# Контрольная работа №6, задание 1

Вершинин Данил Алексеевич

4 июня 2024 г.

## Условие

Дано нелинейное уравнение: $16x^5 + 24x^3 - 2x^2 - 11 = 0$. Отделить корни. Предложить сходящийся метод простой итерации для уточнения корня. Оценить, хотя бы грубо, скорость сходимости. Построить итерационный процесс по методу Ньютона.

## Решение

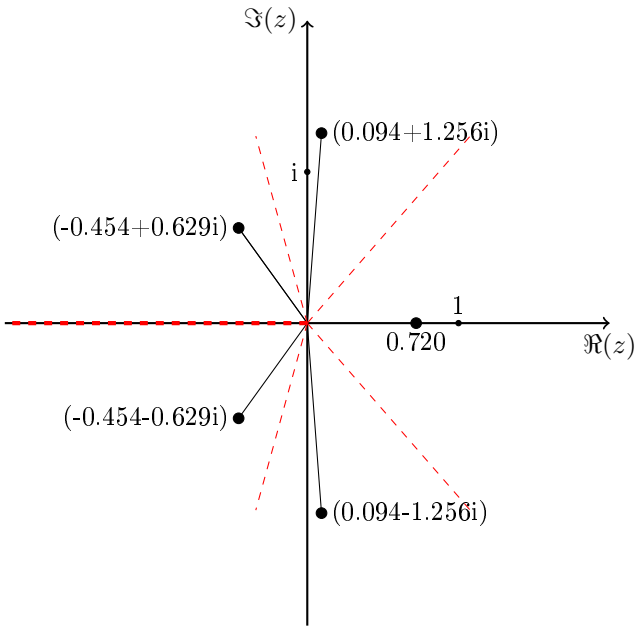


Рис. 1: Корни уравнения. Область поиска разделена.

Предложим метод итерации:

$$16x^5 + 24x^3 - 2x^2 - 11 = 0$$

$$16x^5 + 24x^3 = 2x^2 + 11 \;\Rightarrow\; x^3(16x^2 + 24) = 2x^2 + 11 \;\Rightarrow\; x^3 = \frac{2x^2 + 11}{16x^2 + 24}$$

$$\phi(x_n) = x_{n+1} = \sqrt[3]{\frac{2x_n^2 + 11}{16x_n^2 + 24}}$$

Найдём производную полученной функции:

$$\left(\sqrt[3]{\frac{2x_n^2 + 11}{16x_n^2 + 24}}\right)' = \frac{1}{3} \cdot \left(\frac{2x_n^2 + 11}{16x_n^2 + 24}\right)^{-\frac{2}{3}} \times \frac{4x(16x^2 + 24) - 32x(2x^2 + 11)}{(16x^2 + 24)^2}$$

На Рис. 2 показано, что функция, задающая уравнение, имеет только один вещественный корень. Заметим, что производная предложенной функции итерации по модулю всегда меньше единицы. Что позволяет нам сделать вывод о сходимости предложенной функции в окрестности вещественного корня (см Рис 3.).

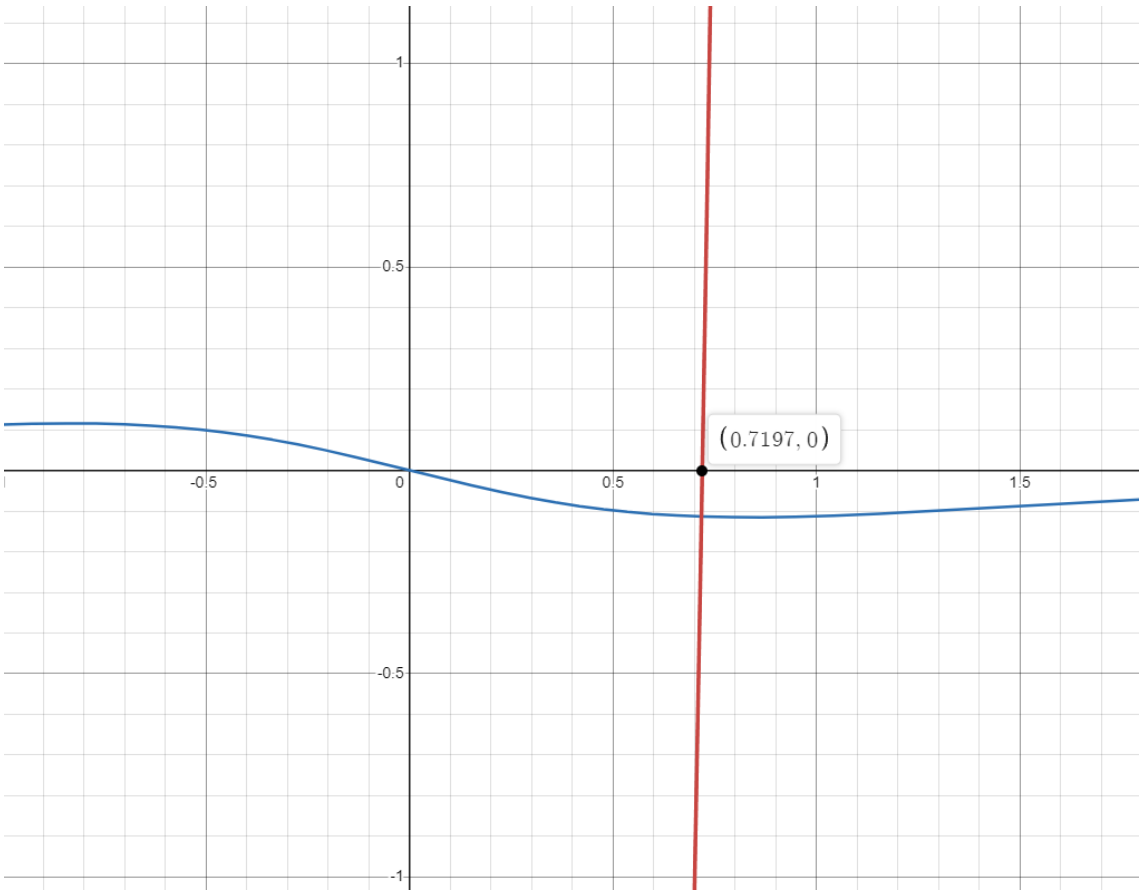


Рис. 2: функция из уравнения (красный). Производная предложенного метода итераций (синий)

## Метод Ньютона

Функцию дважды дифференцируема. Можно применить метод Ньютона.

Для применения метода Ньютона построим итерационную формулу:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Построим нашу функцию:

$$x_{n+1} = x_n - \frac{16x_n^5 + 24x_n^3 - 2x_n^2 - 11}{80x_n^4 + 72x_n^2 - 4x_n}$$

Результаты этого метода (использовалась программа [2]):

```
0.09408799815214919+1.256160052262912i
0.09408799815214919-1.256160052262912i
-0.45393578883367447+0.6292479954643618i
-0.45393578883367447-0.6292479954643618i
0.7196955813634115
```

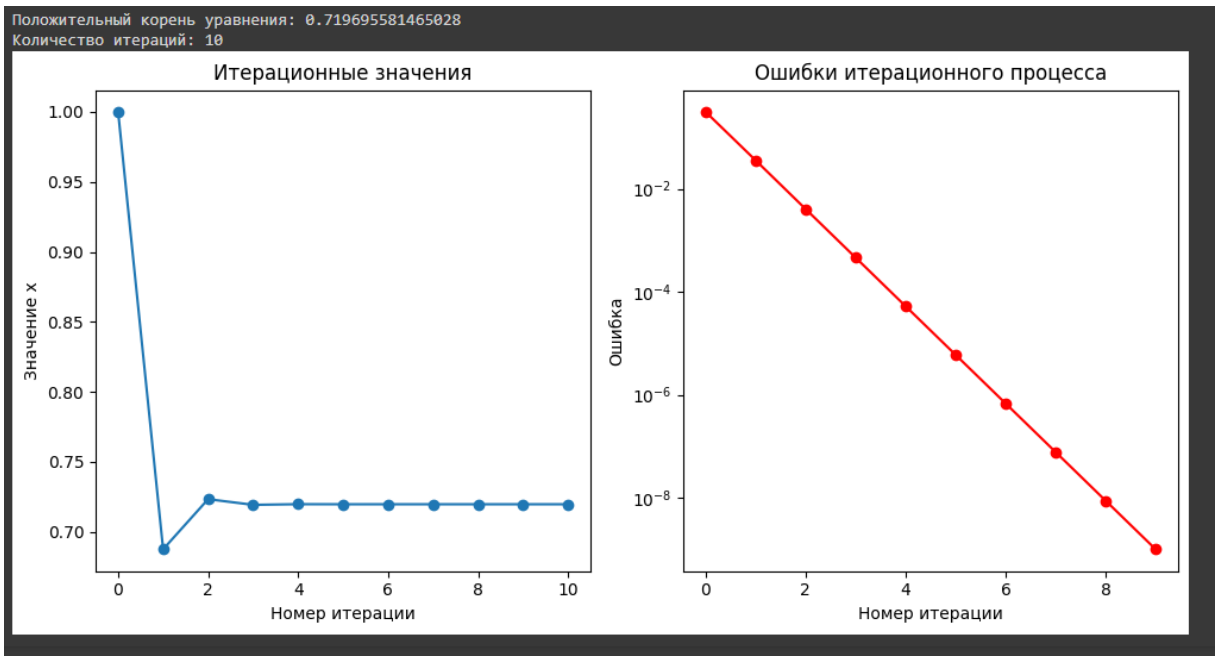


Рис. 3: Вывод программы (см приложение [1]). За счёт того, что производная окрестности корня отрицательная, мы "перескакиваем"через корень. Метод имеет линейную скорость

## Приложение

[1] Код на python для расчетов метода простой итерации

```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp
xx = sp.symbols('x')
f = sp.root((2*xx**2 + 11) / (16*xx**2 + 24), 3)


# Функция итерационного процесса
def phi(x_i):
return f.subs(xx, x_i)

# Параметры итерационного процесса
x0 = 1.0  # Начальное приближение
tolerance = 1e-9  # Допуск
max_iterations = 100  # Максимальное количество итераций

# Массивы для хранения значений
x_values = [x0]
errors = []

# Итерационный процесс
for n in range(max_iterations):
x_next = phi(x_values[-1])
x_values.append(x_next)
error = abs(x_next - x_values[-2])
errors.append(error)
if error < tolerance:
break

# Печать результатов
print(f"Положительный корень уравнения: {x_values[-1]}")
print(f"Количество итераций: {len(x_values) - 1}")
```

```
# Визуализация сходимости
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.plot(x_values, marker='o')
plt.title("Итерационные значения")
plt.xlabel("Номер итерации")
plt.ylabel("Значение x")

plt.subplot(1, 2, 2)
plt.plot(errors, marker='o', color='r')
plt.yscale('log')
plt.title("Ошибки итерационного процесса")
plt.xlabel("Номер итерации")
plt.ylabel("Ошибка")

plt.tight_layout()
plt.show()
```

[2] Код на python для расчетов методом Ньютона

```
f = lambda z: 16*z**5 + 24*z**3 -2*z**2-11
df = lambda z: 80*z**4 + 72*z**2 - 4*z

def newton_method_complex(f, df, z0, tol=1e-6, max_iter=1000):
z = z0
for i in range(max_iter):
z_next = z - f(z) / df(z)
if abs(z_next - z) < tol:
print(i)
return z_next
z = z_next
raise ValueError("Не удалось достичь сходимости")

# Начальные приближения (включая комплексные)
initial_guesses_complex = [0 + 1j, 0 - 1j, -1 + 1j, -1 - 1j, 1 + 0j]

# Нахождение корней комплексным методом Ньютона
complex_roots = []
for guess in initial_guesses_complex:
try:
root = newton_method_complex(f, df, guess)
if not any(np.isclose(root, r, atol=1e-6) for r in complex_roots):
complex_roots.append(root)
except ValueError:
pass

print("Найденные корни методом Ньютона:")
for root in complex_roots:
print(root)
```