МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и прикладной математики

Кафедра прикладной математики и экономико-математических методов

КУРСОВАЯ РАБОТА

по дисциплине:

«Численные методы»

Тема: «Реализация и сравнение стационарных итерационных методов решения СЛАУ»

Направление 01.03.02 «Прикладная Математика и Информатика»
Направленность « Прикладная математика и информатика в экономике и управлении»
Обучающийся Титилин Александр Михайлович
Группа ПМ-2201 Подпись _____

Проверила Соловьёва Наталья Анатольевна
Должность к.ф.-м.н., доцент

Оценка:_____ Дата: _____

Подпись: _____

Санкт-Петербург
2024 г.

Содержание

ВВЕДЕНИЕ **3**

1 ИТЕРАЦИОННЫЕ СТАЦИОНАРНЫЕ ОДНОШАГОВЫЕ МЕТОДЫ РЕШЕНИЯ СЛАУ **4**

1.1 Итерационный метод Якоби 4

1.2 Итерационный метод Гаусса-Зейделя 5

1.3 Методы релаксации 6

1.4 Метод Ричардсона 6

2 СРАВНЕНИЕ АЛГОРИТМОВ **8**

2.1 Особенности реализации алгоритмов 8

2.2 Генерация случайных матриц 8

2.3 Сравнение алгоритмов решения СЛАУ с матрицей, имеющей диагональное преобладание 9

2.3.1 Сравнение времени работы алгоритмов 9

2.3.2 Сравнение точности 30

2.4 Сравнение алгоритмов решения СЛАУ с симметричной и положительно определенной матрицей 31

2.4.1 Сравнение времени работы алгоритмов на случайных симметричных положительно определенных матрицах 31

2.4.2 Сравнение точности 36

2.4.3 Сравнение времени работы алгоритмов на матрицах Гилберта 36

2.4.4 Сравнение точности 39

ЗАКЛЮЧЕНИЕ **40**

БИБЛИОГРАФИЧЕСКИЙ СПИСОК **41**

ВВЕДЕНИЕ

Целью курсовой работы является реализация итерационных алгоритмов решения систем линейных уравнений с использованием языка программирования Python.

Задачи курсовой работы:

1) реализация алгоритмов решения СЛАУ с помощью итерационных методов;
2) создание алгоритмов генерации случайных матриц с необходимыми свойствами;
3) сравнение времени работы и точности алгоритмов между собой и с прямыми методами;
4) визуализация сравнений алгоритмов.

1. ИТЕРАЦИОННЫЕ СТАЦИОНАРНЫЕ ОДНОШАГОВЫЕ МЕТОДЫ РЕШЕНИЯ СЛАУ

Пусть дана СЛАУ вида $Ax = b$, где A квадратная невырожденная матрица порядка n, b столбец длины n, x неизвестный столбец.

Обозначим $x^{(i)}$ i-е приближение к точному решению СЛАУ. T – оператор перехода, отображение, которое переводит приближение в следующее

$$x^{(i+1)} \leftarrow T(x^{(i)}).$$

Все рассмотренные далее методы будут реализовывать процесс последовательного применения оператора перехода, начиная с некоторого начального приближения.

1.1 Итерационный метод Якоби

Рассмотрим данный оператор перехода

$$T(x^{(k)}) = \begin{pmatrix} T_1(x^{(k)}) \\ T_2(x^{(k)}) \\ \vdots \\ T_n(x^{(k)}) \end{pmatrix},$$

где

$$T_i(x^k) = \frac{1}{a_{ii}}(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}), i = 1 \dots n.$$

В листинге 1 представлена реализация данного метода на языке программирования Python.

```
def jacobi_method(A,b,eps=0.001,exit_param=2):
    res = np.zeros(len(b))
    last = np.zeros(len(b))
    k = 0
    while k == 0 or np.linalg.norm(res - last) > eps:
        last = np.copy(res)
        for i in range(len(b)):
```

```
            a = A[i]
            res[i] =  (1/a[i]) * (b[i] - (np.delete(a,i)*np.delete(last,i)).sum())
        k+=1
        if k >= len(b)*exit_param:
            break
    return (res,k)
```

Листинг 1: Реализация метода Якоби.

Метод Якоби сходится при любом начальном приближении, если матрица A в СЛАУ $Ax = b$ имеет диагональное преобладание [1].

1.2 Итерационный метод Гаусса-Зейделя

Рассмотрим оператор перехода

$$T_i(x^{(k)}) = \frac{1}{a_{11}}(b_i - \sum_{j=1}^{i-1} a_{ii}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ii}x_j^{(k)}).$$

В листинге 2 представлена реализация данного метода.

```
def gauss_seildel_method(A,b,eps=0.001,exit_param=2):
    res = np.zeros(b.size)
    last = np.zeros(b.size)
    k = 0
    while np.linalg.norm(res - last) > eps or k == 0:
        last = np.copy(res)
        for i in range(len(b)):
            a = A[i]
            res[i] = (1/a[i]) * (b[i] - (np.delete(a,i) * np.delete(res,i)).sum())
        k+=1
        if k > exit_param*res.size* -np.log10(eps)*exit_param:
            break
    return (res,k)
```

Листинг 2: Реализация метода Гаусса-Зейделя.

Метод Гаусса-Зейделя исправляет недостаток метода Якоби, который не использовал уже вычисленные значения приближения $x^{(k+1)}$.

Метод Гаусса-Зейделя сходится из любого начального приближения, если матрица A в СЛАУ $Ax = b$ имеет диагональное преобладание или является симметричной и положительно определенной [1].

1.3 Методы релаксации

Рассмотрим оператор перехода

$$T_i(x^{(k)}) = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}(b_i - \sum_{j=1}^{i-1})(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1} a_{ij}x_j^{(k)}), \text{ где } \omega \in \mathbb{R}.$$

Число ω называется параметром релаксации. В листинге 3 представлена реализация данного метода.

```
def relaxation_method(A,b,eps=0.001,omega=0.5,exit_param=2):
    res = np.zeros(len(b))
    last = np.zeros(len(b))
    k = 0
    while np.linalg.norm(res - last) > eps or k == 0:
        last = np.copy(res)
        for i in range(len(b)) :
            a = A[i]
            res[i] = (1- omega) * res[i] + (omega/a[i]) * (b[i] - (np.delete(a,i)*
                np.delete(res,i)).sum())
        k+=1
        if k > len(b)*exit_param*-np.log10(eps):
            break
    return (res,k)
```

Листинг 3: Реализация метода релаксации

Для сходимости метода релаксации должно выполнятся неравенство $0 < \omega < 2$. Методы релаксации сходятся из любого начального приближения, если матрица A в СЛАУ $Ax = b$ положительно определенная [1].

1.4 Метод Ричардсона

Рассмотрим оператор перехода

$$T(x^{(k)}) = (E - \tau A)x^{(k)} + \tau b, \text{ где } \tau = \text{const}, \tau \neq 0.$$

В листинге 4 представлена реализация данного метода.

```
def richardson_method(A,b,eps=0.001,exit_param=2):
    eig = np.linalg.eigvals(A).astype("float64")
```

```
    tau = 2/(max(eig) + min(eig))
    res = np.zeros(len(b))
    last = np.zeros(len(b))
    k = 0
    while np.linalg.norm(res - last) > eps or k== 0:
        last= np.copy(res)
        res = (np.eye(len(b)) - tau*A).dot(res) + tau*b
        k+=1
        if k > exit_param*len(b)*-np.log10(eps):
            break
    return (res,k)
```

Листинг 4: Реализация метода Ричарсона

Если в СЛАУ $Ax = b$ матрица A симметрична, положительно определена , то последовательность, порожденная методом Ричардсона с параметром $\tau = \frac{2}{M+\mu}, M > 0, \mu > 0$, где M, μ границы спектра матрицы A, сходится из любого начального приближения [1].

2. СРАВНЕНИЕ АЛГОРИТМОВ

2.1 Особенности реализации алгоритмов

У выбранного языка программирования есть ряд недостатков. Он очень медленный в математических задачах и не имеет встроенной поддержки матриц и функций линейной алгебры. Для решения этих проблем используются сторонние библиотеки.

Список сторонних библиотек:

1) numpy – содержит реализации векторов и матриц [2];
2) scipy – содержит реализации функций линейной алгебры [3];
3) numba – реализует компиляцию, что необходимо для ускорения вычислений [4];
4) pandas – содержит реализацию таблиц, что необходимо для визуализации результатов сравнений [5].

2.2 Генерация случайных матриц

Для проведения сравнений необходимы матрицы, состоящие из случайных чисел. Данные матрицы должны иметь диагональное преобладание или быть симметричными и положительно определенными.

Рассмотрим функции для генерации случайной матрицы с диагональным преобладанием порядка n, представленный в листинге 5. Данный алгоритм достаточно быстро создает матрицы, итерационные методы в применении к данным матрицам быстро сходятся.

```
def generate_dominate_row(n,i):
    x = np.random.randint(10,max(100,n**2))
    nums = []
    s = 0
    m = x//2
    for _ in range(n-1):
        num = randint(-50,m)
```

```
        if s +abs(num) < x:
            nums.append(num)
            s+= abs(num)
            m-=1
        else:
            break
    nums.extend([0]* ((n-1) - len(nums)))
    nums.insert(i,x)
    return np.array(nums)

def generate_diagonal_dominate_matrix(n:int):
    return np.array([generate_dominate_row(n,i) for i in range(n)])
```

Листинг 5: Генерация случайных матриц с диагональным преобладанием.

Рассмотрим функцию для генерации случайной симметричной и положительно определенной матрицы, представленную в листинге 6. Данный алгоритм начинает работать медленно на матрицах порядка больше 1000, но более быстрые алгоритмы создания таких матриц, создают матрицы, итерационные методы в применении к которым сходятся более медленно.

```
def generate_sym_pos_matrix(n):
    nums = np.random.choice(np.arange(2*n),n,replace=False)
    vecs = np.random.randint(100,size=(n,n))
    A = sum(k*v.reshape((n,1))@v.reshape((1,n)) for k,v in zip(nums,vecs))
    return A
```

Листинг 6: Генерация случайных симметричных и положительно определенных матриц.

2.3 Сравнение алгоритмов решения СЛАУ с матрицей, имеющей диагональное преобладание

2.3.1 Сравнение времени работы алгоритмов

Алгоритм сравнения алгоритмов:

1) создаются случайные квадратные матрицы, имеющие диагональное преобладание, порядка $100, 200, \ldots, 5100$;

2) создаются случайные векторы размера $100, 200, \ldots, 5100$;

3) решаются СЛАУ точным методом, запоминаются решения и затраченное время;

4) СЛАУ решаются методом Якоби, Гаусса-Зейделя и релаксации с параметром $\omega = 0.3, 0.5, 1.2$, с компиляцией и без, запоминается затраченное время;

Как можно заметить из таблиц 1 – 10, самым эффективным является метод Гаусса-Зейделя, для которого компиляция ускоряет код в 4.5 раза.

Таблица 1: Результаты работы метода Якоби без компиляции

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	52	0.0443694591522	-0.0417437553406
200	78	0.1281049251556	-0.1263153553009
300	130	0.3254456520081	-0.3212807178497
400	76	0.2703664302826	-0.2578797340393
500	126	0.6120841503143	-0.5879716873169
600	99	0.5401477813721	-0.5048050880432
700	97	0.6299719810486	-0.5812337398529
800	69	0.5371758937836	-0.4641580581665
900	69	0.6227357387543	-0.5164468288422
1000	44	0.4429130554199	-0.3038856983185
1100	80	0.8988883495331	-0.6883490085602
1200	67	0.8679387569427	-0.6290333271027
1300	142	2.0736641883850	-1.7661986351013
1400	59	0.9120345115662	-0.5189483165741
1500	99	1.6660103797913	-1.1691880226135
1600	44	0.8478009700775	-0.2669956684113

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
1700	96	2.0153887271881	-1.3348014354706
1800	63	1.3615150451660	-0.5486998558044
1900	77	1.8149716854095	-0.8589713573456
2000	78	1.9024038314819	-0.6513102054596
2100	51	1.3669714927673	-0.0047633647919
2200	96	2.7012000083923	-1.2116370201111
2300	66	1.9693181514740	-0.2522995471954
2400	80	2.5349364280701	-0.6316859722137
2500	56	1.7136464118958	0.4483952522278
2600	104	3.7057044506073	-1.2810068130493
2700	58	1.9996554851532	0.7516820430756
2800	46	1.7790718078613	1.3243298530579
2900	90	3.4239289760590	0.1136860847473
3000	64	2.7354006767273	1.8070988655090
3100	75	3.0655438899994	1.6953222751617
3200	53	2.4814045429230	2.6817638874054
3300	52	2.3263633251190	3.4485423564911
3400	71	3.7041838169098	2.5708866119385
3500	75	3.6399040222168	4.0877647399902
3600	101	5.5140559673309	5.1446208953857
3700	77	4.0308992862701	4.3328595161438
3800	97	5.7853453159332	3.8730812072754
3900	72	4.2248957157135	9.8103401660919
4000	136	8.5632438659668	6.2643780708313
4100	147	10.1884658336639	1.3673396110535
4200	70	5.2402031421661	6.8190603256226
4300	98	7.2523105144501	5.8085365295410
4400	53	3.8044888973236	10.1782982349396
4500	133	9.8222827911377	5.1262011528015

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
4600	237	19.1437103748322	-3.3280611038208
4700	62	4.8871183395386	12.3196654319763
4800	144	12.4125111103058	5.6006090641022
4900	57	4.7494163513184	14.3763835430145
5000	63	5.8028347492218	14.4247493743896
5100	39	3.4337244033813	18.8006572723389

Таблица 2: Результаты работы метода Якоби с компиляцией

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	52	1.2462661266327	-1.2436404228210
200	78	0.0846045017242	-0.0828149318695
300	130	0.1750047206879	-0.1708397865295
400	76	0.1483836174011	-0.1358969211578
500	126	0.2856369018555	-0.2615244388580
600	99	0.2789611816406	-0.2436184883118
700	97	0.3174338340759	-0.2686955928802
800	69	0.2538394927979	-0.1808216571808
900	69	0.2932667732239	-0.1869778633118
1000	44	0.2109968662262	-0.0719695091248
1100	80	0.4322943687439	-0.2217550277710
1200	67	0.3959450721741	-0.1570396423340
1300	142	0.9175875186920	-0.6101219654083
1400	59	0.4157123565674	-0.0226261615753
1500	99	0.7937390804291	-0.2969167232513
1600	44	0.3854153156281	0.1953899860382

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
1700	96	0.8642973899841	-0.1837100982666
1800	63	0.6015193462372	0.2112958431244
1900	77	0.7820498943329	0.1739504337311
2000	78	0.8397419452667	0.4113516807556
2100	51	0.5938351154327	0.7683730125427
2200	96	1.1946210861206	0.2949419021606
2300	66	0.9122271537781	0.8047914505005
2400	80	1.1049659252167	0.7982845306396
2500	56	0.8440186977386	1.3180229663849
2600	104	1.6135838031769	0.8111138343811
2700	58	0.9647972583771	1.7865402698517
2800	46	0.8365287780762	2.2668728828430
2900	90	1.8695499897003	1.6680650711060
3000	64	1.4069402217865	3.1355593204498
3100	75	1.7986004352570	2.9622657299042
3200	53	1.3548674583435	3.8083009719849
3300	52	1.2982103824615	4.4766952991486
3400	71	1.8741035461426	4.4009668827057
3500	75	2.0641407966614	5.6635279655457
3600	101	2.8348221778870	7.8238546848297
3700	77	2.2926752567291	6.0710835456848
3800	97	2.9539735317230	6.7044529914856
3900	72	2.2821326255798	11.7531032562256
4000	136	4.3838644027710	10.4437575340271
4100	147	5.0060496330261	6.5497558116913
4200	70	2.4042267799377	9.6550366878510
4300	98	3.6198911666870	9.4409558773041
4400	53	1.9632833003998	12.0195038318634
4500	133	5.0482232570648	9.9002606868744

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
4600	237	9.3684167861938	6.4472324848175
4700	62	2.6154508590698	14.5913329124451
4800	144	5.9540500640869	12.0590701103210
4900	57	2.4468579292297	16.6789419651031
5000	63	2.8342251777649	17.3933589458466
5100	39	1.7981827259064	20.4361989498138

Таблица 3: Результаты работы метода Гаусса-Зейделя без компиляции

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	11	0.0193138122559	-0.0166881084442
200	13	0.0280597209930	-0.0262701511383
300	11	0.0442860126495	-0.0401210784912
400	8	0.0369372367859	-0.0244505405426
500	14	0.0744934082031	-0.0503809452057
600	16	0.1111514568329	-0.0758087635040
700	10	0.0775518417358	-0.0288136005402
800	14	0.1128537654877	-0.0398359298706
900	8	0.0812444686890	0.0250444412231
1000	12	0.1285607814789	0.0104665756226
1100	13	0.1541528701782	0.0563864707947
1200	11	0.1443464756012	0.0945589542389
1300	15	0.2733049392700	0.0341606140137
1400	12	0.2351272106171	0.1579589843750
1500	13	0.2502281665802	0.2465941905975
1600	13	0.2620465755463	0.3187587261200

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
1700	12	0.2395026683807	0.4410846233368
1800	9	0.1971158981323	0.6156992912292
1900	9	0.2035489082336	0.7524514198303
2000	14	0.3440911769867	0.9070024490356
2100	10	0.2535443305969	1.1086637973785
2200	11	0.3078474998474	1.1817154884338
2300	10	0.2965457439423	1.4204728603363
2400	10	0.3074960708618	1.5957543849945
2500	16	0.4939227104187	1.6681189537048
2600	9	0.3807876110077	2.0439100265503
2700	11	0.4530491828918	2.2982883453369
2800	13	0.5149223804474	2.5884792804718
2900	15	0.6519904136658	2.8856246471405
3000	10	0.4015343189240	4.1409652233124
3100	15	0.6651651859283	4.0957009792328
3200	10	0.4945995807648	4.6685688495636
3300	10	0.5265166759491	5.2483890056610
3400	15	0.8306963443756	5.4443740844727
3500	14	0.7629737854004	6.9646949768066
3600	15	0.8384444713593	9.8202323913574
3700	10	0.6754627227783	7.6882960796356
3800	12	0.7503695487976	8.9080569744110
3900	9	0.5795228481293	13.4557130336761
4000	13	0.8064048290253	14.0212171077728
4100	12	0.9270818233490	10.6287236213684
4200	9	0.6576917171478	11.4015717506409
4300	10	0.7700955867767	12.2907514572144
4400	9	0.7986357212067	13.1841514110565
4500	11	0.9651329517365	13.9833509922028

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
4600	11	1.0155260562897	14.8001232147217
4700	10	0.9815475940704	16.2252361774445
4800	12	1.1166610717773	16.8964591026306
4900	12	1.4609208106995	17.6648790836334
5000	11	1.6296567916870	18.5979273319244
5100	10	1.7612597942352	20.4731218814850

Таблица 4: Результаты работы метода Гаусса-Зейделя с компиляцией

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	11	0.8528187274933	-0.8501930236816
200	13	0.0114216804504	-0.0096321105957
300	11	0.0152025222778	-0.0110375881195
400	8	0.0139842033386	-0.0014975070953
500	14	0.0313191413879	-0.0072066783905
600	16	0.0441782474518	-0.0088355541229
700	10	0.0334949493408	0.0152432918549
800	14	0.0524339675903	0.0205838680267
900	8	0.0387454032898	0.0675435066223
1000	12	0.0553576946259	0.0836696624756
1100	13	0.0708754062653	0.1396639347076
1200	11	0.0667796134949	0.1721258163452
1300	15	0.0950477123260	0.2124178409576
1400	12	0.0836653709412	0.3094208240509
1500	13	0.0950589179993	0.4017634391785
1600	13	0.1050190925598	0.4757862091064

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
1700	12	0.1085293292999	0.5720579624176
1800	9	0.0917103290558	0.7211048603058
1900	9	0.0961143970490	0.8598859310150
2000	14	0.1583130359650	1.0927805900574
2100	10	0.1257097721100	1.2364983558655
2200	11	0.1329574584961	1.3566055297852
2300	10	0.1396324634552	1.5773861408234
2400	10	0.1433229446411	1.7599275112152
2500	16	0.2383604049683	1.9236812591553
2600	9	0.1377286911011	2.2869689464569
2700	11	0.1775202751160	2.5738172531128
2800	13	0.2212841510773	2.8821175098419
2900	15	0.2562315464020	3.2813835144043
3000	10	0.1712708473206	4.3712286949158
3100	15	0.2680990695953	4.4927670955658
3200	10	0.1988301277161	4.9643383026123
3300	10	0.2039537429810	5.5709519386292
3400	15	0.3246674537659	5.9504029750824
3500	14	0.3032133579254	7.4244554042816
3600	15	0.3314881324768	10.3271887302399
3700	10	0.2457180023193	8.1180408000946
3800	12	0.2866723537445	9.3717541694641
3900	9	0.2322742938995	13.8029615879059
4000	13	0.3418791294098	14.4857428073883
4100	12	0.3655488491058	11.1902565956116
4200	9	0.2557041645050	11.8035593032837
4300	10	0.2934572696686	12.7673897743225
4400	9	0.2666175365448	13.7161695957184
4500	11	0.3482410907745	14.6002428531647

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
4600	11	0.3571336269379	15.4585156440735
4700	10	0.3390650749207	16.8677186965942
4800	12	0.4145016670227	17.5986185073853
4900	12	0.4452347755432	18.6805651187897
5000	11	0.4046220779419	19.8229620456696
5100	10	0.3793203830719	21.8550612926483

Таблица 5: Результаты работы метода релаксации с параметром $\omega = 0.3$ без компиляции

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	69	0.1005275249481	-0.0979018211365
200	57	0.1206607818604	-0.1188712120056
300	49	0.1425139904022	-0.1383490562439
400	56	0.2053191661835	-0.1928324699402
500	65	0.3115594387054	-0.2874469757080
600	63	0.4179477691650	-0.3826050758362
700	58	0.4500634670258	-0.4013252258301
800	63	0.5144131183624	-0.4413952827454
900	56	0.5079410076141	-0.4016520977020
1000	52	0.5479519367218	-0.4089245796204
1100	52	0.5866100788116	-0.3760707378387
1200	60	0.7877483367920	-0.5488429069519
1300	51	0.8253550529480	-0.5178894996643
1400	68	1.0837075710297	-0.6906213760376
1500	61	1.0677292346954	-0.5709068775177
1600	64	1.1952900886536	-0.6144847869873

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
1700	57	1.1720876693726	-0.4915003776550
1800	47	1.0376000404358	-0.2247848510742
1900	50	1.1995959281921	-0.2435956001282
2000	61	1.4129948616028	-0.1619012355804
2100	60	1.6161978244781	-0.2539896965027
2200	72	2.0117256641388	-0.5221626758575
2300	57	1.7831284999847	-0.0661098957062
2400	58	1.8348162174225	0.0684342384338
2500	58	1.9607927799225	0.2012488842010
2600	49	1.7928290367126	0.6318686008453
2700	46	1.7378003597260	1.0135371685028
2800	49	1.9201891422272	1.1832125186920
2900	59	2.3988935947418	1.1387214660645
3000	66	2.7753820419312	1.7671175003052
3100	61	2.5413675308228	2.2194986343384
3200	45	1.8975560665131	3.2656123638153
3300	46	2.2226252555847	3.5522804260254
3400	47	2.4044091701508	3.8706612586975
3500	52	2.7252712249756	5.0023975372314
3600	64	3.5677058696747	7.0909709930420
3700	46	2.6802191734314	5.6835396289825
3800	55	3.2846119403839	6.3738145828247
3900	59	3.7781403064728	10.2570955753326
4000	74	4.7115051746368	10.1161167621613
4100	66	4.5059747695923	7.0498306751251
4200	48	3.2124037742615	8.8468596935272
4300	54	3.9817228317261	9.0791242122650
4400	52	3.9754226207733	10.0073645114899
4500	53	4.1944501399994	10.7540338039398

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
4600	47	3.5656802654266	12.2499690055847
4700	64	5.0644659996033	12.1423177719116
4800	68	5.4379923343658	12.5751278400421
4900	49	4.2294104099274	14.8963894844055
5000	50	4.4580690860748	15.7695150375366
5100	68	5.9131090641022	16.3212726116180

Таблица 6: Результаты работы метода релаксации с параметром $\omega = 0.3$ с компиляцией

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	69	0.0599658489227	-0.0573401451111
200	57	0.0528619289398	-0.0510723590851
300	49	0.0694601535797	-0.0652952194214
400	56	0.1081957817078	-0.0957090854645
500	65	0.1574335098267	-0.1333210468292
600	63	0.1857311725616	-0.1503884792328
700	58	0.2014629840851	-0.1527247428894
800	63	0.2478799819946	-0.1748621463776
900	56	0.3290808200836	-0.2227919101715
1000	52	0.3150672912598	-0.1760399341583
1100	52	0.3454356193542	-0.1348962783813
1200	60	0.4408962726593	-0.2019908428192
1300	51	0.3937158584595	-0.0862503051758
1400	68	0.5974662303925	-0.2043800354004
1500	61	0.5677540302277	-0.0709316730499
1600	64	0.6298723220825	-0.0490670204163

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
1700	57	0.6181175708771	0.0624697208405
1800	47	0.5932829380035	0.2195322513580
1900	50	0.6457016468048	0.3102986812592
2000	61	0.7719609737396	0.4791326522827
2100	60	0.8635165691376	0.4986915588379
2200	72	1.0985848903656	0.3909780979156
2300	57	0.9858880043030	0.7311305999756
2400	58	1.0942242145538	0.8090262413025
2500	58	1.0667393207550	1.0953023433685
2600	49	0.9036757946014	1.5210218429565
2700	46	0.7348537445068	2.0164837837219
2800	49	0.9248378276825	2.1785638332367
2900	59	1.4107966423035	2.1268184185028
3000	66	1.6211378574371	2.9213616847992
3100	61	1.1287012100220	3.6321649551392
3200	45	0.9213261604309	4.2418422698975
3300	46	1.1303019523621	4.6446037292480
3400	47	1.2875993251801	4.9874711036682
3500	52	1.6717109680176	6.0559577941895
3600	64	1.9996886253357	8.6589882373810
3700	46	1.2649641036987	7.0987946987152
3800	55	1.4309070110321	8.2275195121765
3900	59	1.5354316234589	12.4998042583466
4000	74	1.9188091754913	12.9088127613068
4100	66	1.8247318267822	9.7310736179352
4200	48	1.3616037368774	10.6976597309113
4300	54	1.5901432037354	11.4707038402557
4400	52	1.5206227302551	12.4621644020081
4500	53	1.6625690460205	13.2859148979187

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
4600	47	1.5783286094666	14.2373206615448
4700	64	2.3678333759308	14.8389503955841
4800	68	2.5054790973663	15.5076410770416
4900	49	1.9110627174377	17.2147371768951
5000	50	2.0753562450409	18.1522278785706
5100	68	2.9693207740784	19.2650609016418

Таблица 7: Результаты работы метода релаксации с параметром $\omega = 0.5$ без компиляции

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	38	0.0402333736420	-0.0376076698303
200	33	0.0642549991608	-0.0624654293060
300	26	0.0776267051697	-0.0734617710114
400	30	0.1137201786041	-0.1012334823608
500	35	0.1637310981750	-0.1396186351776
600	34	0.1938447952271	-0.1585021018982
700	33	0.2255289554596	-0.1767907142639
800	36	0.3112046718597	-0.2381868362427
900	30	0.2800018787384	-0.1737129688263
1000	29	0.2887110710144	-0.1496837139130
1100	29	0.3312096595764	-0.1206703186035
1200	33	0.4665756225586	-0.2276701927185
1300	27	0.4497785568237	-0.1423130035400
1400	38	0.5659520626068	-0.1728658676147
1500	37	0.6763019561768	-0.1794795989990
1600	35	0.6617882251740	-0.0809829235077

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
1700	31	0.6256315708160	0.0549557209015
1800	27	0.6776878833771	0.1351273059845
1900	28	0.8466567993164	0.1093435287476
2000	34	1.0954561233521	0.1556375026703
2100	33	0.8126471042633	0.5495610237122
2200	39	1.1553461551666	0.3342168331146
2300	32	1.0061981678009	0.7108204364777
2400	33	1.0012252330780	0.9020252227783
2500	32	1.0747253894806	1.0873162746429
2600	28	1.0089309215546	1.4157667160034
2700	26	1.0140163898468	1.7373211383820
2800	27	1.0354535579681	2.0679481029510
2900	34	1.3720691204071	2.1655459403992
3000	36	1.5658690929413	2.9766304492950
3100	34	1.5124974250793	3.2483687400818
3200	25	1.1553020477295	4.0078663825989
3300	25	1.2062094211578	4.5686962604523
3400	26	1.2996444702148	4.9754259586334
3500	30	1.5593328475952	6.1683359146118
3600	34	1.8481411933899	8.8105356693268
3700	28	1.5744004249573	6.7893583774567
3800	29	1.5425746440887	8.1158518791199
3900	32	1.7468125820160	12.2884232997894
4000	41	2.6003525257111	12.2272694110870
4100	37	2.1964769363403	9.3593285083771
4200	27	1.9353318214417	10.1239316463470
4300	30	2.1774110794067	10.8834359645844
4400	29	2.0164768695831	11.9663102626801
4500	29	2.2515411376953	12.6969428062439

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
4600	26	2.0696635246277	13.7459857463837
4700	35	2.9880497455597	14.2187340259552
4800	37	3.1538977622986	14.8592224121094
4900	27	2.3380839824677	16.7877159118652
5000	28	2.5390222072601	17.6885619163513
5100	38	3.5170762538910	18.7173054218292

Таблица 8: Результаты работы метода релаксации с параметром $\omega = 0.5$ с компиляцией

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	38	0.0248558521271	-0.0222301483154
200	33	0.0251035690308	-0.0233139991760
300	26	0.0362539291382	-0.0320889949799
400	30	0.0539436340332	-0.0414569377899
500	35	0.1520664691925	-0.1279540061951
600	34	0.1575365066528	-0.1221938133240
700	33	0.1297938823700	-0.0810556411743
800	36	0.1718380451202	-0.0988202095032
900	30	0.2172434329987	-0.1109545230865
1000	29	0.2170009613037	-0.0779736042023
1100	29	0.1729927062988	0.0375466346741
1200	33	0.2178304195404	0.0210750102997
1300	27	0.1742932796478	0.1331722736359
1400	38	0.2733516693115	0.1197345256805
1500	37	0.2846527099609	0.2121696472168
1600	35	0.3049571514130	0.2758481502533

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
1700	31	0.2814493179321	0.3991379737854
1800	27	0.2532563209534	0.5595588684082
1900	28	0.2941541671753	0.6618461608887
2000	34	0.3561465740204	0.8949470520020
2100	33	0.3768675327301	0.9853405952454
2200	39	0.4861509799957	1.0034120082855
2300	32	0.4221777915955	1.2948408126831
2400	33	0.4465887546539	1.4566617012024
2500	32	0.4615571498871	1.7004845142365
2600	28	0.4367129802704	1.9879846572876
2700	26	0.4402854442596	2.3110520839691
2800	27	0.4604372978210	2.6429643630981
2900	34	0.5776770114899	2.9599380493164
3000	36	0.6508510112762	3.8916485309601
3100	34	0.6246719360352	4.1361942291260
3200	25	0.5412435531616	4.6219248771667
3300	25	0.5218186378479	5.2530870437622
3400	26	0.5811769962311	5.6938934326172
3500	30	0.6890027523041	7.0386660099030
3600	34	0.8069655895233	9.8517112731934
3700	28	0.7306487560272	7.6331100463867
3800	29	0.7524752616882	8.9059512615204
3900	32	0.9109027385712	13.1243331432343
4000	41	1.1246740818024	13.7029478549957
4100	37	1.1245305538177	10.4312748908997
4200	27	0.8497850894928	11.2094783782959
4300	30	1.1448340415955	11.9160130023956
4400	29	0.9561314582825	13.0266556739807
4500	29	0.9992966651917	13.9491872787476

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
4600	26	0.8902773857117	14.9253718852997
4700	35	1.2723178863525	15.9344658851624
4800	37	1.3840119838715	16.6291081905365
4900	27	1.0260970592499	18.0997028350830
5000	28	1.1025667190552	19.1250174045563
5100	38	1.5428266525269	20.6915550231934

Таблица 9: Результаты работы метода релаксации с параметром $\omega = 1.2$ без компиляции

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	12	0.0166332721710	-0.0140075683594
200	15	0.0298423767090	-0.0280528068542
300	15	0.0451591014862	-0.0409941673279
400	21	0.0882899761200	-0.0758032798767
500	17	0.0889542102814	-0.0648417472839
600	25	0.1537625789642	-0.1184198856354
700	12	0.0823657512665	-0.0336275100708
800	22	0.1785562038422	-0.1055383682251
900	14	0.1360051631927	-0.0297162532806
1000	17	0.2198336124420	-0.0808062553406
1100	19	0.2363474369049	-0.0258080959320
1200	15	0.2095620632172	0.0293433666229
1300	25	0.3633921146393	-0.0559265613556
1400	13	0.2153842449188	0.1777019500732
1500	12	0.2089300155640	0.2878923416138
1600	16	0.3037371635437	0.2770681381226

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
1700	17	0.3443384170532	0.3362488746643
1800	14	0.3023431301117	0.5104720592499
1900	13	0.3030085563660	0.6529917716980
2000	19	0.4842064380646	0.7668871879578
2100	13	0.3193805217743	1.0428276062012
2200	16	0.4808347225189	1.0087282657623
2300	12	0.3885960578918	1.3284225463867
2400	14	0.4100711345673	1.4931793212891
2500	23	0.8611750602722	1.3008666038513
2600	12	0.4651086330414	1.9595890045166
2700	14	0.5512661933899	2.2000713348389
2800	18	0.7222700119019	2.3811316490173
2900	21	0.9656829833984	2.5719320774078
3000	13	0.5615544319153	3.9809451103210
3100	25	1.1021821498871	3.6586840152740
3200	14	0.6591060161591	4.5040624141693
3300	15	0.7905263900757	4.9843792915344
3400	23	1.2351336479187	5.0399367809296
3500	19	1.0570447444916	6.6706240177155
3600	26	1.4765775203705	9.1820993423462
3700	15	0.7902989387512	7.5734598636627
3800	16	0.8549127578735	8.8035137653351
3900	12	0.7487325668335	13.2865033149719
4000	16	1.0110464096069	13.8165755271912
4100	14	0.9295926094055	10.6262128353119
4200	12	0.7497112751007	11.3095521926880
4300	16	1.1362178325653	11.9246292114258
4400	12	0.8932015895844	13.0895855426788
4500	14	1.1126706600189	13.8358132839203

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
4600	15	1.2443811893463	14.5712680816650
4700	16	1.3211269378662	15.8856568336487
4800	17	1.4305002689362	16.5826199054718
4900	20	1.7546513080597	17.3711485862732
5000	18	1.5322151184082	18.6953690052032
5100	12	1.1090400218964	21.1253416538239

Таблица 10: Результаты работы метода релаксации с параметром $\omega = 1.2$ с компиляцией

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	12	0.0082290172577	-0.0056033134460
200	15	0.0175986289978	-0.0158090591431
300	15	0.0155405998230	-0.0113756656647
400	21	0.0392782688141	-0.0267915725708
500	17	0.0374767780304	-0.0133643150330
600	25	0.0688302516937	-0.0334875583649
700	12	0.0399148464203	0.0088233947754
800	22	0.0844564437866	-0.0114386081696
900	14	0.0656771659851	0.0406117439270
1000	17	0.0807631015778	0.0582642555237
1100	19	0.1066310405731	0.1039083003998
1200	15	0.0918235778809	0.1470818519592
1300	25	0.1712594032288	0.1362061500549
1400	13	0.0905311107635	0.3025550842285
1500	12	0.1010007858276	0.3958215713501

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
1600	16	0.1719157695770	0.4088895320892
1700	17	0.2272701263428	0.4533171653748
1800	14	0.1994137763977	0.6134014129639
1900	13	0.1798992156982	0.7761011123657
2000	19	0.2744848728180	0.9766087532043
2100	13	0.3218951225281	1.0403130054474
2200	16	0.3746881484985	1.1148748397827
2300	12	0.3302288055420	1.3867897987366
2400	14	0.3215222358704	1.5817282199860
2500	23	0.4818553924561	1.6801862716675
2600	12	0.2393367290497	2.1853609085083
2700	14	0.2901587486267	2.4611787796021
2800	18	0.3887481689453	2.7146534919739
2900	21	0.4525003433228	3.0851147174835
3000	13	0.2949171066284	4.2475824356079
3100	25	0.5747072696686	4.1861588954926
3200	14	0.3570778369904	4.8060905933380
3300	15	0.3736119270325	5.4012937545776
3400	23	0.5868144035339	5.6882560253143
3500	19	0.5423560142517	7.1853127479553
3600	26	0.7527976036072	9.9058792591095
3700	15	0.4427814483643	7.9209773540497
3800	16	0.5006721019745	9.1577544212341
3900	12	0.3766620159149	13.6585738658905
4000	16	0.5179705619812	14.3096513748169
4100	14	0.4637806415558	11.0920248031616
4200	12	0.4140400886536	11.6452233791351
4300	16	0.5790660381317	12.4817810058594
4400	12	0.4403429031372	13.5424442291260

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
4500	14	0.5270836353302	14.4214003086090
4600	15	0.6532554626465	15.1623938083649
4700	16	0.6440670490265	16.5627167224884
4800	17	0.7211799621582	17.2919402122498
4900	20	0.8769676685333	18.2488322257996
5000	18	0.7813432216644	19.4462409019470
5100	12	0.5561645030975	21.6782171726227

2.3.2 Сравнение точности

Сравнение точности будет производиться в два этапа на примере СЛАУ с матрицей порядка 2000. На первом этапе ищется невязка решения. На втором сравнивается относительная погрешность решения с относительной погрешностью возмущенной системы, то есть системы, полученной путем прибавления 0.001 к каждому коэффициенту каждого уравнения включая свободный член.

Как можно заметить из таблиц 11 и 12, самым точным является метод Гаусса-Зейделя.

Таблица 11: Нормы невязки

	Метод	Норма невязки
0	Якоби	1.2763584878147
1	Гаусса-Зейделя	0.0306131739312
2	Релаксации $\omega = 0.3$	4.2002312764003
3	Релаксации $\omega = 0.5$	2.1604302269335
4	Релаксации $\omega = 1.2$	0.1404159486946

Таблица 12: Относительные погрешности

	Метод	δ	δ после возмущения
0	Якоби	0.0000008073422	0.0000042597904
1	Гаусса-Зейделя	0.0000006851802	0.0000043686549
2	Релаксации $\omega = 0.3$	0.0000076816049	0.0000090249221
3	Релаксации $\omega = 0.5$	0.0000032535898	0.0000055289461
4	Релаксации $\omega = 1.2$	0.0000003172812	0.0000042732700

2.4 Сравнение алгоритмов решения СЛАУ с симметричной и положительно определенной матрицей

Все сравнения будут производиться на случайных симметричных положительно определенных матрицах и матрицах Гилберта [1] , которые являются плохо обусловленными.

Алгоритмы сравнения времени работы и точности аналогичны алгоритмам использованным в пункте 2.3.

2.4.1 Сравнение времени работы алгоритмов на случайных симметричных положительно определенных матрицах

Из таблиц 13 – 16 можно заметить, что самым быстрым является метод Гаусса-Зейделя, метод Ричардсона является самым медленным.

Таблица 13: Результаты работы метода Гаусса-Зейделя

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	401	2.3390643596649	-2.3377883434296
200	801	0.7135362625122	-0.7120790481567
300	1201	1.6115331649780	-1.6070280075073
400	1601	2.9337511062622	-2.9233741760254
500	1066	2.4235734939575	-2.4038820266724
600	2401	6.6838998794556	-6.6506719589233
700	560	1.8110439777374	-1.7600324153900
800	476	1.7363913059235	-1.6598703861237
900	720	3.0331983566284	-2.9262795448303
1000	331	1.6133313179016	-1.4683439731598
1100	185	0.9997541904449	-0.8063507080078
1200	192	1.1539945602417	-0.9021136760712
1300	251	1.6482443809509	-1.3270256519318
1400	248	1.7628011703491	-1.3643929958344
1500	111	0.8505480289459	-0.3603699207306
1600	65	0.5391702651978	0.0462150573730
1700	102	0.9039313793182	-0.2014229297638
1800	64	0.6079511642456	0.2231085300446
1900	63	0.6363825798035	0.3379044532776

Таблица 14: Результаты работы метода релаксации с параметром $\omega = 1.2$

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	801	1.1301441192627	-1.1288681030273
200	1601	1.4322702884674	-1.4308130741119
300	2401	3.2345314025879	-3.2300262451172
400	3201	5.8470849990845	-5.8367080688477
500	2601	5.9274573326111	-5.9077658653259
600	4801	13.4231355190277	-13.3899075984955
700	1420	4.5338139533997	-4.4828023910522
800	2216	8.0146768093109	-7.9381558895111
900	1749	7.2374484539032	-7.1305296421051
1000	795	3.8509497642517	-3.7059624195099
1100	394	2.0995259284973	-1.9061224460602
1200	437	2.6578633785248	-2.4059824943542
1300	487	3.2416641712189	-2.9204454421997
1400	1257	9.1039156913757	-8.7055075168610
1500	392	3.0689554214478	-2.5787773132324
1600	140	1.1810905933380	-0.5957052707672
1700	290	2.6101050376892	-1.9075965881348
1800	216	2.0952432155609	-1.2641835212708
1900	159	1.6455583572388	-0.6712713241577

Таблица 15: Результат работы метода релаксации с параметром $\omega = 1.5$

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	801	0.3459229469299	-0.3446469306946
200	1601	1.4861764907837	-1.4847192764282
300	2401	3.4108848571777	-3.4063796997070
400	3201	6.0821325778961	-6.0717556476593
500	4001	9.6253893375397	-9.6056978702545
600	4801	13.9792003631592	-13.9459724426270
700	5601	18.9362955093384	-18.8852839469910
800	6401	24.0277609825134	-23.9512400627136
900	7201	29.8600468635559	-29.7531280517578
1000	4011	19.3795304298401	-19.2345430850983
1100	1915	10.3103036880493	-10.1169002056122
1200	1075	6.4404642581940	-6.1885833740234
1300	1473	9.8529915809631	-9.5317728519440
1400	5953	43.5349123477936	-43.1365041732788
1500	2323	17.9472532272339	-17.4570751190186
1600	862	7.2608704566956	-6.6754851341248
1700	768	6.8267784118652	-6.1242699623108
1800	1042	10.0620710849762	-9.2310113906860
1900	758	8.0394363403320	-7.0651493072510

Таблица 16: Результат работы метода Ричардсона

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	801	0.0877866744995	-0.0874655246735
200	1601	0.1379654407501	-0.1364347934723
300	2401	0.2654173374176	-0.2606511116028
400	3201	0.4632790088654	-0.4527266025543
500	4001	0.9214994907379	-0.9019143581390
600	4801	2.0674915313721	-2.0337567329407
700	5601	6.7622296810150	-6.7098324298859
800	6401	9.3341002464294	-9.2570641040802
900	7201	13.3334608078003	-13.2260453701019
1000	8001	15.0040786266327	-14.8584368228912
1100	8801	22.8945047855377	-22.7024776935577
1200	9601	27.4870173931122	-27.2380068302155
1300	10401	38.6106338500977	-38.2916533946991
1400	11201	46.4673750400543	-46.0744259357452
1500	12001	59.3388886451721	-58.8576273918152
1600	12801	65.2636866569519	-64.6816787719727
1700	13601	94.0645022392273	-93.3679246902466
1800	14401	107.8847098350525	-107.0655002593994
1900	15201	133.7427425384521	-132.7753863334656
2000	16001	243.1282382011414	-242.0031116008759

2.4.2 Сравнение точности

Как можно заметить из таблиц 17 и 18, самым точным методом является метод Гаусса-Зейделя. Точность заметно ниже, чем при решении СЛАУ с матрицей с диагональным преобладанием.

Таблица 17: Нормы невязки

	Метод	Норма невязки
0	Ричардсона	48771.498597651
1	Гаусса-Зейделя	28281.615624683
2	Релаксации $\omega = 1.2$	38669.964384109
3	Релаксации $\omega = 1.5$	44615.878249346

Таблица 18: Относительные погрешности

	Метод	δ	δ после возмущения
0	Ричардсона	0.9997184	0.9997184
1	Гаусса-Зейлделя	0.9915907	0.9915907
2	Релаксации $\omega = 1.2$	0.9767811	0.9767811
3	Релаксации $\omega = 1.5$	0.9688871	0.9688871

2.4.3 Сравнение времени работы алгоритмов на матрицах Гилберта

Из таблиц 19 – 22 можно заметить, что все методы работают на матрицах Гилберта очень медленно, но метод Гаусса-Зейделя быстрее остальных.

Таблица 19: Результат работы метода Гаусса-Зейделя для матриц Гилберта.

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	401	1.9366397857666	-1.9357037544250
200	801	0.8523890972137	-0.8469135761261
300	1201	1.8368988037109	-1.8289024829865
400	1601	3.3037641048431	-3.2885503768921
500	2001	4.9675052165985	-4.9415605068207
600	2401	6.7978944778442	-6.7548944950104
700	2801	9.1904799938202	-9.1319081783295
800	3201	12.0941190719604	-12.0100197792053
900	3601	14.9489524364471	-14.8329596519470

Таблица 20: Результат работы метода релаксации с параметром $\omega = 1.2$ для матриц Гилберта

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	801	1.0987691879272	-1.0978331565857
200	1601	1.5080630779266	-1.5025875568390
300	2401	3.4061763286591	-3.3981800079346
400	3201	6.1902682781219	-6.1750545501709
500	4001	9.5572087764740	-9.5312640666962
600	4801	14.1781008243561	-14.1351008415222
700	5601	18.7976863384247	-18.7391145229340
800	6401	24.5221281051636	-24.4380288124084
900	7201	30.8081021308899	-30.6921093463898

Таблица 21: Результат работы метода релаксации с параметром $\omega = 1.5$ для матриц Гилберта

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	801	0.4468040466309	-0.4458680152893
200	1601	1.5727531909943	-1.5672776699066
300	2401	3.4723470211029	-3.4643507003784
400	3201	6.0610089302063	-6.0457952022552
500	4001	9.2504413127899	-9.2244966030121
600	4801	13.5998442173004	-13.5568442344666
700	5601	19.5377233028412	-19.4791514873505
800	6401	24.2146868705750	-24.1305875778198
900	7201	35.9259448051453	-35.8099520206451

Таблица 22: Результат работы метода Ричардсона для матриц Гилберта

Порядок матрицы	Количество итераций	Затраченное время	Разность времени
100	801	0.0317203998566	-0.0307843685150
200	1601	0.2037823200226	-0.1983067989349
300	2401	0.6605954170227	-0.6525990962982
400	3201	1.6631236076355	-1.6479098796844
500	4001	3.4081640243530	-3.3822193145752
600	4801	6.8010828495026	-6.7580828666687
700	5601	10.9645364284515	-10.9059646129608
800	6401	16.6134688854218	-16.5293695926666
900	7201	34.6125929355621	-34.4966001510620

2.4.4 Сравнение точности

Матрицы Гилберта являются плохо обусловленными, что видно из таблиц 23, 24. Поэтому итерационные методы СЛАУ не дали результата.

Таблица 23: Нормы невязки

	Метод	Норма невязки
0	Ричардсона	1286682.770791929
1	Гаусса-Зейлделя	848369.191361920
2	Релаксации $\omega = 1.2$	842002.442041366
3	Релаксации $\omega = 1.5$	854669.537678260

Таблица 24: Относительные погрешности

	Метод	δ	δ после возмущения
0	Ричардсона	1.0000000	1.0000000
1	Гаусса-Зейлделя	1.0000000	1.0000000
2	Релаксации $\omega = 1.2$	1.0000000	1.0000000
3	Релаксации $\omega = 1.5$	1.0000000	1.0000000

ЗАКЛЮЧЕНИЕ

Все задачи курсовой работы были выполнены, но эту работу можно дополнить. Например:

1) переписать на более эффективный язык программирования, сравнить эффективность;
2) создать более эффективный алгоритм генерации симметричных, положительно определенных матриц;
3) провести расчеты на более производительном персональном компютере;
4) провести расчеты на видеокарте, сравнить эффективность;
5) реализовать другие итерационные методы решения СЛАУ.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Шарый С.П. Курс Вычислительных Методов: 2023.
2. [Электронный ресурс]: URL: numpy.org (дата обращения 05.05.2024).
3. [Электронный ресурс]. URL: scipy.org (дата обращения 05.05.2024).
4. [Электронный ресурс]. URL: numba.pydata.org (дата обращения 05.05.2024).
5. [Электронный ресурс]. URL: pandas.pydata.org (дата обращения 05.05.2024).