Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Вычислительной математике»

«Системы нелинейных алгебраических уравнений»

Выполнил:

Студент группы P32312

Лебедев В.В.

Преподаватель:

Перл О.В.

Санкт-Петербург

2023

Описание методов

Метод Половинного деления для решения нелинейных уравнений

Пусть даны *a* и *b,* при этом *f(a) * f(b) < 0*, а функция имеет вид *f(x) = 0*, значит, между *a* и *b* находится корень *x*, удовлетворяющий условию *f(x) = 0*.

Будем искать его при помощи сокращения расстояния между *a* и *b* до тех пор, пока разница между ними не станет меньше необходимой погрешности. На каждой итерации будем сокращать промежуток вдвое, выбирая из получившихся отрезков те, которые удовлетворяют условию *f(a`) * f(b`) < 0*.

Метод Хорд для решения нелинейных уравнений

Пусть даны *a* и *b,* при этом *f(a) * f(b) < 0*, а функция имеет вид *f(x) = 0*, значит, между *a* и *b* находится корень *x*, удовлетворяющий условию *f(x) = 0*.

Метод ищет приближенную точку *x`* за счет нахождения точки пересечения прямой соединяющей *P1(a, f(a)* и *P2(b, f(b))* и координатной прямой *Ox*. За счет этого расстояние между *a`* и *b`* на каждой итерации сокращается до тех пор, пока не достигнет необходимой погрешности.

Метод Ньютона для решения систем нелинейных уравнений

Метод оперирует системой нелинейных уравнений:

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0, \\ f_2(x_1, \ldots, x_n) = 0, \\ \qquad \vdots \\ f_n(x_1, \ldots, x_n) = 0. \end{cases}$$

Для нее можно вывести итеративную форму для приближения решений

$$x^{(k+1)} = x^{(k)} - W^{-1}(x^{(k)}) \cdot F(x^{(k)})$$

Где *W* - Якобиан (матрица первых производных).

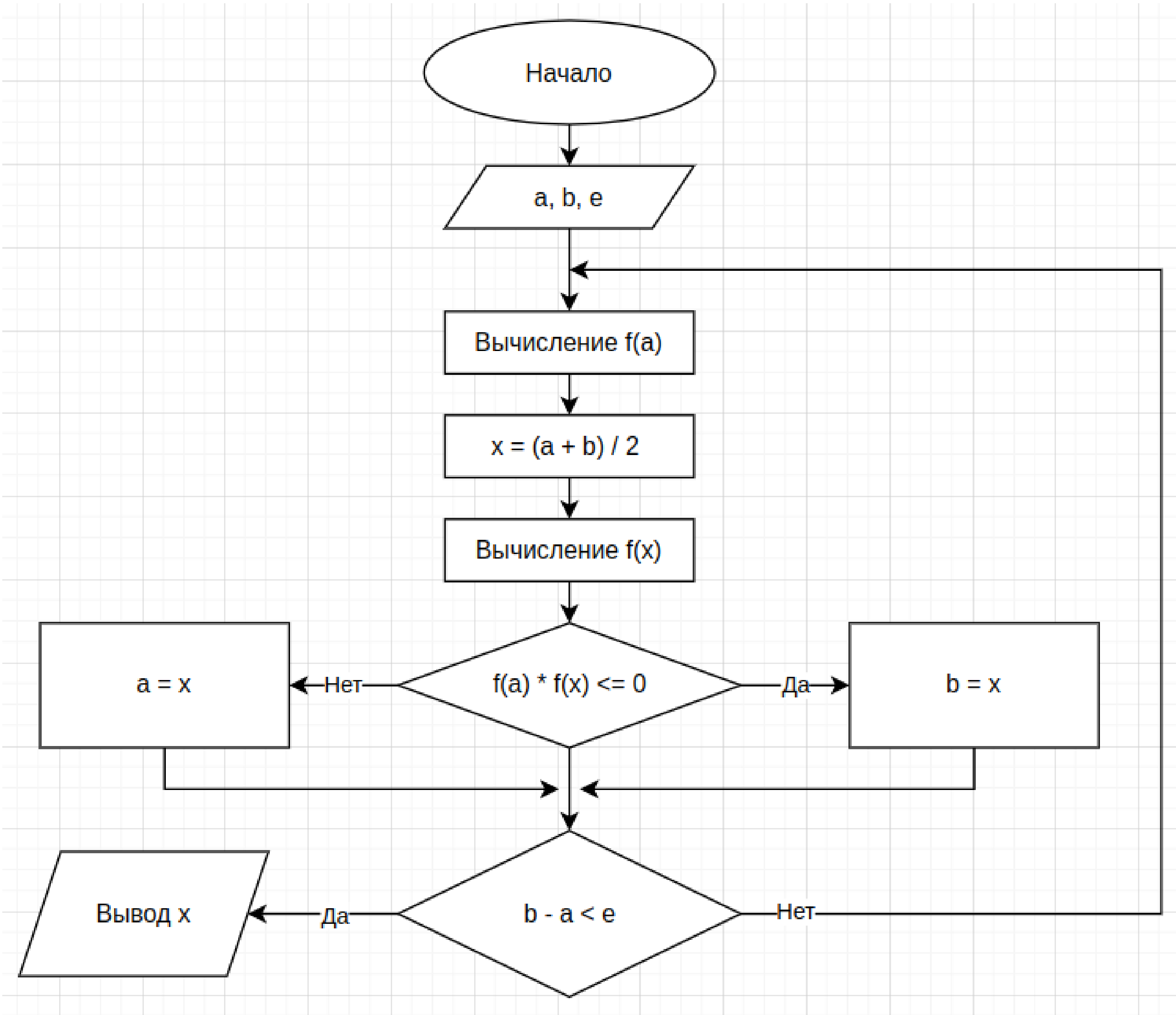
После некоторых преобразований метод сводится к:

$W \cdot \Delta x = - F(x^k)$, где x - разница между значением текущей итерации и следующей, W - Якобиан функции F.

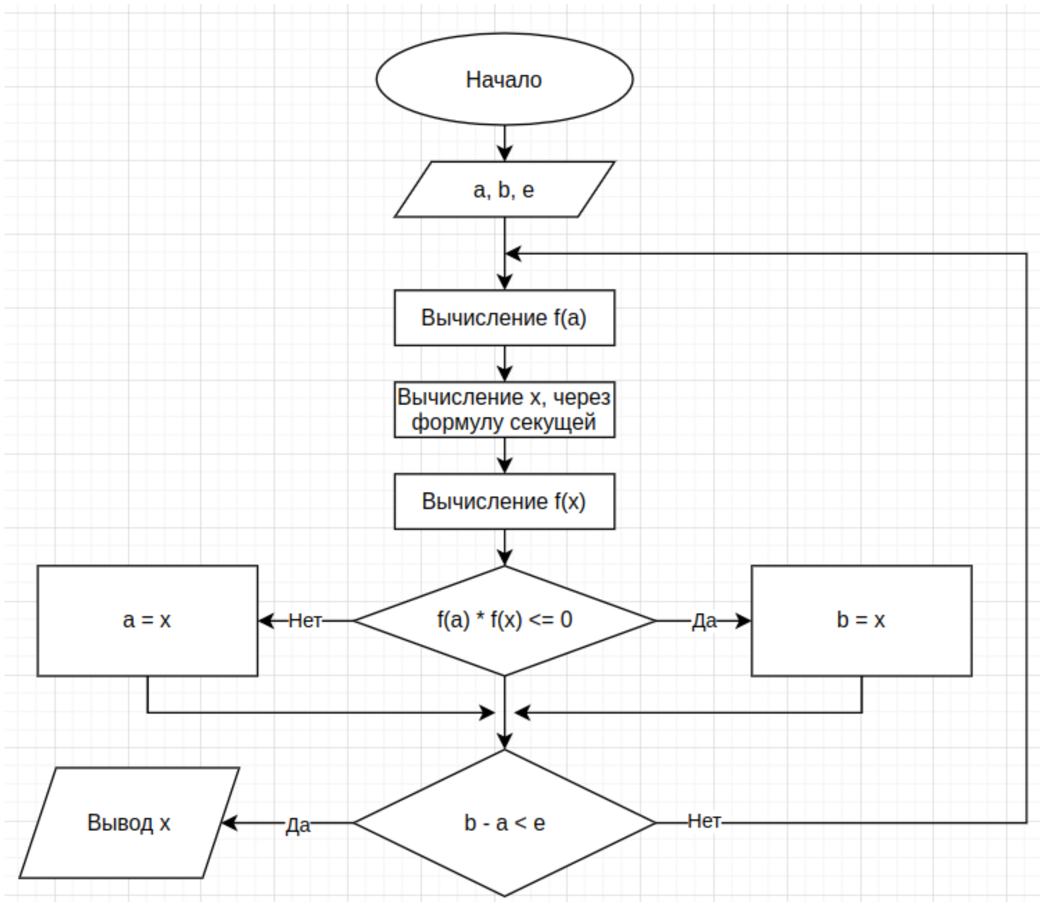
Тогда метод сводится к последовательному решению линейных уравнений, для нахождения Δx ` до тех пор, пока он не станет меньше необходимой погрешности.

Блок-схемы численных методов

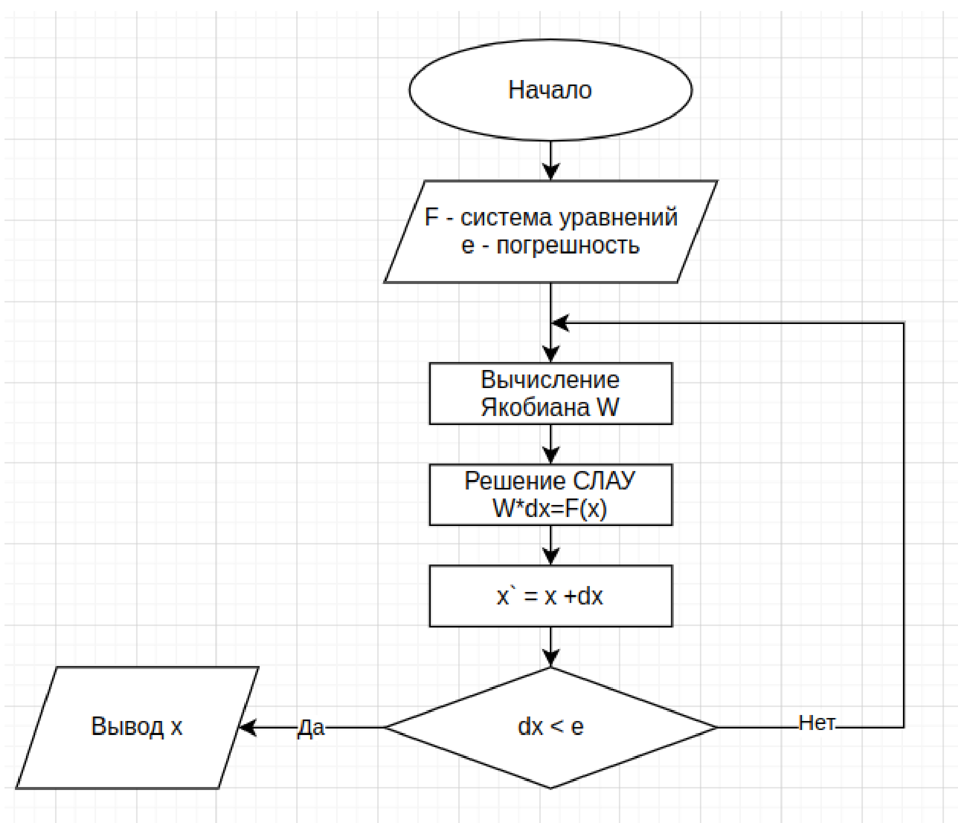
Метод Половинного деления



Метод Хорд



Метод Ньютона



Листинг программы

Метод Половинного деления

```
def bisection_method(f: Callable[[float], float],
                     l: float, r: float, e: float,
                     max_i: int = 1000) -> float:
    i = 0
    while abs(r - l) > e and i < max_i:
        i += 1
        m = (r + l) / 2
        fm = f(m)
        fr = f(r)
        if fm == 0:
            return m
        elif fm * fr > 0:
            r = m
        else:
            l = m
    return (r + l) / 2
```

Метод Хорд

```
def secant_method(f: Callable[[float], float],
                  l: float, r: float, e: float,
                  max_i: int = 1000) -> float:
    i = 0
    x = l
    while abs(r - l) > e and i < max_i:
        i += 1
        x = l - f(l) / (f(l) - f(r)) * (l - r)
        fm = f(x)
        fr = f(r)
        if fm == 0:
            return x
        if fm * fr > 0:
            r = x
        else:
            l = x
    return x
```

Метод Ньютона

```python
def compute_partial_derivative(f: Callable[[list[float]], float],
                               i: int) -> Callable[[list[float]], float]:
    def function(x: list[float]) -> float:
        d = 1e-6
        dx = x[::]
        dx[i] += d
        return (f(dx) - f(x)) / d

    return function


def compute_jacobian(functions: list[Callable[[list[float]], float]],
                     n: int):
    matrix = []
    for i in range(len(functions)):
        row = [compute_partial_derivative(functions[i], j) for j in range(n)]
        matrix.append(row)
    return matrix
```

```python
def newton_method(system: list[Callable[[list[float]], float]],
                  e: float, max_i: int = 1000) -> list[float]:
    n = len(system)
    jacobian = compute_jacobian(system, n)
    x = [0] * n
    s = 10 * e
    i = 0
    while i < max_i and s > e:
        i += 1
        a = [[df(x) for df in row] for row in jacobian]
        b = [-f(x) for f in system]
        dx = gauss_elimination(a, b)
        s = abs(max(dx, key=abs))
        x = [x[i] + dx[i] for i in range(n)]
    return x
```

```python
def max_row(m: list[list[float]], col: int):
    max_i = col
    max_val = abs(m[col][col])
    for row in range(col, len(m)):
        if abs(m[row][col]) > max_val:
            max_i = row
    return max_i


def gauss_forward_elimination(m: list[list[float]]):
    n = len(m)
    for i in range(n):
        i_max = max_row(m, i)
        m[i], m[i_max] = m[i_max], m[i]
        for j in range(i + 1, n):
            factor = m[j][i] / m[i][i]
            for k in range(i + 1, n + 1):
                m[j][k] -= factor * m[i][k]
            m[j][i] = 0
```

```python
def gauss_back_substitution(m: list[list[float]]) -> list[float]:
    n = len(m)
    x = [0.0] * n
    for i in range(n - 1, -1, -1):
        x[i] = m[i][n] / m[i][i]
        for j in range(i - 1, -1, -1):
            m[j][n] -= x[i] * m[j][i]
    return x


def gauss_elimination(a: list[list[float]], b: list[float]) -> list[float]:
    m = copy.deepcopy(a)
    for i in range(len(a)):
        m[i].append(b[i])

    gauss_forward_elimination(m)
    return gauss_back_substitution(m)
```

Примеры

```
Enter 'system' or 'equation' to choose: system
System:

    | (x - 1)^3 - x^2 + y = 0
    | x^2 - y + x^5 = 0

Enter accuracy epsilon: 0.00001
System solution:
x0) 0.58768
x1) 0.41547
```

```
Enter 'system' or 'equation' to choose: equation
Equations:
1) (x - 1)^3 - x^2 + 2 = 0
2) (x^2 - 2)^2 - 2 = 0
Enter number of equation: 1
Enter left bound: 0
Enter right bound: 2
Enter accuracy epsilon: 0.00001
Bisection method result: 1.44504
Secant method result: 1.44504
```

Вывод

Асимптотическая сложность:

- Метод половинного деления: O(L), где L - число итераций
- Метод Хорд: O(L), где L - число итераций
- Метод Ньютона: O(L*n^3), где L - число итераций

Анализ применимости и сравнение:

- Метод половинного деления прост в реализации, всегда сходится, но делает это медленно.
- Метод касательных обладает высокой скоростью сходимости, но его сходимость зависит от вида функции применяется на отрезках малой длины.
- Метод хорд имеет высокую скорость сходимости, но не всегда сходится, в зависимости от выбора границ.
- Метод простой итерации обладает хорошей сходимостью, но перед его использованием требуется преобразование исходного уравнения и проведение дополнительных вычислений.