БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №1
«Метод Гаусса. Метод релаксации»

Студент:

Мирончик Юрий Александрович, 3 курс,
9 группа

Преподаватель: Полевиков Виктор Кузьмич,

Доцент кафедры вычислительной математики

Минск, 2023

Постановка задачи

1. Методом Гаусса и методом релаксации найти решение системы линейных алгебраических уравнений.
2. Построить график зависимости количества итераций от параметра релаксации.
3. Найти нормы исходной и обратной матрицы
4. Вычислить число обусловленности

Теория

Алгоритм решения методом Гаусса

1. Прямой ход:

В прямом ходе метода Гаусса матрица А преобразуется в верхнюю треугольную форму. Это достигается путем элементарных преобразований строк матрицы. Процесс выполняется по алгоритму:

$$\begin{cases} a_{ij}^{k} = a_{ij}^{k-1} - \dfrac{a_{ik}^{k-1} a_{kj}^{k-1}}{a_{kk}^{k-1}}, \quad f_i^{k} = f_i^{k-1} - \dfrac{a_{ik}^{k-1} f_k^{k-1}}{a_{kk}^{k-1}}, \quad k = \overline{1, n-1}, \quad i, j = \overline{k+1, n} \\ a_{ij}^{0} = a_{ij}, \quad f_i^{0} = f_i, \quad i, j = \overline{1, n} \end{cases}$$

2. Обратный ход (обратная подстановка):

$$\begin{cases} x_n = f_n^{n-1} \big/ a_{n,n}^{n-1} \\ x_i = \left(f_i^{i-1} - \sum\limits_{j=i+1}^{n} a_{ij}^{i-1} x_j \right) \Big/ a_{ii}^{i-1}, \quad i = n-1, n-2, \ldots, 1 \end{cases}$$

Теорема: Метод Гаусса применим тогда и только тогда,

когда все главные миноры матрицы А отличны от нуля, т.е.

Метод релаксации

Если матрица A в системе имеет строгое диагональное преобладание, то разрешая уравнения приходим к каноническому виду, где. В этом случае алгоритм метода релаксации можно записать в виде:

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

$$i = \overline{1, n}$$

Теорема 2. Если матрица A симметричная и положительно определенная ($A = A^T > 0$), а параметр релаксации выбирается на интервале $q \in (0, 2)$, то метод релаксации сходится.

3.Критерий окончания: $\left\|Ax^{(k)} - f\right\| < \varepsilon$

Листинг программы

```
import numpy as np
import matplotlib.pyplot as plt


def print_matrix(m):

    max_len = max(len(f"{np.max(m):.17f}"), 17)
    max_len = 8
    for row in m:
        formatted_row = " ".join([f"{x:.5f}".rjust(max_len) for x in row])
        print(formatted_row)


def inverse_matrix(matrix_origin):
    """
    Функция получает на вход матрицу, затем добавляет к ней единичную матрицу,
    проводит элементарные преобразования по строкам с первоначальной, добиваясь
получения слева единичной матрицы.
    В этом случае справа окажется матрица, которая является обратной к заданнй
первоначально
    """
    # Склеиваем 2 матрицы: слева - первоначальная, справа - единичная
    n = matrix_origin.shape[0]
    m = np.hstack((matrix_origin, np.eye(n)))

    for nrow, row in enumerate(m):
        # nrow равен номеру строки
        # row содержит саму строку матрицы
```

```
        divider = row[nrow]  # диагональный элемент
        # делим на диагональный элемент:
        row /= divider
        # теперь вычитаем приведённую строку из всех нижележащих строк:
        for lower_row in m[nrow + 1:]:
            factor = lower_row[nrow]  # элемент строки в колонке nrow
            lower_row -= factor * row  # вычитаем, чтобы получить ноль в колонке nrow
    # обратный ход:
    for k in range(n - 1, 0, -1):
        for row_ in range(k - 1, -1, -1):
            if m[row_, k]:
                # 1) Все элементы выше главной диагонали делаем равными нулю
                m[row_, :] -= m[k, :] * m[row_, k]
    return m[:, n:].copy()




def reform(matrix, vector):
    """
    Функция поворота столбца
    :param matrix: матрица коэффициентов
    :param vector: вектор правой части
    :return:
    """
    p = np.column_stack((matrix, vector))
    row = p.shape[0]
    for m in range(0, row):
        if m < row:
            big = np.argmax(abs(p[m:, m]))
        else:
            big = 0
        b1 = big + m
        c = np.copy(p[b1, :])
        p[b1, :] = p[m, :]
        p[m, :] = c
    return p


def gauss_method(matrix, vector, zeros_matrix):
    """
    Реализация метода Гаусса с частичным выбором ведущего элемента.
    Выполняет прямой и обратный ход, путем обнуления элементов под главной диагональю.
    :param matrix: Матрица коэффициентов a
    :param vector: Правая матрица b
    :param zeros_matrix: Матрица начального значения неизвестного x.
    :return: j - Количество итераций; zeros_matrix -Решение СЛАУ
```

```
    """
    p = reform(matrix, vector)
    row = p.shape[0]
    a0 = p[:, 0: row]
    b0 = p[:, row]
    j = 0
    err = 100.
    while err > 1.e-6 and j < 2500:
        i = 0
        while i < zeros_matrix.size:
            if a0[i, i] == 0:
                print('a[i,i]=0, i=', i)
            zeros_matrix[i] = -(np.dot(a0[i, :], zeros_matrix) - b0[i] - a0[i, i] * zeros_matrix[i]) / a0[i, i]
            i = i + 1
        j = j + 1
        err = find_norma(a0, b0, zeros_matrix)
    return zeros_matrix, j


def relax_method(matrix, right_matrix, zeros_matrix, omiga):
    """
    Реализация метода релаксации для решения системы линейных уравнений.
    :param matrix: Матрица коэффициентов a
    :param right_matrix: Правая матрица b
    :param zeros_matrix: Матрица начального значения неизвестного x.
    :param omiga: Фактор релаксации
    :return:  j - Количество итераций; zeros_matrix -Приблеженное решение СЛАУ
    """
    p = reform(matrix, right_matrix)
    row = p.shape[0]
    a0 = p[:, 0: row]
    b0 = p[:, row]
    err = 100.
    j = 0
    while err > 1.e-6 and j < 2500:
        i = 0
        while i < zeros_matrix.size:
            if a0[i, i] == 0:
                print('a[i,i]=0, i=', i)
            zeros_matrix[i] = (1 - omiga) * zeros_matrix[i] - omiga * (
                    np.dot(a0[i, :], zeros_matrix) - b0[i] - a0[i, i] * zeros_matrix[i]) / a0[i, i]
            i = i + 1
        j = j + 1
        err = find_norma(a0, b0, zeros_matrix)
    return zeros_matrix, j


def find_norma(matrix, right_vector, solution_vector):
```

```
    """
    Это функция для вычисления нормы вектора невязки ax - b.
    Норма невязки используется для определения точности решения.
    :param matrix: Матрица коэффициентов
    :param right_vector: вектор правой части
    :param solution_vector: Вектор решения
    :return: Норма вектора
    """
    axb = np.dot(matrix, solution_vector) - right_vector
    normaxb = np.linalg.norm(axb, ord=2)
    return normaxb


n = 9
A = np.array([[25, 1, 2, 3, 4, 5, 5, 3, 1],
              [0, 23, 5, 3, 1, 1, 2, 10, 2],
              [3, 2, 21, 1.5, 0.5, 4.6, 8.4, 0, 0],
              [0, 8.4, 3, 19, 6.6, 1, 1, 1, 0],
              [3, 2, 1, 2, 17, 1, 1, 0.5, 5.5],
              [2, 4, 1, 2, 1, 18, 1, 0.5, 5.5],
              [3, 4, 2, 2, 1, 1, 20, 0.5, 5.5],
              [2, 4, 3, 2, 1, 3, 0.5, 22, 5.5],
              [5, 4, 2, 2, 1, 3, 0.5, 5.5, 24]], dtype=float)

X = np.array([1.2677, 1.6819, -2.3657, -6.5369, 2.8351, 8, 3, 6, 2])

print('Исходная матрица матрица A:')
print_matrix(A)
print('-' * 160)
print(f'Исходный вектор X: {X}')
print(f'Вектор f={np.dot(A, X)}')
print('-' * 160)

zeros_matrix = np.zeros(n, dtype=float)
result2, j = gauss_method(np.copy(A), np.copy(np.dot(A, X)), np.copy(zeros_matrix))
print('Найденный вектор X по методу Гаусса:')
for i, res in enumerate(result2):
    print(f'\tX{i + 1}= {res:.20f}')
print('-' * 160)

print('Обратная матрица A^(-1) методом Гаусса:')
n_a = inverse_matrix(A)
print_matrix(n_a)
print('-' * 160)

print('Транспонированная матрица:')
trans_A = A.transpose()
print_matrix(trans_A)
```

```
print('-' * 160)

print('Матрица A~ = A^(T)*A:')
A_lambda = np.dot(trans_A, A)
print_matrix(A_lambda)
print('-' * 160)

print('Вектор f~ = At * f: ')
f_lambda = np.dot(trans_A, result2)
print(f_lambda)
print('-' * 160)

norma_A = np.linalg.norm(A)
norma_A_obratn = np.linalg.norm(n_a)
print(f'Норма матрицы A: {norma_A}')
print(f'Норма обратной матрицы A: {norma_A_obratn}')
print(f'Число обусловленности: {norma_A * norma_A_obratn}')
print('-' * 160)

loosefactor = 1.2
result, j = relax_method(np.copy(A), np.copy(np.dot(A, X)), np.copy(zeros_matrix), 1.2)
print('Метод релаксации:\n Количество итераций = ', j)
for i, res in enumerate(result):
   print(f'\tX{i + 1}= {res:.20f}')
print('-' * 160)

print('Обратная матрица умноженная на исходную:')
print_matrix(np.dot(n_a, A))
print('-' * 160)
```

Вывод программы

Матрица матрица А:

25.00000	1.00000	2.00000	3.00000	4.00000	5.00000	5.00000	3.00000	1.00000
0.00000	23.00000	5.00000	3.00000	1.00000	1.00000	2.00000	10.00000	2.00000
3.00000	2.00000	21.00000	1.50000	0.50000	4.60000	8.40000	0.00000	0.00000
0.00000	8.40000	3.00000	19.00000	6.60000	1.00000	1.00000	1.00000	0.00000
3.00000	2.00000	1.00000	2.00000	17.00000	1.00000	1.00000	0.50000	5.50000
2.00000	4.00000	1.00000	2.00000	1.00000	18.00000	1.00000	0.50000	5.50000
3.00000	4.00000	2.00000	2.00000	1.00000	1.00000	20.00000	0.50000	5.50000
2.00000	4.00000	3.00000	2.00000	1.00000	3.00000	0.50000	22.00000	5.50000
5.00000	4.00000	2.00000	2.00000	1.00000	3.00000	0.50000	5.50000	24.00000

Вектор X: [1.2677 1.6819 -2.3657 -6.5369 2.8351 8.   3.   6.   2.   ]

Вектор f =[95.3727   88.0796   11.0994 -81.45858 64.9241 157.6586   77.5606
160.4272 104.596]

Найденный вектор X по методу Гаусса:

X1= 1.2677000(0)

X2= 1.6819000(0)

X3= -2.3656999(9)

X4= -6.5368999(9)

X5= 2.83510000(0)

X6= 7.99999999(9)

X7= 2.99999999(9)

X8= 5.99999999(9)

X9= 1.99999999(9)

Обратная матрица A^{-1}:

0.04252 0.00424 -0.00214 -0.00418 -0.00730 -0.00993 -0.00902 -0.00839 0.00581

0.00262 0.04970 -0.00809 -0.00541 0.00004 0.00266 -0.00154 -0.02291 0.00074

-0.00424 0.00033 0.04983 -0.00086 0.00130 -0.01160 -0.01950 -0.00064 0.00713

0.00153 -0.02050 -0.00374 0.05714 -0.02183 -0.00223 0.00131 0.00564 0.00557

-0.00468 -0.00156 -0.00060 -0.00430 0.06239 0.00018 -0.00107 0.00381 -0.01464

-0.00221 -0.00707 0.00027 -0.00366 -0.00053 0.05809 -0.00135 0.00578 -0.01352

-0.00399 -0.00656 -0.00227 -0.00301 -0.00016 0.00048 0.05292 0.00565 -0.01278

-0.00124 -0.00511 -0.00450 -0.00254 -0.00050 -0.00489 0.00210 0.05074 -0.01040

-0.00823 -0.00523 -0.00098 -0.00164 0.00081 -0.00338 0.00228 -0.00748 0.04422

Транспонированная матрица A^T:

25.00000 0.00000 3.00000 0.00000 3.00000 2.00000 3.00000 2.00000 5.00000

1.00000 23.00000 2.00000 8.40000 2.00000 4.00000 4.00000 4.00000 4.00000

2.00000 5.00000 21.00000 3.00000 1.00000 1.00000 2.00000 3.00000 2.00000

3.00000 3.00000 1.50000 19.00000 2.00000 2.00000 2.00000 2.00000 2.00000

4.00000 1.00000 0.50000 6.60000 17.00000 1.00000 1.00000 1.00000 1.00000

5.00000 1.00000 4.60000 1.00000 1.00000 18.00000 1.00000 3.00000 3.00000

5.00000 2.00000 8.40000 1.00000 1.00000 1.00000 20.00000 0.50000 0.50000

3.00000 10.00000 0.00000 1.00000 0.50000 0.50000 0.50000 22.00000 5.50000

1.00000 2.00000 0.00000 0.00000 5.50000 5.50000 5.50000 5.50000 24.00000

Матрица $A^{\sim} = A^T*A$:

685.00000 85.00000 140.00000 109.50000 164.50000 201.80000 218.70000 150.50000 200.00000

85.00000 672.56000 218.20000 270.60000 133.44000 147.60000 166.20000 356.40000 220.00000

140.00000 218.20000 498.00000 127.50000 68.30000 150.60000 243.90000 138.00000 98.50000

109.50000 270.60000 127.50000 401.25000 183.15000 95.90000 98.60000 116.00000 101.00000

164.50000 133.44000 68.30000 183.15000 353.81000 71.90000 71.80000 65.60000 140.00000

201.80000 147.60000 150.60000 95.90000 71.90000 392.16000 108.64000 118.50000 205.50000

218.70000 166.20000 243.90000 98.60000 71.80000 108.64000 503.06000 60.75000 144.75000

150.50000 356.40000 138.00000 116.00000 65.60000 118.50000 60.75000 625.00000 284.25000

200.00000 220.00000 98.50000 101.00000 140.00000 205.50000 144.75000 284.25000 702.00000

Вектор $\tilde{f} = A^T * f$:

[80.10070047 61.98024037 -19.51039953 -75.23064976 29.62301015

164.43638013 58.12862017 164.00275005 161.72454982]

Норма матрицы А: 69.5186305964092

Норма обратной матрицы A^{-1}: 0.16852100469381576

Число обусловленности: 11.715349473045118

Метод релаксации:

q = 1.2, Критерий остановки: $\left|\left|Ax^{(k)} - f\right|\right| < \varepsilon$, eps = 10^{-9}

Количество итераций = 24

X1= 1.26769998029247599192

X2= 1.68190000005322731532

X3= -2.36570000022176004521

X4= -6.53690000000228928028

X5= 2.83510000343140466939

X6= 8.00000000571660052628

X7= 3.00000000833909297171

X8= 6.00000000763004326387

X9= 2.00000000796202481723

Обратная матрица умноженная на исходную:

1.00000 -0.00000 0.00000 -0.00000 -0.00000 0.00000 0.00000 -0.00000 -0.00000

0.00000 1.00000 -0.00000 -0.00000 0.00000 -0.00000 -0.00000 -0.00000 -0.00000

0.00000 0.00000 1.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000

0.00000 0.00000 0.00000 1.00000 -0.00000 0.00000 -0.00000 0.00000 0.00000

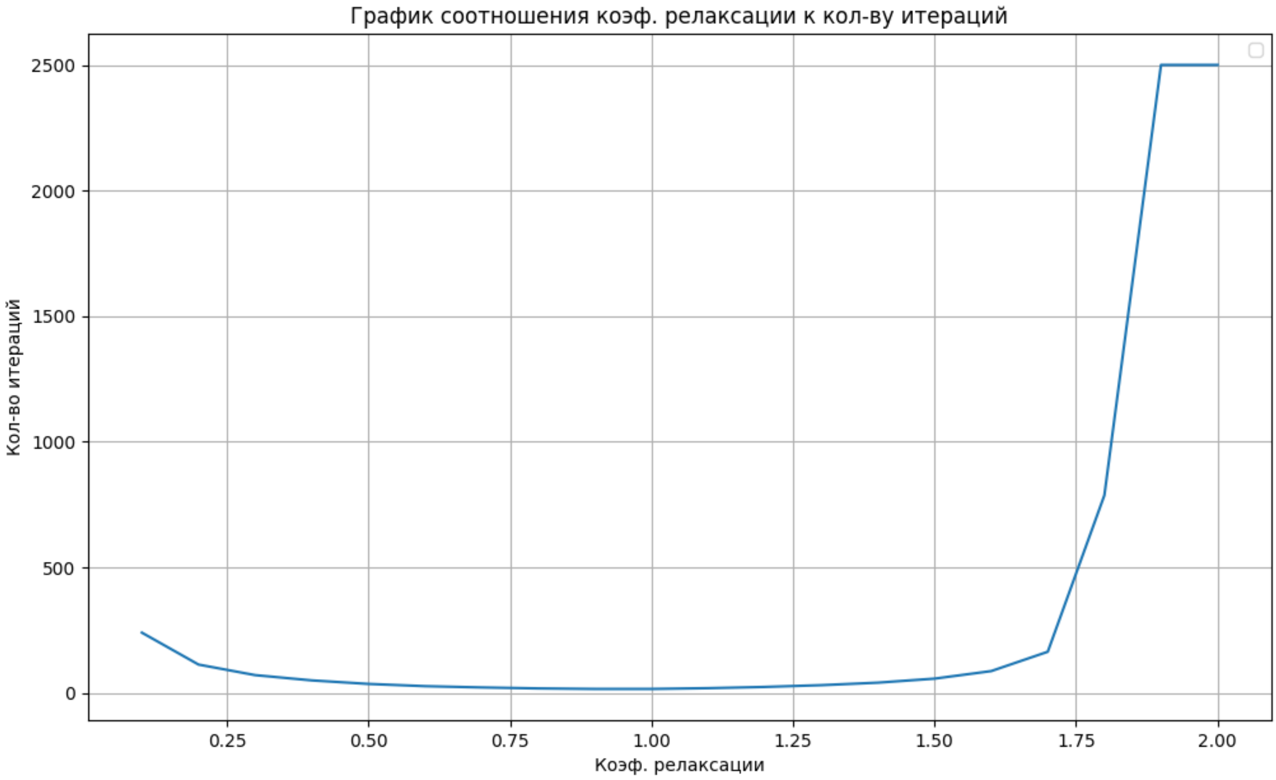
-0.00000 -0.00000 -0.00000 -0.00000 1.00000 -0.00000 -0.00000 -0.00000 0.00000

0.00000 0.00000 -0.00000 -0.00000 0.00000 1.00000 0.00000 0.00000 0.00000

0.00000 0.00000 -0.00000 0.00000 0.00000 0.00000 1.00000 0.00000 0.00000

-0.00000 0.00000 0.00000 0.00000 -0.00000 0.00000 0.00000 1.00000 -0.00000

-0.00000 0.00000 0.00000 -0.00000 -0.00000 0.00000 0.00000 0.00000 1.00000



0.1 < q < 1.9