Spesifikasi Tugas Besar 1

IF2123 Aljabar Linier dan Geometri

Sistem Persamaan Linier, Determinan, dan Penerapannya

Semester I Tahun 2025/2026

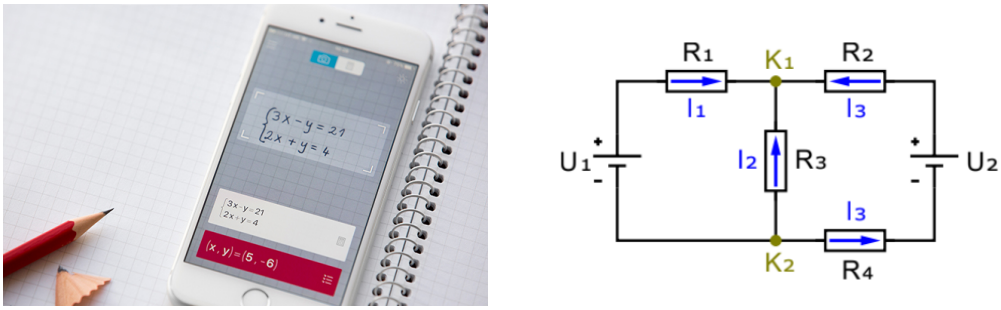
Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

1 Bab 1 (Pendahuluan)

Aplikasi seperti Photomath telah menjadi solusi populer dalam membantu menyelesaikan berbagai permasalahan matematika, termasuk soal matriks dan sistem persamaan linier yang kompleks. Kemampuan aplikasi tersebut dalam memberikan langkah-langkah penyelesaian secara otomatis tidak terlepas dari penerapan algoritma dan konsep matematika yang kuat, khususnya aljabar linier.

Aljabar linier merupakan salah satu cabang matematika yang sangat penting dalam pengembangan ilmu komputer, teknik, fisika, ekonomi, dan berbagai bidang lainnya. Dalam kehidupan sehari-hari, konsep matriks, dan sistem persamaan linier sering kali diaplikasikan untuk menyelesaikan berbagai persoalan nyata, mulai dari pemodelan data, analisis statistik, hingga simulasi sistem dinamis.

Pada tugas besar ini, Anda akan mengimplementasikan berbagai metode penyelesaian sistem persamaan linier, perhitungan determinan, pencarian invers matriks, interpolasi polinomial, interpolasi kurva Bézier kubik, dan regresi polinomial menggunakan bahasa pemrograman Java dalam bentuk pustaka (*library*) yang dapat digunakan secara modular dan terdokumentasi dengan baik.



Gambar 1: Aplikasi Photomath (kiri) dan diagram rangkaian listrik (kanan).

Penyelesaian sistem persamaan linier, misalnya, sangat krusial dalam pemodelan sirkuit listrik, analisis struktur bangunan, hingga pemrosesan citra digital. Sementara itu, interpolasi dan regresi polinomial digunakan untuk memperkirakan nilai di antara data yang tersedia atau memprediksi tren data di masa depan.

Melalui tugas besar ini, diharapkan Anda tidak hanya memahami teori di balik metode-metode tersebut, tetapi juga mampu mengimplementasikannya secara nyata yang dapat digunakan untuk menyelesaikan berbagai kasus uji (*test cases*) yang telah disediakan.

Secara formal, tujuan dari tugas besar ini adalah:

- Mengimplementasikan berbagai metode penyelesaian sistem persamaan linier, perhitungan determinan, invers matriks, interpolasi, dan regresi polinomial secara mandiri dalam bahasa Java.

- Membuat pustaka (*library*) yang dapat digunakan secara modular dan terdokumentasi dengan baik.
- Mengintegrasikan pustaka yang dibuat ke dalam sebuah program yang dapat menerima masukan dari pengguna dan menampilkan hasilnya dengan format yang jelas.
- Menguji pustaka yang dibuat pada berbagai kasus uji dan menganalisis hasilnya.

Dengan demikian, tugas besar ini diharapkan dapat menjadi sarana pembelajaran yang efektif dalam memahami dan mengaplikasikan konsep-konsep aljabar linier secara komputasional, serta membekali Anda dengan keterampilan praktis yang sangat dibutuhkan di dunia profesional.

2 Bab 2 (Dasar Teori)

2.1 Sistem Persamaan Linear

Sistem Persamaan Linear (SPL) adalah himpunan berhingga dari persamaan-persamaan linear dengan variabel-variabel yang sama. Bentuk umum dari SPL dengan m persamaan dan n variabel adalah

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n &= b_2 \\ & \cdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n &= b_n \end{cases}$$

atau, dalam bentuk perkalian matriks (kiri) dan matriks augmented (kanan),

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{x} \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}_{b} \longrightarrow \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right]$$

Solusi dari SPL dapat berupa solusi tunggal, banyak solusi, atau tidak ada solusi. Metode penyelesaian yang umum digunakan antara lain eliminasi Gauss, eliminasi Gauss-Jordan, kaidah Cramer, dan metode matriks balikan.

2.2 Determinan Matriks

Determinan adalah sebuah nilai skalar khusus yang dapat dihitung dari elemen-elemen sebuah matriks persegi. Determinan dari matriks A dinotasikan sebagai $\det(A)$ atau $|A|$. Nilai determinan sangat krusial karena dapat mengindikasikan apakah matriks tersebut memiliki invers. Beberapa metode untuk menghitung determinan matriks mencakup metode ekspansi kofaktor dan metode reduksi baris.

2.3 Invers Matriks

Invers dari sebuah matriks persegi A adalah matriks A^{-1} yang memenuhi properti $AA^{-1} = A^{-1}A = I$, di mana I adalah matriks identitas. Sebuah matriks hanya memiliki invers jika dan hanya jika determinannya tidak nol ($\det(A) \neq 0$). Matriks yang memiliki invers disebut matriks invertible atau non-singular. Perhitungan invers dari suatu matriks dapat dilakukan dengan metode matriks adjoin maupun metode eliminasi Gauss-Jordan.

2.4 Interpolasi Polinomial

Interpolasi polinomial adalah metode untuk menentukan suatu polinomial $P_n(x)$ berderajat n yang melalui $n+1$ titik data $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, x_i diurutkan dengan x_0 terkecil dan x_n terbesar. Secara umum, polinomial tersebut dapat dituliskan sebagai:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

dengan koefisien $a_0, a_1, \ldots, a_n$ yang ditentukan sedemikian rupa sehingga $P_n(x_{perkiraan}) = y_{perkiraan}$ untuk semua nilai $x_0 < x_{perkiraan} < x_n$. Derajat persamaan polinom ditentukan oleh jumlah titik data yang diberikan dikurangi satu.

Koefisien polinomial dapat dicari dengan menyulihkan titik-titik data ke dalam persamaan polinomial dan menyelesaikan sistem persamaan linear yang dihasilkan dengan metode eliminasi Gauss yang dijelaskan sebelumnya. Kita dapat menyusun sistem persamaan linear sebagai berikut.

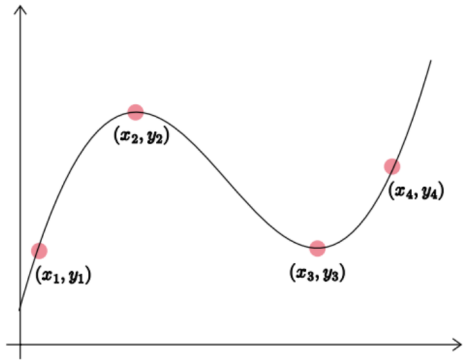
$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n &= y_1 \\ & \ldots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= y_n \end{cases}$$

Persamaan-persamaan di atas dapat dipetakan menjadi sebuah permasalahan sistem persamaan linear, sehingga dapat diselesaikan dengan menggunakan metode eliminasi Gauss untuk menyelesaikan sistem tersebut dan menemukan koefisien polinomial $a_0, a_1, \ldots, a_n$. Dari polinom $P_n(x)$ yang dibentuk dengan koefisien tersebut, kita dapat memperkirakan nilai y untuk nilai x yang berada di antara titik-titik data yang diberikan (interpolasi) maupun di luar titik-titik data tersebut (ekstrapolasi).

2.5 Interpolasi Splina Bézier Kubik

Ketika menggunakan *pen tool* di Adobe Photoshop/Illustrator, kita akan berhadapan dengan kurva Bézier, atau lebih spesifiknya kurva Bézier kubik. Kurva ini sangat berguna untuk menggambar bentuk bentuk halus dan kompleks dengan kontrol yang presisi.

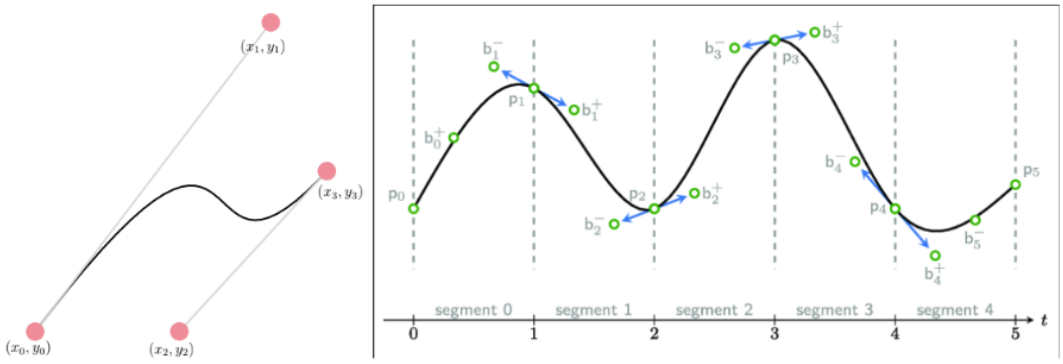
Kurva Bézier kubik adalah kurva parametrik yang didefinisikan oleh empat titik kontrol yang mempengaruhi bentuk kurva tersebut. Titik kontrol pertama (P_0) dan terakhir



Gambar 2: Contoh interpolasi polinomial melalui empat titik data.

(P_3) adalah titik awal dan akhir kurva, sedangkan titik-titik kontrol lainnya (P_1 dan P_2) menentukan arah dan kelengkungan kurva. Secara matematis, kurva Bézier kubik dapat ditulis sebagai persamaan parametrik

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3, \quad 0 \le t \le 1.$$



Gambar 3: Kurva Bézier kubik dengan empat titik kontrol (kiri) dan kurva komposit yang tersusun atas banyak splina Bézier kubik.

Misalkan kita diberikan titik-titik interpolasi $S_0, S_1, \ldots, S_n$ dan kita perlu menemukan titik-titik kontrol $B_0, B_1, \ldots, B_n$ yang sesuai untuk membangun kurva halus yang melewati semua titik S_i. Kita dapat membangun kurvanya dari **splina Bézier kubik**, yakni himpunan kurva Bézier kubik yang didefinisikan pada segmen-segmen $Q_i Q_{i+1}$ untuk $0 \le i \le n-1$.

Masalah ini dapat direduksi menjadi penyelesaian sistem persamaan linear yang didapatkan dari hubungan antara titik-titik S_i dan titik-titik kontrol B_i (silakan baca referensi). Sistem persamaan tersebut adalah

$$\begin{cases} 4b_1 + b_2 &= 6s_1 - s_0 \\ b_1 + 4b_2 + b_3 &= 6s_2 \\ b_2 + 4b_3 + b_4 &= 6s_3 \\ \cdots & \\ b_{n-2} + 4b_{n-1} &= 6s_{n-1} - s_n \end{cases}$$

dengan b_k dan s_k adalah vektor posisi dari titik-titik kontrol B_k dan titik-titik interpolasi S_k, atau dapat dinyatakan dalam bentuk perkalian matriks,

$$\begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & 4 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} 6s_1 - s_0 \\ 6s_2 \\ 6s_3 \\ \vdots \\ 6s_{n-1} - s_n \end{bmatrix}$$

Karena vektor posisi b_k dan s_k memiliki dua komponen (koordinat x dan y, kita dapat memisahkan sistem persamaan di atas menjadi dua sistem persamaan linear yang terpisah, satu untuk komponen x dan satu untuk komponen y. Dengan menyelesaikan kedua sistem persamaan tersebut, kita dapat menentukan posisi titik-titik kontrol B_k yang diperlukan untuk membentuk kurva Bézier kubik yang halus dan sesuai dengan titik-titik interpolasi yang diberikan.

2.6 Regresi Polinomial Berganda (*Multivariate Polynomial Regression*)

Diberikan n titik data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ dengan x_i adalah variabel dan y_i adalah hasil pengukuran ke-i $(1 \leq i \leq n)$, regresi linear sederhana bertujuan untuk menemukan garis lurus

$$y = \beta_0 + \beta_1 x + \epsilon$$

yang paling sesuai dengan data tersebut dengan ϵ adalah galatnya. Namun, dalam banyak kasus, besaran yang kita amati bisa dipengaruhi oleh lebih dari satu variabel. Untuk mengatasi hal ini, kita dapat menggunakan **regresi polinomial berganda** (*multivariate polynomial regression*).

Misalnya, jika y dipengaruhi oleh tiga variabel independen5 x_1, x_2, dan x_3, kita dapat memodelkan hubungan tersebut dengan persamaan linier berganda

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$$

dengan ϵ adalah galat (error) yang mengikuti distribusi normal dengan rata-rata nol dan varians konstan. Tugas kita adalah mengestimasi nilai koefisien $\beta_0, \beta_1, \beta_2$, dan β_3

berdasarkan data yang diberikan. Secara umum, andaikata ada n variabel independen $x_1, x_2, \ldots, x_n$, maka persamaan regresi linier berganda dapat dituliskan sebagai

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

sehingga jika kita memiliki m titik data dan x_{ij} adalah nilai dari variabel x_j pada titik data ke-i serta y_i dan ϵ_i adalah nilai hasil pengukuran dan galatnya pada titik data ke-i, kita dapat menuliskan sistem persamaan linear berikut untuk m titik data,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}_{m\times 1} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}_{m\times(n+1)} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}_{(n+1)\times 1} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_m \end{bmatrix}_{m\times 1},$$

atau

$$Y = X\beta + \epsilon.$$

dengan matriks Y adalah vektor hasil pengukuran, X adalah matriks desain, β adalah vektor koefisien,dan ϵ adalah vektor galat.

Tugas kita pada dasarnya adalah mengestimasi nilai vektor koefisien $\hat{\beta}$ seakurat mungkin, atau dengan kata lain, dengan galat sekecil mungkin. Metode umum yang digunakan adalah metode kuadrat terkecil (*least squares*) yang meminimalkan jumlah kuadrat galat, yaitu $\|\epsilon\|^2 = \|y - X\beta\|^2$. Dengan metode ini, kita dapat menemukan solusi untuk $\hat{\beta}$ dengan menyelesaikan persamaan normal berikut:

$$X^T X\hat{\beta} = X^T y,$$

atau

$$\hat{\beta} = (X^T X)^{-1} X^T y. \tag{1}$$

Sekarang, bagaimana jika hubungan antara y dan variabel-variabel independennya tidak linier? Salah satu pendekatan yang dapat kita gunakan adalah **regresi polinomial berganda** (*multivariate polynomial regression*). Idenya masih mirip: mencari fungsi polinomial yang kurvanya paling sesuai dengan data yang diberikan. Namun, kali ini akan terdapat variabel interaksi, yakni variabel yang merupakan hasil kali dari beberapa variabel independen. Misalnya, untuk tiga variabel independen x_1, x_2, dan x_3 dan derajat 3, fungsi polinomial yang kita hasilkan bisa saja berbentuk

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_1^2 + \beta_5 x_1 x_2 + \beta_6 x_1 x_3 + \beta_7 x_2^2 + \beta_8 x_2 x_3 + \beta_9 x_3^2 + \beta_{10} x_1^3$$
$$+\beta_{11} x_1^2 x_2 + \beta_{12} x_1^2 x_3 + \beta_{13} x_1 x_2^2 + \beta_{14} x_1 x_2 x_3 + \beta_{15} x_1 x_3^2 + \beta_{16} x_2^3 + \beta_{17} x_2^2 x_3 + \beta_{18} x_2 x_3^2 + \beta_{19} x_3^3 + \epsilon.$$

Secara umum, jika kita memiliki k variabel independen dan ingin membuat polinomial dengan derajat d, jumlah total suku dalam polinomial tersebut adalah $p = \binom{d+k}{k}$. Lantas, bagaimana cara kita menemukan seluruh koefisien $\beta_0, \beta_1, \ldots, \beta_p$?

Kita bisa mengadopsi pendekatan serupa dengan yang kita gunakan pada regresi linier berganda. Kali ini, variabel-variabel seperti x_1^2, x_1x_2, x_1^3, dan sebagainya akan dianggap sebagai variabel independen yang berbeda. Dengan demikian, kita dapat menyusun sistem persamaan linear yang mirip dengan yang sebelumnya, tetapi dengan lebih banyak kolom pada matriks desain X untuk mengakomodasi semua suku dalam polinomial. Setelah itu, kita dapat menggunakan metode kuadrat terkecil untuk menemukan estimasi koefisien $\hat{\beta}$ (persamaannya sama persis seperti pada regresi linier berganda).

Misalnya, jika kita memiliki lima sampel data dan tiga variabel independen ($k = 3$) dan ingin membuat polinomial dengan derajat dua ($d = 2$), maka jumlah total suku dalam polinomial tersebut adalah $\binom{2+3}{3} = 10$ dan matriks desain X akan berbentuk seperti ini:

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & x_{11}^2 & x_{11}x_{12} & x_{11}x_{13} & x_{12}^2 & x_{12}x_{13} & x_{13}^2 \\ 1 & x_{21} & x_{22} & x_{23} & x_{21}^2 & x_{21}x_{22} & x_{21}x_{23} & x_{22}^2 & x_{22}x_{23} & x_{23}^2 \\ 1 & x_{31} & x_{32} & x_{33} & x_{31}^2 & x_{31}x_{32} & x_{31}x_{33} & x_{32}^2 & x_{32}x_{33} & x_{33}^2 \\ 1 & x_{41} & x_{42} & x_{43} & x_{41}^2 & x_{41}x_{42} & x_{41}x_{43} & x_{42}^2 & x_{42}x_{43} & x_{43}^2 \\ 1 & x_{51} & x_{52} & x_{53} & x_{51}^2 & x_{51}x_{52} & x_{51}x_{53} & x_{52}^2 & x_{52}x_{53} & x_{53}^2 \end{bmatrix},$$

yang sebenarnya analog dengan matriks desain untuk regresi linier berganda

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & x_{18} & x_{19} \\ 1 & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} & x_{28} & x_{29} \\ 1 & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} & x_{38} & x_{39} \\ 1 & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} & x_{48} & x_{49} \\ 1 & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} & x_{58} & x_{59} \end{bmatrix},$$

dengan x_{ij} adalah nilai dari suku ke-j pada titik data ke-i. Pada intinya, cukup gunakan persamaan (1) untuk menemukan estimasi koefisien $\hat{\beta}$.

3 Bab 3 (Implementasi)

Program aljabar linear ini terdiri dari enam package utama, yaitu matrix, spl, determinan, invers, interpolasi, dan regresi.

3.1 Package Matrix

3.1.1 Matrix.java

Kelas ini berfungsi menyimpan data matrix dalam bentuk array 2 dimensi bertipe double, beserta informasi jumlah baris dan kolomnya. Beberapa metode pada kelas antara lain.

- **Matrix(int rows, int cols)**: konstruktor untuk membuat matriks dengan ukuran $rows \times cols$

- **get(int i, int j)**: mengembalikan nilai pada baris i kolom j
- **set(int i, int j, double value)**: mengubah nilai pada baris i kolom j dengan value
- **copy()**: membuat salinan matriks.
- **print()**: menampilkan keluaran matriks.
- **getSubMatrix(int startRow, int endRow, int startCol, int endCol)**: mengembalikan bagian tertentu dari matriks.
- **multiplyRow(Matrix matrix, int row, double x)**: mengalikan seluruh elemen pada baris row dengan x.
- **subtractMultipliedRow(Matrix matrix, int row1, int row2, double x)**: mengurangi seluruh elemen pada row1 dengan x×row2.
- **matrixTranpos(Matrix matrix)**: mengembalikan transpos dari matrix.
- **matrixToString(Matrix matrix)**: mengembalikan matrix dalam bentuk string.

3.1.2 FileHandler.java

Kelas ini berfungsi untuk menangani operasi pembacaan dan penulisan file untuk matriks dan berbagai hasil perhitungan. Berikut adalah method-method yang tersedia:

- **readAugmentedMatrix(String filename)**: membaca matriks augmented dari file untuk Sistem Persamaan Linier. Method ini memanggil readMatrix() untuk melakukan pembacaan.
- **readMatrix(String filename**: membaca matriks umum dari file dengan format teks. Method ini mendukung:
  - Pemisahan nilai dengan spasi atau tab
  - Format desimal dengan koma (,) atau titik (.)
  - Format pecahan (contoh: 1/2, 3/4)
  - Validasi konsistensi jumlah kolom di setiap baris
  - Mengembalikan objek Matrix yang telah terisi data
- **writeSPLOutput(String filename, SPLResult result, Matrix input)**: menulis hasil penyelesaian Sistem Persamaan Linier ke file. Output mencakup:
  - Header hasil
  - Nama metode yang digunakan
  - Matriks augmented input

- Langkah-langkah penyelesaian

- Hasil akhir (solusi tunggal, tidak ada solusi, atau solusi tak hingga dengan bentuk parametrik)

- **writeInversOutput(String filename, InversResult result, Matrix input)**: menulis hasil perhitungan invers matriks ke file. Output mencakup:

  - Header hasil

  - Matriks input

  - Langkah-langkah perhitungan invers

  - Hasil matriks invers (jika ada)

- **writeDeterminanOutput(String filename, DeterminanResult result, Matrix input)**: menulis hasil perhitungan determinan matriks ke file. Output mencakup:

  - Header hasil

  - Matriks input

  - Langkah-langkah perhitungan determinan

  - Nilai determinan akhir

- **Tambahan**: - Error Handling: semua method dilengkapi dengan try-catch-finally untuk menangani IOException dan memastikan file selalu tertutup dengan benar.

  - Validasi Input: readMatrix() memvalidasi format angka dan konsistensi jumlah kolom untuk menghindari data yang tidak valid.

  - Format Output: semua method penulisan menggunakan format yang konsisten dengan pemisah visual (========) dan struktur yang jelas untuk memudahkan pembacaan.

3.2 Package SPL

3.2.1 SPLSolver.java

Interface ini adalah kontrak untuk semua solver Sistem Persamaan Linier. Setiap kelas yang mengimplementasikan interface ini harus menyediakan implementasi untuk method-method berikut:

- **solve(Matrix augmentedMatrix)**:menyelesaikan sistem persamaan linier Ax = b dengan parameter matriks augmented [A|b]. Method ini:

  - Menerima matriks augmented yang berisi koefisien dan konstanta

  - Memproses sistem persamaan menggunakan algoritma tertentu

- Mengembalikan objek SPLResult yang berisi hasil penyelesaian (solusi tunggal, tidak ada solusi, atau solusi tak hingga)

- **getMethodName()**: mendapatkan nama metode solver yang digunakan. Method ini mengembalikan string yang merepresentasikan nama metode penyelesaian, seperti:

  1. Metode Eliminasi Gauss
  2. Metode Gauss-Jordan
  3. Metode Matriks Balikan
  4. Metode Cramer

3.2.2 SPLResult.java

Kelas ini berfungsi untuk menyimpan hasil penyelesaian SPL dan informasi yang berkaitan. Kelas ini menyediakan struktur data lengkap untuk merepresentasikan berbagai jenis solusi SPL.

- **Enum SolutionType**:

  - UNIQUE: merepresentasikan sistem memiliki solusi tunggal (satu solusi unik)

  - NOSOLUTION: merepresentasikan sistem tidak memiliki solusi (inkonsisten)

  - INFINITE: merepresentasikan sistem memiliki tak hingga banyak solusi (memiliki parameter bebas)

- **Atribut**:

  - type: tipe solusi dari sistem persamaan (UNIQUE, NOSOLUTION, atau INFINITE) - solution: array yang menyimpan nilai solusi untuk setiap variabel (digunakan jika solusi tunggal)

  - parametricSolution: string yang berisi representasi solusi

  - parametrik: (digunakan jika solusi tak hingga)

  - steps: list yang menyimpan langkah-langkah penyelesaian dalam bentuk string

  - methodName: nama metode yang digunakan untuk menyelesaikan SPL item

  SPLResult(): konstruktor untuk membuat objek SPLResult dan menginisialisasi list steps sebagai ArrayList kosong.

- **addStep(String step)**: menambahkan satu langkah penyelesaian ke dalam list steps untuk dokumentasi proses perhitungan.

- **getType()**: mengembalikan tipe solusi sistem persamaan

- **setType(SolutionType type)**: mengubah tipe solusi sistem persamaan dengan value yang diberikan.
- **getSolution()**: mengembalikan array berisi nilai solusi untuk setiap variabel jika solusi tunggal.
- **setSolution(double[] solution)**: mengubah nilai solusi dengan array yang diberikan.
- **getParametricSolution()**: mengembalikan string representasi solusi parametrik jika solusi tak hingga.

  setParametricSolution(String parametricSolution): mengubah representasi solusi parametrik dengan string yang diberikan.

  getSteps(): mengembalikan list berisi semua langkah penyelesaian yang telah dicatat
- **getMethodName()**: mengembalikan nama metode yang digunakan untuk menyelesaikan SPL
- **setMethodName(String methodName)**: mengubah nama metode dengan string yang diberikan.

3.2.3 GaussElimination.java

Kelas ini mengimplementasikan interface SPLSolver untuk menyelesaikan SPL menggunakan metode Eliminasi Gauss. Metode ini terdiri dari dua fase utama: Fase Maju dan Substitusi/Penyulihan Mundur.

- **EPSILON**: nilai toleransi untuk perbandingan floating point (1e-10), digunakan untuk menangani error pembulatan numerik
- **solve(Matrix augmentedMatrix)**: menyelesaikan sistem persamaan linier menggunakan metode Eliminasi Gauss. Method ini:
  1. Menerima matriks augmented [A|b] sebagai input
  2. Melakukan validasi sistem underdetermined (jumlah persamaan $<$ jumlah variabel)
  3. Menjalankan fase maju untuk mengubah matriks menjadi bentuk eselon baris (segitiga atas)
  4. Melakukan partial pivoting untuk meningkatkan stabilitas numerik
  5. Mengeliminasi elemen di bawah pivot menggunakan operasi baris elementer
  6. Memeriksa konsistensi sistem dan menghitung rank matriks

7. Menjalankan substitusi mundur untuk menghitung nilai setiap variabel dari bawah ke atas
8. Mengembalikan objek SPLResult dengan tipe solusi (UNIQUE, NOSOLUTION, atau INFINITE) dan nilai solusi atau representasi parametrik

- **getMethodName()**: mengembalikan string "Eliminasi Gauss" sebagai nama metode solver.
- **swapRows(Matrix m, int row1, int row2)**: menukar posisi dua baris dalam matriks untuk operasi partial pivoting. Method ini menukar semua elemen pada row1 dengan elemen pada row2.
- **matrixToString(Matrix m)**: mengkonversi matriks menjadi representasi string dengan format tertentu untuk ditampilkan dalam langkah-langkah penyelesaian. Setiap elemen diformat dengan lebar 10 karakter dan 4 desimal.
- **buildParametricSolution(Matrix m, int rank)**: membangun representasi string untuk solusi parametrik ketika sistem memiliki tak hingga solusi. Method ini:
  - Menghitung jumlah variabel bebas (numFree = numVars - rank)
  - Mengidentifikasi variabel pivot dan variabel bebas
  - Menyusun persamaan parametrik untuk setiap variabel
  - Variabel pivot dinyatakan dalam fungsi variabel bebas
  - Variabel bebas dinyatakan sebagai parameter t, t, dst
  - Mengembalikan string representasi solusi lengkap

3.2.4 GaussJordan.java

Kelas ini merupakan implementasi dari algoritma eliminasi Gauss-Jordan untuk menyelesaikan (SPL). Kelas ini mengubah matriks augmented menjadi bentuk eselon baris tereduksi untuk menemukan solusi SPL.

- **solve(Matrix augmentedMatrix)**: Metode utama yang menjalankan seluruh proses eliminasi Gauss-Jordan. Metode ini menerima matriks augmented dari sebuah SPL, melakukan operasi baris elementer (penukaran baris, normalisasi, dan eliminasi) untuk mengubahnya menjadi bentuk eselon baris tereduksi. Setelah itu, metode ini menganalisis matriks hasil untuk menentukan tipe solusi (unik, tak hingga, atau tidak ada solusi) dan mengembalikan hasilnya dalam sebuah objek SPLResult.
- **swapRows(Matrix m, int row1, int row2)**: menukar posisi dua baris dalam matriks untuk operasi partial pivoting. Method ini menukar semua elemen pada row1 dengan elemen pada row2.

- **matrixToString(Matrix m)**: mengkonversi matriks menjadi representasi string dengan format tertentu untuk ditampilkan dalam langkah-langkah penyelesaian. Setiap elemen diformat dengan lebar 10 karakter dan 4 desimal.
- **buildParametricSolution(Matrix m, int rank)**: membangun representasi string untuk solusi parametrik ketika sistem memiliki tak hingga solusi.
- **getMethodName()**: mengembalikan string "Eliminasi Gaus-Jordan" sebagai nama metode solver.

3.2.5 CramerRule.java

- **CramerRule()**: ini adalah constructor kelas yang berfungsi untuk menginisialisasi objek detCalculator. Objek ini akan digunakan untuk menghitung determinan matriks A dan matriks-matriks A_i yang diperlukan dalam perhitungan.
- **getMethodName()**: Mengembalikan nama metode sebagai String, yaitu "Kaidah Cramer (Cramer's Rule)". **solve(Matrix augmentedMatrix)**: Metode utama yang menjalankan seluruh proses penyelesaian SPL dengan Kaidah Cramer. Langkah-langkah utamanya adalah:
  1. Memisahkan matriks augmented menjadi matriks koefisien A dan vektor konstanta b.
  2. Menghitung determinan dari matriks koefisien utama, yaitu $\det(A)$.
  3. Jika $\det(A) = 0$, metode akan berhenti karena Kaidah Cramer tidak bisa digunakan (sistem tidak memiliki solusi unik).
  4. Jika $\det(A) \neq 0$, metode akan mengulang untuk setiap variabel x_i: membuat matriks A_i dengan mengganti kolom ke-i dengan vektor b, menghitung determinannya ($\det(A_i)$), lalu mencari nilai x_i dengan membagi $\det(A_i)$ dengan $\det(A)$.
  5. Semua langkah perhitungan, termasuk verifikasi akhir, dan solusi dikemas dalam objek `SPLResult` lalu dikembalikan.
- **isValidForCramer(Matrix augmentedMatrix)**: sebuah metode helper static yang berfungsi untuk memvalidasi apakah sebuah matriks augmented cocok untuk diselesaikan dengan Kaidah Cramer. Metode ini hanya memeriksa apakah jumlah kolom adalah jumlah baris ditambah satu, yang menandakan sebuah sistem n×n.

3.2.6 InversMethod

Kelas ini mengimplementasikan penyelesaian Sistem Persamaan Linear (SPL) dengan menggunakan metode matriks balikan (invers). Prinsip dasarnya adalah mengubah sistem persamaan $Ax = b$ menjadi $x = A^{-1}b$, di mana x adalah vektor solusi, A^{-1} adalah invers

dari matriks koefisien, dan b adalah vektor konstanta. Kelas ini memerlukan matriks koefisien A yang berbentuk persegi ($n \times n$) dan memiliki invers (determinannya tidak nol).

- **InversMethod()**: Ini adalah constructor kelas. Fungsinya adalah untuk membuat dan menginisialisasi objek dari kelas Invers yang akan digunakan nanti untuk menghitung invers dari matriks koefisien A.
- **solve(Matrix augmentedMatrix)**: Metode utama yang menjalankan seluruh logika penyelesaian SPL. Langkah-langkahnya adalah sebagai berikut:
  1. Memvalidasi apakah matriks yang diberikan cocok untuk metode ini (harus sistem $n \times n$).
  2. Memisahkan matriks augmented menjadi dua bagian: matriks koefisien A dan vektor konstanta b.
  3. Memanggil objek `inversCalculator` untuk menghitung matriks balikan dari A (yaitu A^{-1}).
  4. Memeriksa apakah invers berhasil ditemukan. Jika tidak (karena determinan $A = 0$), metode akan menyimpulkan bahwa tidak ada solusi unik.
  5. Jika invers ada, metode akan melakukan perkalian matriks antara A^{-1} dan b untuk mendapatkan vektor solusi x.
  6. Seluruh langkah, proses perhitungan, dan hasil akhir dikemas ke dalam objek `SPLResult` lalu dikembalikan.
- **getMethodName()**: Mengembalikan nama metode yang diimplementasikan, yaitu "Metode Matriks Balikan (Invers)".

3.3 Package Determinan

3.3.1 DeterminanResult.java

Kelas ini berfungsi untuk menyimpan hasil perhitungan determinan dalam bentuk nilai hasil determinan dan string langkah-langkahnya.

- **DeterminanResult(double value, StringBuilder steps)**: konstruktor untuk membuat DeterminanResult.
- **getValue()**: mengembalikan nilai determinan dari DeterminanResult.
- **getSteps()**: mengembalikan langkah-langkah perhitungan dari DeterminanResult.

3.3.2 Determinan.java

Kelas ini berisi operasi-operasi untuk menghitung determinan dengan metode ekspansi kofaktor dan reduksi baris (OBE). Beberapa metode pada kelas ini antara lain.

- **detEkspansiKofaktor(Matrix matrix)**: mengembalikan DeterminanResult dengan metode ekspansi kofaktor.
- **rekursiEkspansiKofaktor(Matrix matrix, StringBuilder steps)**: helper rekursi untuk menghitung detEkspansiKofaktor.
- **minor(Matrix matrix, int i, int j)**: mengembalikan minor dari $matrix_{ij}$
- **detM2x2(Matrix matrix)**: mengembalikan determinan dari matrix ukuran 2×2.
- **detOBE(Matrix matrix)**: mengembalikan DeterminanResult dengan metode reduksi baris.

3.4 Package Invers

3.4.1 InversResult.java

Kelas ini berfungsi untuk menyimpan hasil perhitungan invers dalam bentuk Matrix hasil invers dan string langkah-langkahnya.

- **InversResult(Matrix matrix, StringBuilder steps)**: konstruktor untuk membuat InversResult.
- **getValue()**: mengembalikan Matrix invers dari InversResult.
- **getSteps()**: mengembalikan langkah-langkah perhitungan dari InversResult.

3.4.2 Invers.java

Kelas ini berisi operasi-operasi untuk menghitung invers dengan metode augment dan adjoin. Beberapa metode pada kelas ini antara lain.

- **inversAdjoin(Matrix matrix)**: mengembalikan InversResult dengan metode adjoin.
- **inversAugment(Matrix matrix)**: mengembalikan InversResult dengan metode augment.
- **matrixKofaktor(Matrix matrix)**: mengembalikan matriks kofaktor dari matrix.
- **adjoin(Matrix matrix)**: mengembalikan matriks adjoin dari matrix.

4 Bab 4 (Eksperimen)

4.1 Determinan Matriks

4.1.1 Kasus Uji 1

$$\begin{bmatrix} 5 & 7 & 9 & 11 \\ 10 & 14 & 18 & 22 \\ 15 & 21 & 27 & 33 \\ 20 & 28 & 36 & 44 \end{bmatrix}$$

Determinan yang dihasilkan di kasus uji ini adalah 0. Langkah-langkah dapat dilihat di lampiran.

4.1.2 Kasus Uji 2

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \end{bmatrix}$$

Determinan yang dihasilkan di kasus uji ini adalah 1.6472223857149554E-7. Langkah-langkah dapat dilihat di lampiran.

4.2 Invers Matriks

4.2.1 Kasus Uji 1

$$\begin{bmatrix} 5 & 2 & -3 & 4 \\ 7 & 7 & 8 & 9 \\ 12 & 13 & 2 & 14 \\ 17 & 18 & 19 & 4 \end{bmatrix}$$

Hasil matriks invers:

$$\begin{bmatrix} 0,3141 & -0,0406 & -0,1391 & 0,0813 \\ -0,3032 & 0,1617 & 0,2307 & -0,1403 \\ 0,0037 & -0,0955 & -0,0919 & 0,1031 \\ 0,0118 & -0,1018 & -0,0104 & 0,0458 \end{bmatrix}$$

Langkah-langkah dapat dilihat di lampiran.

4.2.2 Kasus Uji 2

$$\begin{bmatrix} 5 & 7 & 9 & 11 & 13 & 15 \\ 10 & 14 & 18 & 22 & 26 & 30 \\ 15 & 21 & 27 & 33 & 39 & 45 \\ 20 & 28 & 36 & 44 & 52 & 60 \\ 25 & 35 & 45 & 55 & 65 & 75 \\ 30 & 42 & 54 & 66 & 78 & 90 \end{bmatrix}$$

Matriks tidak memiliki invers. Langkah-langkah dapat dilihat di lampiran.

4.3 Sistem Persamaan Linear $Ax = b$

4.3.1 Kasus Uji 1

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

Solusi Tunggal:

x1 = 857828500451524.800000

x2 = -3431314001806094.000000

x3 = -1286742750677284.000000

x4 = -1286742750677285.000000

4.3.2 Kasus Uji 2

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

SOLUSI TAK HINGGA. Sistem memiliki tak hingga banyak solusi.

4.3.3 Kasus Uji 3

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

SOLUSI TAK HINGGA. Sistem memiliki tak hingga banyak solusi.

4.3.4 Kasus Uji 4

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

H adalah matriks Hilbert. Coba uji untuk nilai $n = 6$ dan $n = 10$.

- Untuk n = 6. Solusi Tunggal:

$$\begin{aligned} x_1 &= 36.000000 \\ x_2 &= -630.000000 \\ x_3 &= 3360.000000 \\ x_4 &= -7560.000001 \\ x_5 &= 7560.000001 \\ x_6 &= -2772.000000 \end{aligned}$$

- Untuk n = 10. Solusi Tunggal:

$$\begin{aligned} x_1 &= 99.994297 \\ x_2 &= -4949.766658 \\ x_3 &= 79194.525292 \\ x_4 &= -600553.154407 \\ x_5 &= 2522296.397447 \\ x_6 &= -6305684.920175 \\ x_7 &= 9608590.378181 \\ x_8 &= -8750623.961135 \\ x_9 &= 4375287.438123 \\ x_10 &= -923667.253284 \end{aligned}$$

4.4 Sistem Persamaan Linear Berbentuk Matriks Augmented

4.4.1 Kasus Uji 1

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

SOLUSI TAK HINGGA. Sistem memiliki tak hingga banyak solusi. Sistem memiliki 2 variabel bebas.

Solusi Tunggal:

x1 = -1,0000 + 1,0000*x4

x2 = 0,0000 + 2,0000*x3

x3 = t3 (parameter bebas)

x4 = t4 (parameter bebas)

Langkah-langkah dapat dilihat di lampiran.

4.4.2 Kasus Uji 2

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

TIDAK ADA SOLUSI. Sistem persamaan inkonsisten.Langkah-langkah dapat dilihat di lampiran.

4.5 Sistem Persamaan Linear Berbentuk Umum

4.5.1 Kasus Uji 1

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

Solusi Tunggal:

x1 = -0,2243244

x2 = 0,182432

x3 = 0,7094590

x4 = -0,258108

Langkah-langkah dapat dilihat di lampiran.

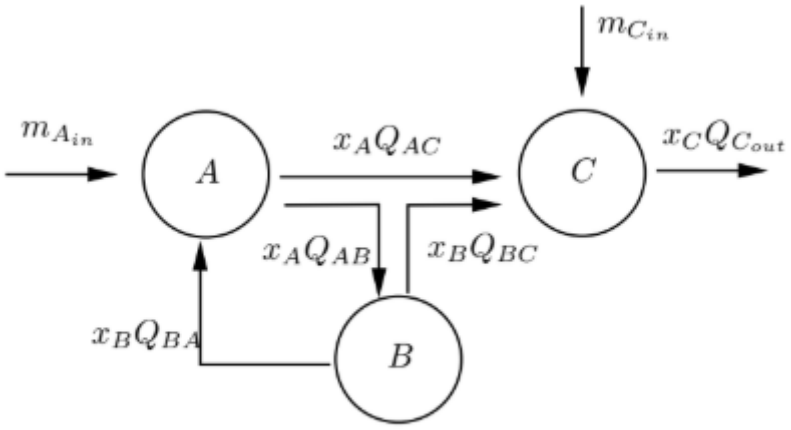
4.5.2 Kasus Uji 2

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

TIDAK ADA SOLUSI. Sistem persamaan inkonsisten.Langkah-langkah dapat dilihat di lampiran.

4.6 Aplikasi Sistem Persamaan Linear

Lihatlah sistem reaktor pada gambar berikut:



Gambar 4: Model rangkaian reaktor

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

$$
\begin{aligned}
&\text{A: } m_{A_{in}} + Q_{BA}x_B - Q_{AB}x_A - Q_{AC}x_A = 0 \\
&\text{B: } Q_{AB}x_A - Q_{BA}x_B - Q_{BC}x_B = 0 \\
&\text{C: } m_{C_{in}} + Q_{AC}x_A + Q_{BC}x_B - Q_{C_{out}}x_C = 0
\end{aligned}
$$

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

Solusi Tunggal:

x1 = 14,444444

x2 = 7,222222

x3 = 10,000000

Langkah-langkah dapat dilihat di lampiran.

4.7 Studi Kasus Interpolasi Polinomial

4.7.1 Studi Kasus 1

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

x	0.1	0.3	0.5	0.7	0.9	1.1	1.3
$f(x)$	0.003	0.067	0.148	0.248	0.370	0.518	0.697

Lakukan pengujian pada nilai-nilai berikut:

$x = 0.2 \quad f(x) = ?$

$x = 0.55 \quad f(x) = ?$

$x = 0.85 \quad f(x) = ?$

$x = 1.28 \quad f(x) = ?$

4.7.2 Studi Kasus 2

Jumlah kasus positif baru Covid-19 di Indonesia semakin fluktuatif dari hari ke hari. Di bawah ini diperlihatkan jumlah kasus baru Covid-19 di Indonesia mulai dari tanggal 17 Juni 2022 hingga 31 Agustus 2022:

Tanggal	Tanggal (desimal)	Jumlah Kasus Baru
17/06/2022	6.567	12.624
30/06/2022	6.967	21.807
08/07/2022	7.258	38.391
14/07/2022	7.451	19.541
21/07/2022	7.548	19.582
01/08/2022	8.032	28.935
08/08/2022	8.258	25.854
15/08/2022	8.484	20.935
22/08/2022	8.709	12.408
31/08/2022	8.997	10.534

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

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

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

$$\text{Tanggal (desimal)} = 6 + \tfrac{17}{30} = 6.567$$

Gunakan 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.

4.8 Studi Kasus Interpolasi Splina Bézier Kubik

Tentukan titik kontrol dari tiap segmen kurva splina Bézier kubik yang melalui titik-titik berikut ini:

$$(0,0), (3,11), (6,-4), (8,0), (11,-10), (17,0).$$

4.9 Studi Kasus Regresi Polinomial Berganda

Wak Rusdi sedang meneliti pertumbuhan alga berdasarkan faktor cahaya, suhu, dan pH. Ia meneliti 50 sampel yang hasilnya dapat dilihat disini.

Silakan proses data tersebut menjadi file txt sesuai dengan format input yang telah dijelaskan se- belumnya lalu tentukan persamaan regresi polinomial berganda yang sesuai dengan data tersebut meng- gunakan polinomial kubik (derajat 3).

5 Bab 5 (Penutup)

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian program yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

a Program aplikasi untuk menyelesaikan Sistem Persamaan Linear (SPL) telah dirancang dan diimplementasikan dalam bahasa pemrograman Java dengan menerapkan tiga metode, yaitu Eliminasi Gauss, Eliminasi Gauss-Jordan, Metode Matriks Balikan (Invers), dan Kaidah Cramer.

b Setiap metode yang diimplementasikan memiliki karakteristik dan batasan yang berbeda. - Eliminasi Gauss mengubah matriks menjadi bentuk eselon baris dan memerlukan tahap substitusi balik (back substitution) untuk mendapatkan nilai variabel.

- Eliminasi Gauss-Jordan mengubah matriks menjadi bentuk eselon baris tereduksi, sehingga nilai setiap variabel dapat langsung diperoleh tanpa substitusi tambahan.

- Metode Matriks Balikan dan Kaidah Cramer hanya dapat diterapkan pada sistem persamaan linear persegi (n×n) dan efektif untuk menemukan solusi unik. Kedua metode ini akan gagal jika determinan matriks koefisien bernilai nol.

c Dari segi komputasi, Kaidah Cramer cenderung menjadi metode yang paling tidak efisien untuk matriks berukuran besar karena memerlukan perhitungan determinan berulang kali, yang memiliki kompleksitas waktu tinggi.

d Implementasi program berhasil menangani kasus-kasus khusus seperti matriks singular (determinan nol) dengan memberikan pesan kesalahan yang sesuai kepada pengguna, sehingga mencegah terjadinya error saat eksekusi. Penanganan presisi angka desimal (karena keterbatasan floating-point) juga menjadi aspek penting yang diperhatikan dalam implementasi untuk menjamin akurasi hasil.

e Untuk mendukung fungsionalitas utama, program juga telah berhasil mengimplementasikan operasi-operasi dasar matriks, secara spesifik:

   (a) Perhitungan determinan matriks, yang dapat dilakukan menggunakan dua metode berbeda: Reduksi Baris dan Ekspansi Kofaktor.

   (b) Perhitungan invers (matriks balikan), yang dapat dilakukan menggunakan dua metode berbeda: Metode Adjoin-Kofaktor dan Eliminasi Gauss-Jordan (Metode Matriks Augmented).

5.2 Saran

Saran yang diberikan pada bagian ini tidak hanya ditujukan untuk pengembangan program, tetapi juga sebagai masukan konstruktif bagi penyelenggara praktikum, khususnya kepada tim asisten dosen, demi optimalisasi proses pembelajaran dan pengerjaan tugas besar di masa mendatang.

a Mengenai Ukuran Kelompok: Melihat luasnya cakupan materi yang harus diimplementasikan—mencakup empat metode penyelesaian SPL, dua metode determinan, dua metode invers, dan lainnya beserta pembuatan laporan—beban kerja yang ditanggung oleh setiap anggota dalam satu kelompok terasa sangat signifikan. Kelompok yang hanya terdiri dari tiga anggota menyebabkan pembagian tugas menjadi sangat padat, sehingga mengurangi kesempatan untuk eksplorasi dan pendalaman

materi secara lebih komprehensif. Oleh karena itu, disarankan agar untuk tugas besar dengan cakupan serupa di masa depan, dapat dipertimbangkan untuk menetapkan jumlah anggota per kelompok sebanyak 4 hingga 5 orang. Hal ini diharapkan dapat membuat pembagian kerja lebih merata dan memberikan ruang yang lebih leluasa bagi setiap anggota untuk berkontribusi secara maksimal.

b Mengenai Proses Pembentukan Kelompok: Untuk memastikan pemerataan kesempatan belajar dan keadilan dalam pengerjaan tugas, kami menyarankan agar proses pembentukan kelompok dapat ditentukan langsung oleh pihak asisten dosen. Proses pemilihan kelompok secara mandiri terkadang dapat menimbulkan ketidakseimbangan distribusi kemampuan antar kelompok. Dengan pembentukan kelompok yang ditentukan oleh asisten dosen, diharapkan setiap kelompok memiliki komposisi anggota yang lebih seimbang. Selain itu, hal ini juga akan melatih mahasiswa untuk dapat bekerja sama secara profesional dengan berbagai macam rekan kerja, yang merupakan sebuah persiapan penting untuk dunia kerja di masa depan.

6 Daftar Pustaka

- Rinaldi Munir. Slide Kuliah IF2123 Aljabar Linier dan Geometri. https://informatika.stei.itb.ac.id 2026/algeo25-26.htm
- https://www.geeksforgeeks.org/dsa/gaussian-elimination/

7 Lampiran

a Untuk mengakses langkah-langkah eksperimen bisa diakses di sini