Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

ОТЧЕТ

ПО ПРЕДМЕТУ “ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ АЛГЕБРЫ”

студентки 2 курса 2 группы

Курец Любови Олеговны

Преподаватель

Горбачева Юлия Николаевна

Постановка задачи

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

Для проведения вычислительного эксперимента необходимо решить систему размерности $n = 10$. Матрицу A и вектор точного решения x заполнить случайными числами (сгенерировать) с двумя знаками после запятой из диапазона от -10 до 10. Правую часть задать умножением матрицы A на вектор x: $f = Ax$.

В результатах выполнения вычислительного эксперимента необходимо привести следующую информацию:

- Матрицу A(построчно), вектор f, точное решение x ,ε.
- Исследовать сходимость метода релаксации в зависимости от параметра релаксации ω. Результаты оформить в виде таблицы.
- Полученный приближенный вектор решений $\hat{x}$ (для какого-нибудь одного параметра ω) и максимум-норму погрешности $||x - \hat{x}||_\infty$.

Краткие теоретические сведенья:

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

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

При ω=1 метод релаксации есть метод Гаусса–Зейделя. Иногда при ω <1 говорят о нижней релаксации, а при ω>1 говорят о верхней релаксации.

Пусть A – симметричная положительно определенная матрица (если нет, то $A^TAx=A^Tf$). Тогда метод верхней релаксации сходится при условии 0<ω<2. В частности, метод Зейделя (ω=1) сходится.

Листинг программы:

```
def generateMatrix(n):
    return np.round(np.random.randint(-10, 10, (10, 10)) + np.random.rand(n, n),2)


def generateVector(n):
    return np.round(np.random.randint(-10, 10, 1) + np.random.rand(n, 1), 2)

    #Максимум-норма
```

```
def maxNorm(vector):
           return max(map(abs, [vector.min(), vector.max()]))


#  решение системы
def solve(A, f, eps, w, count=0):
    n = A.shape[0]
    currentX = np.zeros(n)
    nextX = np.zeros(n)
    norm = float('inf')
    #  итерационный процесс метода релаксации
    while (norm >= eps):
        for i in range(n):
            nextX[i] = (1 - w) * currentX[i] + (w / A[i][i]) * \
                       (f[i] - sum(nextX[:i] * A[i][:i]) -
                        sum(currentX[i + 1:] * A[i][i + 1:]))
        norm = maxNorm(abs(currentX - nextX))
        currentX = copy.deepcopy(nextX)
        count += 1  # +1 итерация
    return nextX, count, norm


#  кол-во итераций для различных параметров
def CountOfIterations(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(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=".5f"))
    return result[0]


# исходные данные
matrix_size = 10
np.random.seed(round(time.time()))
A = generateMatrix(matrix_size)
x = generateVector(matrix_size)
f = np.matmul(A, x)

try:
    accuracy = .1e-4
    generalX = CountOfIterations(A, f, accuracy)
    generalF = np.matmul(A, generalX)
    print("Матрица A:\n", tb.tabulate(A))
    print("Точность: {0}\n".format(accuracy),
          tb.tabulate(zip(x, generalX, f, generalF), headers=["  Вектор x",
"Приближенное решение","Вектор f = A*x","Приближенное f"],tablefmt="fancy_grid",
floatfmt=".5f"))

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

Результаты вычислительного эксперимента:

w	Количество итераций	Максимум-норма
0.20000	2772	0.00001
0.50000	1095	0.00001
0.80000	607	0.00001
1.00000	429	0.00001
1.30000	251	0.00001
1.50000	162	0.00001
1.80000	88	0.00001

Матрица A:

-6.14	6.57	2.22	-3.81	-7.04	6.19	3.46	-4.82	9.64	0.2
7.96	-3.34	1.2	-1.08	-1.53	-8.34	-5.8	-1.41	-8.46	8.72
1.22	-7.31	0.61	9.45	-7.99	9.3	8.3	5.89	-7.85	7.36
5.57	-8.97	-8.19	-0.65	-3.66	-5.52	1.79	-2.12	8.51	4.04
-4.06	-1.61	5.42	-5.43	-6.08	-9.03	-8.31	1.03	-8.58	6.23
1.39	3.7	7.12	1.91	-2.66	-7.71	-1.35	-3.44	7.46	7.06
-8.52	2.2	-9.01	1.62	-9.97	-1.64	2.34	5.14	-7.19	-4.87
9.85	9.61	8.57	9.12	5.59	1.54	9.64	1.28	6.32	-0.59
-4.35	0.38	-1.09	0.76	1.42	9.41	6.79	-9.21	-4.38	-8.01
5.04	7.07	-1.27	8.94	0.19	-8.58	-6.51	9.89	-3.42	-2.54

Точность: 1e-05

Вектор x	Приближенное решение	Вектор f = A*x	Приближенное f
7.33000	7.32997	51.34660	51.34656
7.14000	7.14002	-86.84770	-86.84767
7.22000	7.21996	142.20950	142.20956
7.09000	7.09001	-61.70900	-61.70907
7.32000	7.32004	-222.98340	-222.98350
7.57000	7.57000	100.77020	100.77026
7.37000	7.37000	-224.02450	-224.02448
7.17000	7.16999	443.10530	443.10522
7.53000	7.52999	-61.95900	-61.95907
7.86000	7.86006	55.29400	55.29395

```
Максимум-норма погрешности(w = 1.8): 5.8512976965907626e-05
Поток 0x1 завершился с кодом 0 (0x0).
Программа "python.exe" завершилась с кодом 0 (0x0).
```

Выводы:

Метод релаксации относится к одношаговым итерационным методам, когда для нахождения x^{k+1} требуется помнить только одну предыдущую итерацию

x^{k} . На практике обычно $1<\omega<2$, так как часто именно в таких пределах находится оптимальное значение ω, обеспечивающее наиболее быструю сходимость. При $\omega=1.8$ получаем нужную точность при меньшем количестве операций.