МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

**Отчет по заданию №1**

**по дисциплине**

**«Численные методы»**

Выполнил: студент гр. БЭИ2202

Васильев В. А.

Вариант 8.

Проверил: доц. каф. «Информатика»

Мацкевич А. Г.

Москва, 2023 г.

**1. Индивидуальное задание**

Решить уравнение f(x) = cos(x)–(x + 2)1/2 + 1 = 0 при помощи вспомогательных методов: метода простых итераций и метода Ньютона. Для 2-го заданного метода выполнить решение уравнения с точностью 10-4, создав программу, реализующую заданный метод.

**2. Этап отделения корней**

Используем для этого ЯП Python. Отделение корней произведем графическим методом (график функции) при помощи библиотеки Plotly.

**import** pandas **as** pd

**import** numpy **as** np

**import** plotly.graph\_objects **as** go

**from** sympy **import** \*

**def** f(x):

**return** np.cos(x) - np.sqrt(x+2) + 1

begin = -2

end = 7

step = 0.2

size = 1000

x = np.linspace(begin, end, size)

y\_f = f(x)

fig = go.Figure() # layout\_xaxis\_range = [-2,6]

fig.add\_trace(go.Scatter(x=[begin, end],

y=[0, 0],

name="Ось Х"))

fig.add\_trace(go.Scatter(x=x,

y=y\_f,

name="График функции"))

fig.update\_layout(title="Построение графика функции",

xaxis\_title="Значение аргумента",

yaxis\_title="Значение функции",

margin=dict(l=10, r=70, t=50, b=0))

fig.show()

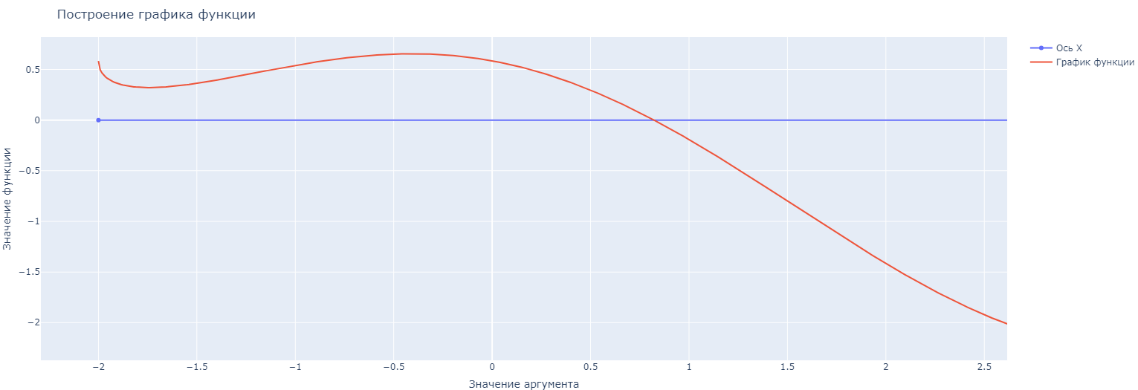


Рисунок 1 – Графическое отделение корня уравнения

Из построенного графика функции f(x) видно, что на отрезке (0, 1.5) есть один корень. На этом графический способ отделения корней заканчивается.

Другой вариант отделения корня – решить задачу аналитически.

Для аналитического отделения корня построена таблица рис.2.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2 – Таблица значений функции и ее производных

В столбцах таблицы выведены некоторые значения аргумента x на заданном отрезке, а также значения функций f(x), при этих значениях x.

Видно, что на отрезке (0, 1.5) функция f(x) меняет знак, значит существует, по крайней мере, один корень.

Значения первой производной в заданных точках отрезка (0, 1.5) не меняет знак, что вызывает некоторую надежду о том, что не меняет знак на всем отрезке (0, 1.5), но делать вывод об этом не совсем корректно с точки зрения математики. Однако анализ аналитического выражения приводит к выводу, что <= -0.354 при любых значениях x. А это значит, что отрицательно на всем отрезке (0, 1.5), и уже из этого следует, что на отрезке (0, 1.5) функция f(x) монотонна и имеет один корень.

Значения первой и второй производной на отрезке (0, 1.5) из таблицы рис.2 будут использованы в последующий методах.

**3. Этап уточнения корня**

**3.1 Метод итераций**

Приведем уравнение f(x)=0 к виду. Тогда рекуррентная формула . Для сходимости процесса простых итерации необходимо, чтобы при. Если то сходимость не обеспечена.

Приведем уравнение f(x) = cos(x) – (x + 2)1/2 + 1 = 0 к виду и проведем исследование.

*# Определим функцию ф*

**def** phi(x):

**return** ((np.cos(x)+1)\*\*2 - 2)

*# Вычислим первую производную функции ф*

z = Symbol('z')

phi\_ = ((cos(z)+1)\*\*2 - 2)

phi\_derivative = phi\_.diff(z)

phi\_derivative

phi1 = lambdify(z, phi\_derivative)

y\_phi = phi(x)

y\_phi1 = phi1(x)

*# Изобразим график первой производной функции ф*

fig = go.Figure()

fig.add\_trace(go.Scatter(x=[begin, end],

y=[0, 0],

name="Ось Х"))

fig.add\_trace(go.Scatter(x=x,

y=y\_phi1,

name="График производной функции фи"))

fig.add\_trace(go.Scatter(x=[a, a],

y=[-2, 1],

name="Ось A"))

fig.add\_trace(go.Scatter(x=[b, b],

y=[-2, 1],

name="Ось B"))

fig.add\_trace(go.Scatter(x=[begin, end],

y=[1, 1],

name="Ограничение сверху"))

fig.add\_trace(go.Scatter(x=[begin, end],

y=[-1, -1],

name="Ограничение снизу"))

fig.update\_layout(title="Построение производной функции фи",

xaxis\_title="Значение аргумента",

yaxis\_title="Значение функции")

fig.show()

Изображение выглядит как текст, линия, График, снимок экрана

Автоматически созданное описание

Рисунок 3 – График функции ф

В приведенном примере на отрезке (0, 1.5) , поэтому построим функцию ϕ(x) = х + λf(x), где параметр λ может быть определен по правилу: λ = , а в знаменатель следует подставить (x), у которого то есть . Тогда рекуррентная формула: .

Для получения решения уравнения методом итерации необходимо воспользоваться следующей рекуррентной формулой:, x0 = 0.

x0 = a

sp\_x = [x0]

sp\_f = [f(x0)]

**for** i **in** range(3):

x\_n = phi(x0)

sp\_x.append(x\_n)

sp\_f.append(f(x\_n))

x0 = x\_n

df = pd.DataFrame(data={'n':range(i+2), 'x': sp\_x, 'f(x)': sp\_f})

df

Результаты вычислений представлены на рисунке 4.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рисунок 4 – Результаты вычислений методом итераций

Сходимость итерационного процесса подтверждается принадлежностью всех выбранному исходному отрезку изоляции корня [0;1.5] и стремлением f() к нулю.

Получим оценку погрешности результата после пяти итераций:

**3.2 Метод Ньютона**

Необходимые и достаточные условия сходимости метода Ньютона:

* f(x) непрерывна на [a; b] и f(a);
* f(x) и f(x) отличны от нуля и сохраняют знаки для.

В нашем случае на отрезке [0;1.5] требования сходимости выполняются.

Начальное приближение аргумента x0 должно удовлетворять условию:, т. е. за начальное приближение следует принять тот конец отрезка, где знак функции и знак второй производной совпадают. Поскольку и (рис. 5), то выберем начальное приближение к корню: x0=0.

Изображение выглядит как линия, снимок экрана, График, текст

Автоматически созданное описание

Рисунок 5 – График второй производной f

Для получения решения уравнения методом Ньютона воспользуемся следующей рекуррентной формулой: .

В нашем случае , x0 = 0.

E = [i **for** i **in** range(-2, -5, -1)]

sp\_i = []

**for** eps **in** E:

x0 = a

i = 0

sp\_x = [x0]

sp\_f = [f(x0)]

x\_n = x0 - f(x0)/f1(x0)

**while** abs(x\_n - x0) > 10\*\*eps:

**if** eps == -4:

sp\_x.append(x\_n)

sp\_f.append(f(x\_n))

x0 = x\_n

x\_n = x0 - f(x0)/f1(x0)

i += 1

sp\_i.append(i)

df = pd.DataFrame(data={'n':range(sp\_i[-1]+1), 'x': sp\_x, 'f(x)': sp\_f})

df

Результат вычисления корня с точностью до представлен на рисунке 6.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 6 – Результаты вычисления при помощи метода Ньютона.

Оценим погрешность после пяти итераций:

График зависимости количества итераций от погрешности представлен на рисунке 7.

Изображение выглядит как текст, линия, График, снимок экрана

Автоматически созданное описание

Рисунок 7 – Зависимости количества итераций от погрешности

**4. Заключение**

Сравнивая метод итераций и метод Ньютона, можно с уверенностью сказать, что при помощи метода Ньютона приближенное значение х сходится к корню уравнения быстрее.