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

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа № 3

Метод простых итераций

Выполнила:

Гафурова Фарангиз Фуркатовна

Группа P3220

Преподаватель:

Перл Ольга Вячеславовна

г. Санкт-Петербург

2025

Оглавление

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.

Основная идея

Метод основан на преобразовании исходного $f(x) = 0$ в эквивалентную форму $x = g(x)$. Затем, начиная с некоторого начального приближения x_0, последовательно строятся приближение $x_{i+1} = g(x_i)$ для $n = 0, 1, 2, \ldots$. Если последовательность $\{x_n\}$ сходится к пределу x^*, то x^* является корнем уравнения $f(x) = 0$.

Алгоритм

- **Выбор начального приближения x_0:** Это значение должно быть достаточно близко к истинному корню уравнения, чтобы обеспечить сходимость метода.
- **Итерационный процесс:** На каждой итерации вычисляется новое приближение $x_{i+1} = g(x_n)$.
- **Проверка сходимости:** Проверяется, удовлетворяет ли последовательность условиям сходимости. Например, можно проверять, уменьшается ли расстояние между последовательными приближениями $|x_{i+1} - x_n|$ меньше некоторого заданного малого числа E. Если условие сходимости выполнено, то процесс останавливается и x_{i+1} принимается в качестве приближения к корню.

Геометрическая интерпретация

В геометрической интерпретации метод простых итераций рассматриваются графики функций $y = g(x)$ и $y = x$. График функции y $= g(x)$ пересекается с прямой $\text{y} = x$ в точке, соответствующей корню уравнения $x = g(x)$. На каждой итерации мы движемся от точки $(x_n,\ x_n)$ к точке (x_{i+1}, x_{i+1}) по линии, параллельной оси x, пока не приблизимся достаточно близко к точке пересечения.

Условия сходимости

- Функция $g(x)$ должна быть непрерывной на некотором интервале, содержащем корень уравнения.
- Таже необходимо, чтобы для всех x из этого интервала выполнялось условие $|g'(x)| \leq q < 1$, где q – некоторая константа. Это условие гарантирует сходимость последовательности итераций.
- Начальное приближение x_0 выбрано достаточно близко к решению системы.

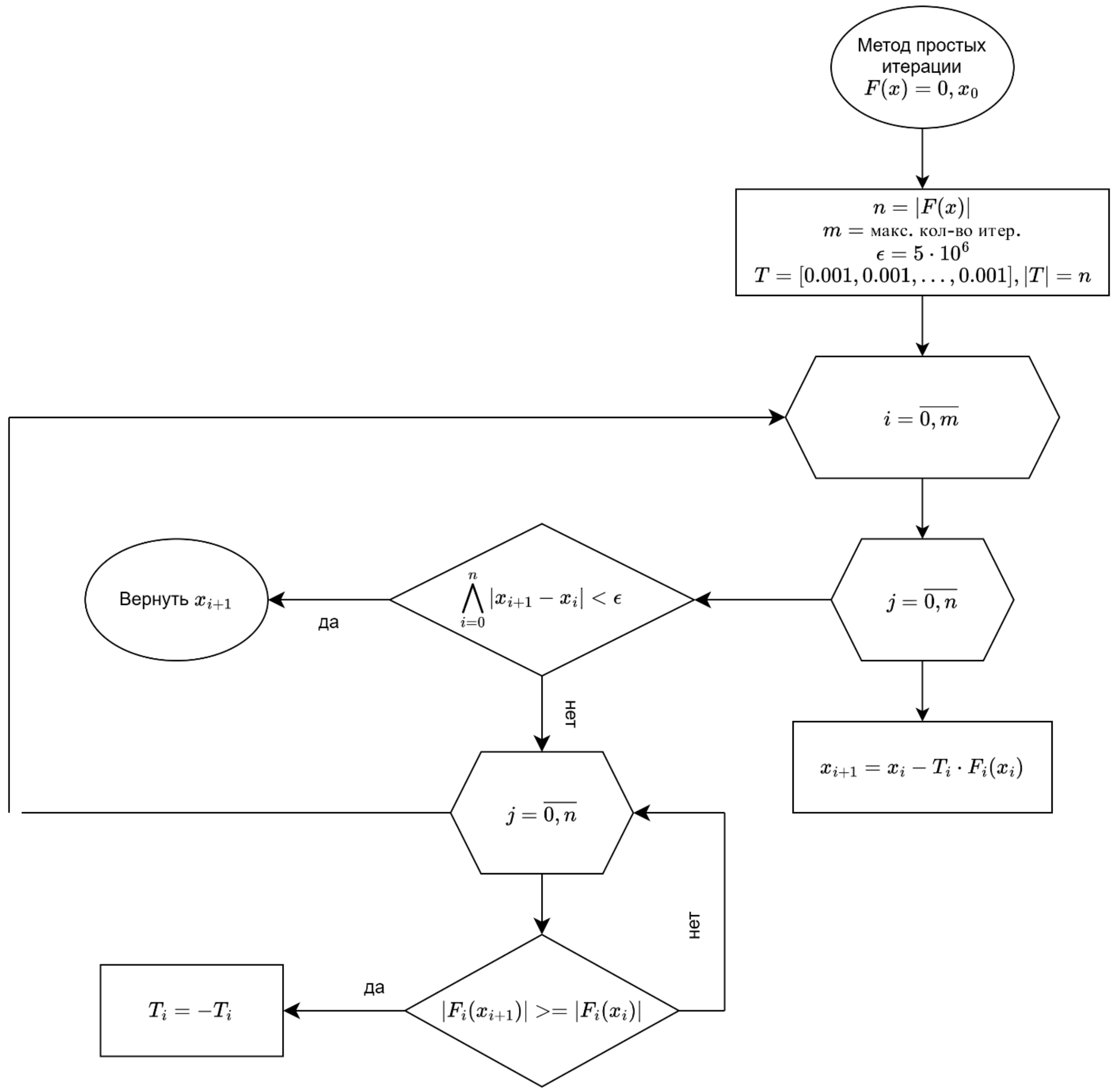
Процесс итерации повторяется до тех пор, пока не будет достигнута нужная точность, то есть || $x^{(k+1)} - x^{(k)}$ ||$<\varepsilon$, где ε – это заданная точность.

Преимущества и недостатки

Преимущества: Метод простых итераций имеет простую структуру и легко реализуется на компьютере. Он может быть эффективен для некоторых классов уравнений, особенно когда функция $g(x)$ имеет простую форму.

Недостатки: Метод не всегда сходится. Он сильно зависит от выбора начального приближения от свойств функции $g(x)$. Кроме того, скорость сходимости может быть относительно медленной, особенно если q близок к 1.

Блок-схема:


Метод простых итерации
$F(x) = 0, x_0$
$n = |F(x)|$
$m =$ макс. кол-во итер.
$\epsilon = 5 \cdot 10^6$
$T = [0.001, 0.001, \dots, 0.001], |T| = n$
$i = \overline{0, m}$
$j = \overline{0, n}$
$x_{i+1} = x_i - T_i \cdot F_i(x_i)$
$\bigwedge_{i=0}^{n} |x_{i+1} - x_i| < \epsilon$
да
Вернуть x_{i+1}
нет
$j = \overline{0, n}$
$|F_i(x_{i+1})| >= |F_i(x_i)|$
да
$T_i = -T_i$
нет

Код

```python
def slve(f, number_of_unknowns, initial_approximations):
    itr = 0
    mxItr = 20000
    eps = 1e-5
    coeff = [0.001] * number_of_unknowns
    crApx = initial_approximations[:]

    while itr < mxItr:
        tmpApxs = crApx.copy()
        for j in range(number_of_unknowns):
            x_1 = f[j](crApx)
            tmpAppox = crApx[j] + coeff[j] * (x_1 - crApx[j])
            tmpApxs[j] = tmpAppox
            if abs(x_1 - crApx[j]) < eps:
                coeff[j] *= 1.1
            else:
                coeff[j] *= 0.9
        if all(abs(tmpApxs[j] - crApx[j]) < eps for jj in range(number_of_unknowns)):
            break
        crApx = tmpApxs

    return [round(res, 5) for res in crApx]

def solve_by_fixed_point_iterations(system_id, number_of_unknowns, initial_approximations):
    f = get_functions(system_id)

    return slve(f, number_of_unknowns, initial_approximations)
```

Примеры работы программы

Тест 1:

Ввод:

```
1
2
300
500
```

Вывод:

```
300.00999848789826
110.30168415871722
```

Тест 2:

Ввод:

```
1
2
0.7
0.9
```

Вывод:

```
0.6935821956668042
0.8968697158536401
```

Тест 3:

Ввод:

```
2
2
-1.5
1
```

Вывод:

```
-1.6620074200640038
1.0084107327483427
```

Тест 4

Ввод:

```
3
3
3
-2
1.8
```

Вывод:

```
2.8106479211145374
-2.1848728193160833
```

```
1.872423859639823
```

Тест 5

Ввод:

```
1
1
0.3
```

Вывод:

```
0.29705873351074513
```

Тест 6

Ввод:

```
2
2
-0.9
1.5
```

Вывод:

```
-0.7897827267330647
1.614741823422567
```

Вывод

В рамках данной лабораторной работы был изучен и реализован численный метод простых итераций для решения систем нелинейных уравнений.

Метод простых итераций для решения систем нелинейных алгебраических уравнений имеет несколько заметных аспектов. С одной стороны, его программная реализация для такой сложной задачи чрезвычайно проста. В отличие от метода Ньютона, этот метод не требует вычисления производных, что делает его применимым к широкому спектру систем нелинейных уравнений.

Сравнение с методом Ньютона:
Метод Ньютона, в отличие от метода простых итераций, использует информацию о производных функций системы уравнений. Это позволяет ему сходиться быстрее, если производные легко вычисляются и удовлетворяют определенным условиям. Например, для систем уравнений с гладкими функциями, у которых можно эффективно вычислить матрицу Якоби, метод Ньютона обычно требует меньшего количества итераций, чем метод простых итераций.

Однако метод Ньютона имеет свои недостатки. Он требует вычисления производных и обратной матрицы Якоби, что может быть вычислительно сложным, особенно для больших систем уравнений. Кроме того, если матрица Якоби вычисляется с ошибкой или является вырожденной, метод может не сойтись. В то время как метод простых итераций, не требующий вычисления производных, может быть более стабилен для систем, где вычисление производных затруднительно или неточным.

Анализ применимости метода:

Метод простых итераций применим к широкому спектру систем нелинейных уравнений. Он особенно полезен для систем, где функция $g(x)$ может быть легко построена и имеет свойства сходимости.

Однако для систем с очень сложными нелинейностями, где трудно построить функцию $g(x)$ со свойствами сходимости, или для систем, где начальные приближения трудно определить, метод может быть неприменим.

Для успешного применения метода простых итераций требуется, чтобы начальные приближения были достаточно близки к истинному решению. Также функции системы уравнений должны быть таковыми, что можно построить функцию $g(x)$ с условиями сходимости. Если данные системы не удовлетворяют этим требованиям, метод может не сойтись или сходиться очень медленно.

Анализ численной ошибки:
Численная ошибка метода простых итераций напрямую связана с заданной точностью, определяемой значением эпсилон. Если метод сходится, то разница между последовательными приближениями должна стать меньше эпсилон. Однако, это не гарантирует, что полученное решение близко к истинному значению на величину, равную эпсилон.

Кроме эпсилона, численная ошибка также зависит от начального приближения и свойств функции $g(x)$. Если начальное приближение находится далеко от истинного решения или функция $g(x)$ имеет плохие свойства сходимости, то ошибка может быть значительно больше эпсилон. Также ошибка может накапливаться на каждой итерации, особенно если метод сходится медленно.

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.