Университет ИТМО
Факультет ПИиКТ

Лабораторная работа №2
Вычислительная математика

Вариант «Метод деления пополам,
Метод простой итерации,
Метод Ньютона»

Выполнил:
Джукашев Д.Р. P3230

Проверила: Перл О.В

Санкт-Петербург 2021

Теория

Метод деления пополам: пусть дано уравнение f(x) и отделен простой корень x*, то есть найден такой отрезок $[a_0,b_0]$, что x* принадлежит $[a_0,b_0]$ и на концах отрезка функция имеет значения, противоположные по знаку ($f(a_0)*f(b_0)<0$). Отрезок $[a_0,b_0]$ называется начальным интервалом неопределенности, потому что известно, что корень ему принадлежит, но его местоположение с требуемой точностью не определено. Уточнение корня состоит в построении последовательности вложенных в друг друга отрезков, каждый из которых содержит корень уравнения. Для этого находится середина текущего интервала неопределенности $c_k = \frac{a_k+b_k}{2}$, k = 0,1,2,3, …, и в качестве следующего интервала неопределенности из двух возможных выбирается тот, на концах которого функция имеет разные знаки

Метод простой итерации: для использования метода простой итерации исходное нелинейное уравнение f(x) = 0 заменяется равносильным уравнением $x = \psi(x)$. Пусть известно начальное приближение корня $x=x_0$. Подставляя это значение в правую часть уравнения, получим новое приближение $x_1 = \psi(x_0)$. Далее, подставляя каждый раз новое значения корня в уравнение, получаем последовательность значений: $x_{i+1} = \psi(x_i), (i = 0,1, \dots)$

Метод Ньютона для решения систем нелинейных уравнений: Формула для нахождения решения:

$$x^{k+1} = x^k - W^{-1}\left(x^{(k)}\right) * F\left(x^{(k)}\right), k = 0,1,2, \dots$$

Листинг

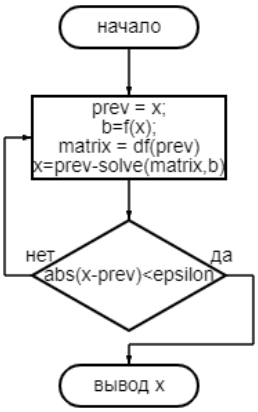
Метод простой итерации:

```
def iter_method(function, eps, x):
    print("Уточнение корня в районе "+str(x))
    rez = x
    x = set_x(function(rez))
    while abs(x-rez) > eps:
        rez = x
        x = set_x(function(x))
    return x
```

Ввод x, ε
$c = x$
$x = f(c)$
до $\lvert x - c \rvert < \varepsilon$
Вывод x

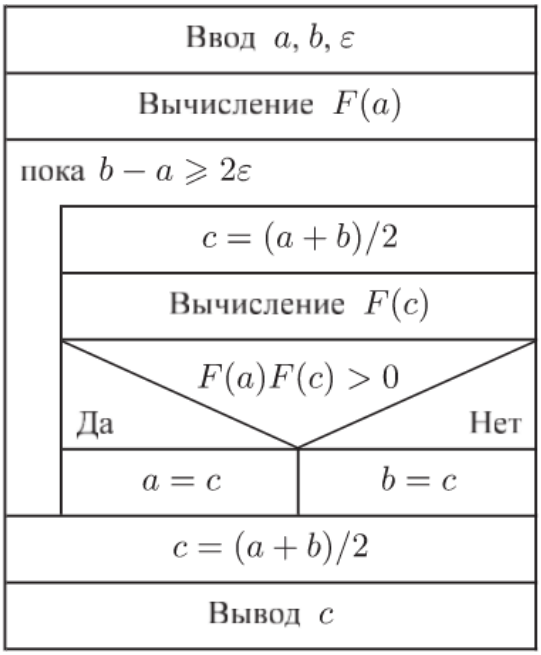
Метод Ньютона:

```
def make_der_matrix(df, x):
    matrix = []
    for i in df:
        row = []
        for a in range(len(df)):
            row.append(i[a](*x))
        matrix.append(row)
    return matrix
def solve_system(f, df, x, epsilon):
    print("Уточнение корней: " + str(x))
    while True:
        prev = x
        b = []
        for i in range(len(f)):
            b.append(f[i](*x))
        matrix = make_der_matrix(df, prev)
        x = prev - solve(matrix, b)
        if all(i < epsilon for i in abs(x - prev)):
            break
    return x
```



Метод бисекции:

```
def bisection_method(function, a, b, eps):
    print("Поиск на промежутке ["+str(a)+";"+str(b)+"]")
    f_a = function(a)
    while b - a >= 2 * eps:
        if f_a == 0:
            return set_c(a)
        if function(b) == 0:
            return set_c(b)
        c = set_c((a + b) / 2)
        f_c = function(c)
        if f_c == 0:
            return c
        if f_a * f_c > 0:
            a = c
        else:
            b = c
    c = set_c((a + b) / 2)
    return c
```



Результат

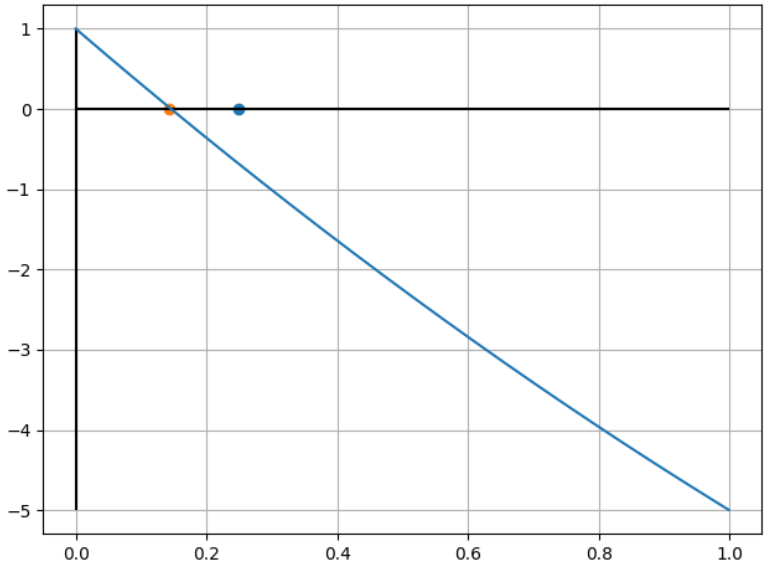
Ввод: 0.5

Вывод:

Результат выполнения метода бисекции: 0.25

Результат выполнения метода простых итераций: 0.14285714285714285

Разность точности их выполнения: 0.10714285714285715



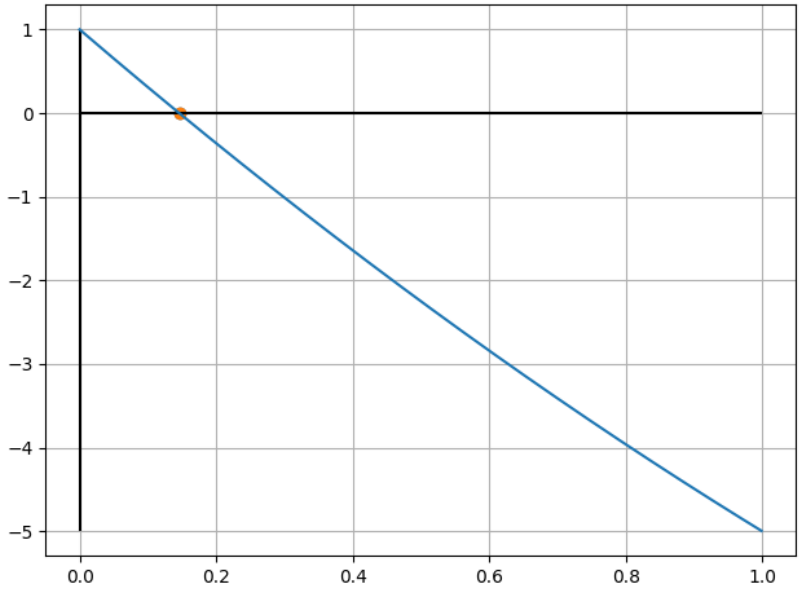
Ввод: 0.0000001

Вывод:

Результат выполнения метода бисекции: 0.14589804410934448

Результат выполнения метода простых итераций: 0.14589803337173055

Разность точности их выполнения: 1.0737613936884216e-08

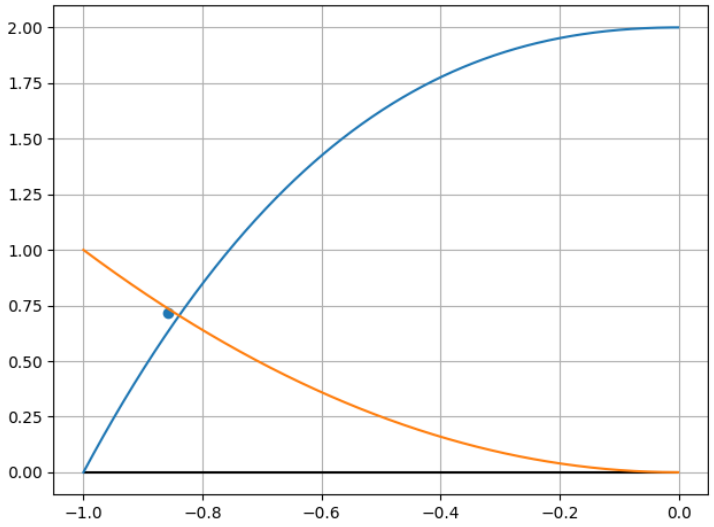


Ввод: 0.5

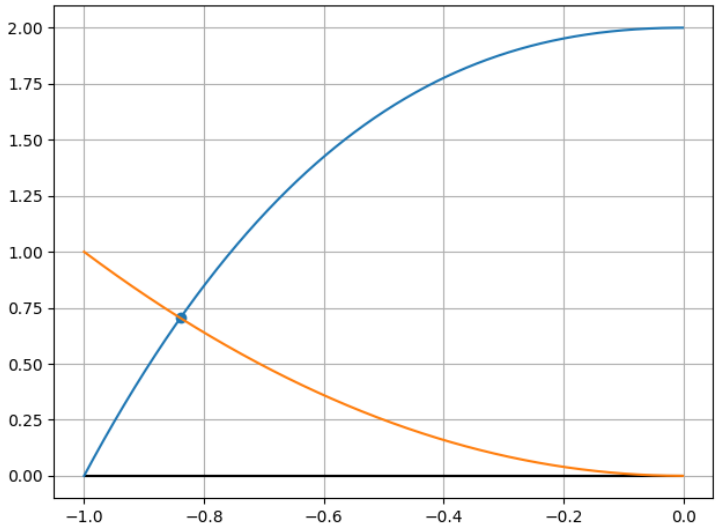
Вывод:

Уточнение корня: (-0.5, 0.5)

Результат выполнения метода Ньютона [-0.85714286 0.71428571]



Ввод: 0.0001
Вывод:
Уточнение корня: (-0.5, 0.5)
Результат выполнения метода Ньютона [-0.83928676 0.70440226]



Вывод: есть несколько методов решения нелинейных уравнений, некоторым необходимо начальное приближение корня, другим только отрезок, на котором есть корень и необходимо отыскать корень с заданной точностью. Они используют различные алгоритмы для уточнения корней. Все используют итерации для достижения необходимой точности. Методы решения систем нелинейных уравнений не сильно отличается от решения одного уравнения, там применяются схожие алгоритмы.