Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

КТиУ, кафедра Информатики и Прикладной Математики

Лабораторная работа №1

по дисциплине

«Вычислительная математика»

«Решение системы линейных алгебраических уравнений методом Гаусса-Зейделя»

Выполнил:

Студент группы P3217

Григорьев Георгий

Санкт-Петербург

2018 г.

1. Описание метода

Метод Зейделя представляет собой некоторую модификацию метода простой итерации. Основная его идея заключается в том, что при вычислении (k+1)-го приближения неизвестной xi учитываются уже вычисленные ранее (k+1) – е приближения неизвестных x1, x2, ...,

Пусть задана система:

$$\begin{cases} a_{11}x_1+\ldots+a_{1n}x_n &= b_1 \\ \ldots \\ a_{m1}x_1+\ldots+a_{mn}x_n &= b_m \end{cases}$$

Тогда с помощью элементарных преобразований над строками систему можно привести к виду

$$\begin{cases} \alpha_{1j_1}x_{j_1}+\alpha_{1j_2}x_{j_2}+\ldots+\alpha_{1j_r}x_{j_r}+\ldots+\alpha_{1j_n}x_{j_n} &= \beta_1 \\ \qquad\alpha_{2j_2}x_{j_2}+\ldots+\alpha_{2j_r}x_{j_r}+\ldots+\alpha_{2j_n}x_{j_n} &= \beta_2 \\ & \ldots \\ \qquad\qquad\alpha_{rj_r}x_{j_r}+\ldots+\alpha_{rj_n}x_{j_n} &= \beta_r \\ 0 &= \beta_{r+1} \\ & \ldots \\ 0 &= \beta_m \end{cases},$$

где коэффициенты под главной диагональю равны нулю.

Рабочие формулы для метода Зейделя для системы трех уравнений имеют следующий вид:

$$\begin{cases} x_1^{(k+1)} = \alpha_{12}x_2^{(k)} + \alpha_{13}x_3^{(k)} + \ldots + \alpha_{1n}x_n^{(k)} + \beta_1. \\ x_2^{(k+1)} = \alpha_{21}x_1^{(k+1)} + \alpha_{23}x_3^{(k)} + \ldots + \alpha_{2n}x_n^{(k)} + \beta_2, \\ \ldots \\ x_n^{(k+1)} = \alpha_{n1}x_1^{(k+1)} + \alpha_{n2}x_2^{(k+1)} + \ldots + \alpha_{n,n-1}x_{n-1}^{(k+1)} + \beta_n. \end{cases}$$

Зададим определенную точность решения e, по достижении которой итерационный процесс завершается, т. е. решение продолжается до тех пор, пока не будет выполнено условие для всех уравнений:

$$\left|x_i^{(k+1)} - x_i^{(k)}\right| \le \varepsilon,$$

где i=1,2,3,…,n.

2. Листинг программы

```
import numpy as np
from math import sqrt

def seidel(A, b, eps):
      """
      :param: A — матрица коэффициентов
      :param: b — столбец свободных членов
      :param: eps — точность
      """
      MAX_ITERATIONS = 1000000
    n = len(A)
    x = [.0 for i in range(n)]

    converge = False
    cur_iter = 0
    while not converge:
      cur_iter += 1
      if cur_iter > MAX_ITERATIONS:
            return None, None
        x_new = np.copy(x)
        for i in range(n):
            s1 = sum(A[i][j] * x_new[j] for j in range(i))
            s2 = sum(A[i][j] * x[j] for j in range(i + 1, n))
            # итерационный шаг
            x_new[i] = (b[i] - s1 - s2) / A[i][i]

        # столбец погрешностей
        norm = x_new[i] - x[i]
        # условие на выход из цикла
        converge = sqrt(sum(norm ** 2 for i in range(n))) <= eps
        x = x_new

    return x, norm.tolist(), n_iter

if __name__ == '__main__':
      n = input("Введите размерность: (Enter для 2): ")
      if not n:
            n = 2
      n = int(n)

      eps = input("Введите точность (Enter для 1e-13): ")
      if not eps:
            eps = 1e-13
      eps = float(eps)

      if input("Хотите ввести свои значения коэффициентов? д/н ")[0].lower() == "д":
            matrix = []
            for _ in range(n):
                  matrix.append(list(map(float, input().split())))
            vector = list(map(float, input().split()))
            x, norm, n_iter = seidel(matrix, vector, eps)
            x_format = ', '.join([f"{a:.4E}" for a in x])
            print(f"Итеративный процесс сходится к [{x_format}].")
            print(f"Заняло {n_iter} итераций.")
            norm_format = ', '.join([f"{a:.4E}" for a in norm])
            print(f"Столбец погрешностей: [{norm_format}].")
            print()
      else:
            if input("Хотите использовать случайные значения коэффициентов? д/н ")
[0].lower() == "д":
                  matrix = np.random.random((n, n)).tolist()
```

```
        vector = np.random.random(n).tolist()
        print("\n".join([" ".join(list(map(str, a))) for a in matrix]))
        print(vector)
        x, norm, n_iter = seidel(matrix, vector, eps)
        x_format = ', '.join([f"{a:.4E}" for a in x])
        print(f"Итеративный процесс сходится к [{x_format}].")
        print(f"Заняло {n_iter} итераций.")
        norm_format = ', '.join([f"{a:.4E}" for a in norm])
        print(f"Столбец погрешностей: [{norm_format}].")
        print()
    else:
        matrix2 = [
        [1, 2],
        [0, 0]]
        vector2 = [3, 0]

        matrix6 = [
        [15, 1, -6, 2, 1, 4],
        [1, 40, 3, -8, 4, 3],
        [2, 7, 62, 4, -7, 3],
        [1, 3, 1, 48, 3, -5],
        [4, 7, 5, 5, 99, 6],
        [1, 1, 1, 1, 1, -7]]
        vector6 = [0, 2, 1, 1, 4, -8]

        n, eps = 6, 1e-13
        x, norm, n_iter = seidel(matrix6, vector6, eps)
        x_format = ', '.join([f"{a:.4E}" for a in x])
        print(f"Итеративный процесс сходится к [{x_format}].")
        print(f"Заняло {n_iter} итераций.")
        norm_format = ', '.join([f"{a:.4E}" for a in norm])
        print(f"Столбец погрешностей: [{norm_format}].")
        print()
        n, eps = 2, 1e-13
        x, norm, n_iter = seidel(matrix2, vector2, eps)
        x_format = ', '.join([f"{a:.4E}" for a in x])
        print(f"Итеративный процесс сходится к [{x_format}].")
        print(f"Заняло {n_iter} итераций.")
        norm_format = ', '.join([f"{a:.4E}" for a in norm])
        print(f"Столбец погрешностей: [{norm_format}].")
        print()
```

3. Примеры

6
15 1 -6 2 1 4
1 40 3 -8 4 3
2 7 62 4 -7 3
1 3 1 48 3 -5
4 7 5 5 99 6
1 1 1 1 1 -7

0 2 1 1 4 -8

-3.3016E-01, 8.9092E-03, -3.9430E-02, 1.4474E-01, -1.9431E-02, 1.1092E+00

2

1 2
0 0

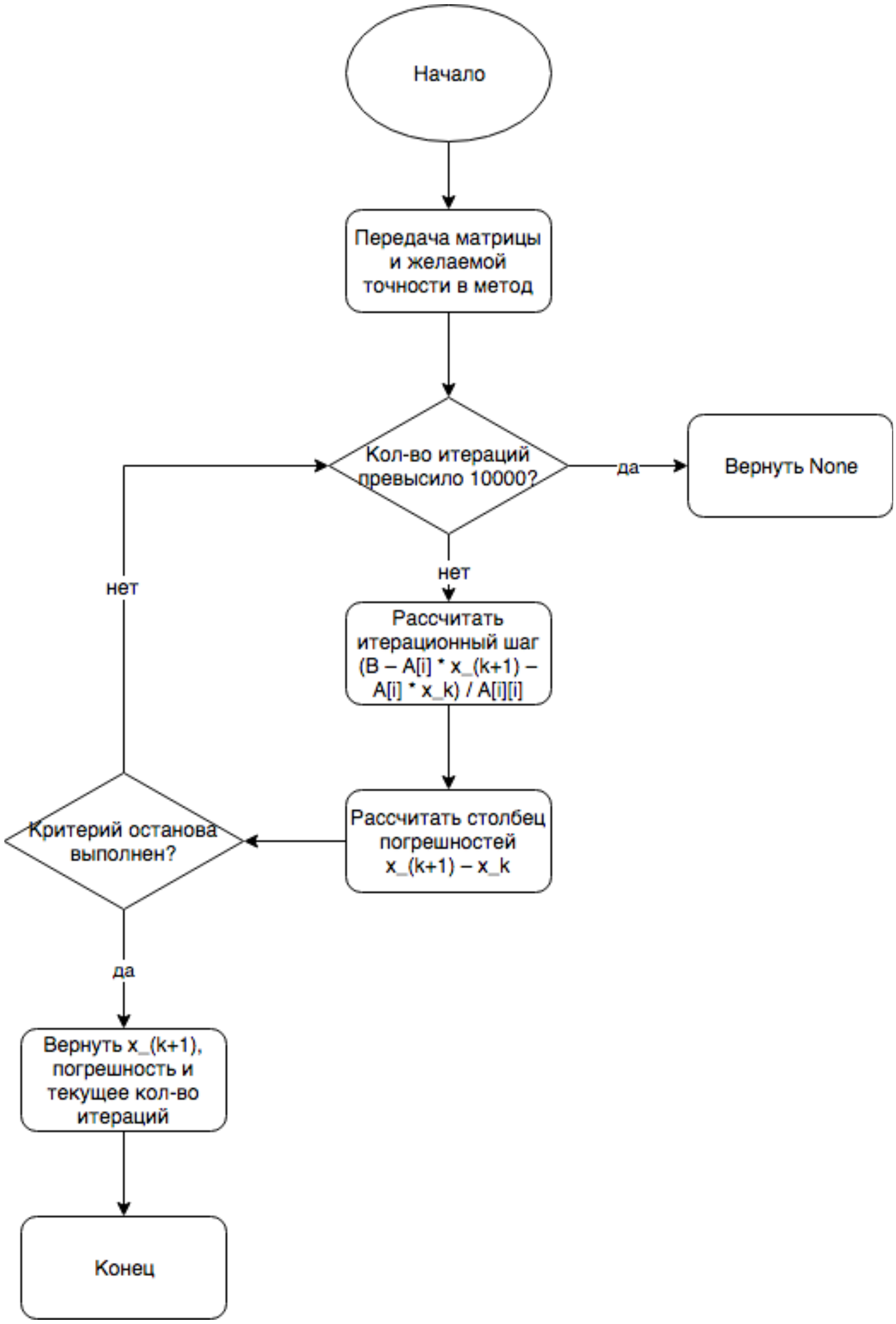
3 0

3.0000E+00, 0.0000E+00

3
0.7286592874963329 0.6842175247955062 0.8634558785876988
0.9218036475487008 0.706502619273745 0.28558474223028596
0.30899201324622816 0.3649644690806928 0.3991518877087774
0.39043399353734254 0.7773724551689634 0.7791683007655045

-5.5031E+00, 9.1518E+00, -2.1559E+00

Вывод: метод Гаусса-Зейделя показал хорошие результаты для выбранных вручную значений коэффициентов, но часто не сходится для случайных значений при высокой точности. Также при бесконечном кол-ве решений он все равно находит одно из них.


Начало
Передача матрицы и желаемой точности в метод
Кол-во итераций превысило 10000?
да
Вернуть None
нет
нет
Рассчитать итерационный шаг (B – A[i] * x_(k+1) – A[i] * x_k) / A[i][i]
Рассчитать столбец погрешностей x_(k+1) – x_k
Критерий останова выполнен?
да
Вернуть x_(k+1), погрешность и текущее кол-во итераций
Конец