Санкт-Петербургский государственный университет

**Курсовая работа**

по дисциплине «Численные методы»

на тему: «Метод квадратного корня»

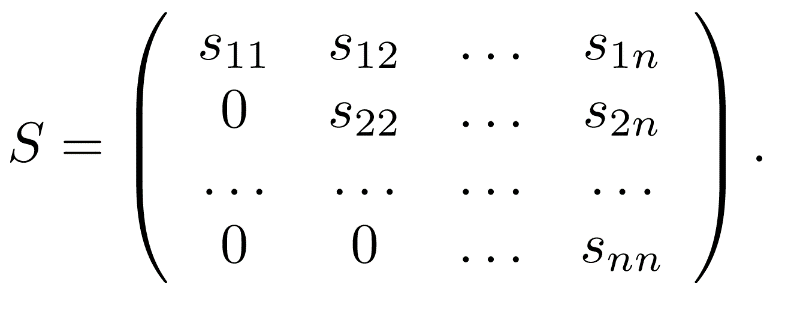
Студент: Марусина Анастасия Петровна

Группа: 209

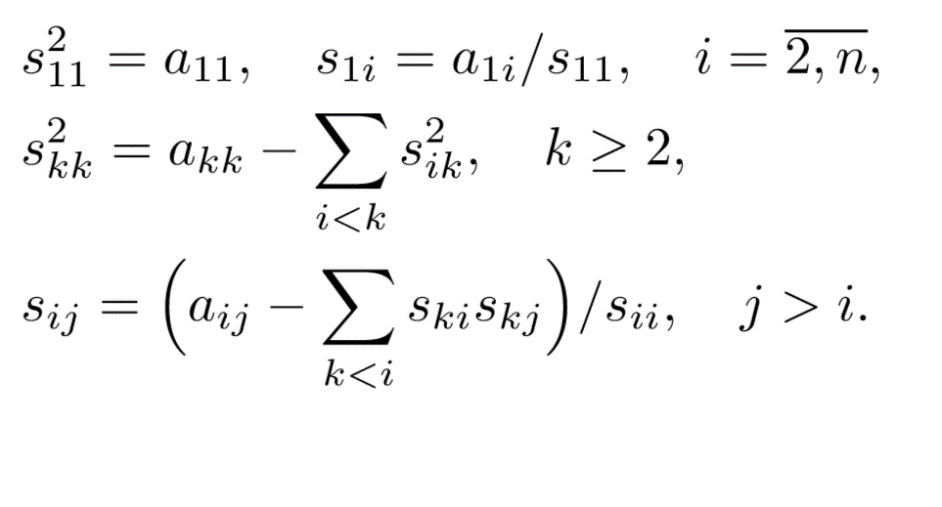
Преподаватель: Перегудин Сергей Иванович

Метод квадратного корня

Будем считать, что матрица A симметричная и положительно определена. Поставим задачу подставить ее в виде произведения: , где S – правая треугольная матрица:



Умножая матрицу на S справа и приравнивая элементы результирующей матрицы соответствующим элементам матрицы A, получим:



Реализация метода на языке C++ используя интегрированную среду разработки Visual Studio 2013:

#include "stdafx.h"

#include <iostream>

using namespace std;

const int HEIGHT = 3;

const int WIDTH = 4;

int \_tmain(int argc, \_TCHAR\* argv[])

{

float Matrix[HEIGHT][WIDTH] = {

{ 10.0, 1.0, 1.0, 12.0 },

{ 1.0, 12.0, 1.0, 14.0 },

{ 1.0, 1.0, 14.0, 16.0 }

};

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

{

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

cout << Matrix[i][j] << "\t";

cout << endl;

}

cout << endl;

float MatrixU[HEIGHT][HEIGHT];

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

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

MatrixU[i][j] = 0;

MatrixU[0][0] = sqrt(Matrix[0][0]);

for (int i = 1; i < HEIGHT; i++)

MatrixU[0][i] = Matrix[0][i] / MatrixU[0][0];

// for Uii

for (int i = 1; i < HEIGHT; i++)

{

float sum = 0;

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

sum = sum + MatrixU[k][i] \* MatrixU[k][i];

MatrixU[i][i] = sqrt(Matrix[i][i] - sum);

if (i + 1 < HEIGHT)

{

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

{

float subsum = 0;

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

subsum = subsum + MatrixU[k][i] \* MatrixU[k][j];

MatrixU[i][j] = 1 / MatrixU[i][i] \* (Matrix[i][j] - subsum);

}

}

}

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

{

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

cout << MatrixU[i][j] << "\t";

cout << endl;

}

cout << endl;

float MatrixUT[HEIGHT][HEIGHT];

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

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

MatrixUT[i][j] = MatrixU[j][i];

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

{

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

cout << MatrixUT[i][j] << "\t";

cout << endl;

}

cout << endl;

float MatrixY[HEIGHT];

MatrixY[0] = Matrix[0][3] / MatrixUT[0][0];

MatrixY[1] = (Matrix[1][3] - MatrixUT[1][0] \* MatrixY[0]) / MatrixUT[1][1];

MatrixY[2] = (Matrix[2][3] - MatrixUT[2][0] \* MatrixY[0] - MatrixUT[2][1] \* MatrixY[1]) / MatrixUT[2][2];

float MatrixX[HEIGHT];

MatrixX[2] = MatrixY[2] / MatrixU[2][2];

MatrixX[1] = (MatrixY[1] - MatrixU[1][2] \* MatrixX[2]) / MatrixU[1][1];

float test = MatrixY[1] - MatrixU[1][2] \* MatrixX[2];

MatrixX[0] = (MatrixY[0] - MatrixU[0][2] \* MatrixX[2] - MatrixU[0][1] \* MatrixX[1]) / MatrixU[0][0];

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

{

cout << "X " << i+1 << "= " << MatrixX[i] << "\t";

cout << endl;

}

cout << endl;

system("pause");

return 0;

}