МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПИИКТ

ЛАБОРАТОРНАЯ РАБОТА № 3

по дисциплине

‘Вычислительная математика’

Вариант:

Метод простой итерации

Выполнил:
Студент группы P3233
Хасаншин Марат Айратович

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2024

Описание численного метода:

Метод простой итерации используется для решения нелинейных систем или уравнений.

Пусть у нас есть система такого вида

$$\begin{cases} F_1(x_1, x_2, \dots, x_n) = 0 \\ F_2(x_1, x_2, \dots, x_n) = 0 \\ \dots \\ F_n(x_1, x_2, \dots, x_n) = 0 \end{cases}$$

Далее, выбрав начальные приближения $x_1, x_2, \dots, x_n$, мы находим новые приближения (y_i) как

$$y_i = x_i + c_i * (F_i(x_1, x_2, \dots, x_n) - x_i)$$

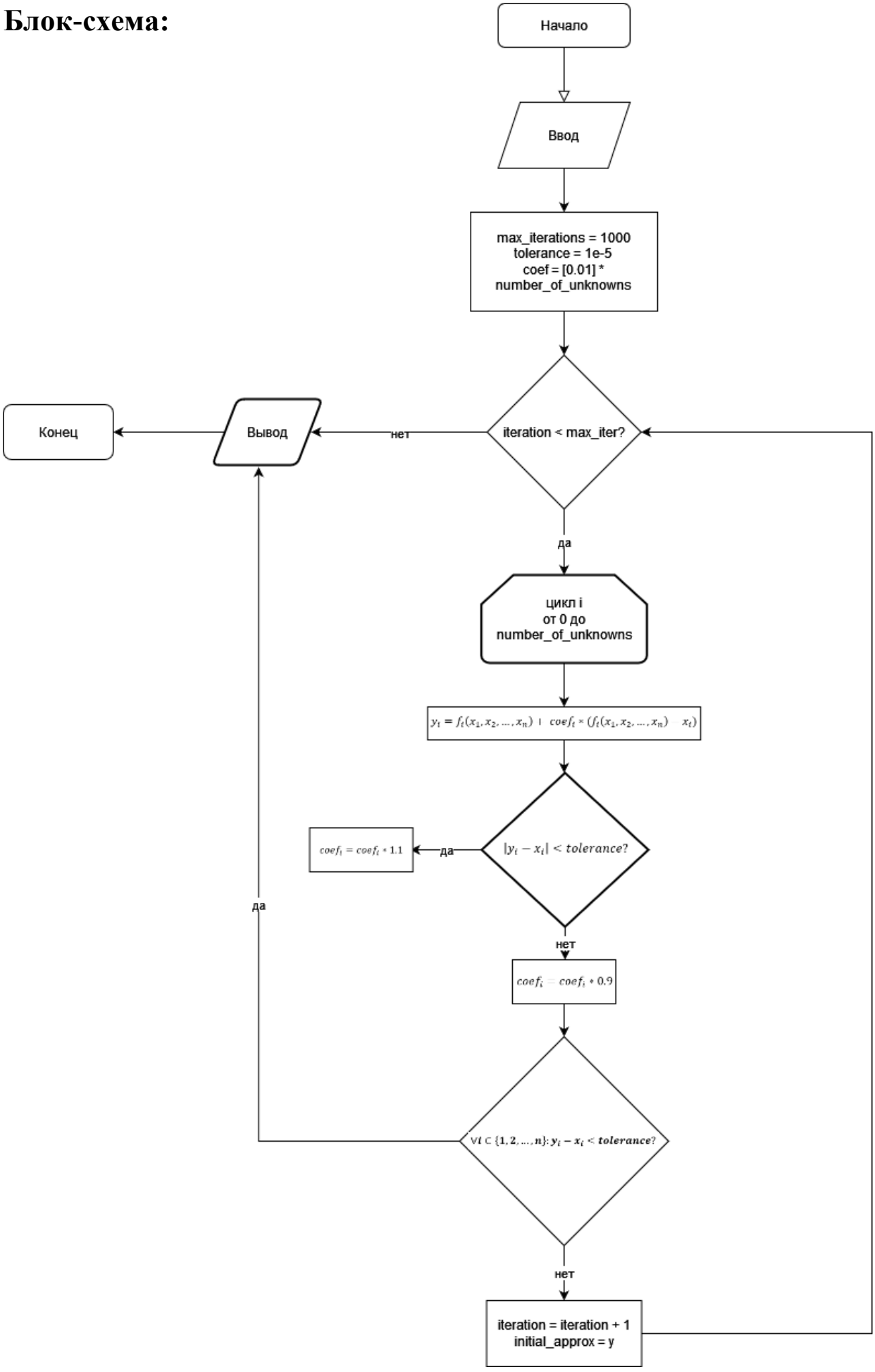
где $c_i = 0.01$, для всех i.

Сравниваем модуль разности нового приближения и старого, если он меньше требуемой погрешности – $c_i = c_i * 1.1$, если нет – $c_i = c_i * 0.9$

Далее, если модуль разности нового приближения и старого меньше требуемой погрешности – возвращаем новое приближение, если же нет – выбираем новые приближения, используя уже новые полученные значения.

Так же ограничиваем количество итераций сверху.

Блок-схема:


Начало
Ввод
max_iterations = 1000
tolerance = 1e-5
coef = [0.01] *
number_of_unknowns
iteration < max_iter?
нет
Вывод
Конец
да
цикл i
от 0 до
number_of_unknowns
$y_i = f_i(x_1, x_2, \ldots, x_n) + coef_i \cdot (f_i(x_1, x_2, \ldots, x_n) - x_i)$
$|y_i - x_i| < tolerance?$
да
$coef_i = coef_i * 1.1$
да
нет
$coef_i = coef_i * 0.9$
$\forall i \subset \{1, 2, \ldots, n\}: y_i - x_i < tolerance?$
нет
iteration = iteration + 1
initial_approx = y

Код:

```
def solve_by_fixed_point_iterations(system_id, number_of_unknowns,
initial_approximations):
    list_of_func = get_functions(system_id)
    tolerance = 1e-5
    max_iter = 1000
    iterations = 0
    new_approximations = [0.0] * number_of_unknowns
    coef = [0.01] * number_of_unknowns
    while iterations < max_iter:
        for i in range(number_of_unknowns):
            new_approximation = list_of_func[i](initial_approximations)
            compare_approx = initial_approximations[i] + coef[i] *
(new_approximation - initial_approximations[i])
            new_approximations[i] = compare_approx

            if (abs(new_approximation - initial_approximations[i]) <
tolerance):
                coef[i] = coef[i] * 1.1
            else:
                coef[i] = coef[i] * 0.9
        flag = True
        for i in range(number_of_unknowns):
            if (abs(new_approximations[i] - initial_approximations[i]) >=
tolerance):
                flag = False
                break
        if (flag):
            return new_approximations
        else:
            iterations += 1
            initial_approximations = new_approximations[:]
    return new_approximations
```

Примеры работы программы:

№1:

```
1
2
1
1
0.984487080488554
0.9508224134762776
```

№2:

```
1
2
0
0
0.0
0.0
```

№3:

```
4
3
2
2
1
2.9864437937256985
2.9710079478014775
1.8207591339314753
```

№4:

```
5
1
1
0.9046067224022992
```

№5:

```
3
2
1
2
0.7497678325362239
2.8497710530674922
```

Вывод:

Тесты 1, 5, 3 показывает решения для систем с 2, 3 и 2 переменными соответственно.
Тест 2 показывает случай, где метод неприменим.

Тест 4 показывает не валидный ввод.

В сравнение с другими методами можно выделить следующие минусы и плюсы:

Минусы:

- Медленно сходится (например, по сравнению с методом Ньютона), в некоторых случаях может вообще не сойтись
- Очень зависит от правильного выбора начальных приближений

Плюсы:

- Метод легок в реализации
- Применим для большого количество СНАУ

Алгоритм может быть применен, если последовательность сходится.

Алгоритмическая сложность метода – O(nk), где n – число переменных, k – количество итераций.

Численная погрешность из-за погрешности при работе с нецелыми числами.