Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică ăi Microelectronică

Departamentul Informatică Sofware și Automate

RAPORT

despre lucrarea de laborator nr. 2

la disciplina Metode și modele de calcul

Tema: Rezolvarea sistemelor de ecuații liniare

A efectuat: st. gr. TI-173 Heghea Nicolae

A verificat: conf. univ Tutunaru Eleonora

Cuprins

[1. Sarcina lucrării 3](#_Toc531675066)

[2. Noțiuni generale metoda iterativă Jacobi 3](#_Toc531675067)

[2.1 Schema bloc 4](#_Toc531675068)

[2.2 Codul Sursă 5](#_Toc531675069)

[2.3 Rezultate 7](#_Toc531675070)

[3. Concluzia 8](#_Toc531675071)

# Sarcina lucrării

1. Să se rezolve sistemul de ecuații liniare în forma matricială .

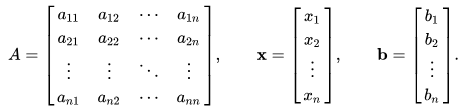
n, m se citesc de la tastieră.

Se afișează matricea Q, și vectorul D, soluția sistemului.

Și dacă condiția nu are loc, să afișeze ce să facem.

# Noțiuni generale metoda iterativă Jacobi

Fie , și un sistem de liniar ecuații, ca :



Atunci soluția se obține iterativ prin formula :



## Schema bloc

Nu

Da

Da

Nu

Da

Nu

Da

Nu

Da

Nu

Da

Nu

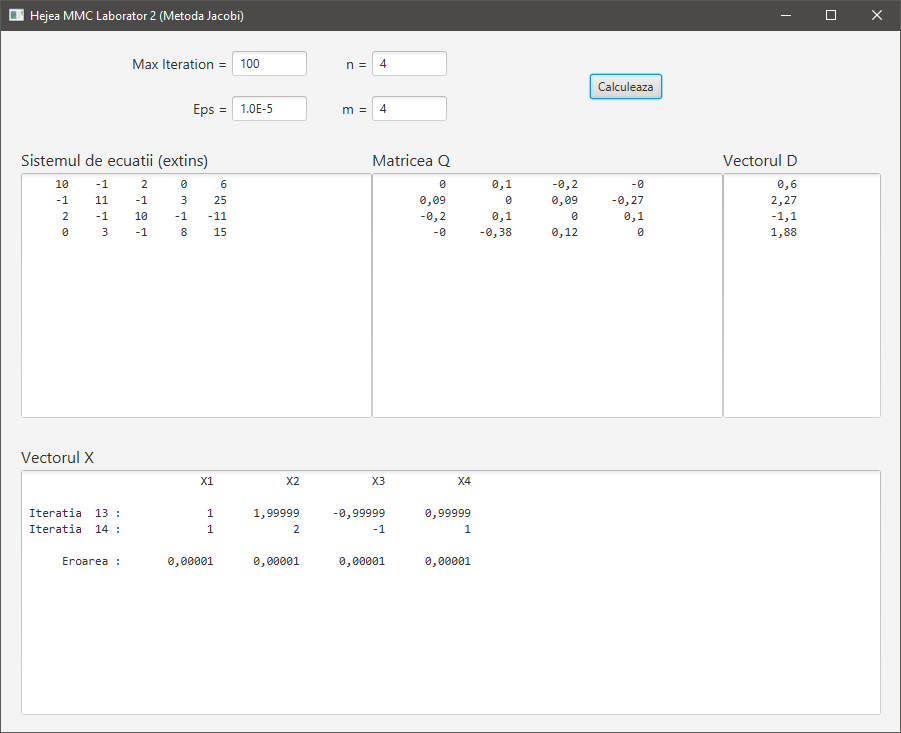
## Codul Sursă

|  |
| --- |
| public void run() throws Exception {    sys = new SystemEQ();  sys.init();    int i, j, k;  double sum;    if (!Convergenta.*diagonalDominant*(sys)) {  throw new Exception("Matricea nu este convergenta");  } else {    k = 0;  do {  double[] x1 = sys.X.get(k).clone();  double[] x = sys.X.get(k).clone();    for (i = 0; i < sys.Eq.length; i++) {    sys.D[i] = sys.Eq[i][sys.Eq.length] / sys.Eq[i][i];  sum = 0.0;    for (j = 0; j < sys.Eq.length; j++) {  if (j != i) {  sys.Q[i][j] = -sys.Eq[i][j] / sys.Eq[i][i];  sum += sys.Eq[i][j] \* x[j];  } else {  sys.Q[i][j] = 0;  }  }  x1[i] = sys.Eq[i][sys.Eq.length] - sum;  x1[i] /= sys.Eq[i][i];  }    sys.X.add(x1);  if (*CondStopEps*(sys)) break;    k++;  } while (k < *nrMaxIteration*);  } } |

|  |
| --- |
| public static boolean CondStopEps(SystemEQ sys) {    int i;  boolean stop = true;  double[] x1 = sys.X.get(sys.X.size() - 1);  double[] x0 = sys.X.get(sys.X.size() - 2);  double max = sys.DifE[0] = *abs*(x1[0] - x0[0]);    for (i = 1; i < sys.Eq.length; i++) {    sys.DifE[i] = *abs*(x1[i] - x0[i]);    if (max > sys.DifE[i]) {  max = sys.DifE[i];  }  }    if (max > *eps*) {  stop = false;  }    return stop; } |
| public static boolean diagonalDominant(SystemEQ sys) {    int i, j;  boolean converge = true;  double aux;    for (i = 0; i < sys.Eq.length; i++) {    aux = 0.0;    for (j = 0; j < sys.Eq.length; j++) {  if (i != j) {  aux += *abs*(sys.Eq[i][j]);  }  }  if (*abs*(sys.Eq[i][i]) <= aux) {  converge = false;  break;  }  }    return converge; } |

## Rezultate

Când are soluții :



Citeva erori care pot să apară :

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Concluzia

Metoda Jacobi este un algoritm iterativ pentru determinarea soluțiilor intr-un sistem de ecuații liniare diagonal dominant. Calculează elementele de pe diagonală, de la o aproximație inițială. Iterează atâta timp cât converge sau nu depășește o eroare dată.

Această metoda converge destul de repede, dar mai poate fi înbinătățit prin :

În aceiași iterație începând cu , pentru calcularea lor se vor folosi din pasul curet și nu din precedent. Atunci această metodă converge extrem de rapid.