Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №2
Численное решение систем линейных уравнений методом простых итераций и методом Зейделя

Выполнил:
студент группы 953501
Кременевский. В.С

Руководитель:
доцент
Анисимов В.Я.

Минск 2021

Содержание

1. Цель работы

1) Изучить итерационные методы решения СЛАУ (метод простых итераций, метод Зейделя).
2) Составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ.
3) Составить программу решения СЛАУ по разработанному алгоритму.
4) Численно решить тестовые примеры и проверить правильность работы программы. Сравнить трудоемкость решения методом простых итераций и методом Зейделя.

2. Теоретический сведения

Прямые методы применяют главным образом для решения задач малой размерности, когда нет ограничений в доступной оперативной памяти ЭВМ или необходимости выполнения чрезмерно большого числа арифметических операций. Большие системы уравнений, возникающие в основном в приложениях, как правило, являются разреженными. Методы исключения для систем с разреженным и матрицами неудобны, например, тем, что при их использовании большое число нулевых элементов превращается в ненулевые, и матрица теряет свойство разреженности. В противоположность им при использовании итерационных методов в ходе итерационного процесса матрица не меняется, и она, естественно, остается разреженной. Большая эффективность итерационных методов по сравнению с прямыми методами тесно связанна с возможностью существенного использования разреженности матриц.

Итерационные методы основаны на построении сходящейся к точному решению x рекуррентной последовательности.

Для решения СЛАУ **методом простых итераций** преобразуем систему от первоначальной формы $Ax = b$ или

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2, \\ \ldots & \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned} \qquad (2.1)$$

к виду

$$x = Bx + c \qquad (2.2)$$

Здесь $\boldsymbol{B}$ – квадратная матрица с элементами b_{ij} $(i, j = 1, 2, ..., n)$, c – вектор-столбец с элементами c_i $(i = 1, 2, ..., n)$.

В развернутой форме записи система (2.2) имеет следующий вид:

Вообще говоря, операция *приведения системы к виду, удобному для итераций*, не является простой и требует специальных знаний, а также существенного использования специфики системы.

Можно, например, преобразовать систему (2.1) следующим образом

$$x_1 = (b_1 - a_{11}x_1 - a_{12}x_2 - \ldots - a_{1n}x_n)/a_{11} + x_1$$

,

$$x_2 = (b_2 - a_{21}x_1 - a_{22}x_2 - \ldots - a_{2n}x_n)/a_{22} + x_2$$

,

если диагональные элементы матрицы $\boldsymbol{A}$ отличны от нуля.

Можно преобразовать систему (2.1) в эквивалентную ей систему

$$x = (\boldsymbol{E} - \boldsymbol{A})\boldsymbol{x} + \boldsymbol{b}$$

Задав произвольным образом столбец начальных приближений $\boldsymbol{x^0} = (x_1^0, x_2^0, ..., x_n^0)^T$, подставим их в правые части системы (2.2) и вычислим новые приближения $\boldsymbol{x^1} = (x_1^1, x_2^1, ..., x_n^1)^T$, которые опять подставим в систему (2.2) и т.д. Таким образом, организуется итерационный процесс

$$\boldsymbol{x^k} = \boldsymbol{B}\boldsymbol{x^{k-1}} + \boldsymbol{c}, \; k = 1, 2, \ldots$$

Известно, что система (2.1) имеет единственное решение $\boldsymbol{x^*}$ и последовательность $\{\boldsymbol{x^k}\}$ сходится к этому решению со скоростью геометрической прогрессии, если $\| \boldsymbol{B} \| < 1$ в любой матричной норме.

Теорема 1. Для того чтобы при любом начальном приближении x^0 итерационная последовательность в методе простых итераций сходилась к Таким образом, получается рекуррентная последовательность в методе простых итераций сходилилась к решению системы, необходимо и достаточно, чтобы все собственные значения матрицы B были по абсолютной величине меньше единицы.

Или спектральный радиус был меньше единицы

$$\mu(A) = \max \{ |\lambda_1|, \ldots., |\lambda_n| \},$$

Также для того, чтобы последовательность простых итераций сходилась к единственному решению достаточно, чтобы выполнялось одно из следующих условий:

1) $$\max_{1\le i\le n}\left(\sum_{j=1}^{n}|b_{ij}|\right)<1$$

2) $$\sum_{i=1}^{n}\sum_{j=1}^{n}b_{ij}^{2}<1$$

3) $$\max_{1\le j\le n}\left(\sum_{i=1}^{n}|b_{ij}|\right)<1$$

Метод Зейделя является модификацией метода простых итераций. Суть его состоит в том, что при вычислении следующего $\boldsymbol{x_i^k}$ в формуле $\boldsymbol{x^k = Bx^{k-1} + c}$, k = 1, 2, … вместо x_1^{k-1}, x_2^{k-1}, …, x_{i-1}^{k-1} используются уже вычисленные x_1^k, x_2^k, …, x_{i-1}^k, т.е.

$$x_i^k = \sum_{j=1}^{i-1} g_{ij}x_j^k + \sum_{j=i+1}^{n} g_{ij}x_j^{k-1} + c_i \qquad (2.3)$$

Такое усовершенствование позволяет ускорить сходимость итерации почти в два раза. Кроме того, данный метод может быть реализован на ЭВМ без привлечения дополнительного массива, т.к. полученное новое x_i^k сразу засылается на место старого.

Схема алгоритма аналогична схеме метода простых итераций. Самый простой способ приведения системы к виду, удобному для итераций, состоит в следующем. Из первого уравнения системы (2.1) выразим неизвестное x_1:

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3 - \ldots - a_{1n}x_n)/a_{11}$$

из второго уравнения выразим неизвестное x_2:

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3 - \ldots - a_{2n}x_n)/a_{22}$$

и т.д. В результате получим систему

$$x_1 = b_{12}x_2 + b_{13}x_3 + \ldots + b_{1n}x_n + c_1$$

$$x_2 = b_{21}x_1 + b_{23}x_3 + \ldots + b_{2n}x_n + c_2$$

,

в которой на главной диагонали матрицы $\boldsymbol{B}$ находятся нули, а остальные

элементы выражаются по формулам

$$b_{ij} = -a_{ij}/a_{ii},\ c_i = b_i/a_{ii},\ (i, j = 1,2, \ldots, n,\ i \neq j)$$

Конечно, для возможности выполнения указанного преобразования необходимо, чтобы диагональные элементы матрицы $\boldsymbol{A}$ были ненулевыми.

Введем нижнюю $\boldsymbol{B_1}$ (получается из $\boldsymbol{B}$ заменой нулями элементов, стоявших на главной диагонали и выше ее) и верхнюю $\boldsymbol{B_2}$ (получается из $\boldsymbol{B}$ заменой нулями элементов, стоявших на главной диагонали и ниже ее) треугольные матрицы.

Заметим, что $\boldsymbol{B} = \boldsymbol{B_1} + \boldsymbol{B_2}$ и поэтому решение $\boldsymbol{x}$ исходной системы удовлетворяет равенству

$$\boldsymbol{x} = \boldsymbol{B}_1\boldsymbol{x} + \boldsymbol{B}_2\boldsymbol{x} + \boldsymbol{c} \quad (2.5)$$

Выберем начальное приближение $\boldsymbol{x}^{(0)} = [x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}]^T$. Подставляя его в правую часть равенства, находим первое приближение

$$\boldsymbol{x}^{(1)} = \boldsymbol{B}_1\boldsymbol{x}^{(1)} + \boldsymbol{B}_2\boldsymbol{x}^{(0)} + \boldsymbol{c} \quad (2.6)$$

Подставляя приближение $\boldsymbol{x}^{(1)}$, получим

$$\boldsymbol{x}^{(2)} = \boldsymbol{B}_1\boldsymbol{x}^{(2)} + \boldsymbol{B}_2\boldsymbol{x}^{(1)} + \boldsymbol{c} \quad (2.7)$$

Продолжая этот процесс далее, получим последовательность x^0), x^, ., x^n), ... приближений к вычисляемых по формуле

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{B}_1\boldsymbol{x}^{(k+1)} + \boldsymbol{B}_2\boldsymbol{x}^{(k)} + \boldsymbol{c} \quad (2.8)$$

Или же

$$\boldsymbol{x}_i^{(k+1)} = \sum_{j=1}^{i-1} \boldsymbol{b}_{ij}\boldsymbol{x}_j^{(k+1)} + \sum_{j=i+1}^{n} \boldsymbol{b}_{ij}\boldsymbol{x}_j^{(k)} + \boldsymbol{c}_i$$

Объединив приведение системы к виду, удобному для итераций и метод Зейделя в одну формулу, получим

$$\boldsymbol{x}_i^{(k+1)} = \boldsymbol{x}_i^{(k)} - \left(\sum_{j=1}^{i-1} \boldsymbol{a}_{ij}\boldsymbol{x}_j^{(k+1)} + \sum_{j=i}^{n} \boldsymbol{a}_{ij}\boldsymbol{x}_j^{(k)} - \boldsymbol{b}_i\right)/\boldsymbol{a}_{ii} \quad (2.9)$$

Тогда достаточным условием сходимости метода Зейделя будет условие *доминирования диагональных элементов в строках или столбцах матрицы* $\boldsymbol{A}$, *т.е.*

$a_{ii} > a_{i1} + \ldots + a_{in}$ *для всех* $i = 1, 2, \ldots, n$
или
$a_{jj} > a_{1j} + \ldots + a_{nj}$ *для всех* $j = 1, 2, \ldots, n$

Методы простой итерации и Зейделя сходятся примерно так же, как геометрическая прогрессия со знаменателем || ***B*** ||.

Ошибку будем высчитывать как:

error = A$\mathbf{x}^{(k)}$ - **b**

Где A - исходная матрица, b - вектор-столбец исходных отвветов, а $\mathbf{x}^{(k)}$ - Приближение на k-ой итерации.

И будем производить вычисления до тех пор, пока точность **error** будет больше заданной по условию Ė.

3. Тестовые примеры

Тестовый пример 1

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение СЛАУ:

$$\begin{cases} 9.2x + 2.5y - 3.7z = -17.5 \\ 0.9x + 9y + 0.2z = 4.4 \\ 4.5x - 1.6y - 10z = -22 \end{cases}$$

Решение:

```
Default matrix
  9.20  2.50 -3.70 --> -17.50
  0.90  9.00  0.20 -->   4.40
  4.50 -1.60-10.30 --> -22.10

Real Accurate solution *X*
[-1.50758786  0.60870501  1.39242006]

-------------------------------------------------
X solution converges by spectrum
Spectrum of matrix: 0.4744078880917683
||B||_0 = 0.673913043478261
||B||_1 = 0.5368932038834952
||B||_F = 0.23054298268802914
-------------------------------------------------
*** Iteration method result ***
X = [[-1.50758856  0.60869464  1.39246606]]
 Done in 12 iterations
-------------------------------------------------
X solution converges by spectrum
Spectrum of matrix: 0.4744078880917683
||B||_0 = 0.673913043478261
||B||_1 = 0.5368932038834952
-------------------------------------------------
*** Seidel method result ***
X = [[-1.5076405   0.60871267  1.39239587]]
Done in 7 iterations
```

Тестовый пример 2

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение СЛАУ:

$$\begin{cases} 0x + 2y + 4z = 16 \\ x + 0y + 2z = 7 \\ 2x + 2y + 0z = -6 \end{cases}$$

Решение:

```
Default matrix
  0.00  2.00  4.00 -->  16.00
  1.00  0.00  2.00 -->   7.00
  2.00  2.00  0.00 -->   6.00

Real Accurate solution *X*
[1. 2. 3.]

-------------------------------------------------
Ahh noo, zeros on main diagonal
X solution not converges
Traceback (most recent call last):
  File "/Users/kremenevskiy/Desktop/labs/Term_4-numerical-a
    x_iteration = iteration_method(A, b, e, 1)
  File "/Users/kremenevskiy/Desktop/labs/Term_4-numerical-a
    raise Exception("Can't find roots, bcs not converges")
Exception: Can't find roots, bcs not converges
```

Как видно, данная система имеет решение прямыми методами, но так как нормы матрицы B не удовлетворяют условиям сходимости, нельзя найти решение ни методом просых итераций ни метод Гаусса-Зейделя.

Тестовый пример 3

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение СЛАУ:

$$\begin{cases} 10x + 1y + 1z = 12 \\ 2x + 10y + 1z = 13 \\ 2x + 2y + 10z = 14 \end{cases}$$

Решение:

```
Default matrix
 10.00  1.00  1.00 -->  12.00
  2.00 10.00  1.00 -->  13.00
  2.00  2.00 10.00 -->  14.00

Real Accurate solution *X*
[1. 1. 1.]

-------------------------------------------------
X solution converges by spectrum
Spectrum of matrix: 0.28473221018630734
||B||_0 = 0.4
||B||_1 = 0.4
||B||_F = 0.07500000000000001
-------------------------------------------------
*** Iteration method result ***
X = [[0.99999717 0.99999644 0.99999551]]
 Done in 9 iterations
-------------------------------------------------
X solution converges by spectrum
Spectrum of matrix: 0.28473221018630734
||B||_0 = 0.4
||B||_1 = 0.4
-------------------------------------------------
*** Seidel method result ***
X = [[0.99999958 0.9999996  1.00000016]]
Done in 5 iterations
```

Тестовый пример 4

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение СЛАУ:

$$\begin{cases} 10x + y - z = 11 \\ 2.7x + 10y + -z = 10 \\ -x + y + 10z = 10 \end{cases}$$

Решение:

```
Default matrix
 10.00  1.00 -1.00 -->  11.00
  2.70 10.00 -1.00 -->  10.00
 -1.00  1.00 10.00 -->  10.00

Real Accurate solution *X*
[1.1232369  0.79995882 1.03232781]

-------------------------------------------------
X solution converges by spectrum
Spectrum of matrix: 0.1896424004376895
||B||_0 = 0.37
||B||_1 = 0.37
||B||_F = 0.06145000000000002
-------------------------------------------------
*** Iteration method result ***
X = [[1.12323285 0.79996262 1.03232375]]
 Done in 6 iterations
-------------------------------------------------
X solution converges by spectrum
Spectrum of matrix: 0.1896424004376895
||B||_0 = 0.37
||B||_1 = 0.37
-------------------------------------------------
*** Seidel method result ***
X = [[1.12323655 0.79995888 1.03232777]]
Done in 5 iterations
```

Тестовый пример 4

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение СЛАУ:

$$\begin{cases} 1.40x_1 + 0.21x_2 + 0.03x_3 + 0.12x_4 - 0.13x_5 = 1.2 \\ -0.06x_1 - 1.26x_2 - 0.03x_3 + 0.17x_4 + 0.12x_5 = 2.2 \\ 0.12x_1 - 0.06x_2 - 1.26x_3 + 0.11x_4 + 0.03x_5 = 4.0 \\ 0.17x_1 + 0.12x_2 - 0.06x_3 - 1.26x_4 + 0.11x_5 = 0 \\ 0.11x_1 + 0.67x_2 + 0.12x_3 - 0.06x_4 - 1.26x_5 = -1.2 \end{cases}$$

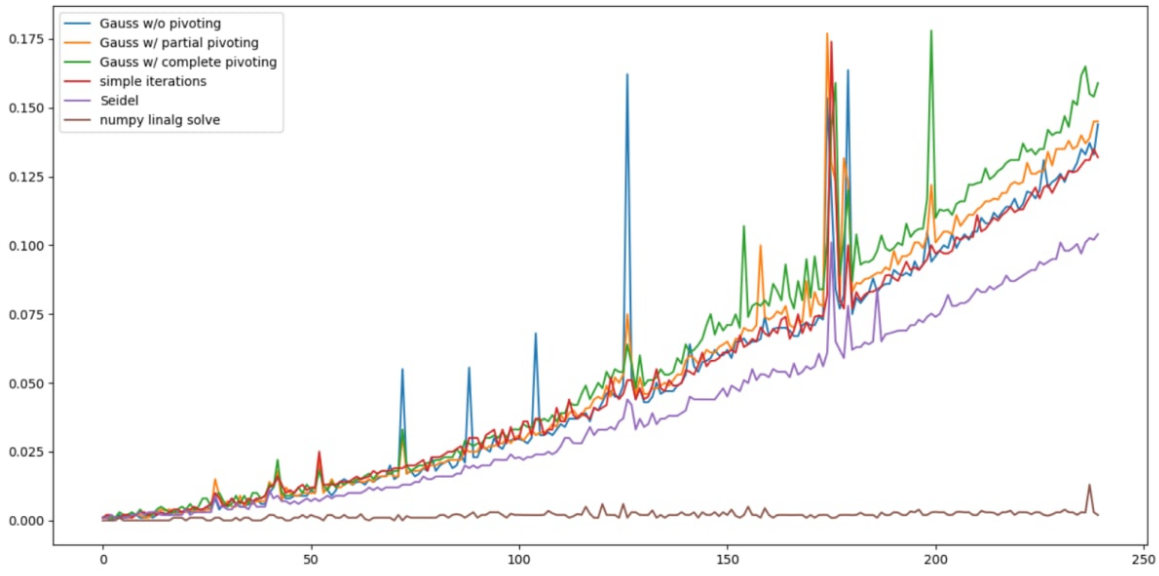
```
Default matrix
  1.40  0.21  0.03  0.12 -0.13 -->   1.20
 -0.06 -1.26 -0.03  0.17  0.12 -->   2.20
  0.12 -0.06 -1.26  0.11  0.03 -->   4.00
  0.17  0.12 -0.06 -1.26  0.11 -->   0.00
  0.11  0.67  0.12 -0.06 -1.26 -->  -1.20

Real Accurate solution *X*
[ 1.15559594 -1.72887854 -2.97548796  0.11940599 -0.15512379]

-------------------------------------------------------------------------------
X solution converges by spectrum
Spectrum of matrix: 0.29232484069739023
||B||_0 = 0.7619047619047619
||B||_1 = 0.8246031746031746
||B||_F = 0.21371976568405138
-------------------------------------------------------------------------------
*** Iteration method result ***
X = [[ 1.15559937 -1.72887308 -2.97548754  0.11940699 -0.15512738]]
 Done in 9 iterations
-------------------------------------------------------------------------------
X solution converges by spectrum
Spectrum of matrix: 0.29232484069739023
||B||_0 = 0.7619047619047619
||B||_1 = 0.8246031746031746
-------------------------------------------------------------------------------
*** Seidel method result ***
X = [[ 1.15560794 -1.72888527 -2.97548961  0.11940443 -0.15512641]]
Done in 5 iterations
```

Метод простых итераций	Метод Зейделя
$x_1 = 1.1556$ $x_2 = -1.7289$ $x_3 = -2.9755$ $x_4 = 0.1194$ $x_5 = -0.1551$	$x_1 = 1.1556$ $x_2 = -1.7289$ $x_3 = -2.9755$ $x_4 = 0.1194$ $x_5 = -0.1551$
Количество итераций	
9	5

Выводы Тестовых примеров:

Как видно из тестовых примеров, данные алгоритмы корректно отрабатывают, если наша матрица B соответсвует одному из условий сходимости (Примеры 1, 3, 4, 5). А также если неудовлетворяет - решение не может быть получено, хотя точными методами легко решается (Пример 2)

Все результаты сверялись с точным решение **X*,** что есть решение методом numpy.linalg.solve(A, b), поэтому можно судить что написанные алгоритмы работают корректно.



На графике изображена зависимость количества переменных в уравнении (x) от времени подсчета решений с точностью E = 0.0001 (y)

Как видно, поставленная функция в numpy.linalg.solve работает стабильно за O(n), в то время как метод Зейделя и метод простых итераций работают гораздо хуже при больших n, но метод Зейделя все таки растет не так быстро как метод простых итераций, что свидетельствует о более быстрой сходимости по методу Зейделя, но оба алгоритма работают за квадратичное время - O (n^2)

Метод Гаусса, дающий точное решение, работает еще медленней, его порядок роста ~ O(n^3) но за счет больших констант при методе зейделя и методе простых итераций в размере до n = 250 разница не такая значительная, возможно эти методы покажут лучший результат при n > 1000.

4. Выполнение задания

Методом простых итераций и методом Зейделя найти с точностью 0,0001 численное решение системы $\boldsymbol{Ax=b}$:

$$\begin{cases} 1.43 * x1 + 0.21 * x2 - 0.03 * x3 + 0.12 * x4 - 0.13 * x5 = 1.2 \\ -0.03 * x1 - 1.23 * x2 - 0.09 * x3 + 0.17 * x4 + 0.12 * x5 = 2.2 \\ 0.12 * x1 - 0.03 * x2 - 1.23 * x3 + 0.11 * x4 - 0.03 * x5 = 4 \\ 0.17 * x1 + 0.12 * x2 - 0.03 * x3 - 1.23 * x4 + 0.11 * x5 = 0 \\ 0.11 * x1 + 0.67 * x2 + 0.12 * x3 - 0.03 * x4 - 1.23 * x5 = -1.2 \end{cases}$$

Решение

```
Default matrix
  1.43  0.21 -0.03  0.12 -0.13 -->   1.20
 -0.03 -1.23 -0.09  0.17  0.12 -->   2.20
  0.12 -0.03 -1.23  0.11 -0.03 -->   4.00
  0.17  0.12 -0.03 -1.23  0.11 -->   0.00
  0.11  0.67  0.12 -0.03 -1.23 -->  -1.20

Real Accurate solution *X*
[ 0.99357836 -1.58890201 -3.1094134   0.04870638 -0.10557848]

-------------------------------------------------------------------------------
X solution converges by spectrum
Spectrum of matrix: 0.2882240463492627
||B||_0 = 0.7560975609756099
||B||_1 = 0.8135198135198136
||B||_F = 0.2213133175285241
-------------------------------------------------------------------------------
*** Iteration method result ***
X = [[ 0.99356839 -1.58890134 -3.10941043  0.04871417 -0.1055505 ]]
 Done in 8 iterations
-------------------------------------------------------------------------------
X solution converges by spectrum
Spectrum of matrix: 0.2882240463492627
||B||_0 = 0.7560975609756099
||B||_1 = 0.8135198135198136
-------------------------------------------------------------------------------
*** Seidel method result ***
X = [[ 0.99358031 -1.58890293 -3.10941325  0.04870591 -0.10557878]]
Done in 6 iterations
```

Как видно из результатов полученное решение совпадает с точным в соответсвии с точностью равной 0.0001.

Также видно, что решение методом Гаусса-Зейделя сходится быстрее:
методом простых итераций требуется 8 итераций,
методом Гаусса-Зейделя всего 6.

5. Вывод

Таким образом, в ходе выполнения лабораторной работы я изучили итерационные методы решения СЛАУ (метод простых итераций, метод Зейделя), составил алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ, составили программу решения СЛАУ по разработанному алгоритму, численно решили тестовые примеры и проверили правильность работы программы, сравнили трудоемкость решения методом простых итераций и методом Зейделя.

Также научился вычислять результаты с заданной точностью. Изучил понятие норм, а также основные нормы, используемые при выяснение сходимости решения к заданным при любых начальных значениях.

Также хочется отметить такой термин как спектрум матрицы, который дает достаточное улсовие сходимости матрицы.

```
    # printing norms
    if verbose == 1:
        print(f'Spectrum of matrix: {find_spectrum(B)}')
        print(f'||B||_0 = {norm_row(B)}')
        print(f'||B||_1 = {norm_column(B)}')
        print(f'||B||_F = {norm_quad(B)}')

    errors_x = np.empty(shape=n)
    errors_y = np.zeros(shape=n)
    e = 1
    x_exact = np.linalg.solve(A, b).reshape((n, 1))

    x_prev = np.empty(shape=(n, 1))
    x_current = c.copy()

    iteration = 0
    while e > error:
        iteration += 1
        x_prev = x_current.copy()

        for i in range(n):
            x_current[i] = c[i]
            for j in range(n):
                x_current[i][0] += B[i, j] * x_prev[j][0]

        if verbose == 2:
            print(f'{iteration}. --- {x_current.reshape((1, n))}')

        # calculate error
        for i in range(n):
            errors_x[i] = abs(x_prev[i][0] - x_current[i][0])
        errors_y = abs(A.dot(x_current).reshape(n, ) - b)
        e = np.amax(errors_x) + np.amax(errors_y)

    return x_current, iteration
```

6. Программная реализация

Алгоритм метода простых итераций:

```python
def iteration_method(matrix_a, answers_b, error=0.0001, verbose=0) -> (np.array, int):
    A = np.array(matrix_a, dtype=float)
    b = np.array(answers_b, dtype=float)

    if A.shape[0] != A.shape[1]:
        raise Exception("Array A must be n*n size")

    n = A.shape[0]

    A = fix_zeros_main_diagonal(A)

    B = np.empty(shape=A.shape)
    for i in range(n):
        for j in range(n):
            if i == j:
                B[i, j] = 0
                continue
            B[i, j] = (-1) * A[i, j] / A[i, i]

    c = np.empty(shape=b.shape[0])
    for i in range(c.shape[0]):
        c[i] = b[i] / A[i, i]
    c = c.reshape((n, 1))

    if verbose == 2:
        # print(f'Eigenvalues = {np.linalg.eig(B)[0]}')
        eigenvalues = np.linalg.eig(B)[0]
        print('Eigenvalues:')
        for i, eig in enumerate(eigenvalues):
            print(f'{i}. {eig}')

    if check_convergence(B):
        if verbose == 1:
            print('X solution converges by spectrum')
    elif norm_convergence(B):
        if verbose == 1:
            print('X solution converges by its norm')
    else:
        if verbose == 1:
            print("X solution not converges")
        raise Exception("Can't find roots, bcs not converges")
```

Метод Гаусса-Зейделя:

```python
def seidel_method(matrix_a, answers_b, error=0.0001, verbose=0) -> (np.array, int):
    A = np.array(matrix_a, dtype=float)
    b = np.array(answers_b, dtype=float)

    # print(A.dot(x) )

    if A.shape[0] != A.shape[1]:
        raise Exception("Array A must be n*n size")

    n = A.shape[0]

    fix_zeros_main_diagonal(A)

    B = np.empty(shape=A.shape)
    for i in range(n):
        for j in range(n):
            if i == j:
                B[i, j] = 0
                continue
            B[i, j] = (-1) * A[i, j] / A[i, i]

    c = np.empty(shape=b.shape[0])
    for i in range(c.shape[0]):
        c[i] = b[i] / A[i, i]
    c = c.reshape((n, 1))

    if verbose == 2:
        eigenvalues = np.linalg.eig(B)[0]
        print('Eigenvalues:')
        for i, eig in enumerate(eigenvalues):
            print(f'{i}. {eig}')

    if check_convergence(B):
        if verbose == 1:
            print('X solution converges by spectrum')
    elif norm_trace(B):
        if verbose == 1:
            print('X solution converges by its norm trace')
    else:
        if verbose == 1:
            print("X solution not converges")
        raise Exception("Can't find roots, has not convenges")
```

```
    raise Exception("Can't find roots, bcs not converges")

# printing norms
if verbose == 1:
    print(f'Spectrum of matrix: {find_spectrum(B)}')
    print(f'||B||_0 = {norm_row(B)}')
    print(f'||B||_1 = {norm_column(B)}')
    if norm_trace(B):
        print(f'for 1<i<n : | B[i][i] | > sum( B[i][j], 1 < j < n)')

errors = np.empty(shape=n)
e = 1
x_exact = np.linalg.solve(A, b).reshape((n, 1))

x_current = np.zeros(shape=c.shape, dtype=float)
x_current[0, 0] = c[0, 0]

iteration = 0
while e > error:
    iteration += 1

    error_x = 0  # set big value for error

    for i in range(n):
        x_prev = x_current[i, 0]  # save prev to calc error
        x_current[i, 0] = c[i, 0]

        for j in range(n):
            x_current[i][0] += B[i, j] * x_current[j][0]

        if abs(x_prev - x_current[i, 0]) > error_x:
            error_x = abs(x_prev - x_current[i, 0])

    if verbose == 2:
        print(f'{iteration}. --- {x_current.reshape((1, n))}')

    errors_y = abs(A.dot(x_current).reshape((n, )) - b)
    e = (np.amax(errors_y) + error_x) / 2

return x_current, iteration
```

Алгоритмы вычисления норм и других вспомогательных функций использовавшихся в программе:

```python
def norm_column(matrix_a):
    A = np.array(matrix_a, dtype=float)
    n = A.shape[0]
    norms = np.zeros(shape=A.shape)

    for j in range(n):
        for i in range(n):
            norms[j] += abs(A[i, j])

    norm = np.amax(norms)
    return norm


def norm_row(matrix_a):
    A = np.array(matrix_a, dtype=float)
    n = A.shape[0]
    norms = np.zeros(shape=A.shape)

    for i in range(n):
        for j in range(n):
            norms[i] += abs(A[i, j])

    norm = np.amax(norms)
    return norm
```

```python
def norm_quad(matrix_a):
    A = np.array(matrix_a, dtype=float)
    n = A.shape[0]
    quad = 0

    for i in range(n):
        for j in range(n):
            quad += abs(A[i, j]) ** 2

    quad = quad ** 1/2

    return quad


def norm_trace(matrix_a):
    A = np.array(matrix_a, dtype=float)
    n = A.shape[0]
    for i in range(n):
        row_sum = 0
        col_sum = 0
        for j in range(n):
            if i == j:
                continue
            row_sum += abs(A[i, j])
            col_sum += abs(A[j, i])
        if (row_sum < abs(A[i, i])) | (col_sum < abs(A[i, i])):
            continue
        else:
            return False
    return True
```

```python
def fix_zeros_main_diagonal(matrix_a):

    A = np.array(matrix_a, dtype=float)
    n = A.shape[0]
    # Check for zeros on main diagonal
    for i in range(n):
        if A[i, i] == 0:
            find = False
            print(f'Ahh noo, zeros on main diagonal')
            for j in range(n):
                if i == j:
                    continue
                if A[j, i] != 0:
                    A[[j, i]] = A[[i, j]]
                    find = True
            if not find:
                raise Exception(f'Zeros (0) on main diagonal!\nPlace: A[{i}, {i}] Can\'t calculate solution')
    return A
```

```python
def norm_convergence(matrix_a):
    A = np.array(matrix_a, dtype=float)

    if (norm_column(A) < 1) | (norm_row(A) < 1) | (norm_quad(A) < 1):
        return True
    return False


def find_spectrum(matrix_a):
    A = np.array(matrix_a, dtype=float)
    eigen_values = np.linalg.eig(A)[0]

    max_eig = np.amax(abs(eigen_values))
    return max_eig


def check_convergence(matrix_a):
    A = np.array(matrix_a, dtype=float)

    if find_spectrum(A) < 1:
        return True
    return False
```