Laporan Tugas Besar 1

IF2123 Aljabar Linier dan Geometri

Oleh kelompok:

Pohon Terbang

13523034	Rafizan Muhammad Syawalazmi
13523063	Syahrizal Bani Khairan
13523100	Aryo Wisanggeni

Bab 1 : Deskripsi Masalah

I. Sistem Persamaan Linier (SPL)

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

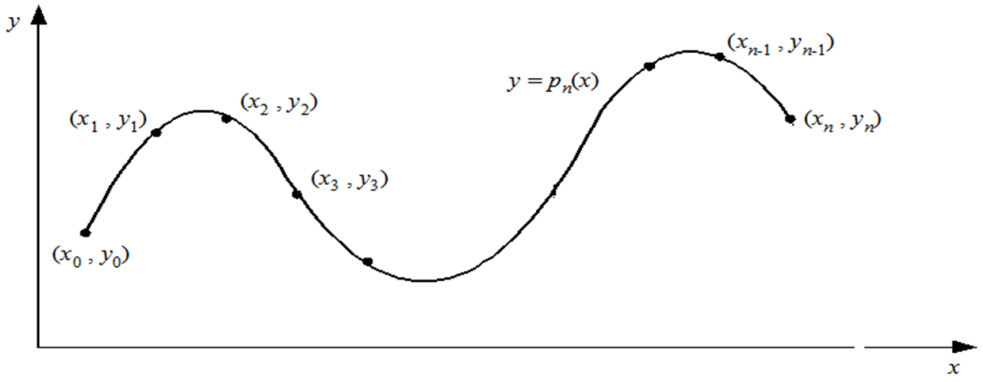
$$\begin{bmatrix} 0 & \mathbf{2} & 1 & -1 \\ 0 & 0 & \mathbf{3} & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad \begin{bmatrix} 0 & \mathbf{1} & 0 & -\frac{2}{3} \\ 0 & 0 & \mathbf{1} & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

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 diminta 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, gunakan *library* tersebut di dalam program Java untuk menyelesaikan berbagai persoalan yang dimodelkan dalam bentuk SPL, menyelesaikan persoalan interpolasi, dan persoalan regresi. Penjelasan tentang interpolasi dan regresi adalah seperti di bawah ini.

II. Interpolasi Polinomial

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



Gambar 2. Ilustrasi beberapa titik yang diinterpolasi secara polinomial.

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

Polinom interpolasi derajat n yang menginterpolasi titik-titik $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$. 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 $a_0, a_1, a_2, \ldots, a_n$,

$$
\begin{aligned}
a_0 + a_1x_0 + a_2{x_0}^2 + \ldots + a_n {x_0}^n &= y_0 \\
a_0 + a_1x_1 + a_2{x_1}^2 + \ldots + a_n {x_1}^n &= y_1 \\
\ldots \qquad\qquad & \qquad \ldots \\
a_0 + a_1x_n + a_2{x_n}^2 + \ldots + a_n {x_n}^n &= y_n
\end{aligned}
$$

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

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

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

III. Regresi Berganda

Regresi (akan dipelajari lebih lanjut di Probabilitas dan Statistika) merupakan salah satu metode untuk memprediksi nilai selain menggunakan Interpolasi Polinom. Pada tugas besar ini, kami diminta untuk membuat 2 jenis regresi yaitu Regresi Linier Berganda dan Regresi Kuadratik Berganda.

1. Regresi Linier Berganda

Meskipun sudah ada persamaan jadi untuk menghitung regresi linear sederhana, terdapat persamaan umum dari regresi linear yang bisa digunakan untuk regresi linear berganda, yaitu.

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

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

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

2. Regresi Kuadratik Berganda

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

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

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

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

N menandakan jumlah peubah, terdapat 2 variabel linier yaitu u_i dan v_i, 2 variabel kuadrat yaitu u_i^2 dan v_i^2, dan 1 variabel interaksi yaitu uv. Untuk setiap n-peubah, akan terdapat 1 konstan N (Terlihat di bagian atas kiri gambar), n variabel linier, n variabel kuadrat, dan C_2^n variabel linier (dengan

syarat n > 1). Tentu dengan bertambahnya peubah n, ukuran matriks akan bertumbuh lebih besar dibandingkan regresi linier berganda tetapi solusi tetap bisa didapat dengan menggunakan SPL.

Kedua model regresi yang dijadikan sistem persamaan linier tersebut diselesaikan dengan menggunakan metode eliminasi Gauss.

IV. *Bicubic Spline Interpolation*

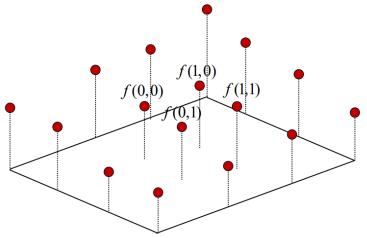
Bicubic spline interpolation adalah metode interpolasi yang digunakan untuk mengaproksimasi fungsi di antara titik-titik data yang diketahui. *Bicubic spline interpolation* melibatkan konsep *spline* dan konstruksi serangkaian polinomial kubik di dalam setiap sel segi empat dari data yang diberikan.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

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

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

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.

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

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

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

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

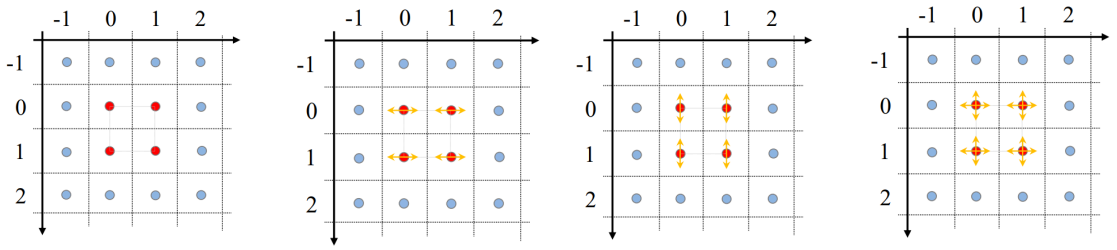
Dengan menggunakan nilai fungsi dan turunan berarah tersebut, dapat terbentuk sebuah matriks solusi *X* yang membentuk persamaan penyelesaian sebagai berikut.

$$y = Xa$$

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

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. Sebagai contoh, elemen matriks *X* pada baris 8 kolom ke 2 adalah koefisien dari a_{10} pada ekspansi sigma untuk $f_x(1, 1)$ sehingga diperoleh nilai konstanta $1 \times 1^{1-1} \times 1^0 = 1$, sesuai dengan isi matriks *X*.

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

Untuk studi kasus ini, akan dibuat matriks *X* menggunakan persamaan yang ada (tidak *hardcode*) serta akan dicari invers matriks *X* dengan *library* yang telah dibuat dalam penyelesaian masalah..

Bab 2 : Teori dasar

2.1. Metode Eliminasi Gauss

Metode eliminasi Gauss mengubah suatu matriks menjadi bentuk matriks eselon baris. Matriks eselon baris adalah matriks yang

1. baris hanya berisi nol berada di bagian bawah matriks dan
2. baris lainnya memiliki pivot (elemen tak nol pertama di baris) di sebelah kanan pivot baris-baris di atasnya.

Sekuens operasi baris elementer dapat mengubah matriks menjadi bentuk matriks eselon baris. Operasi baris elementer adalah

1. Pergantian baris.
2. Pengalian baris dengan konstanta
3. Penjumlahan baris dengan kelipatan baris lain.

2.2. Metode Eliminasi Gauss-Jordan

Metode eliminasi Gauss-Jordan adalah ekstensi dari metode eliminasi Gauss dimana matriks dioperasikan hingga menjadi bentuk eselon baris tereduksi. Matriks eselon baris tereduksi adalah

1. baris hanya berisi nol berada di bagian bawah matriks,
2. baris lainnya memiliki pivot (elemen tak nol pertama di baris) di sebelah kanan pivot baris-baris di atasnya, dan
3. pivot setiap baris bernilai satu dan pivot adalah satu-satunya elemen yang tak nol.

2.3. Determinan

Determinan suatu matriks adalah skalar yang bergantung terhadap elemen-elemen matriks tersebut. Cara pertama untuk menghitung determinan adalah menggunakan ekspansi kofaktor. Ekspansi kofaktor dilakukan dengan mengambil satu baris atau satu kolom. Pada lingkup tugas ini diambil baris pertama dari matriks. Determinan dihitung dengan rumus

$$det(A) = \sum_{j=1}^{n} (-1)^{1+j} a_{1,j} m_{1,j}$$

dimana $a_{1,j}$ adalah elemen matriks baris pertama dan kolom ke-j, dan $m_{1,j}$ adalah determinan dari submatrix A yang berupa matrix A tanpa baris pertama dan kolom ke-j.

Selain itu, ada kasus khusus untuk matriks segitiga atas atau segitiga bawah, determinan memiliki bentuk

$$det(A) = \sum_{i=1}^{n} a_{i,i}$$

2.4. Matriks Balikan

Matriks balikan atau matriks invers dari suatu matriks A adalah matriks A^{-1} *sehingga*

$$AA^{-1} = I$$

dimana I adalah matriks identitas. Matriks dikatakan terbalikkan atau *invertible* jika matriks tersebut memiliki balikan. Matriks yang tidak memiliki balikan disebut matriks *singular*. Matriks tidak memiliki balikan jika $det(A) = 0$.

Matriks balikan dapat ditemukan menggunakan metode eliminasi Gauss-Jordan dengan matriks augmented.

$$[A|I] \sim [I|A^{-1}]$$

2.5. Matriks kofaktor

Matriks kofaktor adalah matriks yang elemennya didefinisikan sebagai berikut

$$C_{ij} = (-1)^{i+j} m_{ij}$$

Matriks ini digunakan untuk menghitung determinan dengan metode ekspansi kofaktor dan sebagai tahap untuk mendapatkan matriks balikan.

2.6. Matriks adjoin

Matriks adjoin adalah transpose dari matriks kofaktor

$$adj(A) = C^T$$

Matriks balikan dari A dapat dibentuk menggunakan matriks adjoin,

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

.

2.7. Kaidah Cramer

Solusi sistem persamaan linier (SPL) dengan jumlah persamaan yang sama dengan jumlah variabel tidak diketahui memiliki rumus eksplisit dengan kaidah Cramer. Dalam kalkulasi solusi SPL dalam n variabel dengan bentuk $Ax = b$, dibutuhkan determinan dari matriks A dan determinan dari n matriks A dengan satu kolomnya diganti dengan b. Misalkan A_i adalah matriks A dengan kolom ke-i diganti dengan b. Maka solusi SPL adalah

$$x_i = \frac{\det(A_i)}{\det(A)} \qquad i = 1, \ldots, n$$

.

2.8. Interpolasi Polinom

Sekumpulan n pasang data berbeda dapat dihubungkan dengan polinom derajat n di bidang kartesius. Polinom ini dapat ditemukan dengan membentuk sistem persamaan linier. Persamaan linier sebanyak n dibangun dengan substitusi n pasang data ke polinom yang belum diketahui koefisien dan konstantanya. Solusi SPL dapat ditemukan dengan cara-cara yang telah disebutkan sebelumnya.

2.9. Interpolasi Bicubic Spline

Metode interpolasi dua dimensi yang digunakan untuk memperhalus data dalam grid atau gambar. Ini melibatkan pemakaian spline kubik (persamaan polinomial derajat tiga) secara berturut-turut dalam dua arah (x dan y) untuk menghasilkan nilai interpolasi di antara titik-titik data yang diketahui.

2.10. Regresi Linier Berganda

Metode statistik yang digunakan untuk memodelkan hubungan antara satu variabel dependen (respons) dan dua atau lebih variabel independen (prediktor). Tujuan dari regresi ini adalah untuk menemukan persamaan linear yang menggambarkan bagaimana variabel dependen dipengaruhi oleh variabel-variabel independen.

2.11. Regresi Kuadrat Berganda

Data yang terdiri variabel terikat dan beberapa variabel bebas dapat juga diregresi secara kuadrat berganda. Fungsi regresi adalah penjumlahan berbobot dari sejumlah kombinasi variabel bebas. Kombinasi variabel meliputi semua perkalian variabel yang derajatnya tidak melebihi dua; kombinasi dapat berupa variabel linier, variabel kuadrat, dan variabel interaksi (pengalian dua variabel derajat satu).

Bab 3 : Implementasi Pustaka

3.1. Matrix.java

Atribut

```
public double[][] mem;
public int rowEff;
public int colEff;
```

Atribut	Keterangan
mem	Menyimpan elemen matriks dengan ukuran

	tertentu. Elemen bertipe *double*.
rowEff	Jumlah baris dari matriks
colEff	Jumlah kolom dari matriks

Konstruktor

```
public Matrix(int nRow,int nCol)
```

Membuat matriks nRow x nCol yang setiap elemennya 0. Nilai rowEff dan colEff diset.

Method

Method	Keterangan
`public boolean IsEqualTo(Matrix m)`	Mengembalikan true jika matriks sama dengan matriks *m*. Kedua matriks sama jika jumlah baris dan jumlah kolom sama dan setiap elemennya sama
`public boolean IsSquare()`	Mengembalikan true jika jumlah baris matriks sama dengan jumlah kolom
`public Matrix GetIdentity()`	Mengembalikan matriks identitas yang dimensinya sama
`public Matrix GetTranspose()`	Mengembalikan transpose dari matriks
`public Matrix GetInverse()`	Mengembalikan invers dari matriks. Memanggil fungsi dari *Inverse.java*
`public Matrix Augment(Matrix m)`	Mengembalikan matriks yang telah digabung matriks *m* di sebelah kanan matriks
`public Matrix GetSubMatrix(int iStart, int jStart, int nRow, int nCol)`	Mengembalikan submatriks berdimensi *nRow* x *nCol* dimulai dari indeks *iStart, jStart* secara kontigu
`public Matrix rightMultiply(Matrix m)`	Mengembalikan perkalian kanan dengan matriks *m*
`public Matrix leftMultiply(Matrix m)`	Mengembalikan perkalian kiri dengan matriks *m*
`public double vectorDot(Matrix m)`	Mengembalikan hasil operasi dot dengan matriks *m*. Asumsi kedua matriks berupa matriks kolom (memiliki satu kolom)

`public Matrix multiplyElement(Matrix m)`	Mengembalikan matriks yang setiap elemennya adalah perkalian elemen kedua matriks. Asumsi kedua matriks berdimensi sama
`public void RowSwap(int row1, int row2)`	Mengganti baris ke-*row1* dengan baris ke-*row2*
`public void RowMultiply(int targetRow, double mult)`	Mengalikan baris ke-*targetRow* dengan konstanta *mult*
`public void RowAddition(int targetRow, int additionRow, double mult)`	Menambahkan kelipatan *mult* dari baris ke-*additionRow* ke baris ke-*targetRow*
`public void DeleteCol(int colIDX)`	Menghapus kolom ke-*colIDX* dan mengurangi jumlah kolom
`public void DeleteRow(int rowIDX)`	Menghapus baris ke-*rowIDX* dan mengurangi jumlah baris
`public static String formatDouble(double value)`	Mengembalikan string berupa format variabel *double*
`public void printMatrix()`	Menunjukkan matriks. Menggunakan method *formatDouble*

3.2. MatrixType.java

Atribut

```
```

Atribut	Keterangan
-	-

Konstruktor

Tidak ada konstruktor eksplisit.

Method

Method	Keterangan
`public static Matrix Identity(int n)`	Mengembalikan matriks identitas n x n.

`public static Matrix Hilbert(int n)`	Mengembalikan matriks Hilbert n x n.

3.3. GaussJordan.java

Atribut

Atribut	Keterangan
-	-

Konstruktor

Tidak ada konstruktor eksplisit.

Method

Method	Keterangan
`public static int FindPotentialLeadingOne(Matrix m, int idx)`	Mengembalikan baris yang berpotensi untuk diubah menjadi baris satu utama dengan satu utama berada di kolom ke-idx
`public static Matrix GaussElimination(Matrix m)`	Mengembalikan matrix hasil eliminasi gauss menjadi matrix eselon baris.
`public static Matrix GaussJordanElimination(Matrix m)`	Mengembalikan matrix hasil eliminasi Gauss Jodan menjadi matrix eselon baris tereduksi.

3.4. Determinant.java

Atribut

```
static double tol = 0.0001;
```

Atribut	Keterangan
tol	Toleransi komparasi antara double

Konstruktor

```
```

Method

Method	Keterangan
`public static double det(Matrix m)`	Mengembalikan determinan matriks *m* dengan metode dekomposisi PLU lalu menghitung determinan masing-masing komponen
`public static Matrix minor(Matrix m, int row, int col)`	Mengembalikan submatriks *m* berukuran *m.rowEff-1* x *m.colEff-1* yang berupa matriks *m* baris ke-*row* dan kolom ke-*col*
`public static double detKofaktor(Matrix m)`	Mengembalikan determinan matriks *m* dengan metode ekspansi kofaktor
`public static double detTriangular(Matrix m)`	Mengembalikan determinan matriks *m* dengan mengalikan semua elemen diagonal. Asumsi *m* adalah matriks segitiga agar hasil valid

3.5. Inverse.java

Atribut

```
```

Atribut	Keterangan
-	-

Konstruktor

Tidak ada konstruktor eksplisit.

Method

Method	Keterangan

`public static Matrix inverse(Matrix m)`	Mengembalikan invers matriks *m* menggunakan method *inverseGaussJordan*
`public static Matrix inverseGaussJordan(Matrix m)`	Mengembalikan invers matriks *m* menggunakan metode eliminasi Gauss-Jordan
`public static Matrix inverseAdj(Matrix m)`	Mengembalikan invers matriks *m* menggunakan metode matriks adjoint

3.6. LinearRegression.java

Atribut

Atribut	Keterangan
-	-

Konstruktor

Tidak ada konstruktor eksplisit.

Method

Method	Keterangan
`public static Matrix Regression(Matrix m)`	Mengembalikan parameter hasil regresi linear berganda dari data di matriks *m*
`public static Matrix AugmentOnesCol(Matrix m, int colIdx)`	Mengembalikan matriks yang disisipkan kolom berisikan nilai 1 di setiap elemennya di kolom ke-*colIdx*
`public static double Solve(Matrix sample, Matrix problem)`	Mengembalikan prediksi dari variabel bebas di matriks *problem* berdasarkan hasil regresi dari data di matriks *sample*

3.7. QuadraticRegression.java

Atribut

Atribut	Keterangan

-	-

Konstruktor

Tidak ada konstruktor eksplisit.

Method

Method	Keterangan
`public static Matrix regress(Matrix data)`	Mengembalikan parameter hasil regresi kuadrat berganda dari data di matriks *data*
`public static Matrix getModel(Matrix data)`	Mengembalikan matriks model regresi menggunakan variabel bebas di *data*
`public static Matrix getTarget(Matrix data)`	Mengembalikan matriks target regresi menggunakan variabel terikat di *data*
`public static Matrix[] getVariables(Matrix data)`	Mengembalikan array berisi matriks-matriks kolom berisi kombinasi variabel bebas di sampel *data*

3.8. BicubicInterpolation.java

Atribut

```
```

Atribut	Keterangan
-	-

Konstruktor

Tidak ada konstruktor eksplisit.

Method

Method	Keterangan
`public static Matrix MatrixX()`	Mengembalikan matriks model untuk bicubic spline
`public static Matrix InterpolationConstant (Matrix X, Matrix f)`	Mengembalikan matriks berisi koefisien yang diperoleh dari matriks augment matrixX dan sampel yang diberi
`public static double InterpolationSolve (Matrix Constant, double x, double y){`	Mengembalikan nilai taksiran bicubic spline interpolation berdasarkan konstanta dari InterpolationConstant dan nilai x dan y yang ingin ditaksir

3.9. Decomposition.java

Atribut

```
public Matrix L, U, P;
```

Atribut	Keterangan
L	Matriks segitiga bawah.
U	Matriks segitiga atas
P	Matriks permutasi

Konstruktor

Tidak ada konstruktor eksplisit.

Method

Method	Keterangan
`public static Matrix MatrixX()`	Mengembalikan matriks model untuk bicubic spline
`public void decomposeLUP(Matrix m)`	Melakukan dekomposisi LUP matriks *m* dan menyimpannya di atribut *L*, *U*, dan *P*.
`public Matrix solve(Matrix target)`	Mengembalikan solusi persamaan *Ptarget = LUx* dengan *L*, *U*, dan *P* yang tersimpan di atribut

3.10. PolinomialInterpolation.java

Atribut

Atribut	Keterangan
-	-

Konstruktor

Tidak ada konstruktor eksplisit.

Method

Method	Keterangan
`public static Matrix GetPolinomialInterpolation(Matrix m)`	Mengembalikan matrix hasil koefisien polinomial interpolasi
`public static double GetEstimate(Matrix a, int x)`	Mengembalikan hasil taksiran dari polinomial interpolasi
`public static void PrintPolinomialInterpolation(Matrix a)`	Meng-output hasil polinomial interpolasi

3.11. MatrixOutput.java

Atribut

Atribut	Keterangan
-	-

Konstruktor

Tidak ada konstruktor eksplisit.

Method

Method	Keterangan
`public static int FindSolutionType(Matrix m)`	Mengembalikan tipe solusi SPL seperti tidak ada solusi, ada satu solusi unik, atau ada banyak solusi.
`public static void GetSPLSingleSolution(Matrix m)`	Mencari dan mengeluarkan solusi SPL yang memiliki satu solusi unik.
`public static void GetSPLManySolution(Matrix m)`	Mencari dan mengeluarkan solusi SPL yang memiliki banyak solusi
`public static void GetSPLSolution(Matrix m)`	Mencari dan mengeluarkan solusi SPL secara general.

3.12. InputMatrix.java

Atribut

Atribut	Keterangan
-	-

Konstruktor

Tidak ada konstruktor eksplisit.

Method

Method	Keterangan
`public BicubicInput InputBicubicKeyBoard()`	Mendapatkan input keyboard / CLI untuk Interpolasi Bicubic Spline
`public BicubicInput InputBicubicFile(String filename)`	Mendapatkan input text file untuk interpolasi bicubic spline
`public Matrix InputMatrixKeyBoard()`	Mendapatkan input keyboard / CLI untuk membuat matrix

`public Matrix InputMatrixFile(String filename)`	Mendapatkan input text file untuk membuat matrix.
`public InterpolationInput InputInterpolationKeyBoard()`	Mendapatkan input keyboard / CLI untuk memlakukan interpolasi
`public InterpolationInput InputInterpolationFile(String filename)`	Mendapatkan input text file untuk melakukan interpolasi
`public RegressionInput InputRegressionKeyBoard()`	Mendapatkan input keyboard / CLI untuk melakukan regresi
`public RegressionInput InputRegressionFile (String filename)`	Mendapatkan input text file untuk melakukan regresi
`static private double parseDouble(String value)`	Mengembalikan string double yang telah digantikan koma ',' menjadi titik '.'
`static private double getDoubleInput(Scanner scanner, String prompt)`	Mengembalikan double dari string yang menggunakan koma ',' ataupun titik '.'

Bab 4 : Eksperimen

Lorem ipsum dolor sit amet

4.1. Sistem Persamaan Linier

```
Jumlah row: 4
Jumlah column: 5
Enter per baris dengan spasi antar kolom:
1 1 -1 -1 1
2 5 -7 -5 -2
2 -1 1 3 4
5 2 -4 2 6
Here is the end matrix from gauss elimination
1 0 0 0.6667 0
0 1 0 -2.6667 0
0 0 1 -1 0
0 0 0 0 1

Tidak ada solusi!
```

```
Jumlah row: 4
Jumlah column: 6
Enter per baris dengan spasi antar kolom:
1 -1 0 0 1 3
1 1 0 -3 0 6
2 -1 0 1 -1 5
-1 2 0 -2 -1 -1
Here is the end matrix from gauss elimination
1 0 0 0 -1 3
0 1 0 0 -2 0
0 0 0 1 -1 -1
0 0 0 0 0 0

Solusi Parametrik:
x1 = 3 + 1t
x2 =  + 2t
x3 = s
x4 = -1 + 1t
x5 = t
```

```
Jumlah row: 3
Jumlah column: 7
Enter per baris dengan spasi antar kolom:
0 1 0 0 1 0 2
0 0 0 1 1 0 -1
0 1 0 0 0 1 1
Here is the end matrix from gauss elimination
0 1 0 0 0 1 1
0 0 0 1 0 1 -2
0 0 0 0 1 -1 1

Solusi Parametrik:
x1 = s
x2 = 1 - 1u
x3 = t
x4 = -2 - 1u
x5 = 1 + 1u
```

4.2. Interpolasi Polinom

4.3. Regresi Berganda

4.4. Bicubic Spline Interpolation

Bab 5 : Kesimpulan

Dari melakukan tugas besar ini, kami menyadari pentingnya ilmu aljabar linier dan geometri. Terdapat banyak penerapan yang bisa dilakukan dari teori yang diterapkan. Dengan mempelajari dan menerapkan berbagai metode ini, kita dapat mengembangkan kemampuan analisis yang lebih baik, memperkuat dasar-dasar matematika, serta memperluas wawasan dalam penggunaan algoritma dan pemrograman untuk memecahkan masalah nyata.

Selain itu, penguasaan teknik-teknik ini membantu meningkatkan kemampuan berpikir logis dan kritis, yang sangat penting dalam berbagai bidang ilmu pengetahuan dan teknologi, termasuk komputasi