СПб НИУ ИТМО

кафедра ИПМ

Вычислительная математика

Лабораторная работа № 5

Обращение симметричной положительно определенной матрицы методом квадратного корня (метод Холецкого)

Работу выполнил:

Студент II курса

Группы № 2120

Журавлев Виталий

Санкт-Петербург

2013 г.

**Цель работы:**

Составить программу, реализующую обращение симметричной положительно определенной матрицы методом квадратного корня (методом Холецкого).

Изучить метод, реализовать и составить отчет.

**Описание метода:**

*Обращение симметричной положительно определенной матрицы методом квадратного корня (метод Холецкого):*

**Разложение Холецкого** — представление симметричной положительно-определённой матрицы  в виде , где  — нижняя треугольная матрица со строго положительными элементами на диагонали. Иногда разложение записывается в эквивалентной форме: , где  — верхняя треугольная матрица. Разложение Холецкого всегда существует и единственно для любой симметричной положительно-определённой матрицы.

Элементы матрицы  можно вычислить, начиная с верхнего левого угла матрицы, по формулам:



, если .

Выражение под корнем всегда положительно, если  — действительная положительно-определённая матрица.

Вычисление происходит сверху вниз, слева направо,т.е. сперва , а затем .

 Матрица C, обратная к матрице A, находится из уравнения
CLLT = E,
где L и LT- треугольные матрицы, E - единичная матрица.
   Вычисление матрицы C проводится в два этапа. На первом этапе вычисляется матрица D исходя из уравнения DLT = E. Расчетные формулы для вычисления матрицы D имеют следующий вид

dii = 1 / ltii
 j-1
dij = -( **∑** diktlkj) / djj    (j = i+1, i+2, ..., N)
 k=i
для i от 1 до N.
   Матрица C вычисляется на втором этапе исходя из уравнения CL = D по формулам
 N
cij = dij - **∑** ciklkj    (i = 1, 2, ..., j)
 k=j+1
 N
wi = - **∑** ciklkj
 k=j+1
cij = wi    (i = j+1, j+2, ..., N)
для j от N-1 до 1.

**Код программы:**

 class Program

 { static void Main(string[] args)

 { HoletskyMethod holetsky;

 Console.Write("Выберите способ ввода иходной матрицы: ");

 string input = Console.ReadLine();

 switch (input)

 { case "1":

 { Console.Write("Введите количество строк (по умолчанию 5): ");

 int NumVar;

 string s = Console.ReadLine();

 if (!Int32.TryParse(s, out NumVar)) NumVar = 5;

 holetsky = new HoletskyMethod(NumVar);

 break; }

 case "2":

 { Console.WriteLine(@"Папка: E:\Matrix\_1#\Gauss\Gauss\bin\Debug");

 Console.Write("Введите имя файла: ");

 string name = Console.ReadLine() + ".txt";

 holetsky = new HoletskyMethod(name);

 break; }

 default: Environment.Exit(0); break; }

 holetsky.HoleskyDecomposition();

 holetsky.SolveMatrix\_1();

 Console.Read(); } }

 class HoletskyMethod

 { int NumEq = 10;

 double[,] Matrix;

 double[,] L;

 double[,] Lt;

 double[,] Matrix\_1;

 public HoletskyMethod(int Num) // ввод с клавиатуры

 { this.NumEq = Num;

 if (Num < 1 || Num > 11) throw new ArgumentException();

 this.Matrix = new double[NumEq, NumEq];

 Console.WriteLine("Введите коэффициенты уравнения:");

 for (int i = 0; i < NumEq; i++)

 { for (int j = 0; j < NumEq; j++)

 { Console.Write("A({0},{1}) = ", i + 1, j + 1);

 this.Matrix[i, j] = Convert.ToDouble(Console.ReadLine()); } }

 OutMatrix(Matrix); }

 public HoletskyMethod(string filename) // Ввод из файла

 { using (StreamReader Reader = new StreamReader(filename))

 { string numb = Reader.ReadLine();

 this.NumEq = int.Parse(numb);

 Matrix = new double[NumEq, NumEq];

 string s = Reader.ReadToEnd();

 char[] separators = new char[] { ' ', '\n', '\t', '\r' };

 string[] mas = s.Split(separators,StringSplitOptions.RemoveEmptyEntries);

 int y = 0;

 for (int i = 0; i < NumEq; i++)

 { for (int j = 0; j < NumEq; j++)

 { if (mas[y] != "") this.Matrix[i, j] = double.Parse(mas[y]);

 y++; } } }

 OutMatrix(Matrix); }

 public void OutMatrix(double[,] Matrix) // вывод уравнений на консоль

 { for (int i = 0; i < NumEq; i++)

 { for (int j = 0; j < NumEq; j++)

 { Console.Write(Matrix[i, j].ToString("F2") + "\t"); }

 Console.Write("\n"); }

 Console.WriteLine(); }

 public void HoleskyDecomposition()

 { L = new double[NumEq, NumEq];

 Lt = new double[NumEq, NumEq];

 for (int i = 0; i < NumEq; i++)

 { double temp;

 for (int j = 0; j < i; j++)

 { temp = 0;

 for (int k = 0; k < j; k++)

 { temp += L[i, k] \* L[j, k]; }

 L[i, j] = (Matrix[i, j] - temp) / L[j, j]; }

 temp = Matrix[i, i];

 for (int k = 0; k < i; k++)

 { temp -= Math.Pow(L[i, k], 2); }

 L[i, i] = Math.Sqrt(temp); }

 for (int i = 0; i < NumEq; i++)

 { for (int j = 0; j < NumEq; j++)

 { Lt[j, i] = L[i, j]; } }

 OutMatrix(L);

 OutMatrix(Lt); }

 public void SolveMatrix\_1()

 { double[,] D = new double[NumEq, NumEq];

 Matrix\_1 = new double[NumEq, NumEq];

 double sum;

 for (int i = 0; i < NumEq; i++) //Нахождение d

 { D[i, i] = 1 / Lt[i, i]; }

 for (int i = 0; i < NumEq; i++)

 { for (int j = i + 1; j < NumEq; j++)

 { sum = 0;

 for (int k = i; k < j; k++)

 { sum += D[i, k] \* Lt[k, j]; }

 D[i, j] = (-sum) / D[j, j]; } }

 for (int j = NumEq-1; j >= 0; j--) //Нахождение A^-1

 { for (int i = 0; i < j+1; i++)

 { sum = 0;

 for (int k = j + 1; k < NumEq; k++)

 { sum += Matrix\_1[i, k] \* L[k, j]; }

 Matrix\_1[i, j] = (D[i, j] - sum); }

 for (int i = j+1; i < NumEq; i++)

 { sum = 0;

 for (int k = j + 1; k < NumEq; k++)

 { sum += Matrix\_1[i, k] \* L[k, j]; }

 Matrix\_1[i, j] = (- sum); } }

 OutMatrix(D);

 OutMatrix(Matrix\_1); } } }

**Вывод:**

В процессе выполнения лабораторной работы было изучено получение обратной матрицы методом квадратного корня (Холецкого). Данный метод достаточно прост и быстр (в сравнении LU-разложения), однако накладывает серьезные ограничения на исходную матрицу.