**נספח**

 התוכנית הבאה מריצה את הרוטינה של אלימינציה של גאוס עם partial pivoting על מטריצה נתונה. על מנת להקל את השימוש בתוכנית מקדמי המטריצה והווקטור b נתונים בתוך קובץ ששמו מועבר כפרמטר לתוכנית ראשית (args[1]). התוכנית מדפיסה את הפתרון אך מדפיסה פלט המאפשר למשתמש לבדוק ולהשתכנע שהפתרון שהוחזר אכן פותר את המערכת.

// gaus1a.java - Partial pivoting

import java.util.Scanner;

public class gaus1a

{

 static void print\_result(double [][]A, double []x,

 double [] b, int n)

 {

 int i, j;

 double sum;

 String temp;

 temp = "";

 System.out.println("Solution X:");

 for(i=0; i < n; i++)

 temp += " X[" + i + "] ";

 System.out.println(temp);

 temp = "";

 for(i=0; i < n; i++)

 temp += " " + x[i] + " ";

 System.out.println(temp);

 System.out.print("\n Verification:\n");

 for(i=0; i < n; i++)

 {

 sum = A[i][0] \* x[0];

 System.out.print(A[i][0] + " \* " + x[0] + " ");

 temp = "";

 for(j=1; j < n; j++)

 {

 sum = sum + A[i][j] \* x[j];

 temp += " + " + A[i][j]+ " \* " + x[j] + " ";

 } // for

 System.out.println(temp);

 System.out.println(" = " + sum +" ?= " + b[i]);

 } // for

 } // print\_result

 static void swap\_rows(double [][]A, int n, int m1, int m2)

 {

 int i;

 double temp;

 for(i=0; i <= n; i++)

 {

 temp = A[m1][i];

 A[m1][i] = A[m2][i];

 A[m2][i] = temp;

 } // for

 } // swap\_rows

 static void gaussian(double [][]A, double []b, int n, double x[])

 {

 int i, j, k, p;

 double [][]W;

 double [][]M;

 double MaxValue;

 M = new double [n][n+1];

 W = new double [n][n+1];

 for(i=0; i < n; i++)

 for(j=0; j < n; j++)

 W[i][j] = A[i][j];

 for(i=0; i < n; i++)

 W[i][n] = b[i];

 for (k=0; k < n; k++)

 {

 p = k;

 MaxValue = Math.abs(W[k][k]);

 for(i=k+1; i < n; i++)

 if (Math.abs(W[i][k]) > MaxValue)

 {

 p = i;

 MaxValue = Math.abs(W[i][k]);

 } // if

 swap\_rows(W, n, k, p);

 for(i=k+1; i < n; i++)

 M[i][k] = W[i][k]/W[k][k];

 for(i=k+1; i < n; i++)

 W[i][k] = 0;

 for(i=k+1; i < n; i++)

 for(j=k+1; j <= n; j++)

 W[i][j] = W[i][j] - M[i][k]\*W[k][j];

 } // for

 x[n-1] = W[n-1][n]/W[n-1][n-1];

 for(i=n-2; i >= 0; i--)

 {

 double temp;

 temp = W[i][n];

 for(k=i+1; k < n; k++)

 temp = temp - W[i][k]\*x[k];

 x[i] = temp/W[i][i];

 } // for \*/

 } // gaussian \*/

 static void read\_file(Scanner fp, double [][]A, double []b, int n)

 {

 int i, j;

 for(i=0; i < n; i++)

 for(j=0; j < n; j++)

 A[i][j] = fp.nextDouble();

 for(i=0; i < n; i++)

 b[i] = fp.nextDouble();

 } // read\_file

 static void print\_original\_system(double [][]A, double []b, int n)

 {

 int i, j;

 String temp;

 System.out.println("Original System:");

 for(i=0; i < n; i++)

 {

 temp = "";

 for(j=0; j < n; j++)

 temp += " " + A[i][j] + " ";

 System.out.println(temp + " " + b[i] + " ");

 } // for

 } // print\_original\_system

 public static void main(String args[])

 {

 int i, n;

 double A[][], b[], x[];

 Scanner sc;

 if (args.length < 1)

 {

 System.out.println("Usage: gaussian filename\n");

 return;

 } // if

 try {

 sc = new Scanner(new java.io.File(args[0]));

 }

 catch(java.io.FileNotFoundException e)

 {

 System.out.println("File not found");

 return;

 }// catch

 n = sc.nextInt();

 A = new double [n+1][n];

 b = new double [n];

 x = new double [n];

 read\_file(sc, A, b, n);

 gaussian(A, b, n, x);

 System.out.println(" A x = b");

 print\_original\_system(A, b, n);

 print\_result(A, x, b, n);

 } // main \*/

} // gaus1a

אם, לדוגמא, תוכן הקובץ הוא

5 🡨------------------ n מימד המטריצה

2 3 -1 0 5 /||\

1 6 2 -3 -1 ||

2 0 1 4 -2 || מקדמי המטריצה

0 5 -2 1 3 ||

3 1 4 -2 7 \||/

17 /||\

10 ||

-24 || b ערכי הוקטור

-2 ||

48 \||/

פלט הריצה הינו

A x = b

Original System:

 2.0 3.0 -1.0 0.0 5.0 17.0

 1.0 6.0 2.0 -3.0 -1.0 10.0

 2.0 0.0 1.0 4.0 -2.0 -24.0

 0.0 5.0 -2.0 1.0 3.0 -2.0

 3.0 1.0 4.0 -2.0 7.0 48.0

Solution X:

 X[0] X[1] X[2] X[3] X[4]

 0.9999999999999964 -0.9999999999999989 2.000000000000002 -4.999999999999998 4.000000000000001

 Verification:

2.0 \* 0.9999999999999964 + 3.0 \* -0.9999999999999989 + -1.0 \* 2.000000000000002 + 0.0 \* -4.999999999999998 + 5.0 \* 4.000000000000001

 = 16.999999999999996 ?= 17.0

1.0 \* 0.9999999999999964 + 6.0 \* -0.9999999999999989 + 2.0 \* 2.000000000000002 + -3.0 \* -4.999999999999998 + -1.0 \* 4.000000000000001

 = 10.000000000000004 ?= 10.0

2.0 \* 0.9999999999999964 + 0.0 \* -0.9999999999999989 + 1.0 \* 2.000000000000002 + 4.0 \* -4.999999999999998 + -2.0 \* 4.000000000000001

 = -24.0 ?= -24.0

0.0 \* 0.9999999999999964 + 5.0 \* -0.9999999999999989 + -2.0 \* 2.000000000000002 + 1.0 \* -4.999999999999998 + 3.0 \* 4.000000000000001

= -1.9999999999999947 ?= -2.0

3.0 \* 0.9999999999999964 + 1.0 \* -0.9999999999999989 + 4.0 \* 2.000000000000002 + -2.0 \* -4.999999999999998 + 7.0 \* 4.000000000000001

 = 48.0 ?= 48.0

התוכנית הבאה מממשת את האלימינציה של גאוס עם scaling בנוסף ל-partial pivoting:

// gaus2a.c - Partial pivoting with scaling

import java.util.Scanner;

public class gaus2a

 {

 static void print\_result(double A[][],

 double x[], double b[], int n)

 {

 int i, j;

 double sum;

 System.out.print("Solution X:\n");

 for(i=0; i < n; i++)

 System.out.print(" X[" + i + "] ");

 System.out.println();

 for(i=0; i < n; i++)

 System.out.print(" " + x[i] + " ");

 System.out.println("\n Verification:");

 for(i=0; i < n; i++)

 {

 sum = A[i][0] \* x[0];

 System.out.print(A[i][0]+ " \* " + x[0] + " ");

 for(j=1; j < n; j++)

 {

 sum = sum + A[i][j] \* x[j];

 System.out.print(" + " + A[i][j] + " \* " +x[j] + " ");

 } // for

 System.out.println();

 System.out.println(" = " + sum + " ?= " + b[i]);

 } // for

 } // print\_result

 static void swap\_rows(double A[][], int n, int m1, int m2)

 {

 int i;

 double temp;

 for(i=0; i <= n; i++)

 {

 temp = A[m1][i];

 A[m1][i] = A[m2][i];

 A[m2][i] = temp;

 } // for

 } // swap\_rows

 static void gaussian(double A[][], double b[], int n, double x[])

 {

 int i, j, k, p;

 double [][]W;

 double [][]M;

 double MaxValue, ScaleValue, temp;

 M = new double [n][n+1];

 W = new double [n][n+1];

 for(i=0; i < n; i++)

 for(j=0; j < n; j++)

 W[i][j] = A[i][j];

 for(i=0; i < n; i++)

 W[i][n] = b[i];

 for(i=0; i < n; i++)

 {

 ScaleValue = Math.abs(W[i][0]);

 for(j=1; j < n; j++)

 {

 temp = Math.abs(W[i][j]);

 if (temp > ScaleValue)

 ScaleValue = temp;

 }

 for(j=0; j <= n; j++)

 W[i][j] = W[i][j]/ScaleValue;

 } // for

 for (k=0; k < n; k++)

 {

 p = k;

 MaxValue = Math.abs(W[k][k]);

 for(i=k+1; i < n; i++)

 if (Math.abs(W[i][k]) > MaxValue)

 {

 p = i;

 MaxValue = Math.abs(W[i][k]);

 } // if

 swap\_rows(W, n, k, p);

 for(i=k+1; i < n; i++)

 M[i][k] = W[i][k]/W[k][k];

 for(i=k+1; i < n; i++)

 W[i][k] = 0;

 for(i=k+1; i < n; i++)

 for(j=k+1; j <= n; j++)

 W[i][j] = W[i][j] - M[i][k]\*W[k][j];

 } // for

 x[n-1] = W[n-1][n]/W[n-1][n-1];

 for(i=n-2; i >= 0; i--)

 {

 double tempd;

 tempd = W[i][n];

 for(k=i+1; k < n; k++)

 tempd = tempd - W[i][k]\*x[k];

 x[i] = tempd/W[i][i];

 } // for

 } // gaussian

 static void read\_file(Scanner fp, double [][]A, double []b, int n)

 {

 int i, j;

 for(i=0; i < n; i++)

 for(j=0; j < n; j++)

 A[i][j] = fp.nextDouble();

 for(i=0; i < n; i++)

 b[i] = fp.nextDouble();

 } // read\_file

 static void print\_original\_system(double [][]A, double []b, int n)

 {

 int i, j;

 String temp;

 System.out.println("Original System:");

 for(i=0; i < n; i++)

 {

 temp = "";

 for(j=0; j < n; j++)

 temp += " " + A[i][j] + " ";

 System.out.println(temp + " " + b[i] + " ");

 } // for

 } // print\_original\_system

 public static void main(String args[])

 {

 int i, n;

 double A[][], b[], x[];

 Scanner sc;

 if (args.length < 1)

 {

 System.out.println("Usage: gaussian filename\n");

 return;

 } // if

 try {

 sc = new Scanner(new java.io.File(args[0]));

 }

 catch(java.io.FileNotFoundException e)

 {

 System.out.println("File not found");

 return;

 }// catch

 n = sc.nextInt();

 A = new double [n+1][n];

 b = new double [n];

 x = new double [n];

 read\_file(sc, A, b, n);

 gaussian(A, b, n, x);

 System.out.println(" A x = b");

 print\_original\_system(A, b, n);

 print\_result(A, x, b, n);

 } // main

 } // gaus2a

פלט ריצה:

A x = b

Original System:

 2.0 17.0 -1.0 0.0 5.0 3.0

 1.0 10.0 2.0 -3.0 -1.0 6.0

 2.0 -24.0 1.0 4.0 -2.0 0.0

 0.0 -2.0 -2.0 1.0 3.0 5.0

 3.0 48.0 4.0 -2.0 7.0 1.0

Solution X:

 X[0] X[1] X[2] X[3] X[4]

 0.9999999999999948 -1.0000000000000002 2.000000000000003 -5.000000000000001 4.000000000000002

 Verification:

2.0 \* 0.9999999999999948 + 17.0 \* -1.0000000000000002 + -1.0 \* 2.000000000000003 + 0.0 \* -5.000000000000001 + 5.0 \* 4.000000000000002

 = 2.9999999999999893 ?= 3.0

1.0 \* 0.9999999999999948 + 10.0 \* -1.0000000000000002 + 2.0 \* 2.000000000000003 + -3.0 \* -5.000000000000001 + -1.0 \* 4.000000000000002

 = 6.000000000000002 ?= 6.0

2.0 \* 0.9999999999999948 + -24.0 \* -1.0000000000000002 + 1.0 \* 2.000000000000003 + 4.0 \* -5.000000000000001 + -2.0 \* 4.000000000000002

 = -7.105427357601002E-15 ?= 0.0

0.0 \* 0.9999999999999948 + -2.0 \* -1.0000000000000002 + -2.0 \* 2.000000000000003 + 1.0 \* -5.000000000000001 + 3.0 \* 4.000000000000002

= 4.999999999999998 ?= 5.0

3.0 \* 0.9999999999999948 + 48.0 \* -1.0000000000000002 + 4.0 \* 2.000000000000003 + -2.0 \* -5.000000000000001 + 7.0 \* 4.000000000000002

 = 1.0 ?= 1.0

התוכנית מממשת את האלימינציה של גאוס עם scaling ו-full pivoting:

// gaus3a.java - scaling and full pivoting

import java.util.Scanner;

public class gaus3a

 {

 static void print\_result(double A[][],

 double x[], double b[], int n)

 {

 int i, j;

 double sum;

 System.out.print("Solution X:\n");

 for(i=0; i < n; i++)

 System.out.print(" X[" + i + "] ");

 System.out.println();

 for(i=0; i < n; i++)

 System.out.print(" " + x[i] + " ");

 System.out.println("\n Verification:");

 for(i=0; i < n; i++)

 {

 sum = A[i][0] \* x[0];

 System.out.print(A[i][0]+ " \* " + x[0] + " ");

 for(j=1; j < n; j++)

 {

 sum = sum + A[i][j] \* x[j];

 System.out.print(" + " + A[i][j] + " \* " +x[j] + " ");

 } // for

 System.out.println();

 System.out.println(" = " + sum + " ?= " + b[i]);

 } // for

 } // print\_result

 static void swap\_rows(double A[][], int n, int m1, int m2)

 {

 int i;

 double temp;

 for(i=0; i <= n; i++)

 {

 temp = A[m1][i];

 A[m1][i] = A[m2][i];

 A[m2][i] = temp;

 } // for

 } // swap\_rows

 static void swap\_cols(double A[][], int n,

 int m1, int m2, int xindex[])

 {

 int i, itemp;

 double dtemp;

 itemp = xindex[m1];

 xindex[m1] = xindex[m2];

 xindex[m2] = itemp;

 for(i=0; i < n; i++)

 {

 dtemp = A[i][m1];

 A[i][m1] = A[i][m2];

 A[i][m2] = dtemp;

 } // for

 } // swap\_cols

 static void gaussian(double A[][], double b[], int n, double x[])

 {

 int i, j, k, p, q;

 double [][]W;

 double [][]M;

 double []y;

 int []xindex;

 double MaxValue, ScaleValue, temp;

 M = new double [n][n+1];

 W = new double [n][n+1];

 for(i=0; i < n; i++)

 for(j=0; j < n; j++)

 W[i][j] = A[i][j];

 for(i=0; i < n; i++)

 W[i][n] = b[i];

 xindex = new int [n];

 for(i=0; i < n; i++)

 xindex[i] = i;

 for(i=0; i < n; i++)

 {

 ScaleValue = Math.abs(W[i][0]);

 for(j=1; j < n; j++)

 {

 temp = Math.abs(W[i][j]);

 if (temp > ScaleValue)

 ScaleValue = temp;

 }

 for(j=0; j <= n; j++)

 W[i][j] = W[i][j]/ScaleValue;

 } // for

 for (k=0; k < n; k++)

 {

 p = k;

 q = k;

 MaxValue = Math.abs(W[k][k]);

 for(i=k; i < n; i++)

 for(j=k; j < n; j++)

 if (Math.abs(W[i][j]) > MaxValue)

 {

 p = i;

 q = j;

 MaxValue = Math.abs(W[i][j]);

 } // if

 swap\_cols(W, n, k, q, xindex);

 swap\_rows(W, n, k, p);

 for(i=k+1; i < n; i++)

 M[i][k] = W[i][k]/W[k][k];

 for(i=k+1; i < n; i++)

 W[i][k] = 0;

 for(i=k+1; i < n; i++)

 for(j=k+1; j <= n; j++)

 W[i][j] = W[i][j] - M[i][k]\*W[k][j];

 } // for

 y = new double [n];

 y[n-1] = W[n-1][n]/W[n-1][n-1];

 for(i=n-2; i >= 0; i--)

 {

 double tempd;

 tempd = W[i][n];

 for(k=i+1; k < n; k++)

 tempd = tempd - W[i][k]\*y[k];

 y[i] = tempd/W[i][i];

 } // for

 for(i=0; i < n; i++)

 x[xindex[i]] = y[i];

 } // gaussian

 static void read\_file(Scanner fp, double [][]A, double []b, int n)

 {

 int i, j;

 for(i=0; i < n; i++)

 for(j=0; j < n; j++)

 A[i][j] = fp.nextDouble();

 for(i=0; i < n; i++)

 b[i] = fp.nextDouble();

 } // read\_file

 static void print\_original\_system(double [][]A, double []b, int n)

 {

 int i, j;

 String temp;

 System.out.println("Original System:");

 for(i=0; i < n; i++)

 {

 temp = "";

 for(j=0; j < n; j++)

 temp += " " + A[i][j] + " ";

 System.out.println(temp + " " + b[i] + " ");

 } // for

 } // print\_original\_system

 public static void main(String args[])

 {

 int i, n;

 double A[][], b[], x[];

 Scanner sc;

 if (args.length < 1)

 {

 System.out.println("Usage: gaussian filename\n");

 return;

 } // if

 try {

 sc = new Scanner(new java.io.File(args[0]));

 }

 catch(java.io.FileNotFoundException e)

 {

 System.out.println("File not found");

 return;

 }// catch

 n = sc.nextInt();

 A = new double [n+1][n];

 b = new double [n];

 x = new double [n];

 read\_file(sc, A, b, n);

 gaussian(A, b, n, x);

 System.out.println(" A x = b");

 print\_original\_system(A, b, n);

 print\_result(A, x, b, n);

 } // main

} // gaus3

פלט ריצה:

A x = b

Original System:

 2.0 -3.0 2.0 5.0 3.0

 1.0 -1.0 1.0 2.0 1.0

 3.0 2.0 2.0 1.0 0.0

 1.0 1.0 -3.0 -1.0 0.0

Solution X:

 X[0] X[1] X[2] X[3]

 -5.000000000000006 6.000000000000008 -2.0000000000000018 7.000000000000008

 Verification:

2.0 \* -5.000000000000006 + -3.0 \* 6.000000000000008 + 2.0 \* -2.0000000000000018 + 5.0 \* 7.000000000000008

 = 3.0 ?= 3.0

1.0 \* -5.000000000000006 + -1.0 \* 6.000000000000008 + 1.0 \* -2.0000000000000018 + 2.0 \* 7.000000000000008

 = 1.0 ?= 1.0

3.0 \* -5.000000000000006 + 2.0 \* 6.000000000000008 + 2.0 \* -2.0000000000000018 + 1.0 \* 7.000000000000008

 = 2.6645352591003757E-15 ?= 0.0

1.0 \* -5.000000000000006 + 1.0 \* 6.000000000000008 + -3.0 \* -2.0000000000000018 + -1.0 \* 7.000000000000008

 = -8.881784197001252E-16 ?= 0.0

התוכנית הבאה היא תוכנית המבצעת אלימינציה של גאוס תוך הבאה בחשבון שהמטריצה עשויה להיות סינגולרית:

// gaus4a.java - Partial pivoting with scaling

import java.util.Scanner;

public class gaus4a

 {

 static void print\_result(double A[][],

 double x[], double b[], int n)

 {

 int i, j;

 double sum;

 System.out.print("Solution X:\n");

 for(i=0; i < n; i++)

 System.out.print(" X[" + i + "] ");

 System.out.println();

 for(i=0; i < n; i++)

 System.out.print(" " + x[i] + " ");

 System.out.println("\n Verification:");

 for(i=0; i < n; i++)

 {

 sum = A[i][0] \* x[0];

 System.out.print(A[i][0]+ " \* " + x[0] + " ");

 for(j=1; j < n; j++)

 {

 sum = sum + A[i][j] \* x[j];

 System.out.print(" + " + A[i][j] + " \* " +x[j] + " ");

 } // for

 System.out.println();

 System.out.println(" = " + sum + " ?= " + b[i]);

 } // for

 } // print\_result

 static void swap\_rows(double A[][], int n, int m1, int m2)

 {

 int i;

 double temp;

 for(i=0; i <= n; i++)

 {

 temp = A[m1][i];

 A[m1][i] = A[m2][i];

 A[m2][i] = temp;

 } // for

 } // swap\_rows

 static int gaussian(double A[][], double b[], int n, double x[])

 {

 int i, j, k, p;

 double [][]W;

 double [][]M;

 double MaxValue, ScaleValue, temp;

 double epsilon = 0.0000001;

 M = new double [n][n+1];

 W = new double [n][n+1];

 for(i=0; i < n; i++)

 for(j=0; j < n; j++)

 W[i][j] = A[i][j];

 for(i=0; i < n; i++)

 W[i][n] = b[i];

 for(i=0; i < n; i++)

 {

 ScaleValue = Math.abs(W[i][0]);

 for(j=1; j < n; j++)

 {

 temp = Math.abs(W[i][j]);

 if (temp > ScaleValue)

 ScaleValue = temp;

 }

 for(j=0; j <= n; j++)

 W[i][j] = W[i][j]/ScaleValue;

 } // for

 for (k=0; k < n; k++)

 {

 // Check if matrix is singular by

 // testng if the current row is zero

 MaxValue = 0;

 for(j=0; j < n; j++)

 {

 temp = Math.abs(W[k][j]);

 if (MaxValue < temp)

 MaxValue = temp;

 } // for

 if (MaxValue < epsilon) // Row of zeros?

 return 0;

 // End of singular check \*/

 p = k;

 MaxValue = Math.abs(W[k][k]);

 for(i=k+1; i < n; i++)

 if (Math.abs(W[i] [k]) > MaxValue)

 {

 p = i;

 MaxValue = Math.abs(W[i] [k]);

 }

 swap\_rows(W, n, k, p);

 for(i=k+1; i < n; i++)

 M[i][k] = W[i][k]/W[k][k];

 for(i=k+1; i < n; i++)

 W[i][k] = 0;

 for(i=k+1; i < n; i++)

 for(j=k+1; j <= n; j++)

 W[i][j] = W[i][j] - M[i][k]\*W[k][j];

 } // for

 x[n-1] = W[n-1][n]/W[n-1][n-1];

 for(i=n-2; i >= 0; i--)

 {

 double tempd;

 tempd = W[i][n];

 for(k=i+1; k < n; k++)

 tempd = tempd - W[i][k]\*x[k];

 x[i] = tempd/W[i][i];

 } // for

 return 1;

 } // gaussian

 static void read\_file(Scanner fp, double [][]A, double []b, int n)

 {

 int i, j;

 for(i=0; i < n; i++)

 for(j=0; j < n; j++)

 A[i][j] = fp.nextDouble();

 for(i=0; i < n; i++)

 b[i] = fp.nextDouble();

 } // read\_file

 static void print\_original\_system(double [][]A, double []b, int n)

 {

 int i, j;

 String temp;

 System.out.println("Original System:");

 for(i=0; i < n; i++)

 {

 temp = "";

 for(j=0; j < n; j++)

 temp += " " + A[i][j] + " ";

 System.out.println(temp + " " + b[i] + " ");

 } // for

 } // print\_original\_system

 public static void main(String args[])

 {

 int i, n;

 double A[][], b[], x[];

 Scanner sc;

 if (args.length < 1)

 {

 System.out.println("Usage: gaussian filename\n");

 return;

 } // if

 try {

 sc = new Scanner(new java.io.File(args[0]));

 }

 catch(java.io.FileNotFoundException e)

 {

 System.out.println("File not found");

 return;

 }// catch

 n = sc.nextInt();

 A = new double [n+1][n];

 b = new double [n];

 x = new double [n];

 read\_file(sc, A, b, n);

 System.out.println(" A x = b");

 print\_original\_system(A, b, n);

 if (gaussian(A, b, n, x) == 0)

 {

 System.out.println("MATRIX IS SINGULAR");

 return;

 } // if

 print\_result(A, x, b, n);

 } // main

 } // gaus4a

פלט ריצה:

A x = b

Original System:

 1.0 -1.0 1.0 -1.0 1.0

 2.0 0.0 3.0 -1.0 -3.0

 3.0 1.0 -1.0 4.0 2.0

 4.0 2.0 1.0 4.0 2.0

MATRIX IS SINGULAR

התוכנית הבאה היא גרסת השדרוג של scaling ו-full pivoting:

// gaus5a.java - scaling and full pivoting

import java.util.Scanner;

public class gaus5a

 {

 static void print\_result(double A[][],

 double x[], double b[], int n)

 {

 int i, j;

 double sum;

 System.out.print("Solution X:\n");

 for(i=0; i < n; i++)

 System.out.print(" X[" + i + "] ");

 System.out.println();

 for(i=0; i < n; i++)

 System.out.print(" " + x[i] + " ");

 System.out.println("\n Verification:");

 for(i=0; i < n; i++)

 {

 sum = A[i][0] \* x[0];

 System.out.print(A[i][0]+ " \* " + x[0] + " ");

 for(j=1; j < n; j++)

 {

 sum = sum + A[i][j] \* x[j];

 System.out.print(" + " + A[i][j] + " \* " +x[j] + " ");

 } // for

 System.out.println();

 System.out.println(" = " + sum + " ?= " + b[i]);

 } // for

 } // print\_result

 static void swap\_rows(double A[][], int n, int m1, int m2)

 {

 int i;

 double temp;

 for(i=0; i <= n; i++)

 {

 temp = A[m1][i];

 A[m1][i] = A[m2][i];

 A[m2][i] = temp;

 } // for

 } // swap\_rows

 static void swap\_cols(double A[][], int n,

 int m1, int m2, int xindex[])

 {

 int i, itemp;

 double dtemp;

 itemp = xindex[m1];

 xindex[m1] = xindex[m2];

 xindex[m2] = itemp;

 for(i=0; i < n; i++)

 {

 dtemp = A[i][m1];

 A[i][m1] = A[i][m2];

 A[i][m2] = dtemp;

 } // for

 } // swap\_cols

 static int gaussian(double A[][], double b[], int n, double x[])

 {

 int i, j, k, p, q;

 double [][]W;

 double [][]M;

 double []y;

 int []xindex;

 double MaxValue, ScaleValue, temp;

 double epsilon = 0.0000001;

 M = new double [n][n+1];

 W = new double [n][n+1];

 for(i=0; i < n; i++)

 for(j=0; j < n; j++)

 W[i][j] = A[i][j];

 for(i=0; i < n; i++)

 W[i][n] = b[i];

 xindex = new int [n];

 for(i=0; i < n; i++)

 xindex[i] = i;

 for(i=0; i < n; i++)

 {

 ScaleValue = Math.abs(W[i][0]);

 for(j=1; j < n; j++)

 {

 temp = Math.abs(W[i][j]);

 if (temp > ScaleValue)

 ScaleValue = temp;

 }

 for(j=0; j <= n; j++)

 W[i][j] = W[i][j]/ScaleValue;

 } // for

 for (k=0; k < n; k++)

 {

 // Check if matrix is singular by

 // testng if the current row is zero

 MaxValue = 0;

 for(j=0; j < n; j++)

 {

 temp = Math.abs(W[k][j]);

 if (MaxValue < temp)

 MaxValue = temp;

 } /\* for \*/

 if (MaxValue < epsilon) // Row of zeros?

 {

 return 0;

 } // if

 // End of singular check

 p = k;

 q = k;

 MaxValue = Math.abs(W[k][k]);

 for(i=k; i < n; i++)

 for(j=k; j < n; j++)

 if (Math.abs(W[i][j]) > MaxValue)

 {

 p = i;

 q = j;

 MaxValue = Math.abs(W[i][j]);

 } // if

 swap\_cols(W, n, k, q, xindex);

 swap\_rows(W, n, k, p);

 for(i=k+1; i < n; i++)

 M[i][k] = W[i][k]/W[k][k];

 for(i=k+1; i < n; i++)

 W[i][k] = 0;

 for(i=k+1; i < n; i++)

 for(j=k+1; j <= n; j++)

 W[i][j] = W[i][j] - M[i][k]\*W[k][j];

 } // for

 y = new double [n];

 y[n-1] = W[n-1][n]/W[n-1][n-1];

 for(i=n-2; i >= 0; i--)

 {

 double tempd;

 tempd = W[i][n];

 for(k=i+1; k < n; k++)

 tempd = tempd - W[i][k]\*y[k];

 y[i] = tempd/W[i][i];

 } // for

 for(i=0; i < n; i++)

 x[xindex[i]] = y[i];

 return 1;

 } // gaussian

 static void read\_file(Scanner fp, double [][]A, double []b, int n)

 {

 int i, j;

 for(i=0; i < n; i++)

 for(j=0; j < n; j++)

 A[i][j] = fp.nextDouble();

 for(i=0; i < n; i++)

 b[i] = fp.nextDouble();

 } // read\_file

 static void print\_original\_system(double [][]A, double []b, int n)

 {

 int i, j;

 String temp;

 System.out.println("Original System:");

 for(i=0; i < n; i++)

 {

 temp = "";

 for(j=0; j < n; j++)

 temp += " " + A[i][j] + " ";

 System.out.println(temp + " " + b[i] + " ");

 } // for

 } // print\_original\_system

 public static void main(String args[])

 {

 int i, n;

 double A[][], b[], x[];

 Scanner sc;

 if (args.length < 1)

 {

 System.out.println("Usage: gaussian filename\n");

 return;

 } // if

 try {

 sc = new Scanner(new java.io.File(args[0]));

 }

 catch(java.io.FileNotFoundException e)

 {

 System.out.println("File not found");

 return;

 }// catch

 n = sc.nextInt();

 A = new double [n+1][n];

 b = new double [n];

 x = new double [n];

 read\_file(sc, A, b, n);

 System.out.println(" A x = b");

 print\_original\_system(A, b, n);

 if (gaussian(A, b, n, x) == 0)

 {

 System.out.println("MATRIX IS SINGULAR");

 return;

 } // if

 print\_result(A, x, b, n);

 } // main

} // gaus5

פלט ריצה:

A x = b

Original System:

 1.0 -1.0 1.0 -1.0 1.0

 2.0 0.0 3.0 -1.0 -3.0

 3.0 1.0 -1.0 4.0 2.0

 4.0 2.0 1.0 4.0 2.0

MATRIX IS SINGULAR