**Федеральное государственное бюджетное образовательное учреждение**

**высшего профессионального образования**

**«Санкт-Петербургский государственный политехнический университет»**

—

**Институт информационных технологий и управления**

**Кафедра «Распределенные вычисления и компьютерные сети»**

**Отчет по лабораторной работе №2**

**На тему: «Решение системы алгебраических уравнений»  
Вариант 23.**

**по дисциплине «Вычислительная математика»**

Выполнил

студент гр. 23507/1 В.Б. Борисов

Руководитель

доц. Т.В. Леонтьева   
  
  
 ≪\_\_\_≫ \_\_\_\_\_\_\_\_\_\_ 2015 г.

Санкт-Петербург

2015

Постановка задачи

Написать процедуру вычисления матрицы Q и вектора z по заданным числам:,,,где

,

Решить систему уравнений Qx=z с помощью **DECOMP** и **SOLVE**, если N=5, , , при следующих значениях параметра 1.0, 0.1, 0.01. Так как – компоненты вектора точного решения (убедиться в этом!), использовать g для оценки погрешности по формуле: , где . Объяснить результаты.

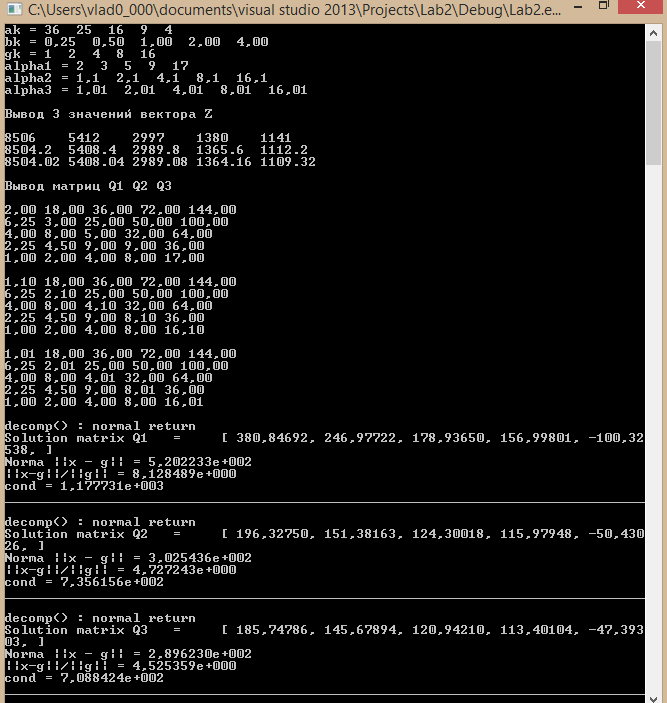
Описание решения

Руководствуясь условию задания, мы вначале решаем три системы, состоящих из трех матриц с помощью программ **DECOMP** и **SOLVE**. Убедившись что нашли правильное решение, мы вычисляем число обусловленности матрицы cond. После мы оцениваем её погрешность.

При выполнении данной работы использовался язык C++, ОСWindows 8.1. Также было проведено тестирование работы программ.

Вывод результатов работы программы:

Результаты тестирования программы были выведены на экран (Результаты работы программы смотри ниже).



Анализ полученных результатов

Для заданной функции было протестировано следующее программное обеспечение: DECOMP и SOLVE. Исходя из полученных результатов, мы можем сделать вывод.  
 Результаты заботы программы показали, что с уменьшением параметра число обусловленности данной матрицы резко возрастает.

Известно, что число обусловленности тем больше, чем больше разница между минимальным и максимальным собственными значениями матрицы. При уменьшении эта разница увеличивается. Следовательно возрастает и число обусловленности, что мы видим в результатах программы.

Величина погрешности при увеличении числа обусловленности убывает.

Исходный текст программы

Files cpp

1) main.cpp

#include <cmath>

#include "cmath.h"

#include <iostream>

#define N 5

#define INDX(i, j) (i) \* N + (j)

using namespace std;

int main()

{

setlocale(LC\_ALL, "RUS");

static double Q1[N \* N], Q2[N \* N], Q3[N \* N], x1[N], x2[N], x3[N],

ak[N], ai[N], gi[N], bi[N], alpha1i[N], alpha2i[N], alpha3i[N], bk[N], alpha1[N], alpha2[N], alpha3[N], cond = 0;

double det = 0, normX\_gk = 0, sigma = 0,sum;

int n = 5, pivot[N], i = 0, j = 0, k = 0, flag = 0;

double gk = 0;

printf("ak = ");

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

printf("%1.0f ", pow(n - k + 1, 2));

printf("\n");

printf("bk = ");

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

printf("%3.2f ", pow(2, k - 2));

printf("\n");

printf("gk = ");

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

printf("%1.0f ", pow(2, k));

printf("\n");

printf("alpha1 = ");

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

printf("%1.0f ", (pow(2, k) + 1));

printf("\n");

printf("alpha2 = ");

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

printf("%1.1f ", (pow(2, k) + 0.1));

printf("\n");

printf("alpha3 = ");

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

printf("%1.2f ", (pow(2, k) + 0.01));

// подсчет значений Z 3 раза

printf("\n\nВывод 3 значений вектора Z\n");

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

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

bi[k] = pow(2, k - 2);

gi[k] = pow(2, k);

}

ai[i] = pow(n - i + 1, 2);

bi[i] = pow(2, i - 2);

alpha1i[i] = pow(2, i) + 1;

alpha2i[i] = pow(2, i) + 0.1;

alpha3i[i] = pow(2, i) + 0.01;

}

sum=bi[1] \* gi[1] + bi[2] \* gi[2] + bi[3] \* gi[3] + bi[4] \* gi[4] + bi[5] \* gi[5];

cout << '\n' << ai[1] \* sum + ((alpha1i[1] - (ai[1] \* bi[1]))\*gi[1]) << '\t' << ai[2] \* sum + ((alpha1i[2] - (ai[2] \* bi[2]))\*gi[2]) << '\t' << ai[3] \* sum + ((alpha1i[3] - (ai[3] \* bi[3]))\*gi[3]) << '\t' << ai[4] \* sum + ((alpha1i[4] - (ai[4] \* bi[4]))\*gi[4]) << '\t' << ai[5] \* sum + ((alpha1i[5] - (ai[5] \* bi[5]))\*gi[5]);

cout << '\n' << ai[1] \* sum + ((alpha2i[1] - (ai[1] \* bi[1]))\*gi[1]) << '\t' << ai[2] \* sum + ((alpha2i[2] - (ai[2] \* bi[2]))\*gi[2]) << '\t' << ai[3] \* sum + ((alpha2i[3] - (ai[3] \* bi[3]))\*gi[3]) << '\t' << ai[4] \* sum + ((alpha2i[4] - (ai[4] \* bi[4]))\*gi[4]) << '\t' << ai[5] \* sum + ((alpha2i[5] - (ai[5] \* bi[5]))\*gi[5]);

cout << '\n' << ai[1] \* sum + ((alpha3i[1] - (ai[1] \* bi[1]))\*gi[1]) << '\t' << ai[2] \* sum + ((alpha3i[2] - (ai[2] \* bi[2]))\*gi[2]) << '\t' << ai[3] \* sum + ((alpha3i[3] - (ai[3] \* bi[3]))\*gi[3]) << '\t' << ai[4] \* sum + ((alpha3i[4] - (ai[4] \* bi[4]))\*gi[4]) << '\t' << ai[5] \* sum + ((alpha3i[5] - (ai[5] \* bi[5]))\*gi[5]);

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

ak[i] = pow(n - i + 1, 2);

bk[i] = pow(2, i - 2);

alpha1[i] = pow(2, i) + 1;

alpha2[i] = pow(2, i) + 0.1;

alpha3[i] = pow(2, i) + 0.01;

}

printf("\n\n");

x1[0] = 8506, x1[1] = 5412, x1[2] = 2997, x1[3] = 1380, x1[4] = 1141;

x2[0] = 8504.2, x2[1] = 5408.4, x2[2] = 2989.8, x2[3] = 1365.6, x2[4] = 1112.2;

x3[0] = 8504.02, x3[1] = 5408.04, x3[2] = 2989.08, x3[3] = 1364.16, x3[4] = 1109.32;

printf("Вывод матриц Q1 Q2 Q3\n\n");

// 1 матрица

Q1[INDX(0, 0)] = alpha1[0]; Q1[INDX(0, 1)] = ak[0] \* bk[1]; Q1[INDX(0, 2)] = ak[0] \* bk[2]; Q1[INDX(0, 3)] = ak[0] \* bk[3]; Q1[INDX(0, 4)] = ak[0] \* bk[4];

Q1[INDX(1, 0)] = ak[1] \* bk[0]; Q1[INDX(1, 1)] = alpha1[1]; Q1[INDX(1, 2)] = ak[1] \* bk[2]; Q1[INDX(1, 3)] = ak[1] \* bk[3]; Q1[INDX(1, 4)] = ak[1] \* bk[4];

Q1[INDX(2, 0)] = ak[2] \* bk[0]; Q1[INDX(2, 1)] = ak[2] \* bk[1]; Q1[INDX(2, 2)] = alpha1[2]; Q1[INDX(2, 3)] = ak[2] \* bk[3]; Q1[INDX(2, 4)] = ak[2] \* bk[4];

Q1[INDX(3, 0)] = ak[3] \* bk[0]; Q1[INDX(3, 1)] = ak[3] \* bk[1]; Q1[INDX(3, 2)] = ak[3] \* bk[2]; Q1[INDX(3, 3)] = alpha1[3]; Q1[INDX(3, 4)] = ak[3] \* bk[4];

Q1[INDX(4, 0)] = ak[4] \* bk[0]; Q1[INDX(4, 1)] = ak[4] \* bk[1]; Q1[INDX(4, 2)] = ak[4] \* bk[2]; Q1[INDX(4, 3)] = ak[4] \* bk[3]; Q1[INDX(4, 4)] = alpha1[4];

// 2 матрица

Q2[INDX(0, 0)] = alpha2[0]; Q2[INDX(0, 1)] = ak[0] \* bk[1]; Q2[INDX(0, 2)] = ak[0] \* bk[2]; Q2[INDX(0, 3)] = ak[0] \* bk[3]; Q2[INDX(0, 4)] = ak[0] \* bk[4];

Q2[INDX(1, 0)] = ak[1] \* bk[0]; Q2[INDX(1, 1)] = alpha2[1]; Q2[INDX(1, 2)] = ak[1] \* bk[2]; Q2[INDX(1, 3)] = ak[1] \* bk[3]; Q2[INDX(1, 4)] = ak[1] \* bk[4];

Q2[INDX(2, 0)] = ak[2] \* bk[0]; Q2[INDX(2, 1)] = ak[2] \* bk[1]; Q2[INDX(2, 2)] = alpha2[2]; Q2[INDX(2, 3)] = ak[2] \* bk[3]; Q2[INDX(2, 4)] = ak[2] \* bk[4];

Q2[INDX(3, 0)] = ak[3] \* bk[0]; Q2[INDX(3, 1)] = ak[3] \* bk[1]; Q2[INDX(3, 2)] = ak[3] \* bk[2]; Q2[INDX(3, 3)] = alpha2[3]; Q2[INDX(3, 4)] = ak[3] \* bk[4];

Q2[INDX(4, 0)] = ak[4] \* bk[0]; Q2[INDX(4, 1)] = ak[4] \* bk[1]; Q2[INDX(4, 2)] = ak[4] \* bk[2]; Q2[INDX(4, 3)] = ak[4] \* bk[3]; Q2[INDX(4, 4)] = alpha2[4];

// 3 матрица

Q3[INDX(0, 0)] = alpha3[0]; Q3[INDX(0, 1)] = ak[0] \* bk[1]; Q3[INDX(0, 2)] = ak[0] \* bk[2]; Q3[INDX(0, 3)] = ak[0] \* bk[3]; Q3[INDX(0, 4)] = ak[0] \* bk[4];

Q3[INDX(1, 0)] = ak[1] \* bk[0]; Q3[INDX(1, 1)] = alpha3[1]; Q3[INDX(1, 2)] = ak[1] \* bk[2]; Q3[INDX(1, 3)] = ak[1] \* bk[3]; Q3[INDX(1, 4)] = ak[1] \* bk[4];

Q3[INDX(2, 0)] = ak[2] \* bk[0]; Q3[INDX(2, 1)] = ak[2] \* bk[1]; Q3[INDX(2, 2)] = alpha3[2]; Q3[INDX(2, 3)] = ak[2] \* bk[3]; Q3[INDX(2, 4)] = ak[2] \* bk[4];

Q3[INDX(3, 0)] = ak[3] \* bk[0]; Q3[INDX(3, 1)] = ak[3] \* bk[1]; Q3[INDX(3, 2)] = ak[3] \* bk[2]; Q3[INDX(3, 3)] = alpha3[3]; Q3[INDX(3, 4)] = ak[3] \* bk[4];

Q3[INDX(4, 0)] = ak[4] \* bk[0]; Q3[INDX(4, 1)] = ak[4] \* bk[1]; Q3[INDX(4, 2)] = ak[4] \* bk[2]; Q3[INDX(4, 3)] = ak[4] \* bk[3]; Q3[INDX(4, 4)] = alpha3[4];

//заполнение матрицы Q1

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

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

Q1[INDX(i, j)];

}

//вывод матрицы Q1 на экран

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

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

printf("%2.2f ", Q1[INDX(i, j)]);

}

printf("\n");

}

//заполнение матрицы Q2

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

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

Q2[INDX(i, j)];

}

printf("\n");

//вывод матрицы Q2 на экран

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

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

printf("%2.2f ", Q2[INDX(i, j)]);

}

printf("\n");

}

printf("\n");

//заполнение матрицы Q3

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

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

Q3[INDX(i, j)];

}

//вывод матрицы Q3 на экран

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

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

printf("%2.2f ", Q3[INDX(i, j)]);

}

printf("\n");

}

printf("\n");

decomp(n, N, Q1, &cond, pivot, &flag);

printf("%s\n", cmathmsg(DECOMP\_C, flag));

if (flag == 0)

{

solve(n, N, Q1, x1, pivot);

printf("Solution matrix Q1 = [ ");

for (i = 0, k = 1; i < n, k <= n; i++, k++)

{

printf("%5.5f, ", x1[i]);

normX\_gk += (x1[i] - pow(2, i))\*(x1[i] - pow(2, i));

}

printf("]");

normX\_gk = sqrt(normX\_gk);

printf("\nNorma ||x - g|| = %e\n", normX\_gk);

sigma = normX\_gk / pow(2, k);

printf("\||x-g||/||g|| = %e\n", sigma);

printf("cond = %e\n", cond);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n", cond);

decomp(n, N, Q2, &cond, pivot, &flag);

printf("%s\n", cmathmsg(DECOMP\_C, flag));

if (flag == 0)

{

solve(n, N, Q2, x2, pivot);

printf("Solution matrix Q2 = [ ");

for (i = 0, k = 1; i < n, k <= n; i++, k++)

{

printf("%5.5f, ", x2[i]);

normX\_gk += (x2[i] - pow(2, i))\*(x2[i] - pow(2, i));

}

printf("]");

normX\_gk = sqrt(normX\_gk);

printf("\nNorma ||x - g|| = %e\n", normX\_gk);

sigma = normX\_gk / pow(2, k);

printf("\||x-g||/||g|| = %e\n", sigma);

printf("cond = %e\n", cond);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n", cond);

decomp(n, N, Q3, &cond, pivot, &flag);

printf("%s\n", cmathmsg(DECOMP\_C, flag));

if (flag == 0)

{

solve(n, N, Q3, x3, pivot);

printf("Solution matrix Q3 = [ ");

for (i = 0,k=1; i < n,k<=n;i++,k++)

{

printf("%5.5f, ", x3[i]);

normX\_gk += (x3[i] - pow(2, i))\*(x3[i] - pow(2, i));

}

printf("]");

normX\_gk = sqrt(normX\_gk);

printf("\nNorma ||x - g|| = %e\n", normX\_gk);

sigma = normX\_gk / pow(2, k);

printf("\||x-g||/||g|| = %e\n", sigma);

printf("cond = %e\n", cond);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n", cond);

}

}

}

system("pause");

}

2)cmathmsg.cpp

3)DECOMPnSolve.cpp

HEADER FILE

1) cmath.h