



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное образовательное учреждение высшего образования  
**«Дальневосточный федеральный университет»**  
**(ДВФУ)**

---

**Институт математики и компьютерных технологий**

**Департамент математического и компьютерного  
моделирования**

Курсовой проект по дисциплине  
«Вычислительная математика»

на тему «Методы релаксации»

Направление подготовки  
01.03.02 «Прикладная математика и информатика»

Выполнил студент  
гр. Б9119-01.03.02 систпро  
Пищиков А.А.  
\_\_\_\_\_  
(ФИО) (подпись)

Проверил доцент, к.ф.-м.н.  
Колобов А.Г.  
\_\_\_\_\_  
(ФИО) (подпись)

« 8 » июня 20 22 г.

**г. Владивосток  
2022**

# Содержание

<b>Введение</b>	<b>3</b>
<b>Основная часть</b>	<b>4</b>
Постановка задачи . . . . .	4
Описание метода . . . . .	4
Вычислительный эксперимент . . . . .	5
Матрица размерности 2 . . . . .	5
Матрица размерности 3 . . . . .	6
Матрица размерности 4 . . . . .	7
Матрица размерности 5 . . . . .	8
<b>Заключение</b>	<b>10</b>
<b>Приложение</b>	<b>11</b>

# Введение

Объектом исследования являются численные методы решения задач линейной алгебры, а также программное обеспечение, реализующее эти методы. Цель работы – ознакомиться с численными методами решения систем линейных алгебраических уравнений, нахождения обратных матриц, решения проблемы собственных значений, решить предложенные типовые задачи, сформулировать выводы по полученным решениям, отметить достоинства и недостатки методов, сравнить удобство использования и эффективность работы каждой использованной программы, приобрести практические навыки и компетенции, а также опыт самостоятельной профессиональной деятельности, а именно:

- создать алгоритм решения поставленной задачи и реализовать его, протестировать программы;
- освоить теорию вычислительного эксперимента; современных компьютерных технологий;
- приобрести навыки представления итогов проделанной работы в виде отчета, оформленного в соответствии с имеющимися требованиями, с привлечением современных средств редактирования и печати.

Работа над курсовым проектом предполагает выполнение следующих задач:

- дальнейшее углубление теоретических знаний обучающихся и их систематизацию;
- получение и развитие прикладных умений и практических навыков по направлению подготовки;
- овладение методикой решения конкретных задач;
- развитие навыков самостоятельной работы;
- развитие навыков обработки полученных результатов, анализа и осмысления их с учетом имеющихся литературных данных;
- приобретение навыков оформления описаний программного продукта;
- повышение общей и профессиональной эрудиции.

Изученный студентом в ходе работы материал должен способство-

вать повышению его качества знаний, закреплению полученных навыков и уверенности в выборе путей будущего развития своих профессиональных способностей.

## Основная часть

### Постановка задачи

Теперь рассмотрим следующую задачу: пусть дана система линейных алгебраических уравнений (СЛАУ):

$$Ax = f,$$

где  $A$  – квадратная матрица размерности  $n$ ,  $f$  – вектор правой части,  $x$  – вектор неизвестных. Необходимо найти решение данной системы, т.е. вектор  $x$ , методом релаксации.

### Описание метода

Метод релаксации – итерационный метод решения СЛАУ. Суть данного метода заключается в том, чтобы вычислять элементы вектора приближения по следующей формуле:

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

где  $\omega$  – релаксационный параметр, до тех пор, пока не будет достигнута требуемая точность. После достижения заданной точности, будет получен вектор, являющийся приближённым решением СЛАУ. Известно, что данный метод сходится для релаксационного параметра  $\omega \in (0, 2)$ , если матрица симметрична и положительно определена.

В зависимости от значения  $\omega$  выделяют несколько разновидностей этого метода. При параметре  $\omega = 1$  данный метод называется методом Зейделя, при  $\omega < 1$  – методом нижней релаксации, а при  $\omega > 1$  – методом верхней релаксации.

## Вычислительный эксперимент

В рамках данного раздела будут рассмотрены СЛАУ, для которых будет найдено численное решение методом релаксации с разными значениями релаксационного параметра  $\omega$  и с разной точностью  $\varepsilon$ . Количество итераций решения будет указано в таблице. В качестве первого приближения для всех систем был выбран нулевой вектор. Точность будет определяться по длине вектора невязки.

### Матрица размерности 2

Дана система:

$$\begin{pmatrix} 3 & 4 \\ 4 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 7 \\ 6 \end{pmatrix}.$$

Решение, полученное аналитически с помощью точного метода решения СЛАУ:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 5 \\ -2 \end{pmatrix}.$$

Решим эту систему численно методом релаксации. Для разных  $\omega$  и  $\varepsilon$  получим следующее количество итераций:

	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$
$\omega = 0.5$	38	65	92
$\omega = 1$	12	21	29
$\omega = 1.5$	6	10	14

Таблица 1: Количество итераций при разных значениях параметра  $\omega$  и точности  $\varepsilon$  для системы размерности 2.

При  $\varepsilon = 0.001$  и  $\omega = 1.5$  численно было получено следующее решение СЛАУ:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4.999995 \\ -1.999944 \end{pmatrix}.$$

### Матрица размерности 3

Дана система:

$$\begin{pmatrix} 5 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 11 \\ 4 \\ -14 \end{pmatrix}.$$

Решение, полученное аналитически с помощью точного метода решения СЛАУ:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ -4 \end{pmatrix}.$$

Решим эту систему численно методом релаксации. Для разных  $\omega$  и  $\varepsilon$  получим следующее количество итераций:

	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$
$\omega = 0.5$	19	30	41
$\omega = 1$	8	10	13
$\omega = 1.5$	11	16	20

Таблица 2: Количество итераций при разных значениях параметра  $\omega$  и точности  $\varepsilon$  для системы размерности 3.

При  $\varepsilon = 0.001$  и  $\omega = 1$  численно было получено следующее решение СЛАУ:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3.000225 \\ 0.999884 \\ -3.999997 \end{pmatrix}.$$

## Матрица размерности 4

Дана система:

$$\begin{pmatrix} 8 & 1 & 3 & 1 \\ 1 & 5 & 0 & 2 \\ 3 & 0 & 6 & 4 \\ 1 & 2 & 4 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 29 \\ 106 \\ -18 \\ 34 \end{pmatrix};$$

Решение, полученное аналитически с помощью точного метода решения СЛАУ:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 18 \\ -9 \\ 6 \end{pmatrix}.$$

Решим эту систему численно методом релаксации. Для разных  $\omega$  и  $\varepsilon$  получим следующее количество итераций:

	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$
$\omega = 0.5$	50	82	113
$\omega = 1$	19	29	39
$\omega = 1.5$	11	14	18

Таблица 3: Количество итераций при разных значениях параметра  $\omega$  и точности  $\varepsilon$  для системы размерности 4.

При  $\varepsilon = 0.001$  и  $\omega = 1.5$  численно было получено следующее решение СЛАУ:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4.000116 \\ 17.999949 \\ -9.000097 \\ 6.000033 \end{pmatrix}.$$

## Матрица размерности 5

Дана система:

$$\begin{pmatrix} 10 & 1 & 5 & 1 & 1 \\ 1 & 7 & 2 & 13 & 2 \\ 5 & 2 & 5 & 5 & 1 \\ 1 & 13 & 5 & 37 & 0 \\ 1 & 2 & 1 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 55 \\ 8 \\ 17 \\ 57 \end{pmatrix};$$



Решение, полученное аналитически с помощью точного метода решения СЛАУ:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} -2 \\ 10 \\ 0 \\ -3 \\ 13 \end{pmatrix}.$$

Решим эту систему численно методом релаксации. Для разных  $\omega$  и  $\varepsilon$  получим следующее количество итераций:

	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$
$\omega = 0.5$	230	477	725
$\omega = 1$	53	134	216
$\omega = 1.5$	41	66	90

Таблица 4: Количество итераций при разных значениях параметра  $\omega$  и точности  $\varepsilon$  для системы размерности 5.

При  $\varepsilon = 0.001$  и  $\omega = 1.5$  численно было получено следующее решение СЛАУ:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} -2.000250 \\ 10.001379 \\ 0.000605 \\ -3.000542 \\ 12.998493 \end{pmatrix}.$$

## Заключение

В рамках курсового проекта был реализован и исследован на матрицах разной размерности численный метод решения СЛАУ, а конкретно – метод релаксации. Из полученных результатов можно сделать вывод о том, что релаксационный параметр  $\omega$  может влиять на скорость сходимости метода как положительно, так и отрицательно, причём значения релаксационного параметра могут оказывать большее влияние на скорость сходимости, чем изменения точности вычисления.

В результате работы над курсовым проектом приобрел практические навыки владения:

- современными численными методами решения задач линейной алгебры;
- основами алгоритмизации для численного решения задач линейной алгебры на одном из языков программирования;
- инструментальными средствами, поддерживающими разработку программного обеспечения для численного решения задач линейной алгебры;

а также навыками представления итогов проделанной работы в виде отчета, оформленного в соответствии с имеющимися требованиями, с привлечением современных средств редактирования и печати.

# Приложение

## Реализация численного алгоритма.

```
1 extern crate nalgebra as na;
2
3 use na::{DMatrix, DVector};
4
5 pub fn sor(a : DMatrix<f32>, f : DVector<f32>, n : usize,
6           omega: f32, epsilon : f32) -> DVector<f32> {
7     let mut xs = DVector::from_element(n, 0.0);
8     let mut xs_prev = xs.clone();
9
10    let mut it_counter = 0;
11
12    loop {
13        it_counter += 1;
14        for i in 0..n {
15            let mut sum1 = 0.0;
16            let mut sum2 = 0.0;
17            for k in 0..i {
18                sum1 += a[(i, k)] * xs[k];
19            }
20            for k in (i + 1)..n {
21                sum2 += a[(i, k)] * xs_prev[k];
22            }
23            xs[i] = omega * (f[i] - sum1 - sum2) / a[(i, i)]
24                    + (1.0 - omega) * xs_prev[i];
25        }
26
27        xs_prev = xs.clone();
28
29        if (&a * &xs - &f).norm() < epsilon {
30            break;
31        }
32    }
33
34    println!("x_{}={}", xs);
35
36    println!("omega_={}, epsilon_={}, iterations_={}\n",
37            omega, epsilon, it_counter);
38
39    return xs;
40 }
```

Код программы для решения СЛАУ размерности 2 методом релаксации для разных значений  $\omega$  и  $\varepsilon$ .

```
1 extern crate nalgebra as na;
2 use na::{DMatrix, DVector};
3
4 use crate::sor::sor;
5
6 pub fn sor2x2() {
7     let n = 2;
8     let matrix_row = [3.0, 4.0,
9                       4.0, 7.0];
10
11     let vector_row = [7.0, 6.0];
12
13
14     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
15     let f = DVector::from_row_slice(&vector_row);
16
17     let epsilon : f32 = 0.1;
18
19     let omega = 0.5;
20
21     sor(a, f, n, omega, epsilon);
22
23
24     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
25     let f = DVector::from_row_slice(&vector_row);
26
27     let omega = 1.0;
28
29     sor(a, f, n, omega, epsilon);
30
31
32     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
33     let f = DVector::from_row_slice(&vector_row);
34
35     let omega = 1.5;
36
37     sor(a, f, n, omega, epsilon);
38
39     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
40     let f = DVector::from_row_slice(&vector_row);
41
42     let epsilon : f32 = 0.01;
43
44     let omega = 0.5;
45
46     sor(a, f, n, omega, epsilon);
```

```

47
48
49     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
50     let f = DVector::from_row_slice(&vector_row);
51
52     let omega = 1.0;
53
54     sor(a, f, n, omega, epsilon);
55
56
57     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
58     let f = DVector::from_row_slice(&vector_row);
59
60     let omega = 1.5;
61
62     sor(a, f, n, omega, epsilon);
63
64     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
65     let f = DVector::from_row_slice(&vector_row);
66
67     let epsilon : f32 = 0.001;
68
69     let omega = 0.5;
70
71     sor(a, f, n, omega, epsilon);
72
73
74     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
75     let f = DVector::from_row_slice(&vector_row);
76
77     let omega = 1.0;
78
79     sor(a, f, n, omega, epsilon);
80
81
82     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
83     let f = DVector::from_row_slice(&vector_row);
84
85     let omega = 1.5;
86
87     sor(a, f, n, omega, epsilon);
88 }

```

Код программы для решения СЛАУ размерности 3 методом релаксации для разных значений  $\omega$  и  $\varepsilon$ .

```
1 extern crate nalgebra as na;
2 use na::{DMatrix, DVector};
3
4 use crate::sor::sor;
5
6 pub fn sor3x3() {
7     let n = 3;
8     let matrix_row = [5.0, 4.0, 2.0,
9                       4.0, 8.0, 4.0,
10                      2.0, 4.0, 6.0];
11
12     let vector_row = [11.0, 4.0, -14.0];
13
14
15     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
16     let f = DVector::from_row_slice(&vector_row);
17
18     let epsilon : f32 = 0.1;
19
20     let omega = 0.5;
21
22     sor(a, f, n, omega, epsilon);
23
24
25     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
26     let f = DVector::from_row_slice(&vector_row);
27
28     let omega = 1.0;
29
30     sor(a, f, n, omega, epsilon);
31
32
33     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
34     let f = DVector::from_row_slice(&vector_row);
35
36     let omega = 1.5;
37
38     sor(a, f, n, omega, epsilon);
39
40     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
41     let f = DVector::from_row_slice(&vector_row);
42
43     let epsilon : f32 = 0.01;
44
45     let omega = 0.5;
46
```

```

47     sor(a, f, n, omega, epsilon);
48
49
50     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
51     let f = DVector::from_row_slice(&vector_row);
52
53     let omega = 1.0;
54
55     sor(a, f, n, omega, epsilon);
56
57
58     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
59     let f = DVector::from_row_slice(&vector_row);
60
61     let omega = 1.5;
62
63     sor(a, f, n, omega, epsilon);
64
65     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
66     let f = DVector::from_row_slice(&vector_row);
67
68     let epsilon : f32 = 0.001;
69
70     let omega = 0.5;
71
72     sor(a, f, n, omega, epsilon);
73
74
75     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
76     let f = DVector::from_row_slice(&vector_row);
77
78     let omega = 1.0;
79
80     sor(a, f, n, omega, epsilon);
81
82
83     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
84     let f = DVector::from_row_slice(&vector_row);
85
86     let omega = 1.5;
87
88     sor(a, f, n, omega, epsilon);
89 }

```

Код программы для решения СЛАУ размерности 4 методом релаксации для разных значений  $\omega$  и  $\varepsilon$ .

```
1 extern crate nalgebra as na;
2 use na::{DMatrix, DVector};
3
4 use crate::sor::sor;
5
6 pub fn sor4x4() {
7     let n = 4;
8     let matrix_row = [8.0, 1.0, 3.0, 1.0,
9                       1.0, 5.0, 0.0, 2.0,
10                      3.0, 0.0, 6.0, 4.0,
11                      1.0, 2.0, 4.0, 5.0];
12
13     let vector_row = [29.0, 106.0, -18.0, 34.0];
14
15
16     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
17     let f = DVector::from_row_slice(&vector_row);
18
19     let epsilon : f32 = 0.1;
20
21     let omega = 0.5;
22
23     sor(a, f, n, omega, epsilon);
24
25
26     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
27     let f = DVector::from_row_slice(&vector_row);
28
29     let omega = 1.0;
30
31     sor(a, f, n, omega, epsilon);
32
33
34     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
35     let f = DVector::from_row_slice(&vector_row);
36
37     let omega = 1.5;
38
39     sor(a, f, n, omega, epsilon);
40
41     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
42     let f = DVector::from_row_slice(&vector_row);
43
44     let epsilon : f32 = 0.01;
45
46     let omega = 0.5;
```



```

47
48     sor(a, f, n, omega, epsilon);
49
50
51     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
52     let f = DVector::from_row_slice(&vector_row);
53
54     let omega = 1.0;
55
56     sor(a, f, n, omega, epsilon);
57
58
59     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
60     let f = DVector::from_row_slice(&vector_row);
61
62     let omega = 1.5;
63
64     sor(a, f, n, omega, epsilon);
65
66     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
67     let f = DVector::from_row_slice(&vector_row);
68
69     let epsilon : f32 = 0.001;
70
71     let omega = 0.5;
72
73     sor(a, f, n, omega, epsilon);
74
75
76     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
77     let f = DVector::from_row_slice(&vector_row);
78
79     let omega = 1.0;
80
81     sor(a, f, n, omega, epsilon);
82
83
84     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
85     let f = DVector::from_row_slice(&vector_row);
86
87     let omega = 1.5;
88
89     sor(a, f, n, omega, epsilon);
90 }

```

Код программы для решения СЛАУ размерности 5 методом релаксации для разных значений  $\omega$  и  $\varepsilon$ .

```
1 extern crate nalgebra as na;
2 use na::{DMatrix, DVector};
3
4 use crate::sor::sor;
5
6 pub fn sor5x5() {
7     let n = 5;
8     let matrix_row = [10.0, 1.0, 5.0, 1.0, 1.0,
9                       1.0, 7.0, 2.0, 13.0, 2.0,
10                      5.0, 2.0, 5.0, 5.0, 1.0,
11                      1.0, 13.0, 5.0, 37.0, 0.0,
12                      1.0, 2.0, 1.0, 0.0, 2.0];
13
14     let vector_row = [0.0, 55.0, 8.0, 17.0, 44.0];
15
16
17     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
18     let f = DVector::from_row_slice(&vector_row);
19
20     let epsilon : f32 = 0.1;
21
22     let omega = 0.5;
23
24     sor(a, f, n, omega, epsilon);
25
26
27     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
28     let f = DVector::from_row_slice(&vector_row);
29
30     let omega = 1.0;
31
32     sor(a, f, n, omega, epsilon);
33
34
35     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
36     let f = DVector::from_row_slice(&vector_row);
37
38     let omega = 1.5;
39
40     sor(a, f, n, omega, epsilon);
41
42     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
43     let f = DVector::from_row_slice(&vector_row);
44
45     let epsilon : f32 = 0.01;
46 }
```

```

47     let omega = 0.5;
48
49     sor(a, f, n, omega, epsilon);
50
51
52     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
53     let f = DVector::from_row_slice(&vector_row);
54
55     let omega = 1.0;
56
57     sor(a, f, n, omega, epsilon);
58
59
60     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
61     let f = DVector::from_row_slice(&vector_row);
62
63     let omega = 1.5;
64
65     sor(a, f, n, omega, epsilon);
66
67     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
68     let f = DVector::from_row_slice(&vector_row);
69
70     let epsilon : f32 = 0.001;
71
72     let omega = 0.5;
73
74     sor(a, f, n, omega, epsilon);
75
76
77     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
78     let f = DVector::from_row_slice(&vector_row);
79
80     let omega = 1.0;
81
82     sor(a, f, n, omega, epsilon);
83
84
85     let a : DMatrix<f32> = DMatrix::from_row_slice(n, n, &matrix_row);
86     let f = DVector::from_row_slice(&vector_row);
87
88     let omega = 1.5;
89
90     sor(a, f, n, omega, epsilon);
91 }

```

Код программы, запускающей вычислительный эксперимент.

```
1 mod sor;
2
3 mod sor2x2;
4 use crate::sor2x2::sor2x2;
5
6 mod sor3x3;
7 use crate::sor3x3::sor3x3;
8
9 mod sor4x4;
10 use crate::sor4x4::sor4x4;
11
12 mod sor5x5;
13 use crate::sor5x5::sor5x5;
14
15 fn main() {
16     sor2x2();
17     sor3x3();
18     sor4x4();
19     sor5x5();
20 }
```