**Nama : Bagaskara Dipowicaksono HP  
NIM : 21120122140119**

**Implementasi Sistem Persamaan Linier Menggunakan Java**

Link Github : https://github.com/Gramthegrob/Implementasi-SIstem-Persamaan-Linear

1. **Matriks Invers**
2. **Source Code**

|  |
| --- |
| import java.util.Scanner;  public class MatrixInverse {   // Fungsi untuk mencetak matriks  public static void printMatrix(double[][] matrix) {  for (int i = 0; i < matrix.length; i++) {  for (int j = 0; j < matrix[i].length; j++) {  System.*out*.print(matrix[i][j] + " ");  }  System.*out*.println();  }  }   // Fungsi untuk mencari invers matriks 2x2  public static double[][] inverseMatrix(double[][] matrix) {  double det = (matrix[0][0] \* matrix[1][1]) - (matrix[0][1] \* matrix[1][0]);  if (det == 0) {  System.*out*.println("Determinan matriks nol, tidak ada invers.");  return null;  }   double[][] inverse = new double[2][2];  inverse[0][0] = matrix[1][1] / det;  inverse[0][1] = -matrix[0][1] / det;  inverse[1][0] = -matrix[1][0] / det;  inverse[1][1] = matrix[0][0] / det;   return inverse;  }   public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);   // Memasukkan elemen matriks  System.*out*.println("Masukkan elemen matriks 2x2:");  double[][] matrix = new double[2][2];  for (int i = 0; i < 2; i++) {  for (int j = 0; j < 2; j++) {  matrix[i][j] = scanner.nextDouble();  }  }   // Mencetak matriks input  System.*out*.println("Matriks input:");  *printMatrix*(matrix);   // Mencari invers matriks  double[][] inverse = *inverseMatrix*(matrix);   // Mencetak matriks invers jika ada  if (inverse != null) {  System.*out*.println("Matriks Invers:");  *printMatrix*(inverse);  }   scanner.close();  } } |

Penjelasan alur :

* Pertama, kita akan menggunakan *library*  ‘scanner’ dengan memasukkan syntax import java.util.Scanner untuk mendapatkan masukan dari pengguna secara langsung sehingga program memiliki kebebasan untuk memasukkan nilai matriks yang diinginkan
* Baris public static void printMatrix(double[][] matrix)digunakan untuk mengindekskan kolom dan baris matriks melalui variabel i dan j. Dalam kelas ini menggunakan array 2 dimensi.
* Baris public static double[][] inverseMatrix(double[][] matrix)digunakan untuk mencari matriks berukuran 2x2. Blok kode ini berguna untuk mengecek apakah masukan yang diberikan memiliki hasil invers atau tidak.
* double det = (matrix[0][0] \* matrix[1][1]) - (matrix[0][1] \* matrix[1][0]) baris ini digunakan untuk mencari determinan dari matriks yang diberikan
* Pada method double[][] inverse = new double[2][2]; matriks yang diberikan akan dihitung sesuai dengan rumus matriks invers dengan mengambil nilai determinan yang sudah terlebih dahulu dihitung.
* Pada method public static void main(String[] args)akan meminta masukan dari pengguna melalui objek scanner, masukan yang diberikan oleh pengguna akan diterjemahkan menuju bentuk matriks melalui double[][] matrix = new double[2][2]

1. **Kode Testing**

|  |
| --- |
| public class MatrixInverseTest {   public static void main(String[] args) {  // Matriks contoh  double[][] matrix = {{2, 3}, {1, 4}};   // Menampilkan matriks input  System.*out*.println("Matriks Input:");  MatrixInverse.*printMatrix*(matrix);   // Mencari invers matriks  double[][] inverse = MatrixInverse.*inverseMatrix*(matrix);   // Menampilkan matriks invers jika ada  if (inverse != null) {  System.*out*.println("Matriks Invers:");  MatrixInverse.*printMatrix*(inverse);  }   // Menggunakan matriks dengan determinan nol  double[][] singularMatrix = {{1, 2}, {2, 4}};  System.*out*.println("Matriks Input (Determinan Nol):");  MatrixInverse.*printMatrix*(singularMatrix);  System.*out*.println("Mencari invers matriks (Determinan Nol):");  MatrixInverse.*inverseMatrix*(singularMatrix);  } } |

* Pada class ini, digunakan method yang dipanggil dari source code yang digunakan untuk mencari hasil invers dari matriks yang diberikan oleh pengguna.

1. **Dekomposisi LU Gauss**
2. **Source code**

|  |
| --- |
| public class GaussElimination {   // Method untuk mencetak matriks  public static void printMatrix(double[][] matrix) {  int rows = matrix.length;  int cols = matrix[0].length;  for (int i = 0; i < rows; i++) {  for (int j = 0; j < cols; j++) {  System.*out*.print(matrix[i][j] + "\t");  }  System.*out*.println();  }  }   // Method untuk melakukan eliminasi Gauss  public static void gaussianElimination(double[][] matrix) {  int rows = matrix.length;  int cols = matrix[0].length;   for (int i = 0; i < rows - 1; i++) {  if (matrix[i][i] == 0) {  System.*out*.println("Pembagian oleh nol terdeteksi. Metode gagal.");  return;  }  for (int k = i + 1; k < rows; k++) {  double factor = matrix[k][i] / matrix[i][i];  for (int j = i; j < cols; j++) {  matrix[k][j] -= factor \* matrix[i][j];  }  }  }  }   public static void main(String[] args) {  double[][] matrix = {  {2, -1, 1},  {-3, 3, 3},  {1, 1, 5}  };   System.*out*.println("Matriks awal:");  *printMatrix*(matrix);   *gaussianElimination*(matrix);   System.*out*.println("\nMatriks segitiga atas:");  *printMatrix*(matrix);   System.*out*.println("\nMatriks segitiga bawah:");  // Cetak matriks dengan mengganti elemen diagonal utama dengan 1  for (int i = 0; i < matrix.length; i++) {  for (int j = 0; j < matrix[0].length; j++) {  if (i == j) {  System.*out*.print("1\t");  } else if (i < j) {  System.*out*.print("0\t");  } else {  System.*out*.print(matrix[i][j] + "\t");  }  }  System.*out*.println();  }  } } |

Penjelasan alur kode :

* Pertama didefiinisikan terlebih dahulu untuk baris dan kolom pada matriks yang ada dalam program. Dengan menggunakan int rows = matrix.length;  
  int cols = matrix[0].length loop bersarang digunakan untuk iterasi baris dan kolom matriks yang akan ditambahkan untuk mencetak baris selanjutnya.
* Method gaussianElimination digunakan untuk melakukan operasi gauss metode ini akan mengecek elemen diagonal pada matriks yang selanjutnya akan mengurangi baris di bawah baris saat ini dikurangi dengan actor yang sesuai uutk membuat elemen di bawah diagonal utama menjadi nol
* double[][] matrix = metode ini digunakan untuk menginisalisasikan matriks awal dengan nilai nilai yang telah ditentukan terlebih dahulu
* hasil perhitungan akan diproses pada matriks segitiga atas dan matriks segitiga bawah. Segitiga bawah menggantikan elemen di atas diagonal utama dengan 0 dan elemen diagonal utama dengan 1

1. **Kode Testing**

|  |
| --- |
| public class GaussEliminationTest {  public static void main(String[] args) {  // Matriks contoh  double[][] matrix = {  {2, -1, 1},  {-3, 3, 3},  {1, 1, 5}  };  // Cetak matriks awal  System.out.println("Matriks Awal:");  GaussElimination.printMatrix(matrix);  // Lakukan eliminasi Gauss  GaussElimination.gaussianElimination(matrix);  // Cetak matriks segitiga atas  System.out.println("\nMatriks Segitiga Atas:");  GaussElimination.printMatrix(matrix);  // Cetak matriks segitiga bawah  System.out.println("\nMatriks Segitiga Bawah:");  for (int i = 0; i < matrix.length; i++) {  for (int j = 0; j < matrix[0].length; j++) {  if (i == j) {  System.out.print("1\t");  } else if (i < j) {  System.out.print("0\t");  } else {  System.out.print(matrix[i][j] + "\t");  }  }  System.out.println();  }  }  } |

1. **Metode Dekomposisi Crout**
2. Source code

|  |
| --- |
| public class CroutDecomposition {  public static void croutDecomposition(double[][] A, double[][] L, double[][] U) {  int n = A.length;  for (int i = 0; i < n; i++) {  // Mengisi elemen diagonal matriks bawah (L) dengan 1  L[i][i] = 1;  // Menghitung elemen matriks atas (U)  for (int k = i; k < n; k++) {  double sum = 0;  for (int j = 0; j < i; j++) {  sum += (L[i][j] \* U[j][k]);  }  U[i][k] = A[i][k] - sum;  }  // Menghitung elemen matriks bawah (L)  for (int k = i + 1; k < n; k++) {  double sum = 0;  for (int j = 0; j < i; j++) {  sum += (L[k][j] \* U[j][i]);  }  L[k][i] = (A[k][i] - sum) / U[i][i];  }  }  }  public static void printMatrix(double[][] matrix) {  int rows = matrix.length;  int cols = matrix[0].length;  for (int i = 0; i < rows; i++) {  for (int j = 0; j < cols; j++) {  System.out.print(matrix[i][j] + "\t");  }  System.out.println();  }  }  public static void main(String[] args) {  double[][] A = {  {2, 3, 1},  {4, 7, 5},  {8, 9, 10}  };  int n = A.length;  double[][] L = new double[n][n];  double[][] U = new double[n][n];  croutDecomposition(A, L, U);  System.out.println("Matriks L:");  printMatrix(L);  System.out.println("\nMatriks U:");  printMatrix(U);  }  } |

Penjelasan alur :

* Definisikan jumlah baris yang nantinya akan saam antara Panjang matriks yaitu jumlah baris dengan lebar matriks yaitu jumlah kolom melalui sintaks int n = A.length
* Gunakan loop yang akan berjalan sepanjang banyak jumlah baris matriks a. matriks ini akan diiterasi melalui setiap baris untuk menghasilkan matriks segitiga bawah dan segitiga atas. Operasi ini direpresentasikan dengan sintaks for (int i = 0; i < n; i++)
* Selanjutnya mengisi elemen diagonal utama dari matriks segitiga bawah dengan 1 sesuai dengan aturan operasi dekomposisi crout. Operasi ini direprensentasikan dengan blok L[i][i] = 1;
* Menghitung elemen matriks segitiga atas di man setiap elemen di atas diagonal utama dihitung dengan mengurangi hasil perkalian elemen yang sesuai degna baris i matriks L dengan kolom K matriks U dari elemen di ats diagonal utama. Operasi ini direpresentasikan dengan blok for (int k = i; k < n; k++) {
* Menghitung elemen matriks di bawah segitiga L di mana setiap elemen di bawah diagonal utama dari matriks L dihitung dengan mengurangi hasil perkalian elemen yang sesuai dengan brais k matriks L dengan kolom i martriks u dari elemen di atas diagonal utama kemudian dibagi dengan elemen diagonal utama dari matriks u. hasil akhir dari operasi ini akan disimpan pada matrks L. operasi ini direpresentasikan oleh blok for (int k = i + 1; k < n; k++)
* Memasukkan nilai matriks tertentu ke dalam program melalui blok double[][] A =
* Deklarasi matriks L dan u yang akan menympan hasil dekomposisi crout int n = A.length;

double[][] L = new double[n][n];

double[][] U = new double[n][n];

1. Kode Testing

|  |
| --- |
| public class CroutDecompositionTest {  public static void main(String[] args) {  // Matriks contoh  double[][] A = {  {2, 3, 1},  {4, 7, 5},  {8, 9, 10}  };  // Mencetak matriks awal  System.out.println("Matriks Awal:");  CroutDecomposition.printMatrix(A);  // Membuat matriks L dan U untuk menyimpan hasil dekomposisi  int n = A.length;  double[][] L = new double[n][n];  double[][] U = new double[n][n];  // Melakukan dekomposisi Crout  CroutDecomposition.croutDecomposition(A, L, U);  // Mencetak matriks segitiga bawah (L)  System.out.println("\nMatriks Segitiga Bawah (L):");  CroutDecomposition.printMatrix(L);  // Mencetak matriks segitiga atas (U)  System.out.println("\nMatriks Segitiga Atas (U):");  CroutDecomposition.printMatrix(U);  }  } |