Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Вычислительная математика

Лабораторная работа №3.

Метод Простых Итерации

Выполнил:

Маликов Глеб Игоревич

Группа № P3224

Преподаватели:

Перл Ольга Вячеславовна

Хохлов Александр Алексеевич

г. Санкт-Петербург

2024

Оглавление

Задание

Метод простых итераций

Дана система нелинейных уравнений. По заданному начальному приближению необходимо найти решение системы с точностью до 5 верного знака после запятой при помощи метода простых итераций.

Формат входных данных:

```
k

n

x0

y0

...
```

где k - номер системы, n - количество уравнений и количество неизвестных, а остальные значения - начальные приближения для соответствующих неизвестных.

Формат выходных данных: список такого же типа данных, как списки входных данных, содержащие значения корня для каждой из неизвестных с точностью до 5 верного знака.

Описание численного метода

Метод простых итераций — это один из численных методов решения систем нелинейных уравнений. Он основан на преобразовании исходной системы уравнений к эквивалентной системе, которая решается последовательным приближением.

Идея метода состоит в том, что система нелинейных уравнений $f(x) = 0$ преобразуется в эквивалентную систему вида $x = g(x)$, где$f(x)$ и $g(x)$ это вектор-функции, а x – это вектор неизвестных. Таким образом, имея некое начальное приближение x_0, с помощью формулы $x_{i+1} = g(x_i)$ можно найти решение системы при условии, что при выбранном $g(x)$ и x_0 последовательность приближении будет сходиться в корни системы.

Условие сходимости метода простых итераций связано с выбором функции $g(x)$ и начального приближения

В общем случае метод простых итераций сходится к решению системы, если выполнены следующие условия:

1. Функция $g(x)$ непрерывна и имеет непрерывную производную на множестве, содержащем последовательность приближений.
2. Существует такое число $q < 1$, что для всех x из этого множества выполняется неравенство $\|g'(x)\| \leq q$, где $\|.\|$ — это норма вектора, а $g'(x)$ — это производная функции $g(x)$. Это условие гарантирует, что каждое следующее приближение x_{i+1} ближе к решению, чем предыдущее x_i
3. Начальное приближение x_0 выбрано достаточно близко к решению системы.

Процесс итерации повторяется до тех пор, пока не будет достигнута нужная точность, то есть $|| \; x^{(k+1)} - x^{(k)} \; || < \varepsilon$, где ε – это заданная точность.

Блок-схемы

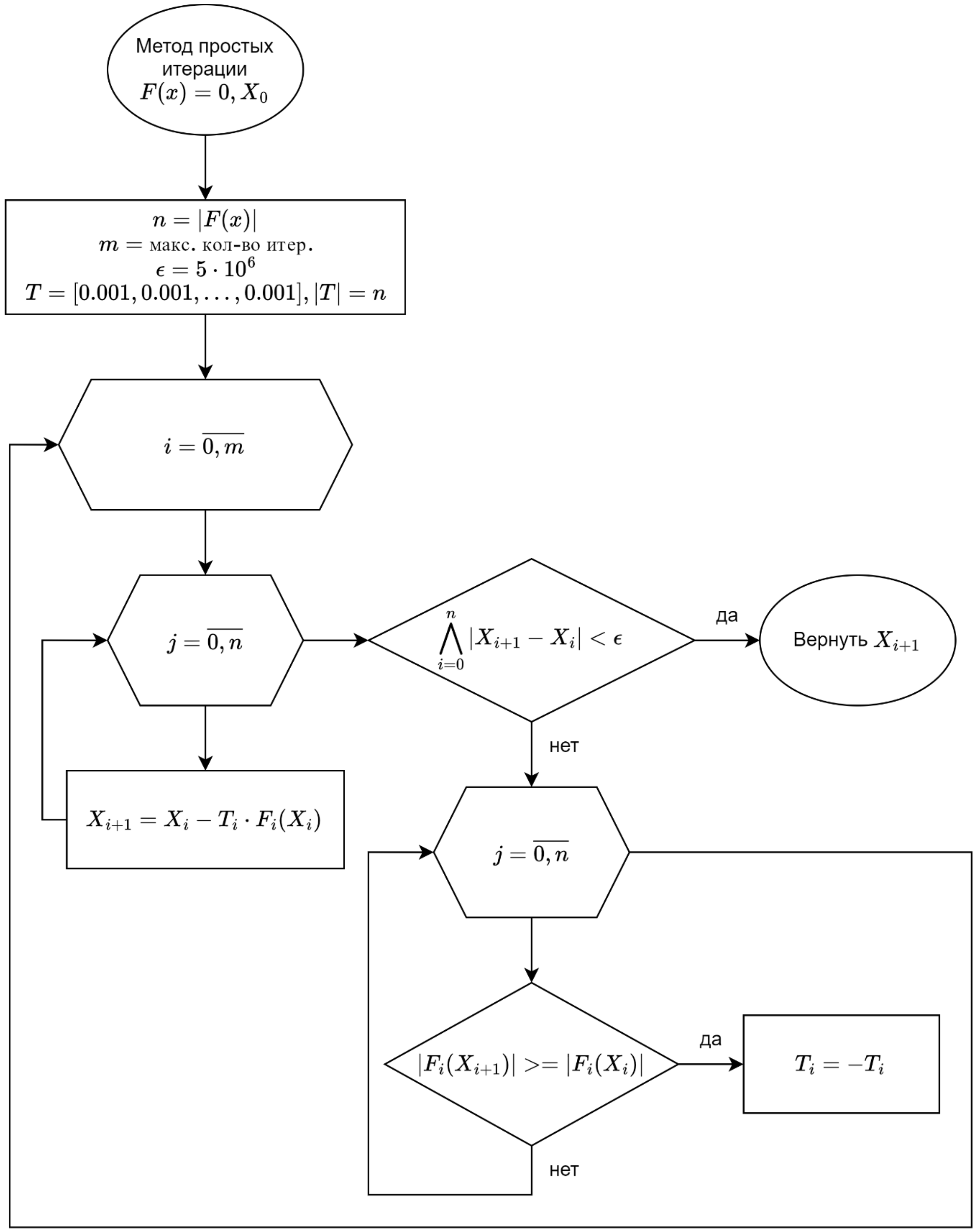

Схема 1 - Метод простых итерации

Код

Исправленный код

Был изменён шаг изменения в каждой итерации (parameter), а также изменение этого параметра в каждой итерации. Изменение обусловлено тем что в отправленном коде идея была в том что при более высоком шаге, скорость сходимости должна была быть более высокой, а для избежания расходимости, делится и-тый шаг на 10. При этом оказалось, что после нескольких итераций, шаг становился на столько малым что изменение $x^{(k+1)}$ становилось меньше ε и программа останавливалась преждевременно.

```
def solve_by_fixed_point_iterations(system_id, number_of_unknowns,
initial_approximations):
    epsilon = 5e-6
    max_iterations = 10000
    funcs = get_functions(system_id)
    current_approximations = initial_approximations.copy()
    next_approximations = [0.0] * number_of_unknowns
    parameter = [0.001] * number_of_unknowns # изменено, чтобы сделать
метод более сходимым

    for _ in range(max_iterations):

        for i in range(number_of_unknowns):
            try:
                next_approximations[i] = current_approximations[i] -
funcs[i](current_approximations) * parameter[i]
            except Exception as e:
                raise ValueError(
                    f"The initial approximations do not converge to a
solution for the given system. Error: {e}")

        if all(abs(next_approximations[i] - current_approximations[i]) <
epsilon for i in range(number_of_unknowns)):
            return next_approximations

        for i in range(number_of_unknowns):
            if abs(funcs[i](next_approximations)) >=
abs(funcs[i](current_approximations)):
                parameter[i] = -parameter[i] # изменено, чтобы сделать
метод более сходимым

        current_approximations = next_approximations.copy()

    raise ValueError("The method did not converge within the maximum number
of iterations")
```

Отправленный код

```
def solve_by_fixed_point_iterations(system_id, number_of_unknowns, initial_approximations):
    epsilon = 5e-6
    max_iterations = 10000
    funcs = get_functions(system_id)
    current_approximations = initial_approximations.copy()
    next_approximations = [0.0] * number_of_unknowns
    parameter = [0.01] * number_of_unknowns

    for _ in range(max_iterations):

        for i in range(number_of_unknowns):
            try:
                next_approximations[i] = current_approximations[i] - funcs[i](current_approximations) * parameter[i]
            except Exception as e:
                raise ValueError(
                    f"The initial approximations do not converge to a solution for the given system. Error: {e}")

        if all(abs(next_approximations[i] - current_approximations[i]) < epsilon for i in range(number_of_unknowns)):
            return next_approximations

        for i in range(number_of_unknowns):
            if abs(funcs[i](next_approximations)) >= abs(funcs[i](current_approximations)):
                parameter[i] = -parameter[i] if parameter[i] > 0 else parameter[i] * 0.1

        current_approximations = next_approximations.copy()

    raise ValueError("The method did not converge within the maximum number of iterations")
```

Примеры работы программы

Пример 1

Система 1 с решением в $(n\pi, 0)$ где n целое число и в $(0, r)$ где r любое действительное число.

Ввод:

```
1

2

0.2

0.5
```

Вывод:

```
0.004991888037816197

0.45344703389505725
```

Информация:

```
Converged after 3691 iterations

Current parameters: [0.001, 0.001]

Function values: [0.004996864107530826, 0.001132914149266958]
```

Пример 2

Система 1 с начальными значениями далёкими от решения системы

Ввод:

```
1

2

300

210
```

Вывод:

`298.4562956562425` Значение примерно равно 95π

`1.5e-323`

Информация:

```
Converged after 5969 iterations

Current parameters: [-0.001, -0.001]

Function values: [-0.0049985429397058434, 2.213e-321]
```

Пример 3

Система 2 с решениями в $(-2,2)$ и примерно $(1.35079, 0.59214)$

Ввод:

```
2
2
-2
2
```

Вывод:

```
-2.0
2.0
```

Информация:

```
Converged after 0 iterations
Current parameters: [0.001, 0.001]
Function values: [0.0, 0.0]
```

Пример 4

Система 2 с начальными значениями приближенными к первому решению системы

Ввод:

```
2
2
-1.9
2.1
```

Вывод:

```
-2.000237888699517
2.000309454231117
```

Информация:

```
Converged after 459 iterations
Current parameters: [0.001, 0.001]
Function values: [-0.004977859237698112, 0.00494356544566088]
```

Пример 5

Система 2 с начальными значениями приближенными ко второму решению системы

Ввод:

```
2
2
1.5
0.5
```

Вывод:

```
1.3504706150380186

0.5923381810705152
```

Информация:

```
Converged after 511 iterations
Current parameters: [-0.001, -0.001]
Function values: [0.004109662270408876, -0.004988785748520463]
```

Пример 6

Система 3. Данна система не имеет действительного решения

Ввод:

```
3

3

-0.1

-0.5

0.2
```

Вывод:

```
-0.02224934763342428

-0.25406117292448266

0.23186518467854403
```

Информация:

```
Converged after 7166 iterations
Current parameters: [0.001, 0.001, 0.001]
Function values: [-0.003941442096798531, -0.00499563452823365, -
0.003070229647883338]
```

Пример 7

Система 3

Ввод:

```
3

3

-3

2

3
```

Вывод:

```
-0.10688478539136449

-0.797530738951318
```

```
0.12008743717101461
```

Информация:

```
Converged after 9789 iterations

Current parameters: [-0.001, 0.001, 0.001]

Function values: [-0.0039025497343009585, 1.7774139987269955e-05, 0.004996201070791795]
```

Пример 8

Система 3 с отдалёнными значениями

Ввод:

```
3

3

5

10

-5
```

Вывод:

```
ValueError: The method did not converge within the maximum number of iterations
```

Вывод

Метод простых итерации для решения систем нелинейных алгебраических уравнении отличается простой программной реализации для такой нетривиальной задачи. Более того, в отличие от метода Ньютона, данный метод не требует производных и может быть применён к широкому спектру систем нелинейных уравнений. Тем не менее, данный метод имеет значительные недостатки. Первый недостаток – это сходимость. метод может не сходиться, если начальное приближение выбрано неправильно или если функция $g(x)$ не удовлетворяет условиям сходимости. Второй недостаток – это скорость сходимости, метод может сходиться медленно, при неверном выборе начального приближения. Так на пример, при выборе параметра (шага) равному 0.001 как показано в коде, решение системы обычно сходится к реальному значению, но про этом требуется во многих случаях более 1000 итерации, и как показано в примере 8, количество итерации превысило 10000 раз. Однако такой выбор шага, имеет более высокую точность чем отправленное решение.

Алгоритмическая сложность метода простых итерации составляет $O(n)$ для каждой итерации, где n – это количество уравнений в системе. Количество итерации будет зависеть от заданной точности, шага и начального приближения. Так на примере 3 можно увидеть, что при выборе начальных приближении равных решению системы, программа не должна была сделать ни одну итерацию.

Код успешно решает задачу нахождения решений систем нелинейных уравнений с использованием метода простых итераций. Он корректно обрабатывает различные входные данные и возвращает ожидаемые результаты, как показано в примерах. Однако, как видно из примеров, метод может не сходиться для некоторых систем или начальных приближений.

Численная ошибка метода простых итераций зависит от выбранной точности (эпсилон). Но при этом решение метода может отличаться от действительного больше чем на эпсилон, так как это число только ограничивает изменение аппроксимаций. С другой стороны, как сказано ранее, сходимость метода зависит от выбранных $g(x)$ и x_0, поэтому численная ошибка может быть больше эпсилон.