
Masukkan nama file: 2_a.txt
SPL memiliki solusi banyak
x1 = -1.0 + x4
x2 = -2.0x3
x3 = t3 dengan t3 adalah bilangan Real
x4 = t4 dengan t4 adalah bilangan Real
```

Determinan

```
1. Matrix Segitiga
2. Determinan Kofaktor
Pilih Metode penyelesaian :2
Anda akan menginput matriks segi empat
Masukkan n: 4
Masukkan matriks dengan ukuran 4 x 4:
1 -1 2 -1
2 1 -2 -2
-1 2 -4 1
3 0 0 -3
Determinan: 0.00000
```

Inverse

```
1. Ekspansi Kofaktor
2. Gauss-Jordan
Pilih Metode penyelesaian : 2
Masukkan nama file: 2_a.txt
Matriks persegi di atas tidak memiliki invers
```

b.

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

Solusi SPL $Ax = b$

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
Pilih Metode penyelesaian :2
Masukkan nama file: 2_b.txt
Solusi dari SPL tersebut adalah:
x2 = 2.0
x3 = 1.0
x4 = 1.0
```

Determinan
Inverse

C. SPL Berbentuk Persamaan Aslinya,

a.
$$8x_1 + x_2 + 3x_3 + 2x_4 = 0$$
$$2x_1 + 9x_2 - x_3 - 2x_4 = 1$$
$$x_1 + 3x_2 + 2x_3 - x_4 = 2$$
$$x_1 + 6x_3 + 4x_4 = 3$$

Ini solusinya x1 beda, ntar mau coba debug lagi, trus kalo pake invers beda ntar coba debug invers lagi sama, ngeprintnya juga nanti coba pake fungsi setPrec() di myUtils

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
Pilih Metode penyelesaian :1
Masukkan nama file: 3_a.txt
Solusi dari SPL tersebut adalah:
x1 = -0.2243243243243243
x2 = 0.18243243243243246
x3 = 0.7094594594594594
x4 = -0.258108108108108
```

```
1. Metode eliminasi Gauss
2. Metode eliminasi Gauss-Jordan
3. Metode matriks balikan
4. Kaidah Cramer
Pilih Metode penyelesaian :2
Masukkan nama file: 3_a.txt
Solusi dari SPL tersebut adalah:
x1 = -0.22432432432432436
x2 = 0.18243243243243246
x3 = 0.7094594594594594
x4 = -0.258108108108108
```

SPL tersebut memiliki solusi unik, akan dibuktikan juga bahwa SPL memiliki nilai determinan tidak sama dengan 0 dan memiliki invers,

```
1. Matrix Segitiga
2. Determinan Kofaktor
Pilih Metode penyelesaian :1
Anda akan menginput matriks segi empat
Masukkan n: 4
Masukkan matriks dengan ukuran 4 x 4:
8 1 3 2
2 9 -1 -2
1 3 2 -1
1 0 6 4
Determinan: 740.00000
```

```
1. Ekspansi Kofaktor
2. Gauss-Jordan
Pilih Metode penyelesaian : 2
Masukkan nama file: 3_a.txt
Matriks Balikan dari matriks persegi di atas adalah:
0.1378 -0.0189 0.0108 -0.0757
-0.0338 0.1419 -0.0811 0.0676
-0.0203 -0.1149 0.3514 0.0405
-0.0041 0.1770 -0.5297 0.2081
```

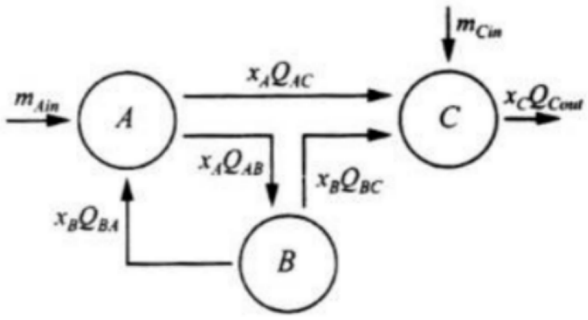
b.

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

.

Solusi SPL $Ax = b$
1. Metode eliminasi Gauss 2. Metode eliminasi Gauss-Jordan 3. Metode matriks balikan 4. Kaidah Cramer Pilih Metode penyelesaian :2 Masukkan nama file: 3_b.txt SPL tidak memiliki solusi
Determinan
Inverse

D. SPL untuk Sistem Reaktor, Determinan dan Invers Konstanta A-nya



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

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

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

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

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 $Q_{Cout} = 150$ m^3/s dan $m_{Ain} = 1300$ dan $m_{Cin} = 200$ mg/s.

Karena SPL tersebut pasti memiliki solusi unik, maka dapat dibuktikan juga bahwa matriks A SPL tersebut memiliki balikan dan determinanya bukan 0,

```
1. Ekspansi Kofaktor
2. Gauss-Jordan
Pilih Metode penyelesaian : 2
Masukkan nama file: 4.txt
Matriks Balikan dari matriks persegi di atas adalah:
-0.0111 -0.0083 0.0000
-0.0056 -0.0167 0.0000
-0.0067 -0.0067 -0.0067
```

```
1. Matrix Segitiga
2. Determinan Kofaktor
Pilih Metode penyelesaian :1
Anda akan menginput matriks segi empat
Masukkan n: 3
Masukkan matriks dengan ukuran 3 x 3:
-120 60 0
40 -80 0
80 20 -150
Determinan: -1080000.00000
```

E. Studi Kasus Interpolasi

1) Pengujian beberapa titik x dari interpolasi pasangan titik-titik pada tabel berikut,

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:

$x = 0.2$ $f(x) = ?$
$x = 0.55$ $f(x) = ?$
$x = 0.85$ $f(x) = ?$
$x = 1.28$ $f(x) = ?$

\

```
Masukkan nama file: 5_a1.txt
Polinom interpolasi yang melalui ke-7 buah titik tersebut adalah
p_6(x) = -0.0000x^6 + 0.0000x^5 + 0.0260x^4 + 0.0000x^3 + 0.1974x^2 + 0.2400x - 0.0230
p_6(0.2000) = 0.0330
```

```
Masukkan nama file: 5_a2.txt
Polinom interpolasi yang melalui ke-7 buah titik tersebut adalah
p_6(x) = -0.0000x^6 + 0.0000x^5 + 0.0260x^4 + 0.0000x^3 + 0.1974x^2 + 0.2400x - 0.0230
p_6(0.5500) = 0.1711
```

```
Masukkan nama file: 5_a3.txt
Polinom interpolasi yang melalui ke-7 buah titik tersebut adalah
p_6(x) = -0.0000x^6 + 0.0000x^5 + 0.0260x^4 + 0.0000x^3 + 0.1974x^2 + 0.2400x - 0.0230
p_6(0.8500) = 0.3372
```

```
Masukkan nama file: 5_a4.txt
Polinom interpolasi yang melalui ke-7 buah titik tersebut adalah
p_6(x) = -0.0000x^6 + 0.0000x^5 + 0.0260x^4 + 0.0000x^3 + 0.1974x^2 + 0.2400x - 0.0230
p_6(1.2800) = 0.6775
```

Keterangan: Pada polinom interpolasi diatas, terdapat konstanta koefisien yang bernilai 0.000, hal tersebut dikarenakan ada format penulisan yaitu 4 angka dibelakang koma yang dimana angka sebenarnya adalah tidak 0 melainkan angka dengan nilai yang sangat kecil.

…

2) Jumlah kasus Covid-19 di Indonesia dengan parameter tanggal x direpresentasikan dan diolah menjadi bentuk desimal,

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

Dengan aturan sebagai berikut dan prediksi 3 data berikut,

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

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

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.

Ini yg testcase d perlu bikin lagi di menu, khusus studi kasus ini (?) hmm ribet, keknya ga perlu.

Apakah kakaknya bakal cek 1 1 studi kasus, apakah semua studi kasus wajib dilakukan?

```
Masukkan nama file: 5_b1.txt
Polinom interpolasi yang melalui ke-10 buah titik tersebut adalah
p_9(x) = -140994.5598x^9 + 9372906.0628x^8 - 275476226.8079x^7 + 4695835438.3742x^6 - 51132198648.8689x^5 + 36
8553169420.9829x^4 - 1756821693899.8840x^3 + 5334238927153.8950x^2 - 9347057986137.9340x + 7187117988941.8590
p_9(7.5160) = 53536.6680
```

```
Masukkan nama file: 5_b2.txt
Polinom interpolasi yang melalui ke-10 buah titik tersebut adalah
p_9(x) = -140994.5598x^9 + 9372906.0628x^8 - 275476226.8079x^7 + 4695835438.3742x^6 - 51132198648.8689x^5 + 36
8553169420.9829x^4 - 1756821693899.8840x^3 + 5334238927153.8950x^2 - 9347057986137.9340x + 7187117988941.8590
p_9(8.3230) = 36292.8398
```

```
Masukkan nama file: 5_b3.txt
Polinom interpolasi yang melalui ke-10 buah titik tersebut adalah
p_9(x) = -140994.5598x^9 + 9372906.0628x^8 - 275476226.8079x^7 + 4695835438.3742x^6 - 51132198648.8689x^5 + 36
8553169420.9829x^4 - 1756821693899.8840x^3 + 5334238927153.8950x^2 - 9347057986137.9340x + 7187117988941.8590
p_9(9.1670) = -667699.8594
```

3) Penyederhanaan fungsi berikut menjadi bentuk fungsi polinom,

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

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 $h = (2 - 0)/5 = 0.4$.

```
Masukkan nama file: 5_c.txt
Polinom interpolasi yang melalui ke-6 buah titik tersebut adalah
p_5(x) = 0.2363x^5 - 1.4213x^4 + 3.2371x^3 - 3.5527x^2 + 2.0353x
p_5(0.0000) = 0.0000
```

...

4) Masukan titik-titik dan nilai x yang ditaksir oleh pengguna melalui terminal,

```
8.0 2.0794
9.0 2.1972
9.5 2.2513
8.3
```

```
1. Masukan dari file
2. Masukan dari keyboard
Pilih Sumber input : 2
Masukkan jumlah titik: 3
8 2.0794
9 2.1972
9.5 2.2513
Masukkan nilai x: 8.3
Polinom interpolasi yang melalui ke-3 buah titik tersebut adalah
p_2(x) = -0.0064x^2 + 0.2266x + 0.6762
p_2(8.3000) = 2.1161
```

F. Studi Kasus Regresi Linear Berganda

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.

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

```
Masukkan nama file: 6.txt
y = -3.50778 + -0.00262 x1 + 0.00080 x2 + 0.15416 x3, f(50.00000, 76.00000, 29.30000) = 0.938434
```

G. Studi Kasus Interpolasi *Bicubic Spline*

1) Dari file txt

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

Tentukan nilai:

$f(0, 0)$ = ?
$f(0.5, 0.5)$ = ?
$f(0.25, 0.75)$ = ?
$f(0.1, 0.9)$ = ?

```
Masukkan nama file: 7_a.txt
21.0 98.0 125.0 153.0
51.0 101.0 161.0 59.0
0.0 42.0 72.0 210.0
16.0 12.0 81.0 96.0
f(0.0, 0.0) = 21.0
```

```
Masukkan nama file: 7_b.txt
21.0 98.0 125.0 153.0
51.0 101.0 161.0 59.0
0.0 42.0 72.0 210.0
16.0 12.0 81.0 96.0
f(0.5, 0.5) = 87.796875
```

```
Masukkan nama file: 7_c.txt
21.0 98.0 125.0 153.0
51.0 101.0 161.0 59.0
0.0 42.0 72.0 210.0
16.0 12.0 81.0 96.0
f(0.25, 0.75) = 82.148193359375
```

```
Masukkan nama file: 7_d.txt
21.0 98.0 125.0 153.0
51.0 101.0 161.0 59.0
0.0 42.0 72.0 210.0
16.0 12.0 81.0 96.0
f(0.1, 0.9) = 91.271267
```

2) Dari input pengguna melalui terminal

Misalnya jika nilai dari $f(0, 0)$, $f(1, 0)$, $f(0, 1)$, $f(1, 1)$, $f_x(0, 0)$, $f_x(1, 0)$, $f_x(0, 1)$, $f_x(1, 1)$, $f_y(0, 0)$, $f_y(1, 0)$, $f_y(0, 1)$, $f_y(1, 1)$, $f_{xy}(0, 0)$, $f_{xy}(1, 0)$, $f_{xy}(0, 1)$, $f_{xy}(1, 1)$ berturut-turut adalah 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 serta nilai a dan b yang dicari berturut-turut adalah 0.5 dan 0.5 maka isi *file text* ditulis sebagai berikut:

```
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
0.5 0.5
```

Luaran yang dihasilkan adalah nilai dari $f(0.5, 0.5)$.

```
Masukkan jumlah titik konfigurasi nilai fungsi dan turunan berarah di sekitarnya,
diikuti dengan nilai a dan b untuk mencari taksiran f(a, b):
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
0.5 0.5
1.0 2.0 3.0 4.0
5.0 6.0 7.0 8.0
9.0 10.0 11.0 12.0
13.0 14.0 15.0 16.0
a = 0.5, b = 0.5
f(0.5, 0.5) = 2.125
```

5. Penutup

A. Kesimpulan

Banyak sekali aplikasi yang dapat kita buat berhubungan dengan Sistem Persamaan Linear, contohnya adalah apa yang sudah kami buat pustaka programnya di atas. Kami berhasil membuat kalkulator matriks yang mampu menyelesaikan permasalahan-permasalahan SPL dengan menggunakan berbagai metode seperti eliminasi Gauss dan Gauss-Jordan dengan prosedur langkah-langkah OBE, nilai determinan, matriks balikan, adjoin, dan lain sebagainya. Selain itu, kami juga berhasil menerapkannya pada berbagai studi kasus SPL, interpolasi polinom, regresi linear berganda, bahkan interpolasi *bicubic spline* yang menghasilkan aproksimasi lebih mulus daripada interpolasi lainnya dalam representasi titik-titik permukaan pada bidang dua dimensi. Akan lebih baik, jika kami memiliki waktu luang untuk menuntaskan bonusnya karena bisa menjadi portofolio tambahan. Karena pemrosesan gambar akan berjalan cukup lama, kami menyimpan tugas ini kemudian untuk mencoba diterapkan pada konsep multi-proses menerapkan *asynchronous code* dan *multi-threading*.

Kami menerapkan ADT yang sudah kami pelajari memakai bahasa C dari mata kuliah Algoritma dan Struktur Data serta menggabungkan implementasinya menggunakan standar yang biasa digunakan dalam *best practice* memprogram bahasa Java. Kami juga berhasil menerapkan *error-handling* dengan memanfaatkan blok-blok *try-catch* dan beberapa objek *Exceptions* sehingga lebih stabil.

B. Saran

Saran untuk kedepannya adalah dimohon bagi asisten tugas besar untuk memberi *guidelines* karena tidak semua kelompok memiliki pengalaman yang cukup dalam mengerjakan suatu projek. Hal ini ditujukan agar proses pengerjaan bisa sesuai dengan *workload* yang dibutuhkan, tidak lebih bahkan bisa dikurangi seefisien (sesangkil) mungkin. Asisten dimohon untuk memberi cara melakukan *automated testing* yang lebih baik karena proses testing ini cukup mengonsumsi banyak waktu. Yang paling penting, kami mohon juga untuk asisten memberi solusi tiap studi kasus agar tidak membebani banyak kelompok mencari tiap solusi yang pasti, atau kami sarankan membuat forum yang bisa mengumpulkan solusi mayoritas yang sudah diselesaikan oleh berbagai kelompok yang sudah melakukan submisi. Saya berharap asisten bisa memberi saran yang bagus saat demo untuk kami mengerjakan tubes-tubes yang akan dirilis setelah UTS. Pertanyaan kepada asisten dan senior,

1. Apa *tools* yang bisa digunakan untuk menghindari *bugs* maupun untuk mempermudah dan mempercepat proses *debugging*?
2. Apa *tools* yang bisa digunakan dalam membantu mengerjakan dokumen, seperti mengotomasi proses memasukkan struktur *class* pada program kami agar bisa lebih otomatis di-*generate* ke laporan?

3. Apa *tools* yang bisa digunakan untuk melakukan *automated testing* supaya proses studi kasus tidak mengonsumsi terlalu banyak waktu karena bisa mengalokasikan waktu lebih banyak mempelajari hal lain, seperti mengerjakan bonus dan mengimplementasikan GUI?
4. Bagaimana *guidelines* dan *best practice* dalam proses pengerjaan maupun cara menggunakan *tools* tersebut sejauh ini dan bisa diringkas untuk mempermudah menangani *common errors*? Mungkin boleh juga disarankan metode-metode baru dalam pengerjaan tugas yang sudah ditemukan oleh berbagai kelompok dari berbagai angkatan untuk bisa diterapkan ke angkatan-angkatan baru seperti kami sehingga generasi penerus bisa berkembang lagi lebih pesat.

C. Komentar

Pada kesempatan tugas besar kali ini, kami tidak sempat menyelesaikan bonus yang cukup seru untuk dikerjakan karena kendala-kendala seperti yang sudah saya sampaikan di bagian saran dan refleksi di bawah walaupun kami sudah berusaha semaksimal mungkin dan mengerjakan secara rajin dan teratur. Apabila kami tidak mengalami kendala tersebut dan memulai dengan baik dan benar sesuai *best practice* dan *guidelines*, kami yakin bisa menuntaskan semua spesifikasi bonus bahkan membuat GUI-nya. Menurut kami, tampilan CLI dan *text-based* masih cukup susah dan tidak efisien untuk digunakan oleh pengguna lain.

D. Refleksi (hasil yang dicapai, saran pengembangan, dan refleksi anda terhadap tugas ini)

Hasil yang dicapai sudah cukup baik, tetapi kami pikir kinerja kami masih kurang baik karena beberapa kendala yang belum kami sadari sebelumnya. Seharusnya kami bisa menerapkan konsep OOP dengan membuat semua metode dalam tiap kelas menjadi *protected* untuk dimanfaatkan pada konsep *inheritance*.

Saran kami kedepannya adalah untuk bisa lebih memanage prioritas dan waktu terhadap tugas-tugas dalam spesifikasi supaya tidak hanya produktif pada ujung *deadline*. Hal ini perlu dilakukan agar tidak menjadi *pressure*/stress/tekanan yang bakal mempengaruhi kesehatan juga. Namun, kami masih belum terbayang bagaimana jika kesalahan-kesalahan dan hambatan-hambatan yang sama mengancam tugas besar selanjutnya. Kami hanya bisa berdoa dan berusaha semaksimal mungkin, selalu mengevaluasi dan merefleksi diri agar bisa mengerjakan lebih efektif dan efisien dari sebelumnya. Kemampuan yang mendorong kita untuk *fast-pace* mengikuti alur perkuliahan sangat berpengaruh menggali potensi kami dalam pemrograman sampai batas, bahkan mengekstensi batasnya dan meningkatkan kapasitas diri, tidak hanya kemampuan saja. Kami juga jadi lebih terbiasa dalam mengerjakan proyek pemrograman berkelompok dan berkolaborasi. Hal-hal tersebut sangat melatih kami untuk mempersiapkan diri dalam dunia industri agar meminimalisasi kesalahan-kesalahan yang akan terjadi dan mampu bekerja lebih produktif sehingga mampu berdampak ke masyarakat luas.

Kami masih mengerjakan kurang rapi karena terlalu banyak catatan komentar yang tidak penting. Sebaiknya, untuk kedepannya kami perlu menulis lebih baik dalam komentar

dokumentasi agar lebih *well-commented* dan menggunakan bahasa yang mudah dan cepat dipahami oleh semua anggota kelompok. Dari sini, kami juga belajar bahwa untuk berkolaborasi, kami perlu kemampuan lebih untuk membaca program anggota lain dalam kelompok agar bisa saling mengevaluasi kesalahan, bahkan menemukan *bugs* yang tidak disadari antar anggota kelompok. Namun, kami berhasil melakukan modularisasi pembagian tugas dengan cukup baik. Akan tetapi, modularisasi pada implementasinya masih cukup kurang. Pada akhirnya, kami bisa memodularisasi lebih baik di akhir-akhir kesempatan sebelum pengumpulan.

Kami menggunakan nama kelompok "cepetBeres" dengan harapan agar bisa selesai lebih cepat dari apa yang kami harapkan. Akan tetapi, kenyataannya kami mengerjakan tugas besar ini lebih dari rata-rata *workload* pengerjaan pada umumnya. Hal ini disebabkan oleh pengetahuan dan pengalaman kami yang minim dalam mengerjakan suatu projek, berkolaborasi dengan github, dan beberapa kesalahan yang tidak kami sadari berdampak pada proses pengerjaan. Bersama tugas besar ini, kami menjadi lebih terbiasa dalam menerapkan berbagai fitur yang ada pada git karena jam terbang yang juga tinggi. Namun, untuk mempersiapkan tugas-tugas kedepannya, kami perlu melakukan *review* ulang konsep alur kerja dan fungsi-fungsi tiap *commands* agar bisa dimanfaatkan seoptimal mungkin, apalagi konsep git sebagai *version control system* untuk kami bisa berpindah ke berbagai versi lebih rapi.

Seharusnya setelah kami memahami apa yang akan kami buat, kami perlu langsung mengerjakannya dan tidak menunda-nunda mulai mengerjakan setelah beberapa hari rilis karena kami tidak akan tahu apa yang akan terjadi setelahnya yang akan kami jelaskan pada paragraf berikut setelahnya. Kami seharusnya langsung melakukan tes kasus setiap selesai membuat suatu metode, entah itu fungsi atau prosedur untuk menghindari *bugs* yang *subtle* dan susah disadari sehingga susah ditemukan dan memakan banyak waktu.

Kami merasa tidak lebih mawas akan *bugs*. Selain itu, karena kami telah menghadapi *workload* jauh lebih tinggi dalam menangani *bugs*, bahkan hampir 90% total waktu, kami menjadi lebih siap untuk menghadapi tantangan kedepannya untuk tidak menunda-nunda. Proses *debugging* tersebut cukup membuat kami frustrasi dan terkadang sangat membosankan. Sangat diperlukan untuk membuat kesepakatan standarisasi pengerjaan kelompok, seperti menggunakan ADT versi yang seperti apa, sebelum pembagian tugas dan pengerjaan dimulai. Tepat setelah tubes dirilis, sangat diharuskan bagi kami untuk segera mulai melakukan pengerjaan dikarenakan *bugs* yang *subtle* bukanlah hal yang bisa diperkirakan karena juga termasuk seberapa beruntung kita bisa menyelesaikan lebih awal. Walaupun spesifikasi tugas besar tidak terlalu banyak, kami sebagai pemula masih mengalami *overwhelming* sehingga terkadang menimbulkan banyak konflik saat *merge*. Pada akhirnya, kami mencoba beradaptasi dan membiasakan diri agar pada tugas-tugas berikutnya bisa kami tangani lebih baik dan santai dengan bertambahnya kemampuan *problem solving* dan *analythical thinking* kami.

6. Daftar Referensi

Anton, H., Rorres, C. (2014). Elementary Linear Algebra: Applications Version. Wiley.

D. B. Rowe. (2018). BiLinear, Bicubic, and In Between Spline Interpolation. Marquette University. Diakses dari PowerPoint Presentation (mu.edu)

Munir, Rinaldi. (2022). IF2123 Aljabar Geometri - Semester I Tahun 2022/2023. Institut Teknologi Bandung. Diakses pada 3 Oktober 2022, dari informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/algeo23-24.htm

Spesifikasi Tugas Besar 1 IF2123 Algeo 2023/2024 - Google Docs

Learn Java - Dev.java

Java Tutorial (w3schools.com)

Matrix calculator

How to Build Your Own Java library? (programcreek.com)

Terlampir juga repository sebagai referensi dan permintaan masukan berupa saran dan komentar dari pihak di luar tim kami 😀

diprayogo/Algeo01-22017: Tugas Besar 1 IF2123 Aljabar Linier dan Geometri Sistem Persamaan Linier, Determinan, dan Aplikasinya Semester I Tahun 2023/2024 (github.com)