**ИНСТИТУТ ТРАНСПОРТА И СВЯЗИ**



ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК И ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНЫХ СИСТЕМ

**Лабораторная работа №3**

по дисциплине

„Численные методы”

Тема: «Методы решения нелинейного уравнения.»

Выполнил: Денис Белов, Андрей Савкин, Евгений Хрущ

Рига.

2020.

**1)** **Формулировка задания**

В данной работе необходимо было реализовать метод бисекции и индивидуальный метод: метод парабол. В зависимости от способа реализации: с использованием скользящего окна или с установкой локальной области поиска, будет меняться входные данные. Общее в низ будет то, что требуется установить точность решения.

Так же для некоторых задач надо предусмотреть смешение функции по оси Y.

**2) Метод бисекций**

 def bisection(self, lower, upper, max\_iterations=500, tolerance=0.0001):

        """ Finds the root of the function by the Bisection Method

        lower -- a

        upper -- b

        """

        self.\_description\_line("Bisection", lower, upper, tolerance)

        print(f'{"Iteration":<10} | {"a":<16} | {"Estimate":<20} | {"b":<16} | {"Error":<16}')

        print('{:-<80}'.format(""))

        initial\_check = self.\_initial\_check(lower, upper, tolerance)

        if initial\_check[0]:

            return initial\_check[1]

        iterations = 0

        while iterations < max\_iterations:

            midpoint = (lower + upper) / 2.0

            print(f'{iterations:<10} | {"%0.8f" % lower:<16} | {"%0.8f" % midpoint:<20} | {"%0.8f" % upper:<16} | '

                  f'{"%0.8f" % abs(upper - lower):<16}')

            if self.function(midpoint) == 0 or abs(upper - lower) <= tolerance:

                print("--- End of Bisection ---\n")

                return midpoint

            if self.function(lower) \* self.function(midpoint) < 0:

                upper = midpoint

            elif self.function(upper) \* self.function(midpoint) < 0:

                lower = midpoint

            else:

                print("--- End of Bisection ---\n")

                return midpoint

            iterations += 1

        self.final\_estimate = (lower + upper) / 2.0

        self.iterations = iterations

        print(f'{iterations:<10} | {"%0.8f" % lower:<16} | {"%0.8f" % self.final\_estimate:<20} | '

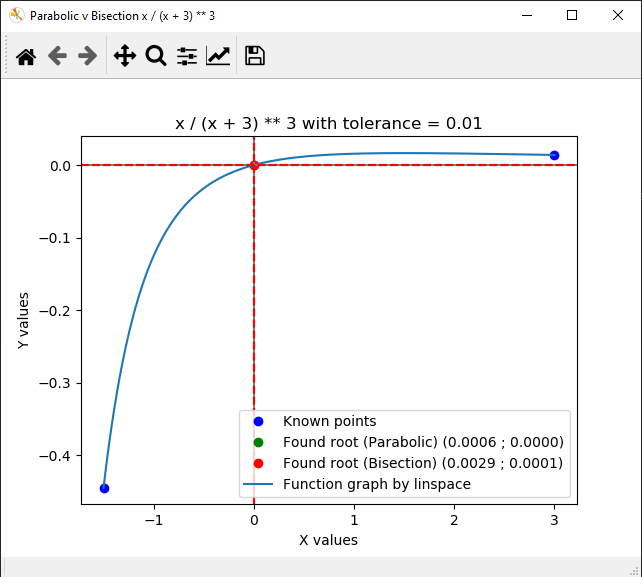
              f'{"%0.8f" % upper:<16} | {"%0.8f" % abs(upper - lower):<16}')

        print("--- End of Bisection ---\n")

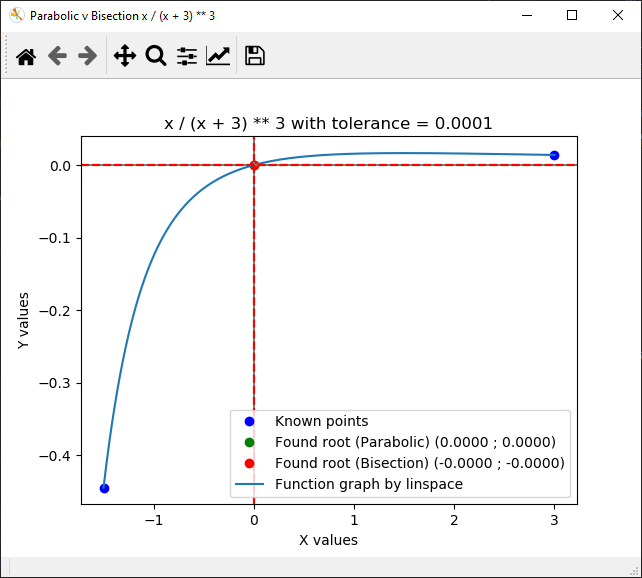
        return self.final\_estimate

**График 1:**

Е=0.01

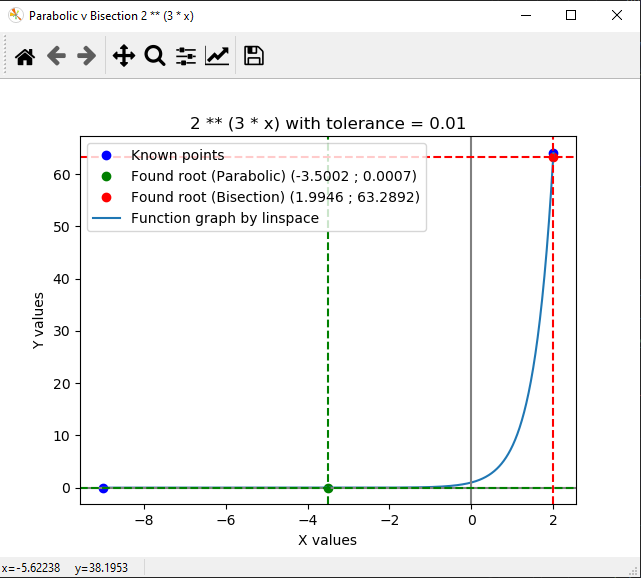


Е=0.0001

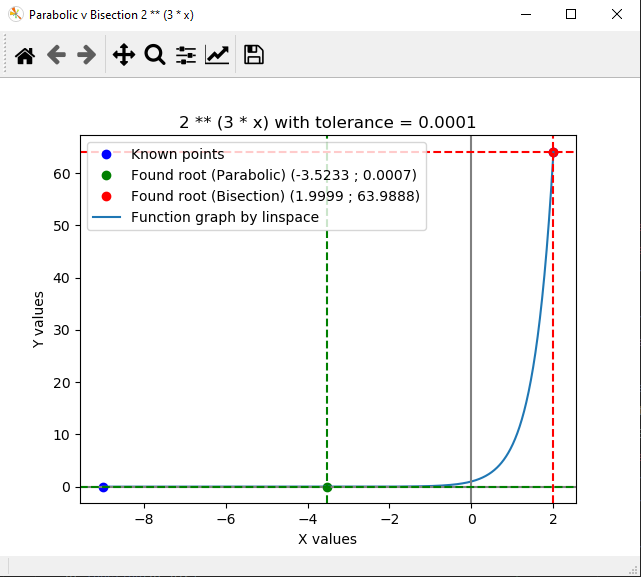


**График 2:**

Е=0.01

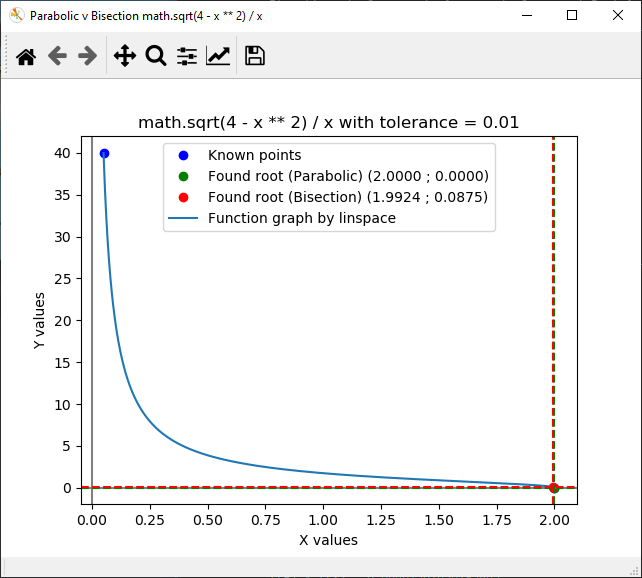


Е=0.0001



**График 3:**

Е=0.01



Е=0.0001

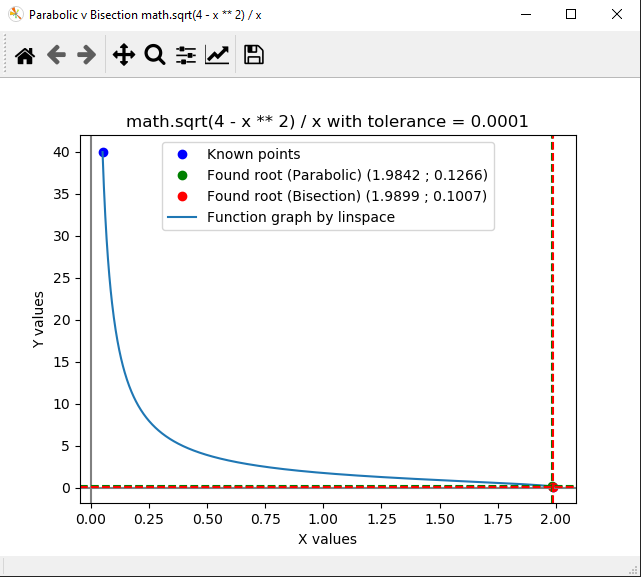


Таблица.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер функции | Точность | Найденный корень | Число итераций |
| 1) x/(x+3)3 | e = 0.01 | x= 0.0029 | n=8 |
| e = 0.0001 | x=-0.000022 | n=15 |
| 2) 23x | e = 0.01 | x=1.99 | n=9 |
| e = 0.0001 | x= 1.99 | n=16 |
| 3) | e = 0.01 | x=1.982 | n=7 |
| e = 0.0001 | x=1.989 | n=14 |

**3)** **Индивидуальный метод: метод парабол**

def parabolic(self, a, b, max\_iterations=50, tolerance=0.0001):

        iterations = 0

        x0 = (a + b) / 2

        self.\_description\_line("Parabolic (Mueller's)", a, b, tolerance)

        print("Initial x's =", a, ",", x0, ",", b)

        print(f'{"Iteration":<10} | {"a0":<8} | {"a1":<8} | {"a2":<8} | {"Root 1":<12} | {"Root 2":<12} | '

              f'{"a":<8} | {"b":<8} | {"Estimate (xi)":<20} | {"Error":<16}')

        print('{:-<136}'.format(""))

        initial\_check = self.\_initial\_check(a, b, tolerance)

        if initial\_check[0]:

            return initial\_check[1]

        current\_x = x0

        prev\_x = 0.0

        while iterations < max\_iterations:

            # Step 3 - get Y values from X ---

            xs = [a, current\_x, b]

            ys = [self.function(a), self.function(x0), self.function(b)]

            # Step 3 - get coefficients of a quadratic function ---

            a0, a1, a2 = self.get\_coefficients\_lagrange(xs, ys)

            # Step 4 - solving quadratic equation ---

            root\_1, root\_2 = self.solve\_quadratic\_equation(a0, a1, a2, current\_x)

            # Update boundaries

            if self.function(a) \* self.function(current\_x) < 0:

                a = a

                b = current\_x

            elif self.function(current\_x) \* self.function(b) < 0:

                a = current\_x

                b = b

            # Chose more appropriate root

            if a <= root\_1 <= b or a >= root\_1 >= b:

                current\_x = root\_1

            elif a <= root\_2 <= b or a >= root\_2 >= b:

                current\_x = root\_2

            print(f'{iterations:<10} | {"%0.4f" % a0:<8} | {"%0.4f" % a1:<8} | {"%0.4f" % a2:<8} | '

                  f'{"%0.4f" % root\_1:<12} | {"%0.4f" % root\_2:<12} | '

                  f'{"%0.4f" % a:<8} | {"%0.4f" % b:<8} | {"%0.8f" % current\_x:<20} | '

                  f'{"%0.8f" % abs(current\_x - prev\_x):<16}')

            if abs(current\_x - prev\_x) <= tolerance:

                print("--- End of Parabolic (Mueller's) ---\n")

                return current\_x

            prev\_x = current\_x

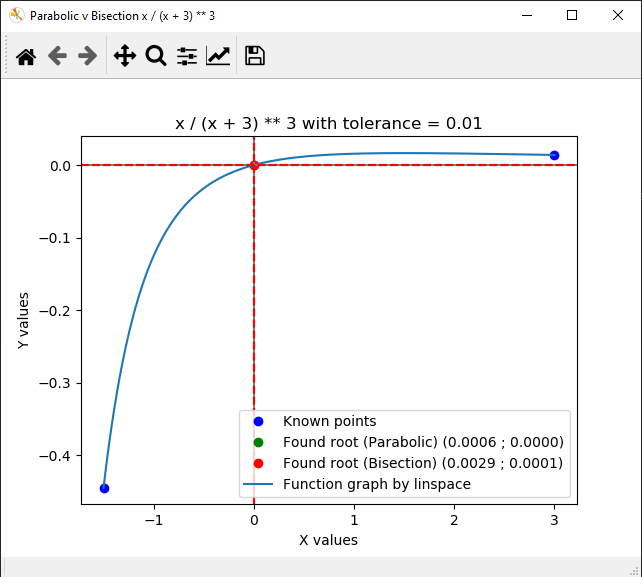
            iterations += 1

        print("--- End of Parabolic (Mueller's) ---\n")

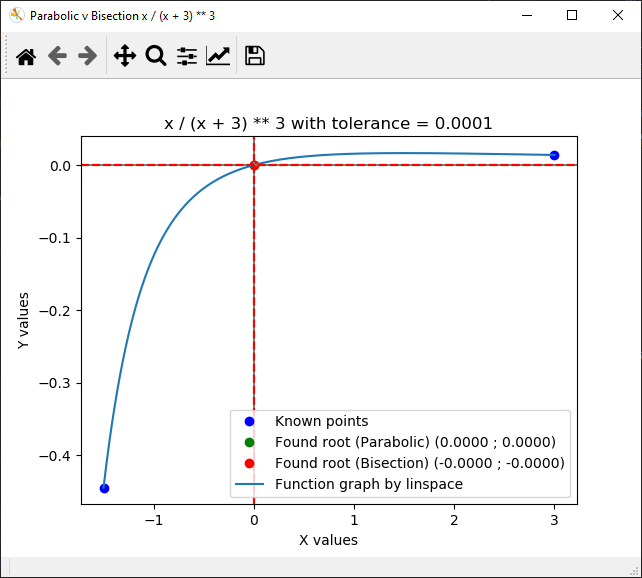
        return current\_x

**График 1:**

Е=0.01

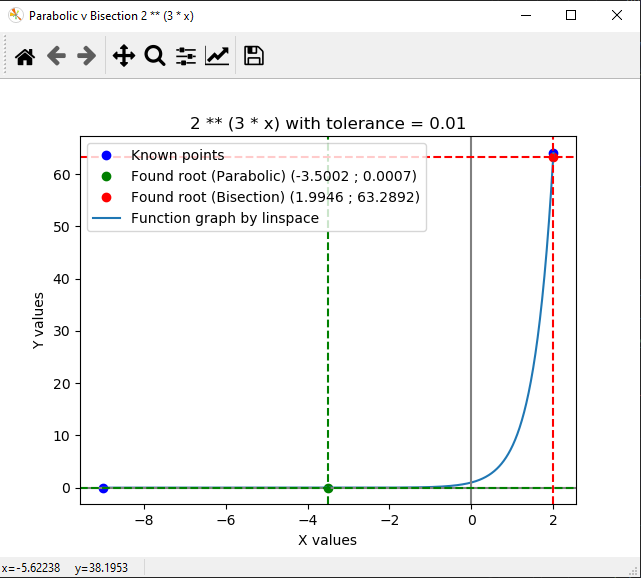


Е=0.0001

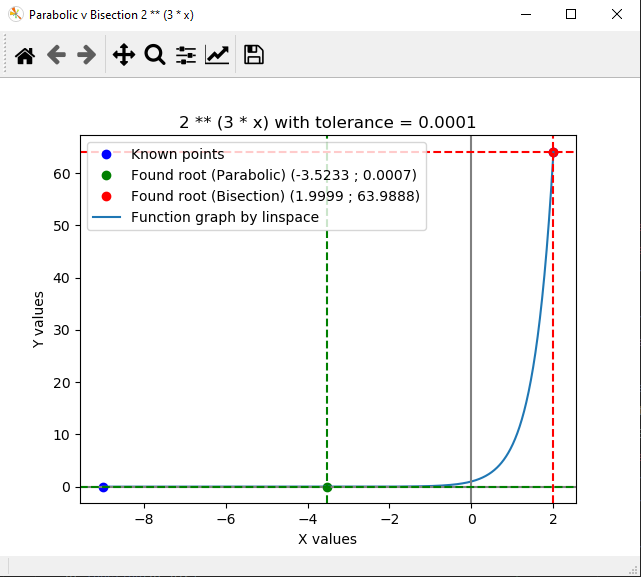


**График 2:**

Е=0.01

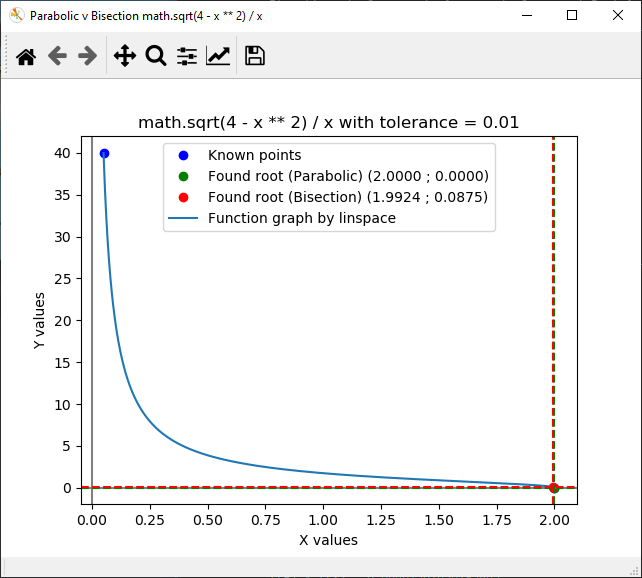


Е=0.0001



**График 3:**

Е=0.01



Е=0.0001

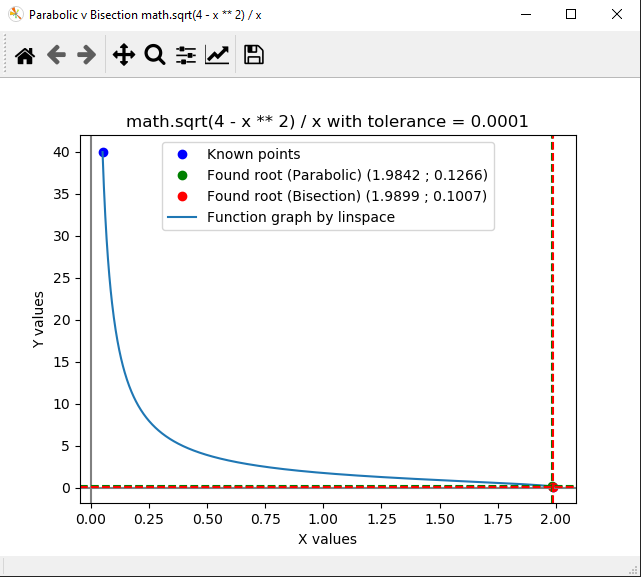


Таблица.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер функции | Точность | Найденный корень | Число итераций |
| 1) x/(x+3)3 | e = 0.01 | x=0.0006 | n=6 |
| e = 0.0001 | x=0.00000003 | n=8 |
| 2) 23x | e = 0.01 | x=-3.5 | n=1 |
| e = 0.0001 | x=-3.55 | n=499 |
| 3) | e = 0.01 | x=1.984 | n=1 |
| e = 0.0001 | x=1.984 | n=1 |

**4) Вывод**

|  |  |  |
| --- | --- | --- |
| Точность  Метод | e = 0.01 | e = 0.0001 |
| Метод бисекции  x/(x+3)3 | n=8  x=0.0029 | n=15  x=-0.000022 |
| Метод парабол  x/(x+3)3 | n=6  x=0.0006 | n=8  x=0.00000003 |

Из полученных значений можно сделать вывод, что метод парабол совершает меньше итераций, чем метод бисекция. При этом корень в методе парабол находится точнее, что делает её лучше по всем аспектам. Правда метод парабол требует больше вычислений, чем бисекция.