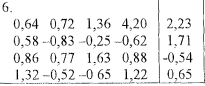
**Постановка задачи**

Решить систему линейных уравнений методом итераций с ускоряющим коэффициентом а с точностью ε = 10^-3. Исследовать зависимость числа итерации от коэффициента 0,1 < α < 1,9, изменяя α с шагом равным 0,1.



**Код программы**

# include <iostream>

# include <iomanip>

# include <cmath>

using namespace std;

const int numPerem = 4;

double norma1(double matrix[numPerem][numPerem])

{

double norma[numPerem];

double max = 0;

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

{

norma[j] = 0;

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

norma[j] += fabs(matrix[i][j]);

if (max < norma[j])

max = norma[j];

}

return max;

}

double norma2(double matrix[numPerem][numPerem])

{

double norma[numPerem];

double max = 0;

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

{

norma[i] = 0;

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

norma[i] += fabs(matrix[i][j]);

if (max < norma[i])

max = norma[i];

}

return max;

}

double norma3(double matrix[numPerem][numPerem])

{

double norma = 0;

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

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

norma += matrix[i][j] \* matrix[i][j];

return sqrt(norma);

}

void comfortability(double matrix[numPerem][numPerem], double rightPart[numPerem], double const speedCoef)

{

double matrixT[numPerem][numPerem];

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

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

matrixT[i][j] = matrix[j][i];

double C[numPerem][numPerem];

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

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

{

C[i][j] = 0;

for (int p = 0; p < numPerem; p++)

C[i][j] += matrixT[i][p] \* matrix[p][j];

}

double norms[3] = { norma1(C), norma2(C), norma3(C) };

double minNorma = norms[0];

for (int i = 1; i < 3; i++)

if (minNorma > norms[i])

minNorma = norms[i];

double G[numPerem][numPerem];

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

{

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

{

G[i][j] = speedCoef \* (-C[i][j]) / minNorma;

}

G[i][i] += 1;

}

double F[numPerem];

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

{

F[i] = 0;

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

F[i] += speedCoef \* matrixT[i][j] / minNorma \* rightPart[j];

}

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

{

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

matrix[i][j] = G[i][j];

rightPart[i] = F[i];

}

}

int iteration\_method(double const matrix[numPerem][numPerem], double const rightPart[numPerem], double XChangable[numPerem], double const epsilon, double const speedCoef = 1)

{

int iterations = 0;

double B[numPerem];

double matrixCount[numPerem][numPerem];

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

{ //Копирование системы

XChangable[i] = 0;

B[i] = rightPart[i];

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

matrixCount[i][j] = matrix[i][j];

}

comfortability(matrixCount, B, speedCoef);

double accuracy = 1;

while (accuracy > epsilon)

{

double XStep[numPerem];

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

{

XStep[i] = XChangable[i];

XChangable[i] = 0;

}

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

{ //Подсчёт X этого шага

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

{

XChangable[i] += matrixCount[i][j] \* XStep[j];

}

XChangable[i] += B[i];

}

double max\_X = 0;

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

{ //Вычисление максимальной невязки

double sum = 0;

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

{

sum += matrix[i][j] \* XChangable[j];

}

sum -= rightPart[i];

if (max\_X < fabs(sum))

max\_X = fabs(sum);

}

accuracy = max\_X;

iterations++;

}

return iterations;

}

int main()

{

setlocale(LC\_ALL, "ru");

const double epsilon = 0.001;

const double matrix[numPerem][numPerem] = {

{ 0.64, 0.72, 1.36, 4.20},

{ 0.58, -0.83, -0.25, -0.62},

{ 0.86, 0.77, 1.63, 0.88},

{ 1.32, -0.52, -0.65, 1.22}

};

const double B[numPerem] = { 2.23, 1.71, -0.54, 0.65 };

double X[numPerem];

cout << "Dannaya matrica: " << endl;

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

{

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

cout << setw(5) << matrix[i][j] << ' ';

cout << " | " << B[i] << endl;

}

cout << endl;

cout << "Kolichestvo iteracii pri znacheniyah yskoryaushego koefficienta" << endl;

for (int alpha = 1; alpha <= 20; alpha++)

cout << setw(3) << alpha / 10. << ": " << iteration\_method(matrix, B, X, epsilon, alpha / 10.) << endl;

cout << endl;

cout << "Result:" << endl;

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

cout << 'X' << i + 1 << " = " << X[i] << endl;

cout << endl;

cout << "Nevyazka:" << endl;

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

{

double result = 0;

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

result += matrix[i][j] \* X[j];

cout << "|B" << i + 1 << " - AX" << i + 1 << "| = " << result - B[i] << endl;

}

cout << endl;

system("pause");

return 0;

}

**Результаты работы программы**