**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Лабораторная работа №1**

**«Численные методы»**

**Вариант 4**

Ёды Никиты Дмитриевича

студента 3 курса, 6 группы

специальность «прикладная математика»

Преподаватель:

Будник А.М.

**Постановка задачи**

Найти корень нелинейного уравнения

*,*

Применяя следующие методы:

1. Метод простой итерации
2. Метод Ньютона
3. Метод Чебышева

Необходимо:

а. Отделить хотя бы один (если их несколько) корень уравнения графическим методом

б. Выбрать начальное приближение , исходя из выполнения условий теоремы о сходимости метода простой итерации

в. Используя выбранное начальное приближение , найти тремя вышеуказанными методами решение данного нелинейного уравнения с точностью . Критерием останова итерационного процесса выбрать следующий: *.*

г. Сравнить методы по скорости сходимости (теоритически и практически) и точности (оценить величину невязки каждого метода)

**Отделение корней**

Пусть *f(x) =* . Вычислим значение функции при *x = 0, 1, -1.*

*f(0) = 1, f(1) = 5, f(-1) = -3*

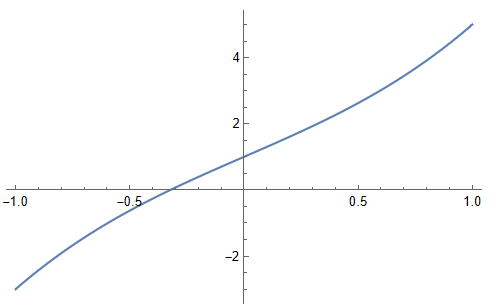


Рисунок 1 - График функции в Wolfram Mathematica

Возьмем отрезок . На отрезке есть корень, т.к. на концах отрезка меняется знак. Ищем корень на промежутке

**Метод половинного деления отрезка**

Найдем середину отрезка :

*c = = = -0.25*

Находим значение *f (-0.25) = 0.234475*. Тогда начальное приближение:

*-0.25*

**Проверка выполнения условий теоремы о сходимости метода Ньютона**

Находим :

Строим интервал *:*

Значения на границах :

*.*

*,*

Находим *M:*

*, M =1.5*

Проверяем условие:

*-0.220588 3.1875*

Выполняется, следовательно, метод Ньютона сходится.

**Метод простой итерации**

Уравнение**:** . Применение метода требует предвари-тельного приведения уравнения *f(x) = 0* к каноническому виду *x = φ(x)*, где *φ(x)* — это заданная функция.   
 Метод простой итерации будет иметь следующий вид:

*k = 0, 1, 2, . . .*

В данном случае, *φ (x) = x =*

Критерием остановки итерационного процесса является условие:

где

**Реализация**

Строим итерационный процесс на языке Python:

|  |
| --- |
| def f(x):      return x\*\*3 + 3\*x + 1  def phi(x):      return (x\*\*3 + 1) / 3  def simple\_iteration\_method(x0, epsilon=1e-7, max\_iter=1000):  iteration = 0      while True:          x\_next = phi(x0)          if abs(x\_next - x0) <= epsilon:              Break          x0 = x\_next          iteration += 1          if iteration >= max\_iter:              print("Максимальное число итераций достигнуто.")              break      residual = f(x\_next)      return x\_next, iteration, residual  initial\_guess = -0.25  solution, iterations, residual = simple\_iteration\_method(initial\_guess)  print("Приближенное решение:", solution)  print("Количество итераций:", iterations)  print("Невязка:", residual) |

**Вывод по полученным результатам**

Корень нелинейного уравнения , на отрезке :

Невязка

*2.08\**

Количество итераций: *7*.

**Метод Ньютона**

Итерационный процесс вида:

*= −*

называется методом Ньютона.

**Реализация**

|  |
| --- |
| def f(x):      return x\*\*3 + 3\*x + 1  def f\_prime(x):      return 3\*x\*\*2 + 3  def newton\_method(x0, epsilon=1e-7, max\_iter=1000):      iteration = 0      while True:          x\_next = x0 - f(x0) / f\_prime(x0)          if abs(x\_next - x0) <= epsilon:              break          x0 = x\_next          iteration += 1          if iteration >= max\_iter:              print("Максимальное число итераций достигнуто.")              break      residual = f(x\_next)      return x\_next, iteration, residual  initial\_guess = -0.25  solution, iterations, residual = newton\_method(initial\_guess)  print("Приближенное решение:", solution)  print("Количество итераций:", iterations)  print("Невязка:", residual) |

**Вывод по полученным результатам**

Корень нелинейного уравнения *,* на отрезке :

Невязка

*1.11\**

Количество итераций: *3*.

**Метод Чебышева**

Итерационный процесс вида:

называется методом Чебышева.

**Реализация**

|  |
| --- |
| import numpy as np  def chebyshev\_method(func, x0, epsilon=1e-7, max\_iter=1000):      x\_prev = x0      x\_curr = x0      iteration = 0      while True:          iteration += 1          x\_next = x\_curr - func(x\_curr) / chebyshev\_derivative(func, x\_curr)          if np.abs(x\_next - x\_curr) <= epsilon or iteration >= max\_iter:              break          x\_prev = x\_curr          x\_curr = x\_next      return x\_next, iteration  def func(x):      return x\*\*3 + 3\*x + 1  def chebyshev\_derivative(func, x):      h = 1e-5      return (func(x + h / 2) - func(x - h / 2)) / h  def residual(func, root):  return np.abs(func(root))  x0 = -0.25  root, iterations = chebyshev\_method(func, x0)  residual\_value = residual(func, root)  print("Корень уравнения:", root)  print("Невязка:", residual\_value)  print("Количество итераций:", iterations) |

**Вывод по полученным результатам**

Корень нелинейного уравнения , на отрезке :

Невязка

*1.11\**

Количество итераций: *4.*

**Анализ методов**

|  |  |
| --- | --- |
| Простая итерация | Необходимо выбрать подходящую функцию (x) длясходимости.  Скорость сходимости зависит от выбора (x). |
| Метод Ньютона | Требует вычисления производной функции.  Имеет квадратичную скорость сходимости при достаточно близком начальном приближении. |
| Метод Чебышева | Не требует вычисления производной.  Имеет быструю сходимость, особенно при сближении к корню. |

На практике методу Ньютона понадобилось меньше всего итераций, а методу простой итерации больше всего. Решения полученные методами Чебышева и Ньютона оказались точнее. Наиболее простой в реализации оказался метод простой итерации.