**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ЛАБОРАТОРНАЯ РАБОТА 2**

Тема: «Итерационные методы решения СЛАУ»

Выполнил студент 1 гр.

Рудь Андрей Владимирович

**Минск 2019**

**Задание**

**Постановка задачи:**

Разработать программу численного решения СЛАУ *Ax = f* методом релаксации, обеспечив сходимость итерационного процесса. В качестве критерия остановки итерационного процесса использовать

|| *x(k+1)* - *x(k)* ||∞ < *ε*, где *ε* = 10-5.

Для проведения вычислительного эксперимента необходимо решить систему размерности *n* =10 . Матрицу A и вектор точного решения x заполнить случайными числами (сгенерировать) с двумя знаками после запятой из диапазона от -10 до 10. Правую часть задать умножением матрицы *A* на вектор *x* : *f = Ax*.

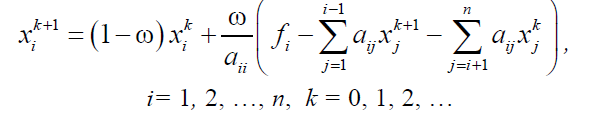
В результатах выполнения вычислительного эксперимента необходимо привести следующую информацию:

Матрицу *A* (построчно), вектор *f*, точное решение *x*, *ε*.

Исследовать сходимость метода релаксации в зависимости от параметра релаксации. Результаты оформить в виде таблицы.

**Краткие теоретические сведения:**

Метод последовательной верхней релаксации является одним из наиболее широко используемых на практике методов для решения СЛАУ. Рассмотрим взвешенную сумму текущего приближения и приближения, построенного по методу Гаусса–Зейделя:



Это алгоритм, который в литературе называют методом релаксации или методом верхней релаксации. Часто его называют методом последовательной верхней релаксации, SOR-методом (SOR – successive over relaxation).

При ω=1 метод релаксации есть метод Гаусса–Зейделя. Иногда при ω<1 говорят о нижней релаксации, а при ω>1 говорят о верхней релаксации. Чаще параметр релаксации ω подбирают экспериментально.

Теорема: Пусть А – симметричная положительно определенная матрица. Тогда метод верхней релаксации сходится при условии 0<ω<2. В частности, метод Гаусса-Зейделя (ω=1) сходится.

Замечание: Как уже отмечалось при рассмотрении метода Якоби, окончание итераций определяется либо максимальным заданным числом итераций , либо условием



где ε >0 – заданное число

Нормы векторов:

Кубическая норма (используется также название максимум-норма):

|| x ||I = max |xi|, 1 ≤ i ≤ n

Для этой нормы используется также обозначение || *x* ||∞

**Листинг программы:**

Класс матрицы с необходимыми конструкторами и методами:

template <typename T>

class Matrix

{

public:

Matrix(int nx, int ny, bool) :size\_x(nx), size\_y(ny)

{

random\_device generator;

//mt19937\_64 generator;

//default\_random\_engine generator;

uniform\_int\_distribution<> distribution(-1000, 1000);

vector < vector<T>> matrix1(size\_x);

for (auto it = matrix1.begin(); it != matrix1.end(); ++it)

{

for (int i = 0; i < size\_y; ++i)

{

int temp = distribution(generator);

T t = temp;

t /= 100;

(\*it).push\_back(t);

}

}

matrix = matrix1;

}

Matrix(int nx, int ny) :size\_x(nx), size\_y(ny)

{

vector < vector<T>> matrix1(size\_x);

for (auto it = matrix1.begin(); it != matrix1.end(); ++it)

{

for (int i = 0; i < size\_y; ++i)

{

(\*it).push\_back(0);

}

}

matrix = matrix1;

}

void matrix\_round(int p)

{

for (int i = 0; i < size\_x; ++i)

for (int j = 0; j < size\_y; ++j)

matrix[i][j] = round(matrix[i][j] \* pow(10,p)) / pow(10, p);

}

Matrix<T> operator\* (Matrix<T> const m)

{

int sizex1 = (\*this).size\_x;

int sizey1 = (\*this).size\_y;

int sizex2 = m.size\_x;

int sizey2 = m.size\_y;

if (sizey1 != sizex2)

return(\*this);

Matrix<T> temp(sizex1, sizey2);

temp.nullify();

for (int i = 0; i < sizex1; i++)

for (int k = 0; k < sizey2; k++)

for (int j = 0; j < sizex2; j++)

temp.matrix[i][k] += ((\*this).matrix[i][j]) \* (m.matrix[j][k]);

return temp;

}

Matrix<T> operator\* (T c)

{

Matrix<T> temp = \*this;

for (int i = 0; i < temp.size\_x; ++i)

for (int j = 0; j < temp.size\_y; ++j)

temp[i][j] \*= c;

return temp;

}

Matrix<T> operator- (Matrix<T> const m)

{

Matrix<T> temp(m.size\_x, m.size\_y);

temp.nullify();

for (int i = 0; i < m.size\_x; ++i)

for (int j = 0; j < m.size\_y; ++j)

temp.matrix[i][j] = matrix[i][j] - m.matrix[i][j];

return temp;

}

void nullify()

{

for (int i = 0; i < size\_x; ++i)

for (int j = 0; j < size\_y; ++j)

matrix[i][j] = 0;

}

friend ostream& operator<< (ostream& os, Matrix<T> const m)

{

for (int i = 0; i < m.size\_x; ++i)

{

for (int j = 0; j < m.size\_y; ++j)

{

os.width(12);

os << m.matrix[i][j] << ' ';

}

os << endl;

}

return os;

}

Matrix transpose()

{

Matrix<T> At(size\_y, size\_x);

for (int i = 0; i < size\_x; ++i)

for (int j = 0; j < size\_y; ++j)

At.matrix[j][i] = matrix[i][j];

return At;

}

vector<T>& operator[] (int i)

{

return matrix[i];

}

int size\_x;

int size\_y;

vector<vector<T>> matrix;

};

Метод вычисления максимум-нормы ветора:

template<typename T>

T maximum\_norm(Matrix<T> v)

{

if (v.size\_y != 1)

{

throw Bad("not a vector!");

return 0.;

}

else

{

T max = v[0][0];

for (int i = 1; i < v.size\_x; ++i)

{

if (abs(v[i][0]) > abs(max))

max = v[i][0];

}

return abs(max);

}

}

Successive over-relaxation - метод релаксации:

template <typename T>

Matrix<T> SOR(Matrix<T> A, Matrix<T> f, T eps, T w, int& counter)

{

if (f.size\_y != 1)

throw Bad();

Matrix<double> x0(f.size\_x, 1, true);

for (int i = 0; i < 10; ++i)

x0[i][0] = f[i][0] / A[i][i];

Matrix<double> xk = x0;

Matrix<double> xk1 = xk;

do {

xk = xk1;

for (int i = 0; i < xk.size\_x; ++i)

{

double sum1 = 0, sum2 = 0;

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

sum1 += A[i][j] \* xk1[j][0];

for (int j = i + 1; j < xk.size\_x; ++j)

sum2 += A[i][j] \* xk[j][0];

xk1[i][0] = (1 - w) \* xk[i][0] + w / A[i][i] \* (f[i][0] - sum1 - sum2);

}

counter++;

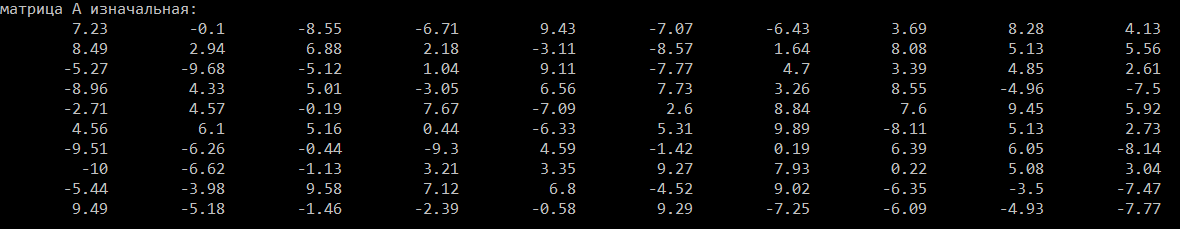
} while (maximum\_norm(xk1 - xk) > eps);

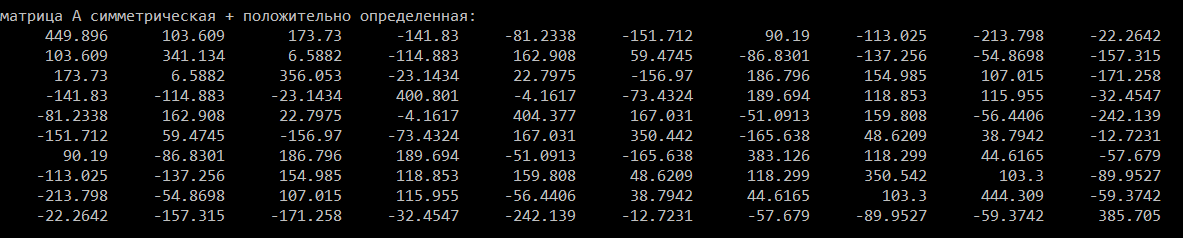
return xk1;

}

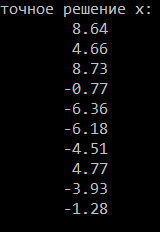
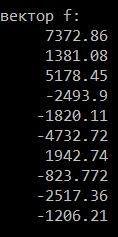
**Результаты:**

Сгенерированная матрица А:





Сгенерированный вектор x: Полученная правая часть f = Ax:

|  |  |  |
| --- | --- | --- |
| Параметр ω | Количество итераций, за которое была достигнута требуемая точность | || *x(k+1)* - *x(k)* ||∞ |
| 0.2 | 564 | 9.92642e-06 |
| 0.5 | 361 | 9.84947e-06 |
| 0.8 | 230 | 9.68884e-06 |
| 1 | 169 | 9.99357e-06 |
| 1.3 | 101 | 9.94974e-06 |
| 1.5 | 62 | 9.43017e-06 |
| 1.8 | 78 | 8.71625e-06 |

Максимум-норма погрешности || *x* - ||∞: 4.76493e-05, ω=1.5

**Вывод:** Достаточное условие сходимости метода релаксации - симметричная положительно определенная матрица, достигается за счет умножения матрицы *A* на . Метод релаксации получил широкое распространение в связи с тем, что он содержит свободный параметр w , изменяя который можно получать различную скорость сходимости итерационного процесса. Но данный параметр зачастую приходится определять экспериментально для каждой отдельной СЛАУ.