Университет ИТМО

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия
Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №2

Вариант: 2аг

Выполнил:

Амири Зикрулло

P32211

Преподаватель:

Перл Ольга

Вячеславовна

Санкт-Петербург, 2023 г

Описание метода

Решение нелинейных уравнений:

метод деления пополам

простейший численный метод для решения нелинейных уравнений вида f(x)=0. Предполагается только непрерывность функции f(x). Поиск основывается на теореме о промежуточных значениях

метод простой итерации

один из простейших численных методов решения уравнений. Метод основан на принципе сжимающего отображения, который применительно к численным методам в общем виде также может называться методом простой итерации или методом последовательных приближений. Где следующий шаг итерации вычисляется при помощи предыдущего. Выходом из итерационного процесса является условие:

x_k1 – x_k0 <= eps

Решение систем нелинейных уравнений:
метод простой итерации

Пусть дана система нелинейных уравнений специального вида

$$\left.\begin{aligned} x_1 &= \varphi_1(x_1, x_2, \ldots, x_n), \\ x_2 &= \varphi_2(x_1, x_2, \ldots, x_n), \\ &\ldots\ldots\ldots\ldots \\ x_n &= \varphi_n(x_1, x_2, \ldots, x_n), \end{aligned}\right\} \qquad (1)$$

где функции $\varphi_1, \varphi_2, \ldots, \varphi_n$ действительны, определены и непрерывны в некоторой окрестности ω изолированного решения $(x_1^*, x_2^*, \ldots, x_n^*)$ этой системы.

Введя в рассмотрение векторы

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \quad \text{и} \quad \boldsymbol{\varphi}(\boldsymbol{x}) = (\varphi_1(\boldsymbol{x}), \varphi_2(\boldsymbol{x}), \ldots, \varphi_n(\boldsymbol{x})),$$

систему (1) можно записать более кратко:

$$\boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{x}). \qquad (2)$$

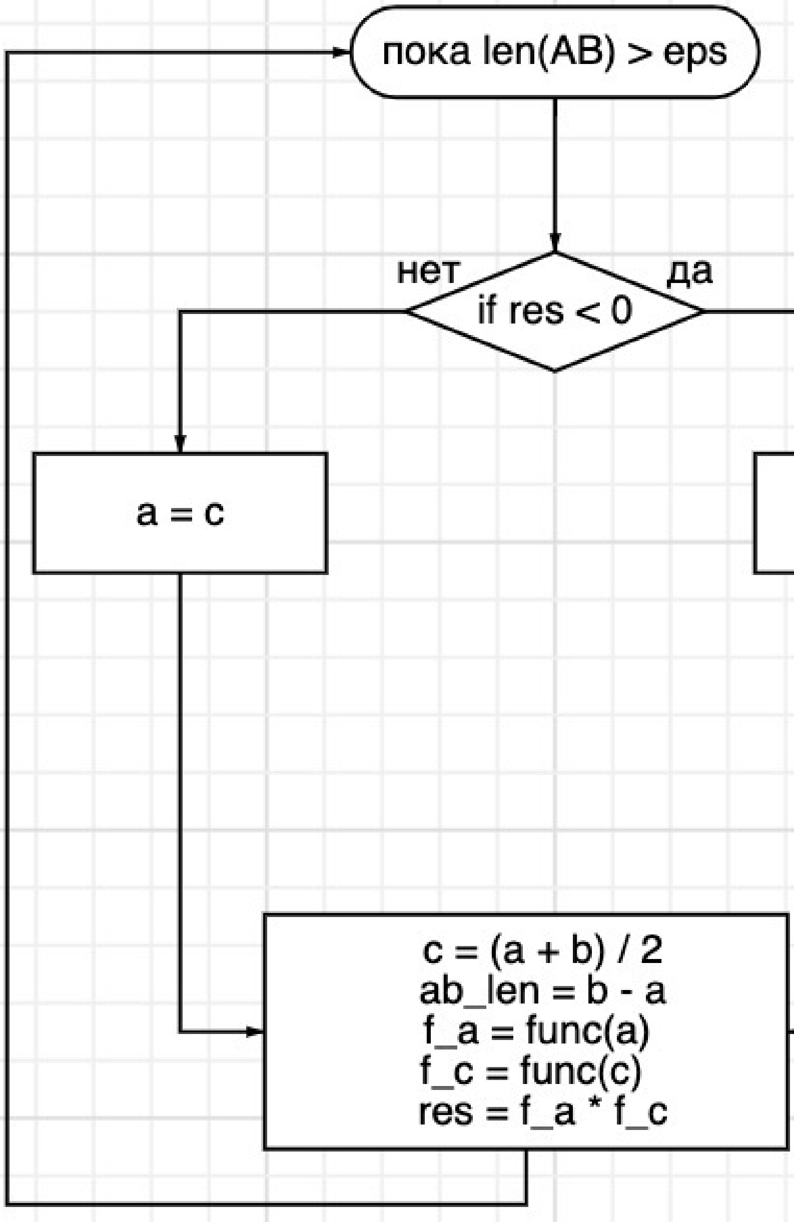
Для нахождения вектор-корня $\boldsymbol{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ уравнения (2) часто удобно использовать *метод итерации*

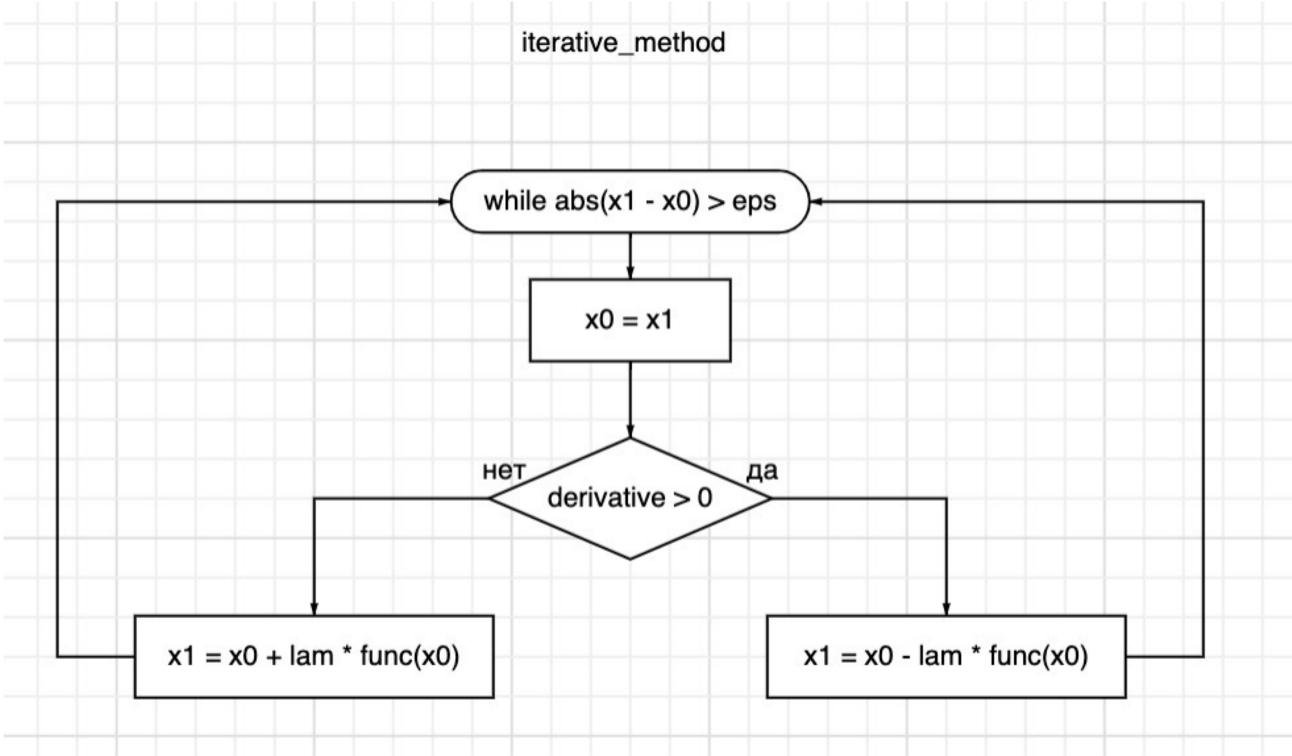
$$\boldsymbol{x}^{(p+1)} = \boldsymbol{\varphi}(\boldsymbol{x}^{(p)}) \qquad (p = 0, 1, 2, \ldots), \qquad (3)$$

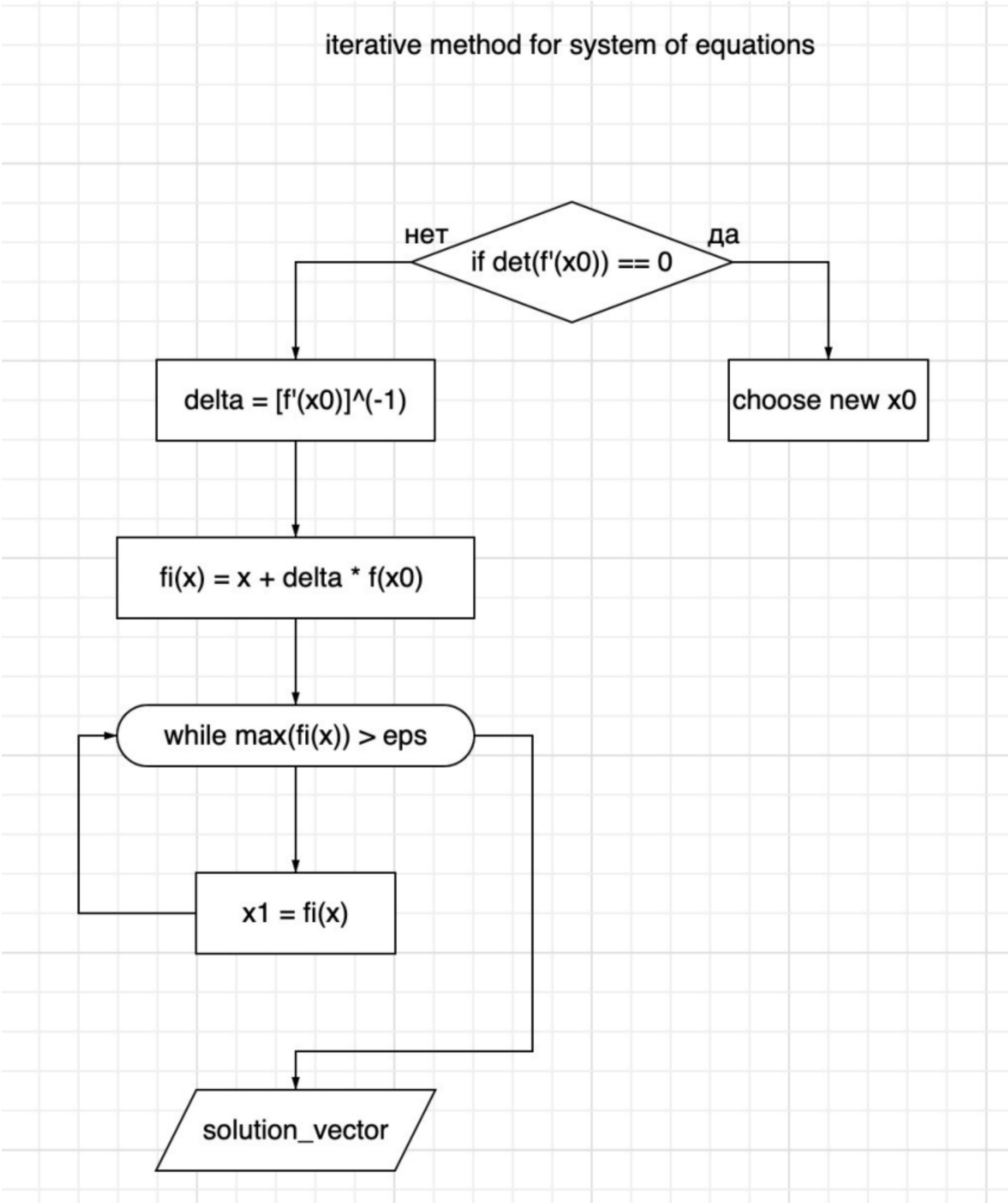
Max(x_k1 – x_k0) <= eps

Блок схема

bisection_method




iterative_method
while abs(x1 - x0) > eps
x0 = x1
нет
да
derivative > 0
x1 = x0 + lam * func(x0)
x1 = x0 - lam * func(x0)


iterative method for system of equations
нет
if det(f'(x0)) == 0
да
delta = [f'(x0)]^(-1)
choose new x0
fi(x) = x + delta * f(x0)
while max(fi(x)) > eps
x1 = fi(x)
solution_vector

Метод деления пополам для решения уравнения.

```
def bisection_method(equation, interval):
    accuracy = 0.001
    left = interval[0]
    right = interval[1]
    mid = (left + right) / 2
    lenth = right - left
    res = equation(left) * equation(mid)

    while lenth >= accuracy:
        if res < 0:
            right = mid
        else:
            left = mid
        mid = (left + right) / 2
        lenth = right - left
        res = equation(left) * equation(mid)

    return mid
```

Метод простой итерации для решения уравнения.

```
def iter_method(equation, interval):
    left, right = interval

    def g(x):
        return x - equation(x)

    x = (left + right) / 2
    max_iterations = 100
    epsilon = 1e-6

    for _ in range(max_iterations):
        try:
            x_next = g(x)
        except:
            return 'Numerical result out of range'
        if abs(x_next - x) < epsilon:
            return x_next
        x = x_next
    return x_next
```

```python
def solve_by_fixed_point_iterations(system_id, number_of_unknowns, initial_approximations):
    funcs = get_functions(system_id)
    result = initial_approximations
    for i in range(100):

        jacobian_matrix = [[det(result, funcs[findx], j) for j in range(number_of_unknowns)] for findx in
                           range(number_of_unknowns)]
        new_f = [fx(result) for fx in funcs]
        for jac in range(number_of_unknowns):
            jacobian_matrix[jac][jac] += 1e-5
        matrix_T = reversion_of_matrix(jacobian_matrix)
        l = vector(matrix_T, new_f)
        result = [result[i] - l[i] for i in range(number_of_unknowns)]
        if all(1e-3 > abs(l[i]) for i in range(number_of_unknowns)):
            return result
    return None
```

Результаты:

```
Введите номер системы: 1
Введите правую и левую границу: 0 3
Результат методом деления пополам: 0.5233154296875
Результат методом простой итерации: 0.5235989167400953
```

```
Введите номер системы: 3
Введите правую и левую границу: 2 5
Результат методом деления пополам: 3.9998779296875
Результат методом простой итерации: 3.984486032507109
```

Вывод:

В ходе выполнения лабораторной работы я изучил метод простых итераций и метод деления пополам. Метод деления пополам основан на принципе разделения интервалов, в котором находится корень, пополам, до тех пор, пока точность не будет достигнута. Если знать знаки функции на концах интервала и они разные, то корень обязательно находится внутри интервала.

Метод простых итераций заключается в поиске корня путем последовательных приближений. Мы выбираем функцию, которая переписывает уравнение в виде $x = g(x)$, итеративно обновляем x до тех пор, пока не достигнем необходимой точности.