Московский государственный университет

Имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Компьютерный практикум по курсу

«Введение в численные методы»

Задание №1

**ИТЕРАЦИОННЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ АЛГЕБРАИЧЕСКИХ ЛИНЕЙНЫХ УРАВНЕНИЙ**

**(на примере методов Зейделя и верхней релаксации)**

Отчет о выполненном задании

Студента 203 учебной группы факультета ВМК МГУ

Чернышев Алексей Александрович

Москва, 2015 г.

Постановка задачи:

Дана система уравнений с невырожденной матрицей. Написать программу численного решения данной системы из n линейных алгебраических уравнений (n – параметр программы), использующую численный алгоритм итерационного метода верхней релаксации: , где D и A(-) – соответственно диагональная и нижняя треугольная матрицы, k – номер текущей итерации, **ω**  – итерационный параметр (параметр программы).

Предусмотреть возможность задания элементов матрицы системы как во входном файле, так и путем задания специальных формул.

Цели и задачи практической работы:

1. Решить заданную СЛАУ итерационным методом Зейделя (или более общим методом верхней релаксации).
2. Разработать критерий остановки итерационного процесса, гарантирующий получение приближенного решения исходной системы СЛАУ с заданной точностью eps (параметр программы).
3. Изучить скорость сходимости итераций к точному решению (при использовании итерационного метода верхней релаксации провести эксперименты с различными значения итерационного параметра ω (в случае симметрической положительно определенной матрицы системы известно, что для сходимости итераций следует выбирать 0 < ω < 2; при ω = 1 метод верхней релаксации совпадает с методом Зейделя)
4. Правильность решения подтвердить системой тестов, используя ресурсы онлайн – систем.

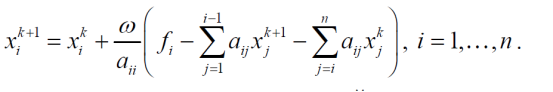
Описание метода:

Рассмотрим произвольную квадратную матрицу:

Разложим её на сумму трех матриц A = A(-) + D + A(+)

где D - диагональная часть матрицы A, которая содержит элементы , стоящие на главной диагонали, A(-) - нижняя треугольная матрица, A(+) - верхняя треугольная матрица. Запишем стандартную каноническую форму линейного одношагового алгоритма решения СЛАУ с введением параметра **ω:**

Для построения алгоритма вычисления очередной итерации нужно разделить в

левой части рекуррентной формулы члены, содержащие и . Перейдем от векторной формы записи рекуррентной формулы к построчной: Очевидно, что искомая погрешность вычислений будет определяться погрешностью задания коэффициентов исходной системы и погрешностью округления.

Запишем разность двух итерационных приближений решения и оценим её минимальное значение



Пусть коэффициенты  и *fi* заданы с некоторой относительной погрешностью . Предположим, что итерационный метод сходится, и невязка убывает с ростом номера итерации *k*, т.е. . Оценка абсолютной погрешности правой части нашей разности может быть представлена в следующем виде

,

здесь .− модуль минимального значения диагонального элемента .Отсюда следует, что задаваемая погрешность метода .

**Описание программы**

На вход в командной строке программы подается число (1 или 2), если sign = 2, то считываем матрицу из файла. Если sign = 1, то матрица коэффициентов и столбец значений задается формулой из приложения 2 (п. 2-4), где же определен и размер матрицы.

Входные данные имеют следующий вид. При sign = 2, сначала вводится значение sign, затем размерность n, и далее сама матрица. В случае, когда sign = 1, вводится число M, а затем размер матрицы n.

Для каждой системы уравнений изучаем скорость сходимости итераций к точному решению. Считаем количество итераций для 0.05 <= ** <= 1.95, с шагом 0.05.

Программа выводит ответ в следующем виде:

“w = 0.05, count = …

w = 0.10, count = …

…

w = 1.95, count = …

Answer:

X1 = …

X2 = …

…

Xn = …“

**Текст программы**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <locale.h>

const double eps = 1e-6;

int

main(void)

{

setlocale(LC\_ALL, "Russian");

FILE \*fr = fopen ("input.txt", "r");

FILE \*fw = fopen ("output.txt", "w");

double \*x, \*\*mas, \*f, \*Diff, w, cur, error, q, M, x\_p, \*\*a, \*\*tran, \*b;

int i, j, n, sign, count = 0, k;

//Считываем sign - 1, если из файла, 2 - если генерируем; и размерность матрицы n

fscanf(fr, "%d %d", &sign, &n);

mas = (double \*\*) calloc(n, sizeof(double\*));

a = (double \*\*) calloc(n, sizeof(double\*));

tran = (double \*\*) calloc(n, sizeof(double\*));

f = (double \*) calloc(n, sizeof(double));

x = (double \*) calloc(n, sizeof(double));

b = (double \*) calloc(n, sizeof(double));

Diff = (double \*) calloc(n, sizeof(double));

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

mas[i] = (double \*) calloc(n, sizeof(double));

tran[i] = (double \*) calloc(n, sizeof(double));

a[i] = (double \*) calloc(n, sizeof(double));

}

//Заполняем нашу матрицу

if (sign == 2){

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

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

fscanf(fr, "%lf", &mas[i][j]);

}

fscanf(fr, "%lf", &f[i]);

}

}

else {

fscanf(fr, "%lf", &M);

q = 1.001 - 2 \* M \* 0.001;

x\_p = M;

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

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

if (i == j) {

mas[i - 1][j - 1] = pow(q - 1, i + j);

} else {

mas[i - 1][j - 1] = pow(q, i + j) + 0.1 \* (j - i);

}

}

f[i - 1] = n \* exp(x\_p / (i + 0.0)) \* cos(x\_p);

}

}

// Делаем транспонированную матрицу tran

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

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

tran[i][j] = mas[j][i];

}

}

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

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

double sum = 0.0;

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

sum += tran[i][k] \* mas[k][j];

}

a[i][j] = sum;

}

}

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

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

mas[i][j] = a[i][j];

}

}

//Домнажаем на транспонированную матрицу tran вектор f

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

double sum = 0.0;

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

sum += tran[i][k] \* f[k];

}

b[i] = sum;

}

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

f[i] = b[i];

}

// Ищем оптимальную w. Реализуем итерационный метод решения СЛАУ

for (w = 0.05; w < 2.0; w += 0.05) {

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

x[i] = 0;

}

, count = 0;

while (1) {

, count++;

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

cur = 0;

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

cur += mas[j][i] \* x[j];

}

x[i] = x[i] + ((f[i] - cur) \* w) / mas[i][i];

}

error = 0.0;

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

Diff[i] = 0.0;

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

Diff[i] += mas[i][j] \* x[j];

}

error += (Diff[i] - f[i]) \* (Diff[i] - f[i]);

}

error = sqrt(error);

if (error < eps)

break;

}

fprintf(fw, "w = %.2lf, count = %d\n", w, count);

}

fprintf(fw, "\n");

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

fprintf(fw, "x%d = %.5lf;\n", i, x[i]);

}

fclose(fr);

fclose(fw);

return 0;

}

# Тесты, подтверждающие правильность работы программы

**Приложение 1. Вариант 12. Слау №1.**

2x1 – 2x2 + x4 = -3

2x1 +3x2 + x3 – 3x4 = -6

3x1 + 4x2 – x3 + 2x4 = 0

x1 + 3x2 + x3 – x4 = 2

Программа вывела:

w = 0,05, count = 3379

w = 0,10, count = 1645

w = 0,15, count = 1067

w = 0,20, count = 778

w = 0,25, count = 605

w = 0,30, count = 489

w = 0,35, count = 406

w = 0,40, count = 344

w = 0,45, count = 296

w = 0,50, count = 257

w = 0,55, count = 226

w = 0,60, count = 199

w = 0,65, count = 177

w = 0,70, count = 158

w = 0,75, count = 141

w = 0,80, count = 127

w = 0,85, count = 114

w = 0,90, count = 102

w = 0,95, count = 92

w = 1,00, count = 83

w = 1,05, count = 74

w = 1,10, count = 67

w = 1,15, count = 60

w = 1,20, count = 53

w = 1,25, count = 46

w = 1,30, count = 40

w = 1,35, count = 32

w = 1,40, count = 24

w = 1,45, count = 27

w = 1,50, count = 29

w = 1,55, count = 32

w = 1,60, count = 38

w = 1,65, count = 45

w = 1,70, count = 53

w = 1,75, count = 64

w = 1,80, count = 83

w = 1,85, count = 110

w = 1,90, count = 173

w = 1,95, count = 348

x0 = -2,00000;

x1 = 1,00000;

x2 = 4,00000;

x3 = 3,00000;

Оптимальный выбор итерационного параметра : ω = 1,40

Совпадает с ответом из [http://www.wolframalpha.com](http://www.wolframalpha.com/).

**Приложение 1. Вариант 12. Слау №2.**

1x1 + 3x2 + 2x3 + x4 = 0

2x1 - x2 + 3x3 – 2x4 = 0

3x1 - 5x2 + 4x3 - 3x4 = 0

x1 + 17x2 + 4x3 – 23x4 = 0

Программа вывела:

w = 0,05, count = 1

w = 0,10, count = 1

w = 0,15, count = 1

w = 0,20, count = 1

w = 0,25, count = 1

w = 0,30, count = 1

w = 0,35, count = 1

w = 0,40, count = 1

w = 0,45, count = 1

w = 0,50, count = 1

w = 0,55, count = 1

w = 0,60, count = 1

w = 0,65, count = 1

w = 0,70, count = 1

w = 0,75, count = 1

w = 0,80, count = 1

w = 0,85, count = 1

w = 0,90, count = 1

w = 0,95, count = 1

w = 1,00, count = 1

w = 1,05, count = 1

w = 1,10, count = 1

w = 1,15, count = 1

w = 1,20, count = 1

w = 1,25, count = 1

w = 1,30, count = 1

w = 1,35, count = 1

w = 1,40, count = 1

w = 1,45, count = 1

w = 1,50, count = 1

w = 1,55, count = 1

w = 1,60, count = 1

w = 1,65, count = 1

w = 1,70, count = 1

w = 1,75, count = 1

w = 1,80, count = 1

w = 1,85, count = 1

w = 1,90, count = 1

w = 1,95, count = 1

x0 = 0,00000;

x1 = 0,00000;

x2 = 0,00000;

x3 = 0,00000;

Система имеет только тривиальное решение. Поэтому любое ω оптимально.

Совпадает с ответом из [http://www.wolframalpha.com](http://www.wolframalpha.com/).

**Приложение 1. Вариант 2. Слау №3.**

45x1 - 28x2 + 34x3 - 52x4 = 9

36x1 - 23x2 + 29x3 – 43x4 = 3

47x1 - 32x2 + 36x3 - 48x4 = -17

27x1  - 19x2 + 22x3 – 35x4 = 6

Программа вывела:

w = 0,05, count = 1327012

w = 0,10, count = 647419

w = 0,15, count = 420863

w = 0,20, count = 307565

w = 0,25, count = 239568

w = 0,30, count = 194222

w = 0,35, count = 161816

w = 0,40, count = 137498

w = 0,45, count = 118569

w = 0,50, count = 103410

w = 0,55, count = 90992

w = 0,60, count = 80626

w = 0,65, count = 71837

w = 0,70, count = 64282

w = 0,75, count = 57712

w = 0,80, count = 51936

w = 0,85, count = 46810

w = 0,90, count = 42219

w = 0,95, count = 38070

w = 1,00, count = 34286

w = 1,05, count = 30803

w = 1,10, count = 27561

w = 1,15, count = 24502

w = 1,20, count = 21557

w = 1,25, count = 18625

w = 1,30, count = 15446

w = 1,35, count = 12386

w = 1,40, count = 12160

w = 1,45, count = 12475

w = 1,50, count = 13101

w = 1,55, count = 15052

w = 1,60, count = 16337

w = 1,65, count = 18862

w = 1,70, count = 21612

w = 1,75, count = 25924

w = 1,80, count = 33350

w = 1,85, count = 44102

w = 1,90, count = 67413

w = 1,95, count = 136513

x0 = 1,00000;

x1 = 2,00000;

x2 = -4,00000;

x3 = -3,00000;

Оптимальный выбор итерационного параметра : ω = 1,40

Совпадает с ответом из [http://www.wolframalpha.com](http://www.wolframalpha.com/).

**Приложение 2. Вариант 4.**

**Задание матрицы формулой**

Программа вывела:

w = 0,05, count = 59225

w = 0,10, count = 28950

w = 0,15, count = 18850

w = 0,20, count = 13775

w = 0,25, count = 10725

w = 0,30, count = 8675

w = 0,35, count = 7225

w = 0,40, count = 6100

w = 0,45, count = 5225

w = 0,50, count = 4500

w = 0,55, count = 3900

w = 0,60, count = 3375

w = 0,65, count = 2900

w = 0,70, count = 2400

w = 0,75, count = 2250

w = 0,80, count = 2000

w = 0,85, count = 1875

w = 0,90, count = 1725

w = 0,95, count = 1650

w = 1,00, count = 1600

w = 1,05, count = 1575

w = 1,10, count = 1575

w = 1,15, count = 1600

w = 1,20, count = 1650

w = 1,25, count = 1725

w = 1,30, count = 1875

w = 1,35, count = 2025

w = 1,40, count = 2150

w = 1,45, count = 2450

w = 1,50, count = 2650

w = 1,55, count = 3075

w = 1,60, count = 3525

w = 1,65, count = 4100

w = 1,70, count = 4900

w = 1,75, count = 5925

w = 1,80, count = 7675

w = 1,85, count = 10375

w = 1,90, count = 15875

w = 1,95, count = 32525

x0 = 3425,29895 x1 = 301,28093 x2 = 61,86547 x3 = -10,13609 x4 = -43,32282

x5 = -62,15298 x6 = -74,22374 x7 = -82,60167 x8 = -88,74666 x9 = -93,43757

x10 = -97,1260 x11 = -100,09124 x12 = -102,51446 x13 = -104,5183 x14 = -106,188

x15 = -107,58718 x16 = -108,7588 x17 = -109,7377 x18 = -110,549 x19 = -111,21443

x20 = -111,74758 x21 = -112,16132 x22 = -112,46528 x23 = -112,66715 x24 = -112,77303

x25 = -112,78778 x26 = -112,71527 x27 = -112,55852 x28 = -112,31990 x29 = -112,00121

x30 = -111,60378 x31 = -111,12853 x32 = -110,57606 x33 = -109,94665 x34 = -109,24035

x35 = -108,45698 x36 = -107,59615 x37 = -106,65730 x38 = -105,63974 x39 = -104,54260

x40 = -103,36490 x41 = -102,10555 x42 = -100,76333 x43 = -99,33693 x44 = -97,82495

x45 = -96,22590 x46 = -94,53819 x47 = -92,76018 x48 = -90,89011 x49 = -88,92619

x50 = -86,86652 x51 = -84,70915 x52 = -82,45207 x53 = -80,09318 x54 = -77,63033

x55 = -75,06130 x56 = -72,38379 x57 = -69,59547 x58 = -66,69392 x59 = -63,67665

x60 = -60,54114 x61 = -57,28477 x62 = -53,90487 x63 = -50,39871 x64 = -46,76350

x65 = -42,99636 x66 = -39,09436 x67 = -35,05452 x68 = -30,87377 x69 = -26,54898

x70 = -22,07694 x71 = -17,45439 x72 = -12,67799 x73 = -7,74433 x74 = -2,64993

x75 = 2,60878 x76 = 8,03541 x77 = 13,63368 x78 = 19,40736 x79 = 25,36032

x80 = 31,49649 x81 = 37,81989 x82 = 44,33462 x83 = 51,04486 x84 = 57,95488

x85 = 65,06904 x86 = 72,39175 x87 = 79,92757 x88 = 87,68109 x89 = 95,65702

x90 = 103,86018 x91 = 112,29544 x92 = 120,96780 x93 = 129,88235 x94 = 139,04427

x95 = 148,45885 x96 = 158,13148 x97 = 168,06765 x98 = 178,27297 x99 = 188,75315

Оптимальный выбор итерационного параметра : ω = 1,10

Совпадает с ответом из [http://www.wolframalpha.com](http://www.wolframalpha.com/).

Так же программа тестировалась на собственных тестах, ниже приведены СЛАУ.

**Собственные тесты.**

1.Исходная матрица:

a00 = 0.78000 a01 = 0.56300 f0 = 0.21700

a10 = 0.91300 a11 = 0.65900 f1 = 0.25400

Программа вывела:

w = 0.05, count = 128

w = 0.10, count = 63

w = 0.15, count = 41

w = 0.20, count = 30

w = 0.25, count = 23

w = 0.30, count = 19

w = 0.35, count = 16

w = 0.40, count = 13

w = 0.45, count = 11

w = 0.50, count = 10

w = 0.55, count = 9

w = 0.60, count = 8

w = 0.65, count = 7

w = 0.70, count = 6

w = 0.75, count = 5

w = 0.80, count = 5

w = 0.85, count = 4

w = 0.90, count = 3

w = 0.95, count = 3

w = 1.00, count = 1

w = 1.05, count = 3

w = 1.10, count = 3

w = 1.15, count = 4

w = 1.20, count = 5

w = 1.25, count = 5

w = 1.30, count = 6

w = 1.35, count = 7

w = 1.40, count = 8

w = 1.45, count = 9

w = 1.50, count = 10

w = 1.55, count = 11

w = 1.60, count = 13

w = 1.65, count = 16

w = 1.70, count = 19

w = 1.75, count = 23

w = 1.80, count = 30

w = 1.85, count = 41

w = 1.90, count = 63

w = 1.95, count = 128

Answer:

X1: 5.56408

X2: -7.32322

Оптимальный выбор итерационного параметра : ω = 1,00

Совпадает с ответом из [http://www.wolframalpha.com](http://www.wolframalpha.com/).

2. Исходная матрица:

x1 + 2x2 + 6x3 + 6x4 + 4x5 = 1

2x1 + 3x2 + 8x3 + 3x5 = 3

3x1 + 4x2 + 4x3 + 5x4 + 2x5 = 8

4x1 + 5x2 + 3x3 + 4x4 = 3

5x1 + 7x2 + 2x3 + 2x4 + 2x5 = 4

Программа вывела:

w = 0,05, count = 2095458

w = 0,10, count = 1020344

w = 0,15, count = 662007

w = 0,20, count = 482868

w = 0,25, count = 375411

w = 0,30, count = 303798

w = 0,35, count = 252669

w = 0,40, count = 214345

w = 0,45, count = 184560

w = 0,50, count = 160755

w = 0,55, count = 141301

w = 0,60, count = 125112

w = 0,65, count = 111435

w = 0,70, count = 99734

w = 0,75, count = 89613

w = 0,80, count = 80778

w = 0,85, count = 73001

w = 0,90, count = 66104

w = 0,95, count = 59947

w = 1,00, count = 54417

w = 1,05, count = 49422

w = 1,10, count = 44886

w = 1,15, count = 40745

w = 1,20, count = 36947

w = 1,25, count = 33447

w = 1,30, count = 30207

w = 1,35, count = 27196

w = 1,40, count = 24386

w = 1,45, count = 21753

w = 1,50, count = 19277

w = 1,55, count = 16940

w = 1,60, count = 14726

w = 1,65, count = 12621

w = 1,70, count = 10612

w = 1,75, count = 8686

w = 1,80, count = 6827

w = 1,85, count = 5014

w = 1,90, count = 3201

w = 1,95, count = 909

x0 = 40,29411;

x1 = -30,04201;

x2 = -1,74790;

x3 = -0,68067;

x4 = 8,84033;

Оптимальный выбор итерационного параметра : ω = 1,95

Совпадает с ответом из [http://www.wolframalpha.com](http://www.wolframalpha.com/).

3. Исходная матрица:

3x1 + x2 + 2x3 + 3x4 + 3x5 = 4

2x1 + 2x2 + 2x3 + 4x4 + 4x5 = 0

6x1 + 4x2 + 3x3 + 6x4 + 5x5 = 6

5x1 + 4x3 + 8x4  + 5x5= 6

4x1 + 4x2 + 5x4 + 4x5 = 4

Программа вывела:

w = 0,05, count = 85845

w = 0,10, count = 41801

w = 0,15, count = 27117

w = 0,20, count = 19774

w = 0,25, count = 15366

w = 0,30, count = 12426

w = 0,35, count = 10324

w = 0,40, count = 8746

w = 0,45, count = 7517

w = 0,50, count = 6531

w = 0,55, count = 5723

w = 0,60, count = 5047

w = 0,65, count = 4473

w = 0,70, count = 3978

w = 0,75, count = 3546

w = 0,80, count = 3164

w = 0,85, count = 2823

w = 0,90, count = 2515

w = 0,95, count = 2234

w = 1,00, count = 1972

w = 1,05, count = 1725

w = 1,10, count = 1484

w = 1,15, count = 1230

w = 1,20, count = 871

w = 1,25, count = 814

w = 1,30, count = 846

w = 1,35, count = 924

w = 1,40, count = 1011

w = 1,45, count = 1109

w = 1,50, count = 1220

w = 1,55, count = 1343

w = 1,60, count = 1587

w = 1,65, count = 1851

w = 1,70, count = 2142

w = 1,75, count = 2695

w = 1,80, count = 3424

w = 1,85, count = 4609

w = 1,90, count = 6992

w = 1,95, count = 14310

x0 = 2,58621;

x1 = -1,24138;

x2 = -1,00000;

x3 = -0,68965;

x4 = 0,51724;

Оптимальный выбор итерационного параметра : ω = 1,25

Совпадает с ответом из [http://www.wolframalpha.com](http://www.wolframalpha.com/).

Вывод

В ходе работы:

1) Мы изучили метод верхней релаксации

2) Проанализировали условия сходимости.

3) Экспериментально подобрали ω

К сожалению данный метод не работает для большинства матриц из примера, так как они не являются положительно определенными. Итерационный метод дает большую точность, в отличии от метода Гаусса. Я заметил, что при выборе w близким к 0 или к 2 возрастало количество итераций, следовательно, увеличивалась скорость сходимости. Метод верхней релаксации прост и удобен для вычислений. Он не требует никаких действий с матрицей А, не требует дополнительного резерва памяти, что существенно при решении больших систем