Министерство образования и науки Российской Федерации
Новосибирский государственный технический университет
Кафедра прикладной математики

Численные методы

Практическая работа №2

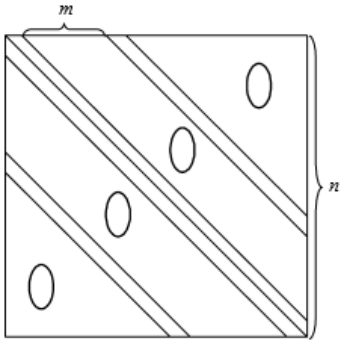
Факультет: прикладной математики и информатики
Группа: ПМ-63
Студент: Кожекин М.В.
Преподаватели: Задорожный А. Г.
Персова М.Г.

Новосибирск
2018

1. Цель работы

Разработать программы решения СЛАУ методами Якоби, Гаусса-Зейделя, блочной релаксации с хранением матрицы в диагональном формате. Исследовать сходимость методов для различных тестовых матриц и её зависимость от параметра релаксации. Изучить возможность оценки порядка числа обусловленности матрицы путем вычислительного эксперимента.

Вариант 10: 7-ми диагональная матрица с параметрами m – количество нулевых диагоналей, n – размерность матрицы. Размер блока в реализации блочной релаксации переменный. Исследовать зависимость скорости сходимости от размеров блока.



2. Анализ

Пусть дана система линейных алгебраических уравнений: Ax = F. Выбирается начальное приближение $x^{(0)} = (x_1^0, x_2^0, \dots, x_n^0)$

Метод Якоби с параметром релаксации

Каждое следующее приближение в методе Якоби с параметром релаксации рассчитывается по формуле:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}}\left[f_i - \sum_{j=1}^{n} a_{ij}x_j^{(k)}\right], 0 < \omega \leq 1$$

Метод Гаусса-Зейделя с параметром релаксации

Каждое последующее приближение в методе Гаусса-Зейделя с параметром релаксации рассчитывается по формуле:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}}\left[f_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\right], 0 < \omega < 2$$

Условия выхода из итерационного процесса для рассмотренных методов:

1. Выход по относительной невязке:

$$\frac{||F - Ax^{(k)}||}{||F||} < \varepsilon$$

2. Защита от зацикливания: k > maxiter, maxiter - максимальное количество итераций.

3. Текст программы

Для удобства программа была разбита на следующие модули:

head.h – заголовочный файл, в котором определяется точность вычислений

matrix.h и matrix.cpp – Класс 7-ми диагональной матрицы

vect.h и vect.cpp – класс векторов $x^{(k)}$ и F

slae.h и slae.cpp – класс СЛАУ

main.cpp – файл с исследованиями

head.h

```
#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <fstream>
#include <iostream>
#include <vector>
#include <iomanip>

using std::vector;
using std::string;
using std::cout;
using std::endl;

// float || double
typedef double real;
```

matrix.h

```
#include "head.h"


class matrix {

public:
	int readMatrixFromFile(char* fileName);
	void writeMatrixToFile(char* fileName);
	int getDimention() { return n; }
	void setE(real new_E) { E = new_E; }
	void setMaxiter(real new_maxiter) { maxiter = new_maxiter; }
	void generateMatrixWith7Diagonals(int new_n, int new_m);
	void invertSigns();

protected:
	real calcAii(int i);
	vector <real> di, au1, au2, au3, al1, al2, al3;
	int n, m, maxiter;
	real E;
};
```

matrix.cpp

```
#include "matrix.h"


// Ввод 7-диагональной матрицы из файла
int matrix::readMatrixFromFile(char * fileName) {

	std::ifstream fin;
	fin.open(fileName);

	fin >> n >> m;
	fin >> E >> maxiter;

	di.resize(n);
```

```
        for (int i = 0; i < n; ++i)
                fin >> di[i];

        al1.resize(n - 1);
        for (int i = 0; i < al1.size(); ++i)
                fin >> al1[i];

        al2.resize(n - m - 2);
        for (int i = 0; i < al2.size(); ++i)
                fin >> al2[i];

        al3.resize(n - m - 3);
        for (int i = 0; i < al3.size(); ++i)
                fin >> al3[i];


        au1.resize(n - 1);
        for (int i = 0; i < au1.size(); ++i)
                fin >> au1[i];

        au2.resize(n - m - 2);
        for (int i = 0; i < au2.size(); ++i)
                fin >> au2[i];

        au3.resize(n - m - 3);
        for (int i = 0; i < au3.size(); ++i)
                fin >> au3[i];




        fin.close();
        return 0;
}


// Вывод 7-ми диагональной матрицы в файл
void matrix::writeMatrixToFile(char * fileName) {

        std::ofstream fout;
        fout.open(fileName);

        fout << n << " " << m << endl;
        fout << E << " " << maxiter << endl;

        for (int i = 0; i < n; ++i)
                fout << di[i] << " ";
        fout << endl;


        for (int i = 0; i < al1.size(); ++i)
                fout << al1[i] << " ";
        fout << endl;

        for (int i = 0; i < al2.size(); ++i)
                fout << al2[i] << " ";
        fout << endl;

        for (int i = 0; i < al3.size(); ++i)
                fout << al3[i] << " ";
        fout << endl;



        for (int i = 0; i < au1.size(); ++i)
                fout << au1[i] << " ";
        fout << endl;

        for (int i = 0; i < au2.size(); ++i)
                fout << au2[i] << " ";
        fout << endl;
```

```
        for (int i = 0; i < au3.size(); ++i)
                fout << au3[i] << " ";
        fout << endl;


        fout.close();
}


// Создаём матрицу A(k)
void matrix::generateMatrixWith7Diagonals(int new_n, int new_m) {

        n = new_n;
        m = new_m;
        di.clear();
        di.resize(n, 0);
        au1.resize(n - 1);
        al1.resize(n - 1);

        au2.resize(n - m - 2);
        al2.resize(n - m - 2);

        au3.resize(n - m - 3);
        al3.resize(n - m - 3);

        for (int i = 0; i < al1.size(); ++i) {
                au1[i] = -rand() % 5;
                al1[i] = -rand() % 5;
        }

        for (int i = 0; i < al2.size(); ++i) {
                au2[i] = -rand() % 5;
                al2[i] = -rand() % 5;
        }

        for (int i = 0; i < al3.size(); ++i) {
                au3[i] = -rand() % 5;
                al3[i] = -rand() % 5;
        }


        for (int i = 0; i < di.size(); ++i)
                di[i] = -calcAii(i);

        di[0]++;
}


// Рассчёт суммы элементов строки при генерации матрицы A(k)
real matrix::calcAii(int i) {

        real sum = 0;

        if (i >= 1) { // Нижний треугольник
                sum += al1[i - 1];
                if (i >= m + 2) {
                        sum += al2[i - m - 2];
                        if (i >= m + 3)
                                sum += al3[i - m - 3];
                }
        }

        sum += di[i];

        if (i < n - 1) { // Верхний треугольник
                sum += au1[i];
                if (i < n - m - 2) {
                        sum += au2[i];
                        if (i < n - m - 3)
                                sum += au3[i];
                }
        }
```

```
	return sum;
}


// Меняем знак внедиагональных элементов на противоположный
void matrix::invertSigns() {

	for (int i = 0; i < al1.size(); ++i) {
		au1[i] = abs(au1[i]);
		al1[i] = abs(al1[i]);
	}

	for (int i = 0; i < al2.size(); ++i) {
		au2[i] = abs(au2[i]);
		al2[i] = abs(al2[i]);
	}

	for (int i = 0; i < al3.size(); ++i) {
		au3[i] = abs(au3[i]);
		al3[i] = abs(al3[i]);
	}
}
```

vect.h

```
#include "head.h"


// Векторы xk and F
class vect {

public:

	void getVectX(vector <real> &x) { x = F; };

	void generateVectX(int size);
	// Создаём начальное приближение (нулевой вектор)
	void generateInitualGuess(int size) { xk.clear(); xk.resize(size, real(0)); }
	void writeTableToFile(std::ofstream& fout, int presision, real w, int iterations, int condNumber);
	void writeFToFile(char *fileName);
	bool isXcorrect();

protected:
	vector <real> xk, F;
};
```

vect.cpp

```
#include "head.h"
#include "vect.h"



// Создаём вектор xk* = (1,2,...n)'
void vect::generateVectX(int size) {

	xk.resize(size);
	for (int i = 0; i < size; ++i) {
		xk[i] = i + 1;
	}
}


// Вывод вектора F в файл
void vect::writeFToFile(char *fileName) {

	std::ofstream fout;
	fout.open(fileName);

	for (int i = 0; i < F.size();++i)
```

```
                fout << int(F[i]) << endl;
        fout.close();
}


// generates 1/3 of table in research
void vect::writeTableToFile(std::ofstream& fout, int presision, real w, int iterations, int condNumber) {

        fout << std::fixed << std::setprecision(presision) << w << "\t";
        fout << std::fixed << std::setprecision(std::numeric_limits<real>::digits10 + 1);
        for (int i = 0; i < xk.size();++i)
                fout << xk[i] << " ";
        fout << " \t";

        fout << std::scientific;
        for (int i = 0; i < xk.size();++i)
                fout << xk[i] - real(i + 1) << " ";
        fout << " \t";

        fout << iterations << "\t";
        fout << condNumber << endl;
}




// Проверяем насколько xk  близко к вектору xk* = (1,2,...,n)'
bool vect::isXcorrect() {

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

                if (abs(xk[i] - (real)(i + 1)) > std::numeric_limits<real>::digits10 + 2)
                        return false;
        }

        return true;
}
```

slae.h

```
#include "head.h"
#include"matrix.h"
#include "vect.h"


class SLAE : public matrix, public vect {

public:
        void convMatrixToDense();
        void writeDenseMatrixToFile(char* fileName);

        real multLine(vector<real>& line, int i, int mode);
        void mult();

        void Jacobi(real w);
        void GaussSeildel(real w);
        int calcIterative(int mode, real w);
        real findOptimalW(int mode, std::ofstream& fout);


protected:
        real calcNormE(vector <real> &x);
        real calcRelativeDiscrepancy();
        int calcCondNumber();
        vector <vector <real>> A;

};
```

slae.cpp

```cpp
#include "slae.h"

// Преобразование 7-ми диагональной матрицы в плотный формат
void SLAE::convMatrixToDense() {

	A.clear();
	A.resize(n);

	for (int i = 0; i < n; ++i) {
		A[i].resize(n, 0);
		A[i][i] = di[i];
	}

	int j = 1;
	for (int i = 0; i < al1.size(); ++i, ++j) {

		A[i][j] = au1[i];
		A[j][i] = al1[i];
	}

	j = m + 2;
	for (int i = 0; i < al2.size(); ++i, ++j) {

		A[i][j] = au2[i];
		A[j][i] = al2[i];
	}

	j = m + 3;
	for (int i = 0; i < al3.size(); ++i, ++j) {

		A[i][j] = au3[i];
		A[j][i] = al3[i];
	}
}


// Вывод плотной матрицы в файл
void SLAE::writeDenseMatrixToFile(char * fileName) {

	std::ofstream fout;
	fout.open(fileName);


	for (int i = 0; i < n;++i) {
		for (int j = 0; j < n; ++j)
			fout << A[i][j] << "\t";
		fout << endl;
	}

	fout.close();
}


// Умножение i-й строки матрицы на вектор
real SLAE::multLine(vector <real> &line, int i, int mode) {

	real sum = 0;
	if (mode == 1 || mode == 3) {   // Нижний треугольник

		if (i > 0) {

			sum += al1[i - 1] * line[i - 1];
			if (i > m + 1) {
				sum += al2[i - m - 2] * line[i - m - 2];
				if (i > m + 2)
					sum += al3[i - m - 3] * line[i - m - 3];
			}
		}
	}
```

```
        if (mode == 2 || mode == 3) {   // Главная диагональ
                                                                        // и верхний треугольник
                sum += di[i] * line[i];
                if (i < n - 1) {

                        sum += au1[i] * line[i + 1];

                        if (i < n - m - 2) {
                                sum += au2[i] * line[i + m + 2];
                                if (i < n - m - 3)
                                        sum += au3[i] * line[i + m + 3];
                        }
                }
        }

        return sum;
}


// Умножение матрицы на вектор
void SLAE::mult() {

        int index;
        F.clear();
        F.resize(n, 0);
        // Нижний треугольник
        index = 1;
        for (int i = 0; i < al1.size(); ++i, ++index)
                F[index] += al1[i] * xk[i];
        index = m + 2;
        for (int i = 0; i < al2.size(); ++i, ++index)
                F[index] += al2[i] * xk[i];
        index = m + 3;
        for (int i = 0; i < al3.size(); ++i, ++index)
                F[index] += al3[i] * xk[i];


        // Главная диагональ
        for (int i = 0; i < di.size(); ++i)
                F[i] += di[i] * xk[i];


        // Верхний треугольник
        index = 1;
        for (int i = 0; i < au1.size(); ++i, ++index)
                F[i] += au1[i] * xk[index];
        index = m + 2;
        for (int i = 0; i < au2.size(); ++i, ++index)
                F[i] += au2[i] * xk[index];
        index = m + 3;
        for (int i = 0; i < au3.size(); ++i, ++index)
                F[i] += au3[i] * xk[index];
}


// Метод Якоби. 0 < w < 1
// Используется общая память для xk и xk1
void SLAE::Jacobi(real w) {

        real sum;
        vector <real> xk1;
        xk1.resize(n);

        for (int i = 0; i < n; ++i) {
                sum = multLine(xk, i, 3);
                //xk[i] += w * (F[i] - sum) / di[i];
                xk1[i] = xk[i] + w * (F[i] - sum) / di[i];
        }
        xk = xk1;
}
```

```
// Метод Гаусса-Зейделя. 0 < w < 2
void SLAE::GaussSeildel(real w) {

	real sum;
	vector <real> xk1 = xk;

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

		sum = multLine(xk1, i, 1);
		sum += multLine(xk, i, 2);

		xk1[i] = xk[i] + w * (F[i] - sum) / di[i];
	}
	xk = xk1;
}


// Решение СЛАУ итерационным методом
// 1  Метод Якоби
// 2  Метод Гаусса-Зейделя
int SLAE::calcIterative(int mode, real w) {
	int i = 0;
	while (i < maxiter && calcRelativeDiscrepancy() >= E) {

		if (mode == 1)
			Jacobi(w);
		else
			GaussSeildel(w);
		++i;
	}
	return i;
}



// Поиск оптимального веса
real SLAE::findOptimalW(int mode, std::ofstream& fout) {

	real optimalW, tmpW;
	int max_i, min_i = maxiter, tmp_i;
	if (mode == 1) max_i = 101;
	else max_i = 200;

	for (int i = 0; i < max_i; ++i) {

		generateInitualGuess(getDimention());
		tmpW = real(i) / 100;
		tmp_i = calcIterative(mode, tmpW);
		if (tmp_i < min_i) {
			min_i = tmp_i;
			optimalW = tmpW;
		}
		if (i % 10 == 0) // Выводим таблицу с точность 0.1
			writeTableToFile(fout, 1, tmpW, tmp_i, calcCondNumber());
	}
	generateInitualGuess(getDimention());
	min_i = calcIterative(mode, optimalW);
	writeTableToFile(fout, 2, optimalW, min_i, calcCondNumber());

	return optimalW;
}


// Вычисление нормы в Евклидовом пространстве
real SLAE::calcNormE(vector <real> &x) {

	real normE = 0;
	for (int i = 0; i < n; i++)
		normE += x[i] * x[i];

	return sqrt(normE);
```

```
}


// Рассчёт относительной невязки
real SLAE::calcRelativeDiscrepancy() {

	vector <real> numerator, denominator = F;
	numerator.resize(n);

	mult(); // F = A*xk

	for (int i = 0; i < n; ++i)
		numerator[i] = denominator[i] - F[i]; // F - A*xk

	// || F - A*xk || / || F ||
	real res = calcNormE(numerator) / calcNormE(denominator);
	F = denominator;
	return res;
}


// Рассчёт числа обусловленности
int SLAE::calcCondNumber() {

	vector <real> numerator, denominator;
	numerator.resize(n);
	denominator.resize(n);
	for (int i = 0; i < n; ++i) {
		numerator[i] = xk[i] - (i + 1); // x - x*
		denominator[i] = i + 1;         // x
	}

	return calcNormE(numerator) / (calcNormE(denominator) * calcRelativeDiscrepancy());
}
```

main.cpp

```
#include "head.h"
#include "slae.h"


void main() {

	std::ofstream fout_res;
	SLAE slae;

	/*slae.generateMatrixWith7Diagonals(10, 3);
	slae.setE(1e-10);
	slae.setMaxiter(200000);
	slae.writeMatrixToFile("A.txt");
	slae.convMatrixToDense();
	slae.writeDenseMatrixToFile("A_dense.txt");

	slae.invertSigns();
	slae.writeMatrixToFile("B.txt");
	slae.convMatrixToDense();
	slae.writeDenseMatrixToFile("B_dense.txt");*/

	fout_res.open("A_Jacobi.txt");
	slae.readMatrixFromFile("A.txt");
	slae.generateVectX(slae.getDimention());
	slae.mult();
	slae.writeFToFile("F_A.txt");
	cout << slae.findOptimalW(1, fout_res) << endl;
	fout_res.close();


	fout_res.open("A_GaussSeidel.txt");
	slae.generateVectX(slae.getDimention());
	slae.mult();
	cout << slae.findOptimalW(2, fout_res) << endl;
	fout_res.close();
```

```
        fout_res.open("B_Jacobi.txt");
        slae.readMatrixFromFile("B.txt");
        slae.generateVectX(slae.getDimention());
        slae.mult();
        slae.writeFToFile("F_B.txt");
        cout << slae.findOptimalW(1, fout_res) << endl;
        fout_res.close();


        fout_res.open("B_GaussSeidel.txt");
        slae.generateVectX(slae.getDimention());
        slae.mult();
        cout << slae.findOptimalW(2, fout_res) << endl;
        fout_res.close();
}
```

4. Исследования на матрице с диагональным преобладанием

Матрица A:

7	-1	0	0	0	-2	-3	0	0	0		1		-28
-2	9	-4	0	0	0	-1	-2	0	0		2		-19
0	0	7	-4	0	0	0	-2	-1	0		3		-20
0	0	-4	13	-3	0	0	0	-2	-4		4		-33
0	0	0	-3	6	-2	0	0	0	-1	*	5	=	-4
-1	0	0	0	-4	5	0	0	0	0		6		9
0	-4	0	0	0	0	5	-1	0	0		7		19
0	-1	-3	0	0	0	-2	7	-1	0		8		22
0	0	-3	-2	0	0	0	-1	6	0		9		29
0	0	0	-2	-1	0	0	0	-2	5		10		19

Матрица B:

7	1	0	0	0	2	3	0	0	0		1		42
2	9	4	0	0	0	1	2	0	0		2		55
0	0	7	4	0	0	0	2	1	0		3		62
0	0	4	13	3	0	0	0	2	4		4		137
0	0	0	3	6	2	0	0	0	1	*	5	=	64
1	0	0	0	4	5	0	0	0	0		6		51
0	4	0	0	0	0	5	1	0	0		7		51
0	1	3	0	0	0	2	7	1	0		8		90
0	0	3	2	0	0	0	1	6	0		9		79
0	0	0	2	1	0	0	0	2	5		10		81

Рассчёт числа обусловленности через MathCad

conde(A) = 56.165
cond1(A) = 59.33
cond2(A) = 32.684
$\frac{\lambda_{max}}{\lambda_{min}} = 29.051$

conde(B) = 22.443
cond1(B) = 20.17
cond2(B) = 9.994
$\frac{\lambda_{max}}{\lambda_{min}} = 9.221$

Метод Якоби:

Матрица А					Матрица В				
ω	x	ω	N	ν_a	ω	X	ω	N	ν_a
0.0	0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000	-1.0000000000000000e+00 -2.0000000000000000e+00 -3.0000000000000000e+00 -4.0000000000000000e+00 -5.0000000000000000e+00 -6.0000000000000000e+00 -7.0000000000000000e+00 -8.0000000000000000e+00 -9.0000000000000000e+00 -1.0000000000000000e+01	200000	1	0.0	0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000	-1.0000000000000000e+00 -2.0000000000000000e+00 -3.0000000000000000e+00 -4.0000000000000000e+00 -5.0000000000000000e+00 -6.0000000000000000e+00 -7.0000000000000000e+00 -8.0000000000000000e+00 -9.0000000000000000e+00 -1.0000000000000000e+01	200000	1
0.1	0.9999999086871355 1.9999998943391433 2.9999998896802715 3.9999998895363134 4.9999998907211065 5.9999998940223316 6.9999998933957253 7.9999998910905843 8.9999998895630728 9.9999998894794793	-9.1312864491932544e-08 -1.0566085673069381e-07 -1.1031972846353710e-07 -1.1046368664224815e-07 -1.0927889348977260e-07 -1.0597766841868861e-07 -1.0660427474107337e-07 -1.0890941570806945e-07 -1.1043692715873021e-07 -1.1052052073523555e-07	64347	172	0.1	0.9999999994023329 2.0000000025239726 2.9999999952948841 4.0000000039983785 4.9999999947566538 6.0000000059475322 6.9999999962053341 8.0000000036666350 9.0000000005634035 9.9999999989300790	-5.9766713800257776e-10 2.5239725864878437e-09 -4.7051158524880066e-09 3.9983785171671116e-09 -5.2433462016665544e-09 5.9475322444768608e-09 -3.7946659148246908e-09 3.6666349956249178e-09 5.6340354603889864e-10 -1.0699210406528437e-09	558	6
0.2	0.9999999086837773 1.9999998943352566 2.9999998896762126 3.9999998895322491 4.9999998907170866 5.9999998940184351 6.9999998933918048 7.9999998910865777 8.9999998895590103 9.9999998894754132	-9.1316222694537430e-08 -1.0566474339945842e-07 -1.1032378743891513e-07 -1.1046775094669670e-07 -1.0928291338530016e-07 -1.0598156485741583e-07 -1.0660819516061792e-07 -1.0891342228092071e-07 -1.1044098968682192e-07 -1.1052458681604094e-07	32169	172	0.2	0.9999999994002374 2.0000000025325595 2.9999999952789231 4.0000000040119579 4.9999999947387597 6.0000000059678698 6.9999999961924351 8.0000000036790730 9.0000000005653042 9.9999999989264872	-5.9976257293925528e-10 2.5325594954495045e-09 -4.7210768627792277e-09 4.0119578770259068e-09 -5.2612403322882528e-09 5.9678697539311543e-09 -3.8075649300139958e-09 3.6790730462143983e-09 5.6530424785705691e-10 -1.0735128341821110e-09	275	6
0.3	0.9999999086804374 1.9999998943313917 2.9999998896721771 3.9999998895282083 4.9999998907130889 5.9999998940145582 6.9999998933879048 7.9999998910825925 8.9999998895549691 9.9999998894713720	-9.1319562578462410e-08 -1.0566860830785174e-07 -1.1032782287756504e-07 -1.1047179171441712e-07 -1.0928691107636723e-07 -1.0598544175621782e-07 -1.0661209515205883e-07 -1.0891740753748991e-07 -1.1044503089863156e-07 -1.1052862802785057e-07	21443	172	0.3	0.9999999994163842 2.0000000024641471 2.9999999954064975 4.0000000039035584 4.9999999948808354 6.0000000058067489 6.9999999962953003 8.0000000035796557 9.0000000005500169 9.9999999989555306	-5.8361582233601439e-10 2.4641471085828925e-09 -4.5935024672871805e-09 3.9035583654367656e-09 -5.1191646477377617e-09 5.8067488595270333e-09 -3.7046996581580061e-09 3.5796556829836845e-09 5.5001692089717835e-10 -1.0444693998579169e-09	181	6
0.4	0.9999999086771151 1.9999998943275474 2.9999998896681630 3.9999998895241893 4.9999998907091125 5.9999998940107018 6.9999998933840262 7.9999998910786312 8.9999998895509510 9.9999998894673503	-9.1322884920863601e-08 -1.0567245256609681e-07 -1.1033183699993288e-07 -1.1047581072176627e-07 -1.0929088745115223e-07 -1.0598929822691616e-07 -1.0661597382721766e-07 -1.0892136881324177e-07 -1.1044904901780228e-07 -1.1053264969973498e-07	16080	172	0.4	0.9999999994339085 2.0000000023899727 2.9999999955448033 4.0000000037860381 4.9999999950348917 6.0000000056320282 6.9999999964068262 8.0000000034718752 9.0000000005334506 9.9999999989870041	-5.6609150700381861e-10 2.3899726642184760e-09 -4.4551966560391065e-09 3.7860381496557238e-09 -4.9651083244839356e-09 5.6320281771604641e-09 -3.5931737585315204e-09 3.4718752317530743e-09 5.3345061701293162e-10 -1.0129959093774232e-09	134	6

0.5	0.9999999087744764 1.9999998944402071 2.9999998897857911 3.9999998896419706 4.9999998908256309 5.9999998941237003 6.9999998934976926 7.9999998911947552 8.9999998896687039 9.9999998895851920	-9.1225523579652190e-08 -1.0555979290671758e-07 -1.1021420887047384e-07 -1.1035802938152983e-07 -1.0917436910062861e-07 -1.0587629972746981e-07 -1.0650230741759970e-07 -1.0880524481393650e-07 -1.1033129609927528e-07 -1.1041480796336600e-07	12863	172	0.5	0.9999999994687327 2.0000000022427988 2.9999999958191794 4.0000000035528807 4.9999999953406107 6.0000000052852664 6.9999999966281008 8.0000000032580569 9.0000000005005916 9.9999999990494128	-5.3126725241270378e-10 2.2427988355389061e-09 -4.1808205786253438e-09 3.5528806563434046e-09 -4.6593893188173752e-09 5.2852664467195609e-09 -3.3718992042963691e-09 3.2580569353513056e-09 5.0059156819770578e-10 -9.5058716453877423e-10	106	6
0.6	0.9999999086705368 1.9999998943199353 2.9999998896602151 3.9999998895162316 4.9999998907012406 5.9999998940030670 6.9999998933763470 7.9999998910707850 8.9999998895429947 9.9999998894593887	-9.1329463214329110e-08 -1.0568006469924285e-07 -1.1033978486452156e-07 -1.1048376835631757e-07 -1.0929875937648603e-07 -1.0599693300861190e-07 -1.0662365301783439e-07 -1.0892921498140140e-07 -1.1045700532008595e-07 -1.1054061133108917e-07	10717	172	0.6	0.9999999994730626 2.0000000022244202 2.9999999958534587 4.0000000035237555 4.9999999953787713 6.0000000052419962 6.9999999966557365 8.0000000032313423 9.0000000004964829 9.9999999990572199	-5.2693738261666567e-10 2.2244202035892613e-09 -4.1465413325170175e-09 3.5237555096045980e-09 -4.6212287330149593e-09 5.2419961704686102e-09 -3.3442635327673997e-09 3.2313423048435652e-09 4.9648285482817300e-10 -9.4278007622961013e-10	87	6
0.7	0.9999999087176388 1.9999998943744386 2.9999998897171216 3.9999998895732118 4.9999998907576098 5.9999998940577344 6.9999998934313359 7.9999998911269641 8.9999998895999624 9.9999998895163973	-9.1282361225353270e-08 -1.0562556140847335e-07 -1.1028287838499296e-07 -1.1042678815798013e-07 -1.0924239024490134e-07 -1.0594226562687936e-07 -1.0656866411551391e-07 -1.0887303591999853e-07 -1.1040003755624639e-07 -1.1048360271104229e-07	9185	172	0.7	0.9999999994290811 2.0000000024100117 2.9999999955075105 4.0000000038177488 4.9999999949931908 6.0000000056793823 6.9999999963767161 8.0000000035009347 9.0000000005379039 9.9999999989785735	-5.7091886773719125e-10 2.4100117457237502e-09 -4.4924894915254754e-09 3.8177487837742774e-09 -5.0068091894672762e-09 5.6793822977851960e-09 -3.6232838951377744e-09 3.5009346532888230e-09 5.3790394360930804e-10 -1.0214264989372168e-09	73	6
0.8	0.9999999087647489 1.9999998944289514 2.9999998897740388 3.9999998896302031 4.9999998908139895 5.9999998941124097 6.9999998934863363 7.9999998911831529 8.9999998896569391 9.9999998895734183	-9.1235251131749351e-08 -1.0557104856978583e-07 -1.1022596124732331e-07 -1.1036979685741244e-07 -1.0918601045517562e-07 -1.0588759025154104e-07 -1.0651366366687398e-07 -1.0881684708863304e-07 -1.1034306091062263e-07 -1.1042658165649755e-07	8036	172	0.8	0.9999999994538367 2.0000000023056472 2.9999999957021970 4.0000000036523948 4.9999999952101302 6.0000000054334102 6.9999999965337336 8.0000000033493013 9.0000000005146408 9.9999999990228829	-5.4616333677870443e-10 2.3056472286953067e-09 -4.2978030023732572e-09 3.6523948310218657e-09 -4.7898698340986812e-09 5.4334101662334433e-09 -3.4662663850326680e-09 3.3493012807639388e-09 5.1464077444052236e-10 -9.7711705393521697e-10	63	6
0.9	0.9999999087363747 1.9999998943961186 2.9999998897397582 3.9999998895958768 4.9999998907800318 5.9999998940794788 6.9999998934532091 7.9999998911493098 8.9999998896226234 9.9999998895390760	-9.1263625323634301e-08 -1.0560388141733767e-07 -1.1026024182569927e-07 -1.1040412317697701e-07 -1.0921996818069601e-07 -1.0592052124280826e-07 -1.0654679094557196e-07 -1.0885069023913729e-07 -1.1037737657204616e-07 -1.1046092396327367e-07	7142	172	0.83	0.9999999993365503 2.0000000025028610 2.9999999951482041 4.0000000039993227 4.9999999946010005 6.0000000059849068 6.9999999960762365 8.0000000036628212 9.0000000005059793 9.9999999988451087	-6.6344973959076015e-10 2.5028610295407816e-09 -4.8517958539662231e-09 3.9993226508272528e-09 -5.3989994697190014e-09 5.9849067923778421e-09 -3.9237635363065237e-09 3.6628211574907255e-09 5.0597925849160674e-10 -1.1548912937087152e-09	60	6
1.0	0.9999999087583639 1.9999998944215627 2.9999998897663240 3.9999998896224778 4.9999998908063468 5.9999998941049988 6.9999998934788810 7.9999998911755368 8.9999998896492173 9.9999998895656894	-9.1241636135386273e-08 -1.0557843732605932e-07 -1.1023367596507683e-07 -1.1037752223330699e-07 -1.0919365323047714e-07 -1.0589500121227502e-07 -1.0652111903652894e-07 -1.0882446321858197e-07 -1.1035078273380350e-07 -1.1043431058510578e-07	6427	172	0.9	0.9999999995649510 1.9999999994965956 2.9999999994743898 3.9999999994737170 4.9999999994793489 5.9999999994950919 6.9999999994920916 7.9999999994811208 8.9999999994738378 9.9999999994734399	-4.3504899682744735e-10 -5.0340442925289608e-10 -5.2561022201302876e-10 -5.2628301716595161e-10 -5.2065107780663311e-10 -5.0490811531744839e-10 -5.0790838201919541e-10 -5.1887916185933136e-10 -5.2616222490087239e-10 -5.2656012883289804e-10	102	0

					1.0	1.00000000004370451 2.00000000005057172 3.00000000005280150 4.00000000005287060 5.00000000005230341 6.00000000005072325 7.00000000005102310 8.00000000005212666 9.00000000005285781 10.00000000005289760	4.3704506680342092e-10 5.0571724585779521e-10 5.2801496508436685e-10 5.2870596789489355e-10 5.2303406050668855e-10 5.0723247824180362e-10 5.1023096858671124e-10 5.2126658545148530e-10 5.2857807020245673e-10 5.2897597413448239e-10	8363	0

Метод Гаусса-Зейделя:

Матрица А					Матрица В				
ω	x	ω	N	ν_a	ω	X	ω	N	ν_a
0,00	0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000	-1.0000000000000000e+00 -2.0000000000000000e+00 -3.0000000000000000e+00 -4.0000000000000000e+00 -5.0000000000000000e+00 -6.0000000000000000e+00 -7.0000000000000000e+00 -8.0000000000000000e+00 -9.0000000000000000e+00 -1.0000000000000000e+01	200000	1	0,00	0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000 0.0000000000000000	-1.0000000000000000e+00 -2.0000000000000000e+00 -3.0000000000000000e+00 -4.0000000000000000e+00 -5.0000000000000000e+00 -6.0000000000000000e+00 -7.0000000000000000e+00 -8.0000000000000000e+00 -9.0000000000000000e+00 -1.0000000000000000e+01	200000	1
0,10	0.9999999093196420 1.9999998950664688 2.9999998904450242 3.9999998903162330 4.9999998915035464 5.9999998947920208 6.9999998941427544 7.9999998918614708 8.9999998903518303 9.9999998902814120	-9.0680358000305716e-08 -1.0493353119755966e-07 -1.0955497575793061e-07 -1.0968376695785764e-07 -1.0849645359911619e-07 -1.0520797921742542e-07 -1.0585724563583199e-07 -1.0813852924229650e-07 -1.0964816965497448e-07 -1.0971858799280199e-07	61184	171	0,10	0.9999999995005951 2.0000000025938678 2.9999999952014713 4.0000000040164974 4.9999999948627396 6.0000000057219482 6.9999999961539912 8.0000000036633914 9.0000000006115854 9.9999999988800159	-4.9940485080668395e-10 2.5938677872261451e-09 -4.7985286855123377e-09 4.0164973569289941e-09 -5.1372603948607320e-09 5.7219482485493245e-09 -3.8460088447322960e-09 3.6633913680361729e-09 6.1158544895079103e-10 -1.1199841054576609e-09	545	6
0,20	0.9999999101525973 1.9999998960253174 2.9999998914520067 3.9999998913400079 4.9999998925282316 5.9999998957976386 6.9999998951245983 7.9999998928729612 8.9999998913852135 9.9999998913293133	-8.9847402739984261e-08 -1.0397468264145004e-07 -1.0854799326054376e-07 -1.0865999211517874e-07 -1.0747176837355710e-07 -1.0420236140618044e-07 -1.0487540169634713e-07 -1.0712703879534047e-07 -1.0861478649815126e-07 -1.0867068667153035e-07	29008	169	0,20	0.9999999996218625 2.0000000026163027 2.9999999952066458 4.0000000039440664 4.9999999950951111 6.0000000053517635 6.9999999961809234 8.0000000035772807 9.0000000006501910 9.9999999988532178	-3.7813752129522982e-10 2.6163027300185604e-09 -4.7933541580391648e-09 3.9440664068024489e-09 -4.9048889394498474e-09 5.3517634768240896e-09 -3.8190766105117291e-09 3.5772806938894064e-09 6.5019101214147668e-10 -1.1467822247368531e-09	262	6
0,30	0.9999999112249558 1.9999998972607309 2.9999998927482907 3.9999998926548805 4.9999998938419745 5.9999998970846606 6.9999998963867993 7.9999998941716424 8.9999998927105267 9.9999998926706137	-8.8775044204680853e-08 -1.0273926909221132e-07 -1.0725170929148931e-07 -1.0734511945997838e-07 -1.0615802548130659e-07 -1.0291533936168662e-07 -1.0361320068597024e-07 -1.0582835763273124e-07 -1.0728947330562733e-07 -1.0732938626745181e-07	18284	167	0,30	0.9999999997523229 2.0000000026867784 2.9999999951375567 4.0000000039232599 4.9999999952750187 6.0000000050351581 6.9999999961472206 8.0000000035380516 9.0000000007020873 9.9999999988093844	-2.4767710105066953e-10 2.6867783553541358e-09 -4.8624433368615883e-09 3.9232599391425538e-09 -4.7249812951122294e-09 5.0351580682672648e-09 -3.8527794288256700e-09 3.5380516294480913e-09 7.0208727720455499e-10 -1.1906156061058937e-09	167	5

0,40	0.9999999127005458 1.9999998989622769 2.9999998945318063 3.9999998944590494 4.9999998956408591 5.9999998988432548 6.9999998981206319 7.9999998959529313 8.9999998945259225 9.9999998945036292	-8.7299454221145822e-08 -1.0103772307701320e-07 -1.0546819373402627e-07 -1.0554095064563285e-07 -1.0435914088446907e-07 -1.0115674520250195e-07 -1.0187936805294839e-07 -1.0404706873856640e-07 -1.0547407747196758e-07 -1.0549637075030205e-07	12924	165	0,40	0.9999999999034912 2.0000000027765958 2.9999999950529839 4.0000000039015964 4.9999999954730860 6.0000000046929456 6.9999999960991719 8.0000000034980765 9.0000000007607639 9.9999999987625365	-9.6508800950800833e-11 2.7765958421355208e-09 -4.9470161300746440e-09 3.9015963793076480e-09 -4.5269139548054227e-09 4.6929455876920656e-09 -3.9008281049746074e-09 3.4980764951342280e-09 7.6076389632362407e-10 -1.2374634650313965e-09	119	5
0,50	0.9999999143968442 1.9999999009186771 2.9999998965820494 3.9999998965319543 4.9999998977068660 5.9999999008621616 6.9999999001131465 7.9999998979994000 8.9999998966110422 9.9999998966080330	-8.5603155786095897e-08 -9.9081322924021720e-08 -1.0341795064761072e-07 -1.0346804568683865e-07 -1.0229313396337147e-07 -9.9137838383001053e-08 -9.9886853455188884e-08 -1.0200059996634536e-07 -1.0338895783945645e-07 -1.0339196698794240e-07	9708	161	0,50	1.0000000000786253 2.0000000027404501 2.9999999952149183 4.0000000036733718 4.9999999959227095 6.0000000040898609 6.9999999962410477 8.0000000032744456 9.0000000007844889 9.9999999987800976	7.8625328470138811e-11 2.7404500890781947e-09 -4.7850816642380778e-09 3.6733718289383432e-09 -4.0772905052222086e-09 4.0898608943962245e-09 -3.7589522605685488e-09 3.2744456035516123e-09 7.8448891827065381e-10 -1.2199024013170856e-09	90	5
0,60	0.9999999165601673 1.9999999034149827 2.9999998991966050 3.9999998991714749 4.9999999003346041 5.9999999034270148 6.9999999026518536 7.9999999006047222 8.9999998992636296 9.9999998992817059	-8.3439832709863992e-08 -9.6585017317352140e-08 -1.0080339496454371e-07 -1.0082852508475071e-07 -9.9665395936199275e-08 -9.6572985164300462e-08 -9.7348146432807425e-08 -9.9395277786129554e-08 -1.0073637035645788e-07 -1.0071829414926015e-07	7565	157	0,60	1.0000000002807767 2.0000000028157268 2.9999999952120024 4.0000000035566838 4.9999999962618871 6.0000000035997791 6.9999999962463093 8.0000000031532856 9.0000000008382788 9.9999999987596446	2.8077673519533164e-10 2.8157267628614591e-09 -4.7879975539899533e-09 3.5566838363365605e-09 -3.7381129303071248e-09 3.5997791414388303e-09 -3.7536906916102453e-09 3.1532856326066394e-09 8.3827877972453280e-10 -1.2403553739659401e-09	70	5
0,70	0.9999999192455081 1.9999999065148426 2.9999999024419179 3.9999999024440989 4.9999999035898144 5.9999999066015297 6.9999999058009337 7.9999999038344454 8.9999999025501296 9.9999999025911031	-8.0754491871282141e-08 -9.3485157393047302e-08 -9.7558082057247475e-08 -9.7555901135137901e-08 -9.6410185612683108e-08 -9.3398470291106150e-08 -9.4199066325018066e-08 -9.6165554630545103e-08 -9.7449870395394100e-08 -9.7408896948536494e-08	6035	152	0,70	1.0000000005488765 2.0000000030293013 2.9999999950298748 4.0000000035407224 4.9999999965337558 6.0000000031642529 6.9999999960994463 8.0000000031260612 9.0000000009321663 9.9999999986965804	5.4887649980628339e-10 3.0293012542870201e-09 -4.9701252002876117e-09 3.5407223819561295e-09 -3.4662441805721755e-09 3.1642528597330966e-09 -3.9005536578429201e-09 3.1260611876859912e-09 9.3216634411419363e-10 -1.3034195944783278e-09	55	5
0,80	0.9999999225215240 1.9999999102977470 2.9999999064009342 3.9999999064327452 4.9999999075544146 5.9999999104650366 6.9999999096404606 7.9999999077702997 8.9999999065533469 9.9999999066189620	-7.7478475968284499e-08 -8.9702252958900885e-08 -9.3599065831284634e-08 -9.3567254833004654e-08 -9.2445585408995612e-08 -8.9534963443327342e-08 -9.0359539406392742e-08 -9.2229700321411201e-08 -9.3446653082196462e-08 -9.3381038013262696e-08	4888	146	0,80	1.0000000008763987 2.0000000031033407 2.9999999951607190 4.0000000032525724 4.9999999971326075 6.0000000024182469 6.9999999961740178 8.0000000028739144 9.0000000009797372 9.9999999987193551	8.7639873136424740e-10 3.1033406955316423e-09 -4.8392809759434385e-09 3.2525724336096573e-09 -2.8673925456246252e-09 2.4182469360312098e-09 -3.8259821977248976e-09 2.8739144397604832e-09 9.7973718027333234e-10 -1.2806449234403772e-09	43	4
0,90	0.9999999264008406 1.9999999147783454 2.9999999110889077 3.9999999111525630 4.9999999122432941 5.9999999150318679 6.9999999141851035 7.9999999124272048 8.9999999112883486 9.9999999113802431	-7.3599159433612726e-08 -8.5221654622102960e-08 -8.8911092266386049e-08 -8.8847436963135351e-08 -8.7756705902108934e-08 -8.4968132085805337e-08 -8.5814896522151685e-08 -8.7572795237633727e-08 -8.8711651358153176e-08 -8.8619756866137323e-08	3996	139	0,90	1.0000000038005359 2.0000000000939746 3.0000000002875491 3.9999999984831325 5.0000000041375783 5.9999999955099641 6.9999999994701430 7.9999999999332410 9.0000000004779661 9.9999999994667217	3.8005358860004890e-09 9.3974605874791450e-11 2.8754909564554509e-10 -1.5168675204790816e-09 4.1375782799946137e-09 -4.4900358986410538e-09 -5.2985704712682491e-10 -6.6759042738340213e-11 4.7796611113426479e-10 -5.3327831039950979e-10	29	3

1,00	0.9999999308334052 1.9999999198987219 2.9999999164453777 3.9999999165430151 4.9999999175965577 5.9999999202439271 6.9999999193764344 7.9999999177454706 8.9999999166946054 9.9999999168143585	-6.9166594784952906e-08 -8.0101278054911518e-08 -8.3554622332115969e-08 -8.3456984878438334e-08 -8.2403442291933970e-08 -7.9756072857151139e-08 -8.0623565601456448e-08 -8.2254529409908628e-08 -8.3305394582566805e-08 -8.3185641486238637e-08	3282	130	1,00	0.9999999985349012 1.9999999971977958 3.0000000034906695 3.9999999981323957 5.0000000010057626 5.9999999994884101 7.0000000029451535 7.9999999983118562 8.9999999991585575 10.0000000008824657	-1.4650988200415327e-09 -2.8022042464215247e-09 3.4906695312031388e-09 -1.8676042934373527e-09 1.0057625843273854e-09 -5.1158988156885243e-10 2.9451534544477909e-09 -1.6881438469340537e-09 -8.4144247125550464e-10 8.8246565610461403e-10	28	3
1,10	0.9999999362195577 1.9999999261226320 2.9999999229539593 3.9999999230867722 4.9999999240904813 5.9999999265618751 6.9999999256808847 7.9999999242007576 8.9999999232536627 9.9999999234018109	-6.3780442260963355e-08 -7.3877368000907495e-08 -7.7046040658501624e-08 -7.6913227786690186e-08 -7.5909518670869147e-08 -7.3438124914559921e-08 -7.4319115306309413e-08 -7.5799242438279180e-08 -7.6746337285271693e-08 -7.6598189124865712e-08	2698	120	1,10	1.0000000010966072 2.0000000029960363 2.9999999964432269 4.0000000018857520 4.9999999988940127 6.0000000005321850 6.9999999971985645 8.0000000014561454 9.0000000008195524 9.9999999992303810	1.0966072494511536e-09 2.9960363079339913e-09 -3.5567730982677404e-09 1.8857519989978755e-09 -1.1059873017416066e-09 5.3218496276485894e-10 -2.8014355279992742e-09 1.4561454264594431e-09 8.1955242592357536e-10 -7.6961903516803432e-10	22	3
1,20	0.9999999425487642 1.9999999334382499 2.9999999306018923 3.9999999307699348 4.9999999317104402 5.9999999339706465 6.9999999330854870 7.9999999317792065 8.9999999309508993 9.9999999311269487	-5.7451235768724018e-08 -6.6561750111659990e-08 -6.9398107704898848e-08 -6.9230065236070004e-08 -6.8289559784773246e-08 -6.6029353540386637e-08 -6.6914513041638202e-08 -6.8220793458806384e-08 -6.9049100659412943e-08 -6.8873051262130502e-08	2211	108	1,13	1.0000000011769521 2.0000000024572766 2.9999999987437191 4.0000000000882867 5.0000000008843353 5.9999999992031521 6.9999999980970173 8.0000000002200746 9.0000000003504788 9.9999999997705533	1.1769520913418319e-09 2.4572766044173022e-09 -1.2562808571203732e-09 8.8286711275031848e-11 8.8433527167808279e-10 -7.9684792098078105e-10 -1.9029826603400579e-09 2.2007462519013643e-10 3.5047875712734822e-10 -2.2944668387481215e-10	21	2
1,30	0.9999999492451012 1.9999999411787075 2.9999999386934681 3.9999999388973002 4.9999999397697534 5.9999999418053385 6.9999999409187517 7.9999999397955817 8.9999999390921648 9.9999999392963090	-5.0754898839322493e-08 -5.8821292503452582e-08 -6.1306531851101909e-08 -6.1102699788762038e-08 -6.0230246567982704e-08 -5.8194661534116676e-08 -5.9081248338088699e-08 -6.0204418339537824e-08 -6.0907835219836670e-08 -6.0703690962782275e-08	1797	96	1,20	0.9999999987847872 2.0000000005149943 3.0000000001319984 4.0000000000621121 5.0000000002193437 6.0000000001492202 6.9999999991679802 8.0000000000782556 8.9999999997793747 10.0000000000433005	-1.2152128192610689e-09 5.1499426945156301e-10 1.3199841220057351e-10 6.2112093246469158e-11 2.1934365435072323e-10 1.4922019175855894e-10 -8.3201978640090601e-10 7.8255624202938634e-11 -2.2062529581035051e-10 4.3300474317220505e-11	29	1
1,40	0.9999999564638040 1.9999999495241467 2.9999999474161978 3.9999999476550396 4.9999999484513458 5.9999999502420742 6.9999999493609737 7.9999999484332092 8.9999999478625519 9.9999999480936701	-4.3536195981630499e-08 -5.0475853274889459e-08 -5.2583802201411345e-08 -5.2344960366212945e-08 -5.1548654234068181e-08 -4.9757925779658763e-08 -5.0639026305532298e-08 -5.1566790837398457e-08 -5.2137448136591047e-08 -5.1906329900930359e-08	1440	82	1,30	0.9999999984195528 2.0000000008202838 3.0000000002062284 4.0000000000733174 5.0000000002873657 6.0000000002086127 6.9999999985998640 8.0000000002518110 8.9999999995897291 10.0000000001871019	-1.5804472175418027e-09 8.2028384085219841e-10 2.0622836771622133e-10 7.3317352189405938e-11 2.8736568680187702e-10 2.0861268268390631e-10 -1.4001360071347335e-09 2.5181101648286131e-10 -4.1027092834156065e-10 1.8710188953718898e-10	53	1
1,50	0.9999999637912922 1.9999999579953760 2.9999999562706012 3.9999999565450235 4.9999999572638885 5.9999999588057529 6.9999999579305943 7.9999999572012142 8.9999999567652704 9.9999999570235065	-3.6208707809848306e-08 -4.2004624001279467e-08 -4.3729398768732608e-08 -4.3454976506041021e-08 -4.2736111538488331e-08 -4.1194247124565209e-08 -4.2069405736810950e-08 -4.2798785848674470e-08 -4.3234729574237463e-08 -4.2976493475066491e-08	1127	67	1,40	1.0000000020057507 1.9999999988120774 2.9999999996972115 3.9999999999082676 4.9999999996309450 5.9999999997256737 7.0000000021516833 7.9999999994792335 9.0000000006961685 9.9999999995649347	2.0057506766590905e-09 -1.1879226491373629e-09 -3.0278846097075984e-10 -9.1732399454258484e-11 -3.6905500877537634e-10 -2.7432633942225948e-10 2.1516832759971294e-09 -5.2076654100119413e-10 6.9616845621567336e-10 -4.3506531710590934e-10	160	1

1,60	0.9999999717603811 1.9999999672109119 2.9999999659002792 3.9999999662056220 4.9999999668341486 5.9999999680993437 6.9999999672460653 7.9999999667281232 8.9999999664345918 9.9999999667150092	-2.8239618865555371e-08 -3.2789088111329079e-08 -3.4099720824798396e-08 -3.3794377962692579e-08 -3.3165851398564428e-08 -3.1900656338734734e-08 -3.2753934675611163e-08 -3.3271876809237710e-08 -3.3565408230629146e-08 -3.3284990763604583e-08	849	52					
1,70	0.9999999799884447 1.9999999767280119 2.9999999758433669 3.9999999761745211 4.9999999767045651 5.9999999776785895 6.9999999768610621 7.9999999765580494 8.9999999764081782 9.9999999767050856	-2.0011555301735484e-08 -2.3271988114359488e-08 -2.4156633138261441e-08 -2.3825478923100718e-08 -2.3295434914416546e-08 -2.2321410497738725e-08 -2.3138937876865384e-08 -2.3441950602887118e-08 -2.3591821829427317e-08 -2.3294914441862602e-08	596	37					
1,80	0.9999999959034590 1.9999999947579647 2.9999999942299200 3.9999999943088089 4.9999999945928586 5.9999999948459752 6.9999999949697207 7.9999999951899721 8.9999999947141749 9.9999999944148072	-4.0965409953130916e-09 -5.2420352503190770e-09 -5.7700799693805038e-09 -5.6911910739643190e-09 -5.4071414012923924e-09 -5.1540247625325719e-09 -5.0302793042078520e-09 -4.8100279315121952e-09 -5.2858251109455523e-09 -5.5851927527328371e-09	368	9					

5. Вывод:

В ходе работы стало понятно, что при разных параметрах релаксации методы Якоби и Гаусса-Зейделя сходятся с разной скоростью, а при при некоторых даже расходится. Наиболее быстро метод Якоби сходится при w = 1 (матрица A), w = 0.83 (матрица B), а метод Гаусса-Зейделя при w = 1.8 (матрица A), w = 1.13 (матрица B).

Кроме параметра релаксации на число итераций влияют также относительная невязка, число обусловленности матрицы и начальное приближение.