БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Отчет по лабораторной работе №2
Тема: «Итерационные методы решения СЛАУ»

Выполнил:
Гаргома А. О.

Преподаватель:
Горбачева Ю. Н.

Минск 2019

Оглавление

1 Постановка задачи

Разработать программу численного решения СЛАУ $Ax = f$ методом релаксации обеспечив сходимость итерационного процесса. В качестве критерия остановки итерационного процесса использовать $\|x^{(k+1)} - x^{(k)}\|_\infty < \epsilon$, где $\epsilon = 10^{-5}$.

2 Теория

Метод последовательной верхней релаксации является одним из наиболее широко используемых на практике методов для решения СЛАУ.

Рассмотрим взвешенную сумму текущего приближения и приближения, построенного по методу Гаусса–Зейделя:

$$x^{k+1}(i) = (1-\omega)x^k(i) + \frac{\omega}{a(i,i)}\left(f(i) - \sum_{j=1}^{i-1} a(i,j)x^{k+1}(j) - \sum_{j=i+1}^{n} a(i,j)x^k(j)\right),$$
$$i = 1, 2, \ldots, n,\ k = 0, 1, 2, \ldots \quad (1)$$

При $\omega = 1$ метод релаксации (1) есть метод Гаусса–Зейделя. Иногда при $\omega < 1$ говорят о нижней релаксации, а при $\omega > 1$ говорят о верхней релаксации. На практике обычно $1 < \omega < 2$, так как часто именно в таких пределах находится оптимальное значение ω, обеспечивающее наиболее быструю сходимость.

3 Листинг программы

```
import tabulate as tb
import numpy as np
import time
import copy


def generate_matrix(n):
    """Генерация случайной матрицы"""
    return np.round(np.random.randint(-10, 10, (10, 10)
        ) + np.random.rand(n, n),
                    2)


def generate_vector(n):
    """Генерация случайного вектора"""
    return np.round(np.random.randint(-10, 10, 1) + np.
        random.rand(n, 1), 2)


def max_norm(vector):
    """Максимум-норма"""
    return max(map(abs, [vector.min(), vector.max()]))


# Решение СЛАУ
def solve_lin_system(A, f, eps, w, count=0):
    n = A.shape[0]
    x_cur = np.zeros(n)
    x_next = np.zeros(n)
    norm = float('inf')
    # Итерационный процесс метода релаксации
    while (norm >= eps):
        for i in range(n):
            x_next[i] = (1 - w) * x_cur[i] + (w / A[i][
                i]) * \
                        (f[i] - sum(x_next[:i] * A[i][:
                            i]) - sum(
                             x_cur[i + 1:] * A[i][i +
```

```
                                        1:]))
        norm = max_norm(abs(x_cur - x_next))
        x_cur = copy.deepcopy(x_next)
        count += 1  # +1 итерация
    return x_next, count, norm


# Решение СЛАУ с различными параметрами релаксации w
def test(A, f, eps):
    params = [0.2, 0.5, 0.8, 1, 1.3, 1.5, 1.8]
    amount_of_iterations = [0] * len(params)
    norm = [0] * len(params)
    result = ()
    for i in range(len(params)):
        count = 0
        result = solve_lin_system(np.matmul(np.
            transpose(A), A),
                                  np.matmul(np.
                                      transpose(A), f),
                                      eps, params[i],
                                  count)
        amount_of_iterations[i] = result[1]
        norm[i] = result[2]
    print(tb.tabulate(zip(params, amount_of_iterations,
        norm),
                      headers=["w", "Количество
                          итераций", "Максимум-норма",
                          ],
                      tablefmt="fancy_grid", floatfmt="
                          .14f"))
    return result[0]


# Исходные данные
matrix_size = 10
np.random.seed(round(time.time()))
A = generate_matrix(matrix_size)
x = generate_vector(matrix_size)
f = np.matmul(A, x)

try:
```

```
    accuracy = .1e-4
    approximate_x = test(A, f, accuracy)
    approximate_f = np.matmul(A, approximate_x)
    print("Матрица A: \n", tb.tabulate(A, tablefmt="
        fancy_grid"))
    print("Точность: {0}\n".format(accuracy),
          tb.tabulate(zip(x, approximate_x, f,
              approximate_f),
                      headers=["Вектор x", "
                          Приближенное решение",
                              "Вектор f = A*x",
                              "Приближенное f"],
                      tablefmt="fancy_grid", floatfmt="
                          .14f"))

    max_norm_of_error = max_norm(
        x.T - approximate_x)  # Максимум-норма
            погрешности
    print("Максимум-норма погрешности (w = 1.8): {0}".
        format(max_norm_of_error))
except Exception as error:
    print(error.args)
```

4 Результат работы

w	Количество итераций	Максимум-норма
0.20000000000000	2518	0.00000999310916
0.50000000000000	1079	0.00000998766243
0.80000000000000	627	0.00000988394922
1.00000000000000	453	0.00000997930774
1.30000000000000	272	0.00000996917665
1.50000000000000	178	0.00000967427544
1.80000000000000	99	0.00000785260049

Матрица A:

-8.92	-1.74	4.86	6.39	-3.48	-5.61	-3.03	1.41	3.44	-9.75
7.42	-4.82	-3.08	-4.15	-0.46	-1.17	-3.69	4.27	8.02	2.16
8.45	8.53	-1.48	9.01	-3.22	-1.35	0.63	-6.98	4.96	8.1
-8.99	-7.43	-1.36	0.83	-7.13	-9.45	-8.88	-2.99	-0.84	0.54
2.58	4.11	3.54	9.49	-8.27	5.88	-0.64	2.1	0.92	-2.2
-8.98	2.83	-7.89	-2.53	6.25	3.47	7.07	-0.12	4.92	-4.09
-4.69	0.7	-7.63	5.56	8.26	7.88	9.58	3.98	9.24	-3.08
-6.6	-2.79	5.31	8.35	-4.21	9.27	-5.7	-3.33	-8.3	-6.23
-9.1	-1.89	-1.43	7.69	6.8	8.58	-6.87	-0.98	2.5	-7.55
-5.98	2.85	-9.21	-5.96	8.6	-0.04	-1.88	-2.12	-1.96	-2.79

Точность: 1e-05

Вектор x	Приближенное решение	Вектор f = A*x	Приближенное f
-5.59000000000000	-5.58996050066819	89.37770000000000	89.37774447672079
-5.14000000000000	-5.14002879517229	-23.40270000000000	-23.40264075902812
-5.78000000000000	-5.78003301280200	-143.28469999999999	-143.28467028475978
-5.38000000000000	-5.37998275155046	253.86959999999999	253.86956698468060
-5.71000000000000	-5.71000046815563	-92.19350000000000	-92.19352614335430
-5.41000000000000	-5.41001425505555	-7.10850000000000	-7.10853674212464
-5.97000000000000	-5.96998251052883	-165.83290000000000	-165.83282526757458
-5.13000000000000	-5.13002578818488	79.41330000000001	79.41338863804455
-5.56000000000000	-5.56001768640937	13.93430000000000	13.93427072507863
-5.24000000000000	-5.24004774640300	102.80460000000001	102.80466918753864

Максимум-норма погрешности(w = 1.8): 4.774640300198030e-05

5 Вывод

Реализован алгоритм для решения СЛАУ методом релаксации. Количество итераций уменьшается при параметре релаксации $\omega \longrightarrow 2$