МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Новосибирский государственный технический университет»

НГТУ НЭТИ

Кафедра прикладной математики

Практическая работа №2

по дисциплине «Численные методы»

ФПМИ

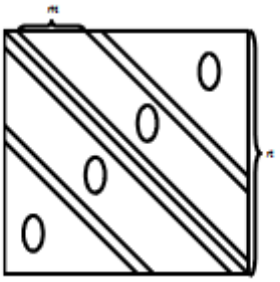
Группа	ПМ-92
Вариант	10
Студенты	Кутузов Иван
	Иванов Владислав
Преподаватель	Задорожный А. Г.
Дата	08.10.2021

Новосибирск

Цель работы

Разработать программу решения СЛАУ методами Гаусса-Зейделя, Якоби с хранением матрицы в диагональном формате. Исследовать сходимость методов для тестовых матриц и ее 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.



Анализ

Пусть имеется система линейных алгебраических уравнений:

$$Ax = F$$

Выбирается начальное приближение $x^{(0)} = (x_1^0, x_2^0, ..., x_n^0)$ (при отсутствии априорных данных для выбора приближения в качестве начального можно выбрать нулевой вектор).

Метод Якоби:
Каждое следующее приближение в методе Якоби рассчитывается по формуле:

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

где k - номер текущей итерации.

Метод Гаусса-Зейделя:
Каждое следующее приближение рассчитывается по формуле:

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

Для ускорения сходимости итерационного процесса можно использовать параметр релаксации.

Метод Якоби с параметром релаксации:

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

Метод Гаусса-Зейделя с параметром релаксации:

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

Условие выхода из итерационного процесса для рассмотренных методов:

- выход по относительной невязке:

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

- защита от зацикливания: $k < maxiter$, $maxiter$- максимальное количество итераций.

Текст программы

Программа была разбита на следующие модули:

common.h — содержит макрос, позволяющий быстро изменить точность вычисления (double или float).

io.h — содержит процедуры для чтения и записи данных в файл.

AlgebraicStructures.h — содержит определения структур данных векторов и матриц.

Source.c — содержит основные процедуры решения СЛАУ.

common.h

```
#ifndef FLOAT
	typedef double real;
	typedef double real_sum;
#endif
```

io.h

```
#pragma once

#include "common.h"
#include "AlgebraicStructures.h"

#define FLOAT

void ReadVectorReal(struct VectorReal* vector, const char* filePath);

void ReadBlockMatrix(struct BlockMatrix* matrix, const char* filePath);

void WriteVectorReal(struct VectorReal* vector, const char* filePath);

void WriteTable(real weight, int iteration, int conditionality, const char*
filePath);
```

io.c

```c
#include "io.h"
#include <stdio.h>
#include <corecrt_malloc.h>

void ReadVectorReal(struct VectorReal* vector, const char* filePath)
{
	FILE* stream = NULL;

	fopen_s(&stream, filePath, "r");

	if (stream != NULL)
	{
		fscanf_s(stream, "%d", &vector->n);
		vector->data = malloc(sizeof(real) * vector->n);
		for (int i = 0; i < vector->n; i++)
		{
			fscanf_s(stream, "%lf", (vector->data + i));
		}

		fclose(stream);
	}
}

void ReadBlockMatrix(struct BlockMatrix* matrix, const char* filePath)
{
	FILE* stream = NULL;

	fopen_s(&stream, filePath, "a+");

	if (stream != NULL)
	{
		fscanf_s(stream, "%d", &matrix->n);
		fscanf_s(stream, "%d", &matrix->m);

		fclose(stream);
	}
}

void WriteVectorReal(struct VectorReal* vector, const char* filePath)
{
	FILE* stream = NULL;

	fopen_s(&stream, filePath, "a+");

	if (stream != NULL)
	{
		fprintf_s(stream, "--------------\n");
		for (int i = 0; i < vector->n; i++)
		{
			fprintf_s(stream, "%.15lf\n", vector->data[i]);
		}
		fprintf_s(stream, "--------------\n");

		fclose(stream);
	}
}
```

```
void WriteTable(real weight, int iteration, int conditionality, const char*
filePath)
{
	FILE* stream = NULL;

	fopen_s(&stream, filePath, "a+");

	if (stream != NULL)
	{
		fprintf_s(stream, "--------------\n");
		fprintf_s(stream, "weight: %.2lf\niteration: %d\nnumber of
conditionality: %d", weight, iteration, conditionality);
		fprintf_s(stream, "--------------\n");

		fclose(stream);
	}
}
```

AlgebraicStructures.h

```
#pragma once

#include "common.h"

struct VectorReal
{
	int n;
	real* data;
};

struct BlockMatrix
{
	int n;
	int m;
	struct VectorReal* di;
	struct VectorReal* al1;
	struct VectorReal* al2;
	struct VectorReal* al3;
	struct VectorReal* au1;
	struct VectorReal* au2;
	struct VectorReal* au3;
};

struct VectorReal* AllocateMemoryForVector();
struct BlockMatrix* AllocateMemoryForMatrix();

void FreeVectorMemory(struct VectorReal* vector);
void FreeMatrixMemory(struct BlockMatrix* matrix);
```

AlgebraicStructures.c

```
#include "AlgebraicStructures.h"
#include <stddef.h>

struct VectorReal* AllocateMemoryForVector()
{
	struct VectorReal* vector = malloc(sizeof(struct VectorReal));
	vector->n = 0;

	return vector;
}
```

```
struct BlockMatrix* AllocateMemoryForMatrix()
{
	struct BlockMatrix* matrix = malloc(sizeof(struct BlockMatrix));

	matrix->n = 0;
	matrix->m = -1;

	matrix->di = AllocateMemoryForVector();
	matrix->al1 = AllocateMemoryForVector();
	matrix->al2 = AllocateMemoryForVector();
	matrix->al3 = AllocateMemoryForVector();
	matrix->au1 = AllocateMemoryForVector();
	matrix->au2 = AllocateMemoryForVector();
	matrix->au3 = AllocateMemoryForVector();

	return matrix;
}

void FreeVectorMemory(struct VectorReal* vector)
{
	free(vector->data);
	vector->data = NULL;
	free(vector);
	vector = NULL;
}

void FreeMatrixMemory(struct BlockMatrix* matrix)
{
	FreeVectorMemory(matrix->di);
	FreeVectorMemory(matrix->al1);
	FreeVectorMemory(matrix->al2);
	FreeVectorMemory(matrix->al3);
	FreeVectorMemory(matrix->au1);
	FreeVectorMemory(matrix->au2);
	FreeVectorMemory(matrix->au3);

	free(matrix);
	matrix = NULL;
}
```

Source.c

```
#include "io.h"
#include <stdio.h>
#include <math.h>
#include <corecrt_malloc.h>

struct BlockMatrix* matrix;
struct VectorReal* F;
struct VectorReal* Fk;
struct VectorReal* xk;
struct VectorReal* xk1;
real FNorm;
const int maxiter = 100000;
const real error = 0.0000000000001;
const int iterations = 200;

real MultiplyLineByVector(int line)
{
	real result = 0.0;
	int m = matrix->m;
	int n = matrix->n;

	if (line > 0)
	{
		result += matrix->al1->data[line - 1] * xk->data[line - 1];

		if (line > m + 1)
		{
			result += matrix->al2->data[line - m - 2] * xk->data[line - m
- 2];

			if (line > m + 2)
			{
				result += matrix->al3->data[line - m - 3] *
xk->data[line - m - 3];
			}
		}
	}

	result += matrix->di->data[line] * xk->data[line];

	if (line < n - 1)
	{
		result += matrix->au1->data[line] * xk->data[line + 1];

		if (line < n - m - 2)
		{
			result += matrix->au2->data[line] * xk->data[line + m + 2];

			if (line < n - m - 3)
			{
				result += matrix->au3->data[line] * xk->data[line + m
+ 3];
			}
		}
	}

	return result;
}

real CalculateEuclideanNorm(struct VectorReal* vector)
{
	real euclideanNorm = 0;

	for (int i = 0; i < vector->n; i++)
	{
```

```
                euclideanNorm += vector->data[i] * vector->data[i];
        }

        return (real)sqrt(euclideanNorm);
}

real CalculateRelativeDiscrepancy()
{
        return CalculateEuclideanNorm(Fk) / FNorm;// || F - A*xk || / || F ||
}

int CalculateConditionalityNumber()//xk1 - становится вектором ошибки
{
        for (int i = 0; i < xk->n; i++)
        {
                xk1->data[i] = xk->data[i] - (real)(i + 1);// x - x*
        }

        return CalculateEuclideanNorm(xk1) / (FNorm *
CalculateRelativeDiscrepancy());
}

void SolveByJacobi(real w)
{
        for (int i = 0; i < matrix->n; i++)
        {
                Fk->data[i] = F->data[i] - MultiplyLineByVector(i);

                xk1->data[i] = xk->data[i] + w * (Fk->data[i]) /
matrix->di->data[i];
        }

        real* swap = xk;
        xk = xk1;
        xk1 = swap;
}

void SolveByGaussSeidel(real w)
{
        for (int i = 0; i < matrix->n; i++)
        {
                Fk->data[i] = F->data[i] - MultiplyLineByVector(i);

                xk->data[i] = xk->data[i] + w * (Fk->data[i]) / matrix->di->data[i];
        }
}

int CalculateByIterationMethod(real w, void(*method)(real w))
{
        int i = 0;

        while (CalculateRelativeDiscrepancy() >= error && i < maxiter)
        {
                method(w);
                i++;
        }

        return i;
}

real FindOptimalWeight(void (*method) (real w))
{
        real optimalWeight, weight;
        int min = maxiter + 1;
        int iter;

        for (int i = 0; i <= iterations; i++)
```

```
	{
		weight = (real)i / 100.0;

		for (int j = 0; j < xk->n; j++)
		{
			xk->data[j] = 0.0;
			xk1->data[j] = 0.0;
			Fk->data[j] = F->data[j];
		}

		iter = CalculateByIterationMethod(weight, method);

		if (iter < min)
		{
			min = iter;
			optimalWeight = weight;
		}

		if (i % 10 == 0)
		{
			int conditionality = CalculateConditionalityNumber();
			WriteTable(weight, iter, conditionality, "Test\\table.txt");
			WriteVectorReal(xk, "Test\\result.txt");
		}
	}

	CalculateByIterationMethod(optimalWeight, method);
	int conditionality = CalculateConditionalityNumber();
	WriteTable(optimalWeight, min, conditionality, "Test\\table.txt");

	return optimalWeight;
}

int main()
{
	matrix = AllocateMemoryForMatrix();
	ReadBlockMatrix(matrix, "Test\\matrix.txt");
	ReadVectorReal(matrix->di, "Test\\di.txt");
	ReadVectorReal(matrix->al1, "Test\\al1.txt");
	ReadVectorReal(matrix->al2, "Test\\al2.txt");
	ReadVectorReal(matrix->al3, "Test\\al3.txt");
	ReadVectorReal(matrix->au1, "Test\\au1.txt");
	ReadVectorReal(matrix->au2, "Test\\au2.txt");
	ReadVectorReal(matrix->au3, "Test\\au3.txt");

	F = AllocateMemoryForVector();
	ReadVectorReal(F, "Test\\vector.txt");

	Fk = AllocateMemoryForVector();
	Fk->n = F->n;
	Fk->data = malloc(sizeof(real) * Fk->n);
	xk = AllocateMemoryForVector();
	xk->n = F->n;
	xk->data = malloc(sizeof(real) * xk->n);
	xk1 = AllocateMemoryForVector();
	xk1->n = F->n;
	xk1->data = malloc(sizeof(real) * xk1->n);

	FNorm = CalculateEuclideanNorm(F);

	FindOptimalWeight(&SolveByJacobi);
	//FindOptimalWeight(&SolveByGaussSeidel);

	return 0;
}
```

Исследования на матрице с диагональным преобладанием

Исходная матрица:

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 с обратными знаками:

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

Число обусловленности для тестируемых матриц:

$norm_e(A) = 56.17$

$norm_{l1}(A) = 59.33$

$norm_{l2}(A) = 32.68$

$\frac{\lambda max}{\lambda min} = 29.05$

$norm_e(B) = 22.44$

$norm_{l1}(B) = 20.17$

$norm_{l2}(B) = 9.99$

$\frac{\lambda max}{\lambda min} = 9.22$

Метод Якоби:

Исходная матрица					Матрица с обратными знаками				
ω	x	$x - x^*$	max i	$v(A)$	ω	x	$x - x^*$	max i	$v(A)$
0.0	0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000	1.000000000000000 2.000000000000000 3.000000000000000 4.000000000000000 5.000000000000000 6.000000000000000 7.000000000000000 8.000000000000000 9.000000000000000 10.000000000000000	100000	1	0.0	0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000	1.000000000000000 2.000000000000000 3.000000000000000 4.000000000000000 5.000000000000000 6.000000000000000 7.000000000000000 8.000000000000000 9.000000000000000 10.000000000000000	100000	1
0.1	0.999999999908502 1.999999999894126 2.999999999889455 3.999999999889309 4.999999999890496 5.999999999893806 6.999999999893179 7.999999999890870 8.999999999889333 9.999999999889246	0,000000000091498 0,000000000105880 0,000000000110550 0,000000000110700 0,000000000109510 0,000000000106200 0,000000000106830 0,000000000109130 0,000000000110671 0,000000000110759	89419	49	0.1	0.999999999999401 2.000000000002527 2.999999999995288 4.000000000004005 4.999999999994748 6.000000000005957 6.999999999996201 8.000000000003674 9.000000000000563 9.999999999998929	0,000000000000599 0,000000000002520 0,000000000004720 0,000000000004000 0,000000000005260 0,000000000005950 0,000000000003800 0,000000000003670 0,000000000000560 0,000000000001080	806	6
0.2	0.999999999908663 1.999999999894312 2.999999999889652 3.999999999889507 4.999999999890693 5.999999999893995 6.999999999893368 7.999999999891062 8.999999999889534 9.999999999889448	0,000000000091337 0,000000000105690 0,000000000110350 0,000000000110500 0,000000000109310 0,000000000106010 0,000000000106640 0,000000000108940 0,000000000110470 0,000000000110560	44703	48	0.2	0.999999999999391 2.000000000002572 2.999999999995205 4.000000000004075 4.999999999994655 6.000000000006063 6.999999999996132 8.000000000003737 9.000000000000574 9.999999999998911	0,000000000000609 0,000000000002570 0,000000000004800 0,000000000004070 0,000000000005350 0,000000000006060 0,000000000003870 0,000000000003730 0,000000000000570 0,000000000001091	397	6
0.3	0.999999999908567 1.999999999894200 2.999999999889535 3.999999999889391 4.999999999890576 5.999999999893883 6.999999999893256 7.999999999890948 8.999999999889418 9.999999999889333	0,000000000091433 0,000000000105800 0,000000000110470 0,000000000110610 0,000000000109430 0,000000000106120 0,000000000106750 0,000000000109060 0,000000000110591 0,000000000110671	29796	48	0.3	0.999999999999399 2.000000000002537 2.999999999995271 4.000000000004019 4.999999999994729 6.000000000005978 6.999999999996185 8.000000000003686 9.000000000000567 9.999999999998925	0,000000000000601 0,000000000002530 0,000000000004730 0,000000000004010 0,000000000005280 0,000000000005970 0,000000000003820 0,000000000003681 0,000000000000560 0,000000000001080	261	6
0.4	0.999999999908797 1.999999999894466 2.999999999889813 3.999999999889670 4.999999999890854 5.999999999894150 6.999999999893523 7.999999999891221 8.999999999889697 9.999999999889614	0,000000000091203 0,000000000105540 0,000000000110190 0,000000000110330 0,000000000109150 0,000000000105850 0,000000000106480 0,000000000108780 0,000000000110310 0,000000000110390	22346	48	0.4	0.999999999999409 2.000000000002493 2.999999999995353 4.000000000003949 4.999999999994822 6.000000000005874 6.999999999996254 8.000000000003620 9.000000000000558 9.999999999998943	0,000000000000591 0,000000000002490 0,000000000004650 0,000000000003940 0,000000000005180 0,000000000005870 0,000000000003750 0,000000000003620 0,000000000000551 0,000000000001060	193	6

0.5	0.999999999908764 1.999999999894428 2.999999999889773 3.999999999889629 4.999999999890814 5.999999999894112 6.999999999893485 7.999999999891182 8.999999999889656 9.999999999889573	0,000000000091236 0,000000000105580 0,000000000110230 0,000000000110380 0,000000000109190 0,000000000105890 0,000000000106520 0,000000000108820 0,000000000110351 0,000000000110431	17874	48	0.5	0.999999999999405 2.000000000002511 2.999999999995319 4.000000000003978 4.999999999994782 6.000000000005918 6.999999999996225 8.000000000003649 9.000000000000560 9.999999999998936	0,000000000000595 0,000000000002510 0,000000000004690 0,000000000003970 0,000000000005220 0,000000000005910 0,000000000003780 0,000000000003640 0,000000000000560 0,000000000001069	152	6
0.6	0.999999999908774 1.999999999894440 2.999999999889785 3.999999999889641 4.999999999890824 5.999999999894123 6.999999999893498 7.999999999891195 8.999999999889669 9.999999999889585	0,000000000091226 0,000000000105560 0,000000000110220 0,000000000110360 0,000000000109180 0,000000000105880 0,000000000106510 0,000000000108810 0,000000000110340 0,000000000110420	14893	48	0.6	0.999999999999437 2.000000000002375 2.999999999995573 4.000000000003762 4.999999999995066 6.000000000005597 6.999999999996430 8.000000000003450 9.000000000000529 9.999999999998993	0,000000000000563 0,000000000002370 0,000000000004430 0,000000000003760 0,000000000004940 0,000000000005590 0,000000000003570 0,000000000003450 0,000000000000520 0,000000000001011	125	6
0.7	0.999999999908653 1.999999999894300 2.999999999889639 3.999999999889495 4.999999999890681 5.999999999893983 6.999999999893356 7.999999999891051 8.999999999889521 9.999999999889438	0,000000000091347 0,000000000105700 0,000000000110370 0,000000000110510 0,000000000109320 0,000000000106020 0,000000000106650 0,000000000108950 0,000000000110481 0,000000000110569	12763	48	0.7	0.999999999999503 2.000000000002100 2.999999999996085 4.000000000003326 4.999999999995637 6.000000000004949 6.999999999996843 8.000000000003050 9.000000000000469 9.999999999999110	0,000000000000497 0,000000000002100 0,000000000003920 0,000000000003320 0,000000000004370 0,000000000004940 0,000000000003160 0,000000000003050 0,000000000000460 0,000000000000890	106	6
0.8	0.999999999908638 1.999999999894282 2.999999999889621 3.999999999889478 4.999999999890663 5.999999999893966 6.999999999893338 7.999999999891032 8.999999999889505 9.999999999889422	0,000000000091362 0,000000000105720 0,000000000110380 0,000000000110530 0,000000000109340 0,000000000106040 0,000000000106670 0,000000000108970 0,000000000110500 0,000000000110580	11166	48	0.83	0.999999999999438 2.000000000002376 2.999999999995573 4.000000000003766 4.999999999995065 6.000000000005600 6.999999999996430 8.000000000003453 9.000000000000533 9.999999999998995	0,000000000000562 0,000000000002370 0,000000000004430 0,000000000003760 0,000000000004940 0,000000000005600 0,000000000003570 0,000000000003450 0,000000000000529 0,000000000001011	87	6
0.9	0.999999999908651 1.999999999894297 2.999999999889636 3.999999999889492 4.999999999890677 5.999999999893980 6.999999999893352 7.999999999891046 8.999999999889519 9.999999999889434	0,000000000091349 0,000000000105710 0,000000000110370 0,000000000110510 0,000000000109330 0,000000000106020 0,000000000106650 0,000000000108960 0,000000000110489 0,000000000110569	9924	48	0.84	0.999999999999450 2.000000000002249 2.999999999995760 4.000000000003571 4.999999999995277 6.000000000005323 6.999999999996578 8.000000000003276 9.000000000000488 9.999999999999023	0,000000000000550 0,000000000002240 0,000000000004240 0,000000000003570 0,000000000004730 0,000000000005320 0,000000000003430 0,000000000003270 0,000000000000480 0,000000000000981	86	6
1.0	0.999999999908809 1.999999999894480 2.999999999889827 3.999999999889683 4.999999999890867 5.999999999894164 6.999999999893538 7.999999999891235 8.999999999889710 9.999999999889626	0,000000000091191 0,000000000105520 0,000000000110180 0,000000000110320 0,000000000109140 0,000000000105840 0,000000000106470 0,000000000108770 0,000000000110290 0,000000000110379	8931	48	0.85	0.999999999999441 2.000000000001475 2.999999999996696 4.000000000002441 4.999999999996343 6.000000000003734 6.999999999997304 8.000000000002226 9.000000000000174 9.999999999999096	0,000000000000559 0,000000000001470 0,000000000003310 0,000000000002440 0,000000000003660 0,000000000003730 0,000000000002700 0,000000000002220 0,000000000000171 0,000000000000909	88	6
1.14	0.999999999908944 1.999999999894636 2.999999999889989 3.999999999889847 4.999999999891028 5.999999999894318 6.999999999893696 7.999999999891397	0,000000000091056 0,000000000105370 0,000000000110020 0,000000000110160 0,000000000108980 0,000000000105690 0,000000000106310 0,000000000108610	7833	48	0.9	1.000000000000392 2.000000000000453 3.000000000000473 4.000000000000473 5.000000000000469 6.000000000000453 7.000000000000457 8.000000000000467	0,000000000000390 0,000000000000450 0,000000000000470 0,000000000000470 0,000000000000460 0,000000000000450 0,000000000000450 0,000000000000460	133	0

	8.999999999889873 9.999999999889789	0,000000000110131 0,000000000110219				9.000000000000474 10.000000000000474	0,000000000000471 0,000000000000400		
1.15	0.999999999908957 1.999999999894654 2.999999999890008 3.999999999889865 4.999999999891047 5.999999999894338 6.999999999893713 7.999999999891415 8.999999999889891 9.999999999889807	0,000000000091043 0,000000000105350 0,000000000110000 0,000000000110140 0,000000000108960 0,000000000105670 0,000000000106290 0,000000000108590 0,000000000110109 0,000000000110200	7765	48	1.0	0.999999999999562 1.999999999999493 2.999999999999472 3.999999999999472 4.999999999999477 5.999999999999491 6.999999999999488 7.999999999999479 8.999999999999471 9.999999999999471	0,000000000000438 0,000000000000510 0,000000000000530 0,000000000000530 0,000000000000530 0,000000000000510 0,000000000000520 0,000000000000530 0,000000000000529 0,000000000000529	10874	0
1.16	-1656867222756877379118126613547504710914184903323692805652480.000000000000000 6993480948271112215264421015239305573324424204991828957593600.000000000000000 -1303639865558288742497295727197602615984994171166677733998592.000000000000000 1107846122570978688685898394131662188912598446406869234325913600.000000000000000 -1452912056039865438436216866509233881766077361662227212582912.000000000000000 1648093845717776712289700104939435237165219458724782176993280.000000000000000 -1051418057672219047112716184666854572177336116377309465804800.000000000000000 1015908614300947410141870620836403783964424434232973796048896.000000000000000 1560869852737431271062701603028914271337379423696317736550400.000000000000000 -2963905069438828013193973318889482283090616511202069619671040.000000000000000		100000	0	1.1	-799045666911135943999 2627546276440791024475592429311343348752449 2945874050866596044012295197681175120764556 4437999024142075325462067120938541601106862 7874144567545477340246666398448924204924698079290830122059340238746546117213899945449512212269577842248273318217666247085570593493766159824466969576381632.000000000000000 -924599735091685746322111376333232799192242150366937892566112749309601431810574282811870551940596492368802136 745585701996275360046109067914187791148549094545725526069279876 948687894980199404537376854453517705401296 926726517414429191460 010763851396483819151 277164080774324869571 554692695887302994084 038180864.000000000000000 -inf -nan(ind) -inf -9273720281922885194491495405685650877368968482482926139891174257 7189673708202199581011 094331247660020808637 057105122709866773726 941057033636480073165 961683173705453602087 899842794567485527647 807265693333083582367 4783617487426721150931 013603859454890182291 366653716772581941602 521810917176621431836 1547898880.000000000000000 -932855238915755908370244197558132238677611704154660778945926330831456318884995748144 785384263184265215813 065779233718322063791141619989690294103237241376528253623521463 491902841475360066920 464853698759255976558 964281016639942100542 1691133808113366302157 202974181022498275067		3922	-2147483648

						645248569014979834565 5743152128.00000000000 0000 -953026702585374715212 717227819108834848690 167815446269834986209 899083572551445062827 992612805034567553925 186305324732197767882 338614203509781013067 380960734538977453705 2115795113998048315889 513253818413635421445 321513373348351714806 764173864462214019206 057368219802196695891 453363891842433258189 682978586624.000000000 000000 -inf -inf			

Метод Гаусса-Зейделя:

Исходная матрица					**Матрица с обратными знаками**				
ω	x	$x - x^*$	max i	$v(A)$	ω	x	$x - x^*$	max i	$v(A)$
0.0	0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000	1.000000000000000 2.000000000000000 3.000000000000000 4.000000000000000 5.000000000000000 6.000000000000000 7.000000000000000 8.000000000000000 9.000000000000000 10.000000000000000	100000	1	0.0	0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000 0.000000000000000	1.000000000000000 2.000000000000000 3.000000000000000 4.000000000000000 5.000000000000000 6.000000000000000 7.000000000000000 8.000000000000000 9.000000000000000 10.000000000000000	100000	1
0.1	0.999999999909138 1.999999999894855 2.999999999890221 3.999999999890091 4.999999999891280 5.999999999894576 6.999999999893930 7.999999999891640 8.999999999890125 9.999999999890051	0,000000000090862 0,000000000105150 0,000000000109780 0,000000000109910 0,000000000108720 0,000000000105430 0,000000000106070 0,000000000108360 0,000000000109880 0,000000000109949	84997	46	0.1	0.999999999999490 2.000000000002641 2.999999999995115 4.000000000004089 4.999999999994769 6.000000000005828 6.999999999996085 8.000000000003730 9.000000000000622 9.999999999998861	0,000000000000510 0,000000000002640 0,000000000004890 0,000000000004080 0,000000000005240 0,000000000005820 0,000000000003920 0,000000000003730 0,000000000000620 0,000000000001140	783	6
0.2	0.999999999910082 1.999999999895944 2.999999999891367 3.999999999891255 4.999999999892443 5.999999999895715 6.999999999895042 7.999999999892789 8.999999999891299 9.999999999891244	0,000000000089918 0,000000000104060 0,000000000108640 0,000000000108750 0,000000000107560 0,000000000104290 0,000000000104960 0,000000000107220 0,000000000108709 0,000000000108759	40282	44	0.2	0.999999999999603 2.000000000002737 2.999999999994986 4.000000000004126 4.999999999994867 6.000000000005602 6.999999999996003 8.000000000003745 9.000000000000680 9.999999999998799	0,000000000000397 0,000000000002730 0,000000000005020 0,000000000004120 0,000000000005140 0,000000000005600 0,000000000004000 0,000000000003739 0,000000000000680 0,000000000001210	374	6
0.3	0.999999999911195 1.999999999897227 2.999999999892712 3.999999999892619 4.999999999893806 5.999999999897050 6.999999999896353 7.999999999894137 8.999999999892674 9.999999999892633	0,000000000088805 0,000000000102780 0,000000000107290 0,000000000107390 0,000000000106200 0,000000000102950 0,000000000103650 0,000000000105870 0,000000000107329 0,000000000107370	25379	41	0.3	0.999999999999757 2.000000000002605 2.999999999995288 4.000000000003803 4.999999999995416 6.000000000004887 6.999999999996266 8.000000000003428 9.000000000000680 9.999999999998847	0,000000000000243 0,000000000002600 0,000000000004720 0,000000000003800 0,000000000004590 0,000000000004880 0,000000000003740 0,000000000003419 0,000000000000680 0,000000000001160	238	5
0.4	0.999999999912623 1.999999999898873 2.999999999894438 3.999999999894365 4.999999999895548 5.999999999898754 6.999999999898031 7.999999999895861 8.999999999894431 9.999999999894410	0,000000000087377 0,000000000101130 0,000000000105570 0,000000000105640 0,000000000104460 0,000000000101250 0,000000000101970 0,000000000104140 0,000000000105571 0,000000000105590	17929	39	0.4	0.999999999999905 2.000000000002613 2.999999999995346 4.000000000003671 4.999999999995738 6.000000000004421 6.999999999996330 8.000000000003290 9.000000000000716 9.999999999998836	0,000000000000095 0,000000000002610 0,000000000004660 0,000000000003670 0,000000000004270 0,000000000004420 0,000000000003670 0,000000000003290 0,000000000000711 0,000000000001171	169	5

0.5	0.999999999914426 1.999999999900953 2.999999999896617 3.999999999896568 4.999999999897742 5.999999999900896 6.999999999900147 7.999999999898034 8.999999999896646 9.999999999896643	0,000000000085574 0,000000000099050 0,000000000103390 0,000000000103440 0,000000000102260 0,000000000099110 0,000000000099860 0,000000000101970 0,000000000103361 0,000000000103361	13460	36	0.5	1.000000000000071 2.000000000002640 2.999999999995392 4.000000000003539 4.999999999996068 6.000000000003946 6.999999999996381 8.000000000003153 9.000000000000755 9.999999999998826	0,000000000000070 0,000000000002640 0,000000000004610 0,000000000003530 0,000000000003940 0,000000000003940 0,000000000003620 0,000000000003149 0,000000000000750 0,000000000001180	127	5
0.6	0.999999999916579 1.999999999903437 2.999999999899219 3.999999999899194 4.999999999900357 5.999999999903449 6.999999999902674 7.999999999900627 8.999999999899286 9.999999999899304	0,000000000083421 0,000000000096570 0,000000000100790 0,000000000100810 0,000000000099650 0,000000000096560 0,000000000097330 0,000000000099380 0,000000000100719 0,000000000100700	10481	34	0.6	1.000000000000287 2.000000000002953 2.999999999994982 4.000000000003729 4.999999999996074 6.000000000003784 6.999999999996066 8.000000000003304 9.000000000000878 9.999999999998703	0,000000000000280 0,000000000002950 0,000000000005020 0,000000000003720 0,000000000003930 0,000000000003780 0,000000000003940 0,000000000003300 0,000000000000870 0,000000000001300	98	5
0.7	0.999999999919303 1.999999999906582 2.999999999902511 3.999999999902514 4.999999999903658 5.999999999906668 6.999999999905868 7.999999999903903 8.999999999902620 9.999999999902661	0,000000000080697 0,000000000093420 0,000000000097490 0,000000000097490 0,000000000096350 0,000000000093340 0,000000000094140 0,000000000096100 0,000000000097380 0,000000000097341	8354	31	0.7	1.000000000000523 2.000000000002957 2.999999999995152 4.000000000003457 4.999999999996603 6.000000000003105 6.999999999996195 8.000000000003050 9.000000000000908 9.999999999998732	0,000000000000520 0,000000000002950 0,000000000004850 0,000000000003450 0,000000000003400 0,000000000003100 0,000000000003810 0,000000000003050 0,000000000000901 0,000000000001270	77	5
0.8	0.999999999922567 1.999999999910350 2.999999999906456 3.999999999906488 4.999999999907609 5.999999999910517 6.999999999909693 7.999999999907824 8.999999999906608 9.999999999906674	0,000000000077433 0,000000000089650 0,000000000093550 0,000000000093520 0,000000000092400 0,000000000089490 0,000000000090310 0,000000000092180 0,000000000093401 0,000000000093330	6759	29	0.8	1.000000000000853 2.000000000003162 2.999999999995070 4.000000000003326 4.999999999997034 6.000000000002515 6.999999999996109 8.000000000002926 9.000000000000993 9.999999999998705	0,000000000000850 0,000000000003160 0,000000000004930 0,000000000003320 0,000000000002970 0,000000000002510 0,000000000003900 0,000000000002920 0,000000000000989 0,000000000001300	60	4
0.9	0.999999999926269 1.999999999914625 2.999999999910929 3.999999999910993 4.999999999912085 5.999999999914879 6.999999999914031 7.999999999912269 8.999999999911129 9.999999999911221	0,000000000073731 0,000000000085380 0,000000000089080 0,000000000089010 0,000000000087920 0,000000000085130 0,000000000085970 0,000000000087740 0,000000000088880 0,000000000088781	5518	26	0.9	0.999999999999229 1.999999999997430 3.000000000003723 3.999999999997612 5.000000000001988 5.999999999998439 7.000000000002953 7.999999999997963 8.999999999999217 10.000000000000940	0,000000000000771 0,000000000002570 0,000000000003720 0,000000000002390 0,000000000001980 0,000000000001570 0,000000000002950 0,000000000002040 0,000000000000790 0,000000000000901	43	4
1.0	0.999999999931005 1.999999999920099 2.999999999916653 3.999999999916751 4.999999999917802 5.999999999920442 6.999999999919578 7.999999999917950 8.999999999916902 9.999999999917021	0,000000000068995 0,000000000079910 0,000000000083350 0,000000000083250 0,000000000082200 0,000000000079560 0,000000000080430 0,000000000082050 0,000000000083100 0,000000000082981	4526	24	1.0	0.999999999999073 1.999999999998438 3.000000000001903 3.999999999999027 5.000000000000432 5.999999999999841 7.000000000001642 7.999999999999084 8.999999999999527 10.000000000000492	0,000000000000927 0,000000000001570 0,000000000001900 0,000000000000980 0,000000000000430 0,000000000000160 0,000000000001640 0,000000000000920 0,000000000000480 0,000000000000400	38	3
1.1	0.999999999936101 1.999999999925986 2.999999999922811 3.999999999922945 4.999999999923951 5.999999999926427 6.999999999925543 7.999999999924062 8.999999999923110 9.999999999923260	0,000000000063899 0,000000000074020 0,000000000077190 0,000000000077060 0,000000000076050 0,000000000073580 0,000000000074460 0,000000000075940 0,000000000076890 0,000000000076740	3713	21	1.1	1.000000000001366 2.000000000001120 2.999999999999677 3.999999999999641 5.000000000000960 5.999999999999078 6.999999999999238 7.999999999999944 9.000000000000204 9.999999999999931	0,000000000001360 0,000000000001120 0,000000000000330 0,000000000000360 0,000000000000960 0,000000000000930 0,000000000000770 0,000000000000060 0,000000000000201 0,000000000000069	29	3

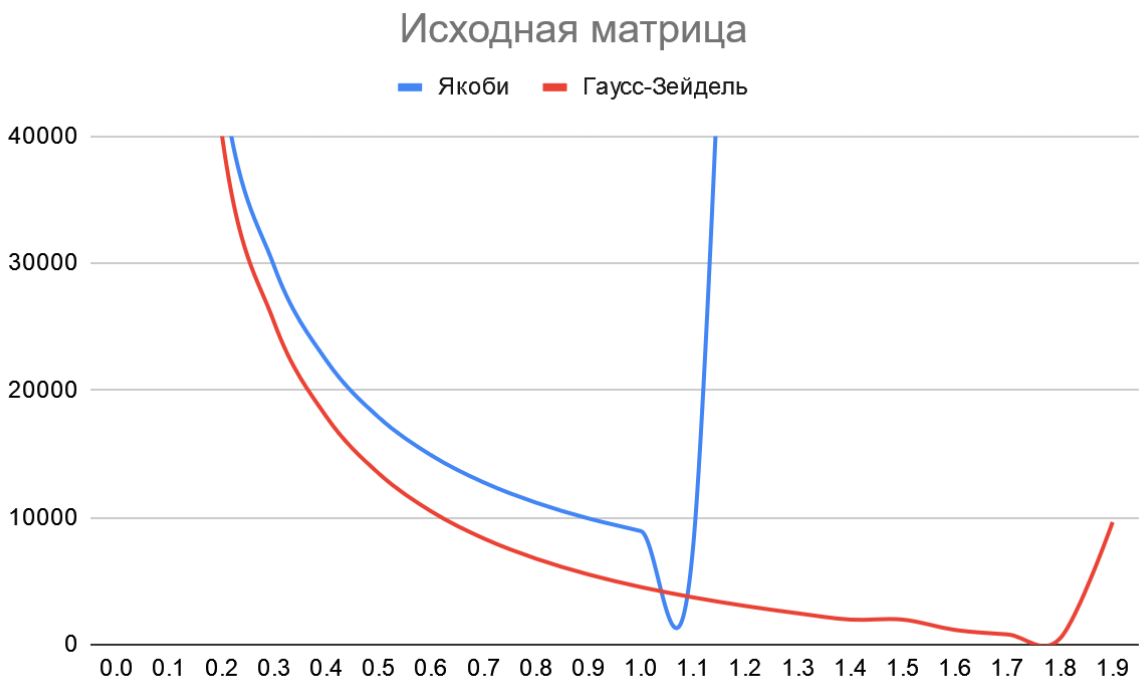
1.2	0.999999999942113 1.999999999932933 2.999999999930074 3.999999999930245 4.999999999931193 5.999999999933470 6.999999999932577 7.999999999931259 8.999999999930427 9.999999999930603	0,000000000057887 0,000000000067070 0,000000000069930 0,000000000069760 0,000000000068810 0,000000000066530 0,000000000067430 0,000000000068750 0,000000000069580 0,000000000069400	3035	19	1.11	1.000000000000741 1.999999999999016 3.000000000002448 3.999999999998073 5.000000000001982 5.999999999998539 7.000000000001309 7.999999999998766 8.999999999999583 10.000000000000544	0,000000000000740 0,000000000000990 0,000000000002440 0,000000000001930 0,000000000001980 0,000000000001470 0,000000000001300 0,000000000001240 0,000000000000419 0,000000000000499	28	3
1.3	0.999999999948901 1.999999999940780 2.999999999938278 3.999999999938483 4.999999999939361 5.999999999941411 6.999999999940518 7.999999999939387 8.999999999938680 9.999999999938884	0,000000000051099 0,000000000059220 0,000000000061730 0,000000000061520 0,000000000060640 0,000000000058590 0,000000000059490 0,000000000060620 0,000000000061320 0,000000000061121	2460	16	1.12	0.999999999998889 1.999999999997059 3.000000000003395 3.999999999998272 5.000000000000904 5.999999999999647 7.000000000002690 7.999999999998678 8.999999999999204 10.000000000000718	0,000000000001111 0,000000000002950 0,000000000003390 0,000000000001730 0,000000000000900 0,000000000000360 0,000000000002690 0,000000000001330 0,000000000000799 0,000000000000700	31	3
1.4	0.999999999956381 1.999999999949427 2.999999999947315 3.999999999947555 4.999999999948352 5.999999999950146 6.999999999949264 7.999999999948333 8.999999999947764 9.999999999947995	0,000000000043619 0,000000000050580 0,000000000052690 0,000000000052450 0,000000000051650 0,000000000049860 0,000000000050740 0,000000000051670 0,000000000052241 0,000000000052010	1965	14	1.2	0.999999999998374 2.000000000000689 3.000000000000177 4.000000000000083 5.000000000000294 6.000000000000201 6.999999999998886 8.000000000000103 8.999999999999705 10.000000000000057	0,000000000001626 0,000000000000680 0,000000000000170 0,000000000000080 0,000000000000290 0,000000000000200 0,000000000001120 0,000000000000099 0,000000000000300 0,000000000000057	37	1
1.5	0.999999999963968 1.999999999958201 2.999999999956482 3.999999999956758 4.999999999957472 5.999999999959006 6.999999999958136 7.999999999957410 8.999999999956977 9.999999999957234	0,000000000036032 0,000000000041800 0,000000000043520 0,000000000043250 0,000000000042530 0,000000000041000 0,000000000041870 0,000000000042590 0,000000000043030 0,000000000042769	1965	11	1.3	1.000000000002367 1.999999999998771 2.999999999999691 3.999999999999891 4.999999999999571 5.999999999999687 7.000000000002098 7.999999999999623 9.000000000000615 9.999999999999719	0,000000000002360 0,000000000001230 0,000000000000310 0,000000000000110 0,000000000000430 0,000000000000320 0,000000000002090 0,000000000000380 0,000000000000609 0,000000000000290	68	1
1.6	0.999999999972010 1.999999999967500 2.999999999966199 3.999999999966503 4.999999999967125 5.999999999968378 6.999999999967534 7.999999999967021 8.999999999966727 9.999999999967008	0,000000000027990 0,000000000032500 0,000000000033810 0,000000000033500 0,000000000032880 0,000000000031630 0,000000000032470 0,000000000032980 0,000000000033280 0,000000000032999	1148	9	1.4	1.000000000001958 1.999999999998841 2.999999999999706 3.999999999999913 4.999999999999639 5.999999999999734 7.000000000002098 7.999999999999490 9.000000000000677 9.999999999999577	0,000000000001950 0,000000000001160 0,000000000000300 0,000000000000090 0,000000000000370 0,000000000000270 0,000000000002090 0,000000000000510 0,000000000000670 0,000000000000430	210	1
1.7	0.999999999980293 1.999999999977087 2.999999999976217 3.999999999976545 4.999999999977066 5.999999999978026 6.999999999977219 7.999999999976919 8.999999999976772 9.999999999977065	0,000000000019707 0,000000000022920 0,000000000023790 0,000000000023460 0,000000000022940 0,000000000021980 0,000000000022790 0,000000000023090 0,000000000023229 0,000000000022940	800	6	1.5	-inf inf -nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind)		7083	-2147 48364 8
1.79	0.999999999987351 1.999999999985252 2.999999999984746 3.999999999985107 4.999999999985549 5.999999999986263 6.999999999985475 7.999999999985359 8.999999999985336 9.999999999985651	0,000000000012649 0,000000000014750 0,000000000015260 0,000000000014900 0,000000000014460 0,000000000013740 0,000000000014530 0,000000000014650 0,000000000014669 0,000000000014349	501	1					

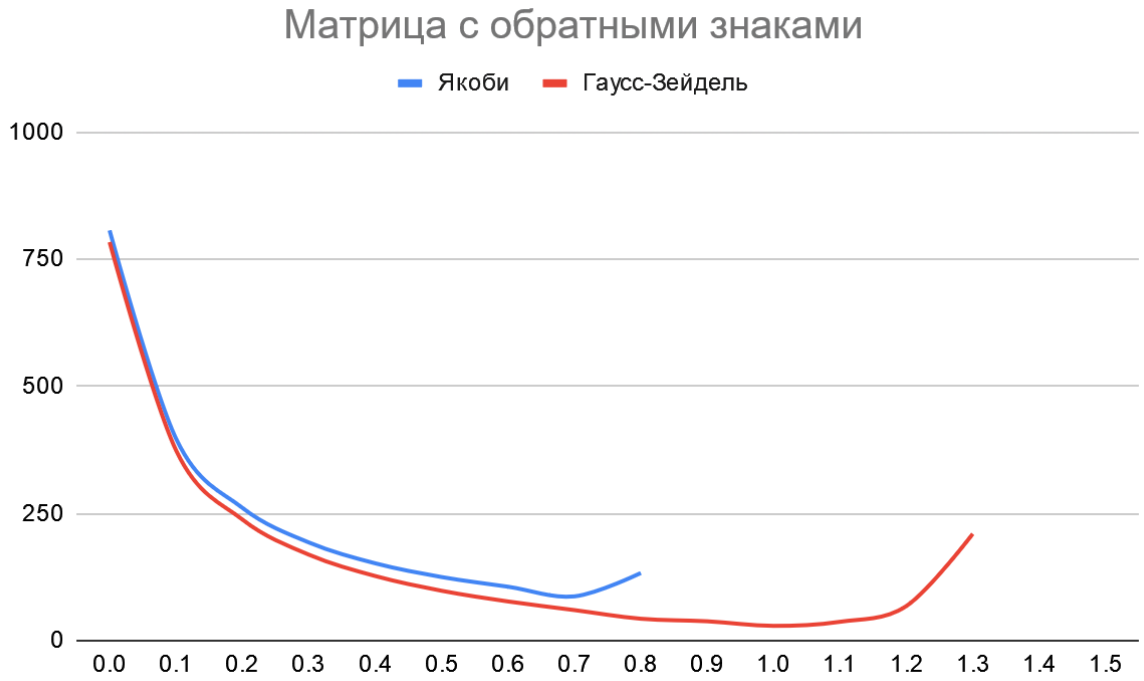
1.8	0.999999999996670 1.999999999996597 2.999999999996721 3.999999999996896 4.999999999996885 5.999999999997105 6.999999999996616 7.999999999996382 8.999999999996659 9.999999999997035	0,000000000003330 0,000000000003410 0,000000000003280 0,000000000003110 0,000000000003120 0,000000000002900 0,000000000003390 0,000000000003620 0,000000000003350 0,000000000002970	491	1					
1.81	0.999999999999334 1.999999999999799 3.000000000000187 4.000000000000234 5.000000000000069 6.000000000000123 6.999999999999713 7.999999999999408 8.999999999999911 10.000000000000368	0,000000000000666 0,000000000000210 0,000000000000180 0,000000000000230 0,000000000000060 0,000000000000120 0,000000000000290 0,000000000000600 0,000000000000091 0,000000000000300	633	0					
1.9	inf inf -nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind) -nan(ind)		9625	-2147 48364 8					

Вывод:

В ходе работы было выяснено, что скорость сходимости зависит от параметра релаксации, причем для разных матриц оптимальный параметр релаксации отличается. Для метода Якоби в случае с первой матрицей наилучшим весом оказался $\omega = 1.15$, для второй матрицы - $\omega = 0.85$. Первая матрица методом Гаусса-Зейделя наиболее быстро сходится при $\omega = 1.8$, вторая - при $\omega = 1.12$.
Следует также отметить, что в случае с методом Якоби наилучший вес оказался за пределами допустимых значений ($\omega = 1.15$ для первой матрицы), а метод Гаусса-Зейделя расходился при параметре, лежащим в допустимом интервале ($\omega = 1.5$для второй матрицы).

Исследуем более подробно зависимость количества итераций от параметра релаксации:





По графикам можно наблюдать, что имеется один ярко выраженный минимум, что позволяет более точно вычислить значение оптимального веса.

В ходе работы было оценено число обусловленности с помощью относительной невязки. Как и ожидалось, результат отличается от реального значения числа обусловленности. Итеративный метод оценки обусловленности позволяет вычислить

не само число обусловленности, а всего лишь его оценку снизу. Взглянув на формулу оценки числа обусловленности,

$$\nu_A \geq \frac{\|x - x^*\|}{\|x^*\|} / \frac{\|F - Ax\|}{\|F\|} .$$

становится понятно, что оно зависит от точности получаемого решения, причем чем точнее решение, тем ниже граница оценки. Поэтому в таблице мы наблюдаем, что для разного значения ω, мы получаем разное значение оценки $v(A)$. Также на точность оценки влияет и вычислительная ошибка.