РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ С ПОМОЩЬЮ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Дударева О.В., к.ф.-м. н., доцент
Муллаянова А.Р., бакалавриат
Бирский филиал УУНиТ, г.Бирск, Россия

Аннотация.В статье рассматривается численное решение нелинейных уравнений с использованием языка программирования Python. На примере уравнения демонстрируется реализация метода Ньютона (касательных). Приведён листинг программы на Python, которая автоматически вычисляет значения функции и её производных, выбирает начальное приближение, выполняет итерационный процесс до достижения заданной точности и выводит найденные корни. Результаты работы программы подтверждаются графическим анализом.

Ключевые слова:язык программирования python, нелинейные уравнения, метод Ньютона(касательных).

В инженерной и научной практике часто встречаются уравнения, которые невозможно решить аналитически. Это так называемые нелинейные уравнения, содержащие переменную в степенях, под знаками тригонометрических, экспоненциальных или логарифмических функций. Для их решения применяются численные методы. Среди них особое место занимает метод Ньютона, также известный как метод касательных — один из самых быстрых и эффективных алгоритмов уточнения корней.

Язык программирования Python благодаря своей простоте и наличию мощных математических библиотек является идеальным инструментом для реализации метода Ньютона [1-2]. Рассмотрим этот подход на конкретном примере.

Пусть дано нелинейное уравнение:

$$3x^2\text{-}2e^{\cos(x\text{-}0,2)}=0 \quad (1)$$

Требуется найти его корни с заданной точностью ε=0,0001.

Метод Ньютона основан на линеаризации функции в окрестности текущего приближения. Итерационная формула имеет вид:

$$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)},\ n = 0,1,2,... \quad (2)$$

где $F'(x_n)$ - производная функции в точке x_n.

Для гарантии сходимости метода Ньютона начальное приближение должно выбираться специальным образом. Используется следующий критерий:если $F(b)\cdot F''(x) > 0$ на $[a;b]$, то $x_0 = b$;иначе $x_0 = a$.Итерационный процесс продолжается до тех пор, пока изменение приближения не станет меньше заданной точности:

$$|x_{n+1} - x_n| < \varepsilon, \quad (3)$$

где ε - заданная точность, в нашем случае она равно 0,0001

На Рис.1 представлен полный код программы, реализующей метод Ньютона для решения уравнения (1).Программа вычисляет значения исходной функции $f(x)$, её первой производной $f'(x)$ и второй производной $f''(x)$ с помощью заранее определённых математических выражений. Затем, используя представленный выше критерий, программа автоматически выбирает начальное приближение c — либо левую, либо правую границу отрезка. После этого запускается итерационный цикл: на каждом шаге текущее приближениеc_0 уточняется по формуле (2), а счётчик итераций n увеличивается. Программа выводит промежуточные значения, позволяя наблюдать за процессом сходимости. Цикл продолжается до тех пор, пока модуль разности между двумя последовательными приближениями $|c - c_0|$ не станет меньше заданной точности ε. По завершении программа выводит найденный корень с семью знаками после запятой, количество выполненных итераций и значение функции в корне, которое должно быть близко к нулю, подтверждая корректность решения.

```
import math
def f(x: float) -> float:
    return 3*(x**2)-2*(math.exp(math.cos(x-0.2)))
def df(x: float) -> float:
    return 6*x+2*(math.exp(math.cos(x-0.2)))*(math.sin(x-0.2))
def ddf(x: float) -> float:
    return 6+2*(math.cos(x-0.2)-(math.sin(x-0.2)**2))*math.exp(math.cos(x-
   0.2))
print("Введите значения a и b:")
a = float(input("a = "))
b = float(input("b = "))
eps = float(input("Введите точность eps: "))
if f(a) * ddf(a) > 0:
    c = a
else:
    c = b
n = 0
while True:
    c0 = c
    c = c0 - f(c0) / df(c0)
    n += 1
    if abs(c - c0) < eps:
        break
print(f"Корень x = {c:.7f}")
print(f"Количество итераций = {n}")
```

Рис.1. Код программы, реализующей метод Ньютона

При решении уравнения (1) для разныхотрезков программа показывает результаты, представленные на Рис.2. На отрезке [−1;−0,9] первый корень уравнения x_1=−0,9853583. На отрезке [1,1; 1,2] второй корень уравнения x_2= 1,1098304.

```
Введите значения a и b:
a = -1
b = -0.9
Введите точность eps: 0.0001
Корень x = -0.9853583
Количество итераций = 2
>>>
====== RESTART: C:/Users/user/A
Введите значения a и b:
a = 1.1
b = 1.2
Введите точность eps: 0.0001
Корень x = 1.1098304
Количество итераций = 3
>>>
```

Рис. 2.Результат работы программы

Для проверки правильности решения построим график функции(1). Из графика, представленного на Рис.3 видно, что на отрезках [−1;−0,9] и [1,1; 1,2] имеется два корня уравнения.

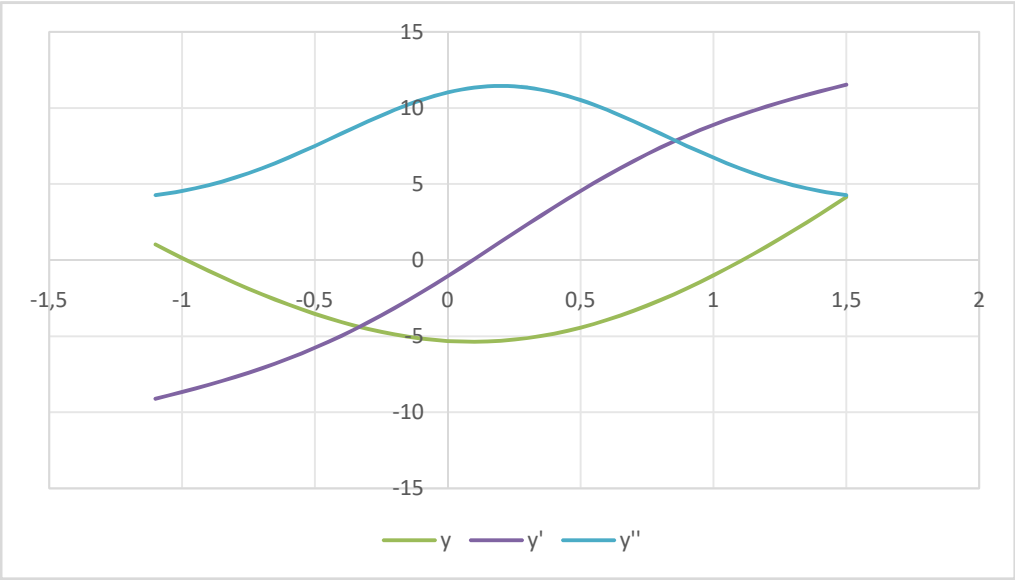

Рис.3. График функции

Представленный пример наглядно демонстрирует преимущества описанного подхода. Прежде всего, метод Ньютона обеспечивает высокую скорость сходимости: уже через 4–5 итераций он даёт результат с точностью до 10^{-4}, что значительно быстрее многих других численных методов. При этом важным достоинством является простота реализации. На языке программирования Python метод Ньютона записывается буквально в несколько строк, что делает его доступным даже для начинающих программистов. Кроме того, подход отличается гибкостью: с помощью библиотеки matplotlib можно легко отделить корни графически, а затем уточнить их численно тем же методом. Программа выводит протокол каждой итерации, что обеспечивает наглядность и позволяет детально отслеживать процесс сходимости на каждом шаге вычислений.

Python — мощный и удобный язык для численного решения нелинейных уравнений. С его помощью можно быстро реализовать метод Ньютона, отделить корни графически и получить решение с заданной точностью за минимальное число итераций. Решенная задачанаглядно показывает, что метод касательных является не только теоретически обоснованным, но и практически

эффективным инструментом, который легко программируется и даёт отличные результаты даже для таких сложных функций, как комбинация степенных и экспоненциально-тригонометрических выражений.

Литература:

1. Лапчик М. П. Численные методы: Учеб. пособие для студ. вузов / М. П. Лапчик, М. И. Рагулина, Е. К. Хеннер; Под ред. М. П. Лапчика. – М.: Издательский центр «Академия», 2004. – 384 с.

2. Маккини У. Pythonи анализ данных: первичная обработка данных с помощью pandas, NumPyиIPython/ У.Маккини - 2-е изд. - Москва: ДМК Пресс, 2020.