

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа по
Вычислительной математике №3
Вариант «Метод простых итераций»

Работу выполнила:

Касьяненко В.М.

Группа:

P3220

Санкт-Петербург,

2024

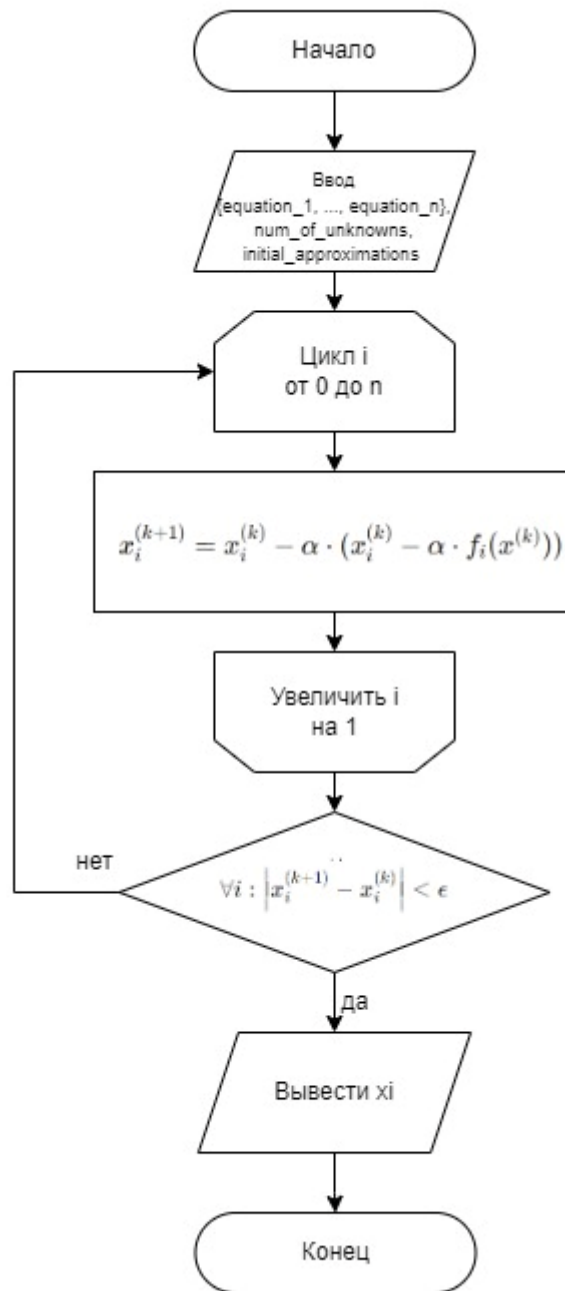
Описание численного метода.

Метод простых итераций – это численный метод для нахождения корней системы нелинейных уравнений, основанный на принципе последовательных итераций, в которых текущее приближение корней системы используется для вычисления следующего приближения.

Для данной задачи у нас есть система уравнений $F(x) = 0$, где $F(x)$ – вектор-функция, содержащая нелинейные уравнения системы, а x – вектор неизвестных. На каждой итерации рассчитывается новое приближение $x^{(k+1)}$ на основе предыдущего $x^{(k)}$ и параметра α , используя выражение $x^{(k+1)} = x^{(k)} - \alpha F(x^{(k)})$, где $F(x^{(k)})$ – вектор значений функций в точке $x^{(k)}$. Затем проверяется условие сходимости: если разница между предыдущим и новым приближениями меньше заданной точности для всех компонент вектора x , то итерационный процесс завершается.

Таким образом, метод простых итераций продолжает итерационный процесс до тех пор, пока не будет достигнут критерий окончания, который в данном случае определяется точностью

Блок-схема



Код численного метода

```
def solve_by_fixed_point_iterations(system_id, number_of_unknowns, initial_approximations):
    max_iterations = 100
    tolerance = 1e-5
    step_length = 0.0001

    def fixed_point_iteration(func, initial_guesses, max_iter, tol, step_length):
        iterations = 0
        x_prev = initial_guesses
        while iterations < max_iter:
            x_next = func(x_prev, step_length)
            if all(abs(x_next[i] - x_prev[i]) < tol for i in range(number_of_unknowns)):
                break
            x_prev = x_next
            iterations += 1
        return x_next, iterations

    functions = get_functions(system_id)

    def system(x, step_length):
        return [x[i] - step_length * functions[i](x) for i in range(number_of_unknowns)]

    solution, iterations = fixed_point_iteration(system, initial_approximations, max_iterations,
tolerance, step_length)
    return solution
```

Примеры работы программы

1)

Входные данные:

1

2

1.0

2.0

Выходные данные:

0.9916078527646449

1.9900660838909958

2)

Входные данные:

3

3

0.5

-1.0

2.0

Выходные данные:

0.4538072181466193

-1.0304576235085257

1.9538281206568267

3)

Входные данные:

3

3

0.0

0.0

0.0

Выходные данные:

0.0009950238612479096

-0.001990174530636173

0.0029851821351471173

4)

Входные данные:

1

1

0.0

Выходные данные:

0.0

5)

Входные данные:

2

2

1.0

2.0

Выходные данные:

1.445721981124815

2.1435695839180386

Выводы

В результате выполнения лабораторной работы был реализован метод простых итераций для решения систем нелинейных уравнений, основанный на идее последовательного уточнения начального приближения до достижения требуемой точности. Метод простых итераций легко реализуем и понятен, а также дает возможность решения большого класса нелинейных систем уравнений. Метод простых итераций преобразует исходную систему уравнений в эквивалентную систему, где каждое уравнение содержит одну из неизвестных в явном виде, а остальные неизвестные выражены через уже найденные значения на предыдущей итерации. Это позволяет свести решение к поиску неподвижной точки функции итерации.

Сравнение с методом Ньютона и методом бисекции. Метод простых итераций является простым в реализации. Он не требует вычисления производных функций и может быть применен к широкому спектру задач. Однако его сходимость может быть медленной, что может привести к необходимости выполнять большое количество итераций для достижения требуемой точности. Метод Ньютона, с другой стороны, обеспечивает быструю сходимость, особенно вблизи решения. Однако для его применения требуется вычисление производных функций, что может быть сложно или даже невозможно в некоторых случаях. Кроме того, метод Ньютона может быть неустойчивым при выборе неправильного начального приближения или для некоторых видов систем. Метод бисекции является простым и гарантирует сходимость к решению на отрезке, где функция меняет знак. Он не требует вычисления производных и легко реализуется. Однако метод бисекции медленно сходится и применим только к уравнениям с одной переменной.

Анализ применимости метода: метод простых итераций хорошо подходит для простых систем нелинейных уравнений, особенно если есть хорошее начальное приближение и известно, что решение находится вблизи этого приближения. В случае многомерных систем уравнений (с большим числом неизвестных) метод простых итераций может быть менее эффективным из-за необходимости итерировать по всем переменным сразу. Также метод простых итераций склонен к нахождению только локальных решений системы уравнений. Если необходимо найти глобальные решения, может потребоваться использовать другие методы, такие как метод Ньютона.

Общая алгоритмическая сложность метода простых итераций может быть выражена как $O(nm)$, где n – количество неизвестных, а m – максимальное количество итераций.

Метод простых итераций может иметь ограниченную точность из-за приближенных вычислений. Это может привести к накоплению погрешности на каждой итерации, что в итоге может ухудшить точность результата. Число итераций, необходимых для достижения заданной точности, также важно для анализа численной ошибки. Большое число итераций может быть затратным с точки зрения вычислительных ресурсов, в то время как недостаточное количество итераций может привести к неточному результату. Метод простых итераций также может быть чувствителен к

начальным приближениям. Небольшие изменения в начальных условиях могут привести к значительным изменениям в итоговом решении.

Таким образом, метод простых итераций является хорошим инструментом для численного решения систем нелинейных уравнений, но требует внимательной настройки и анализа для достижения точных и стабильных результатов.