**Исламов Радмир Лабораторная 2**

**Задание 1**

function [ res ] = lab2\_1( x )

res = x.^5 - 19\*x.^4 + 106\*x.^3 - 70\*x.^2 - 539\*x - 343;

end

x = -10:0.1:10;

>> plot(x, lab\_2\_1(x))

>> grid on

>> p = [1 -19 106 -70 -539 -343];

>> roots(p)

ans =

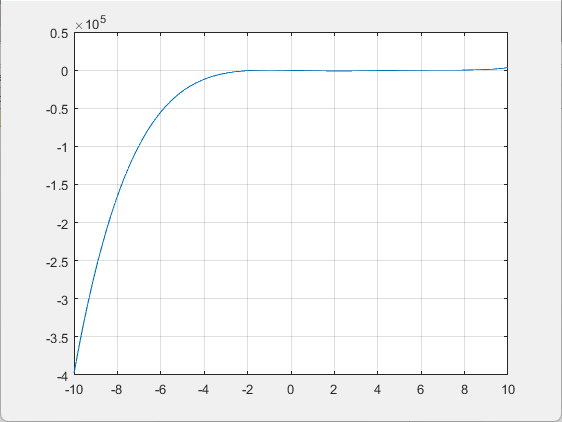
7.0000

7.0000 + 0.0000i

7.0000 - 0.0000i

-1.0000

-1.0000



**Задание 2**

function [root] = meth\_dih(otr)

x1 = otr(1); x2 = otr(2);x = 0;c = 0;eps = 10^-16;

if (lab\_2\_1(x1)\*lab\_2\_1(x2) > 0)

print('Некорректный отрезок')

else

while(abs(lab\_2\_1(x)) >= eps)

x = (x1 + x2)/2;

if (lab\_2\_1(x1)\*lab\_2\_1(x) > 0)

x1 = x;

else

x2 = x;

end

c = c + 1;

end

root = x;

c

end

end

>> otr = -10:20:10;

>> meth\_dih(otr)

c =

17

ans =

7.0000

**Задание 3**

function [ root ] = nuton(f, x0)

syms x\_s

x = x0 - f(x0)/subs(diff(f(x\_s)), x0); c = 0;

while( f(x) ~= 0 )

x = x - f(x)/subs(diff(f(x\_s)), x);

c = c + 1;

end

root = x;

c

end

>> f = @(x) x^5-19\*x^4+106\*x^3-70\*x^2-539\*x-343;

>> nuton(f, 8)

c =

16

ans =

7

>> nuton(f, -2)

c =

11

ans =

-1

**Задание 4**

function [ root ] = nuton\_mod(f, x0, p)

syms x\_s

x = x0 - p\*f(x0)/subs(diff(f(x\_s)), x0); c = 0;

while( f(x) ~= 0 )

x = x - p\*f(x)/subs(diff(f(x\_s)), x);

c = c + 1;

end

root = x;

c

end

>> nuton\_mod(f, 8, 2)

c =

5

ans =

7

**Задание 5**

def square\_root(a):

x = 1.0 # Начальное значение x

while True:

next\_x = 0.5 \* (a / x + x) # Вычисляем следующее значение x по формуле метода

if abs(next\_x - x) < 1e-9: # Проверяем условие остановки: разница между текущим и следующим значением x достаточно мала

break

x = next\_x # Обновляем значение x для следующей итерации

return x

# Пример использования:

a = 16 # Число, из которого нужно извлечь квадратный корень

result = square\_root(a)

print("Квадратный корень из", a, "равен", result)

Квадратный корень из 16 равен 4.000000000000051

**Задание 6**

def square\_root(a):

convergence\_region = find\_convergence\_region(a)

if len(convergence\_region) == 0:

print("Метод не сходится для данного значения a")

return None

x = 1.0 # Начальное значение x

if x < convergence\_region[0]:

print("Начальное значение x находится за пределами области сходимости")

print("Выполняется расчет вне области сходимости")

while True:

next\_x = 0.5 \* (x + a / x)

if abs(next\_x - x) < 1e-9:

break

x = next\_x

return x

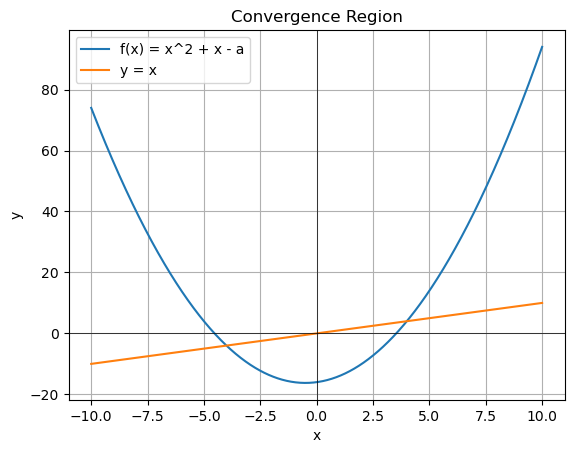
# Пример использования:

a = 16

result = square\_root(a)

if result is not None:

print("Квадратный корень из", a, "равен", result)



Начальное значение x находится за пределами области сходимости

Выполняется расчет вне области сходимости

Квадратный корень из 16 равен 4.000000000000051

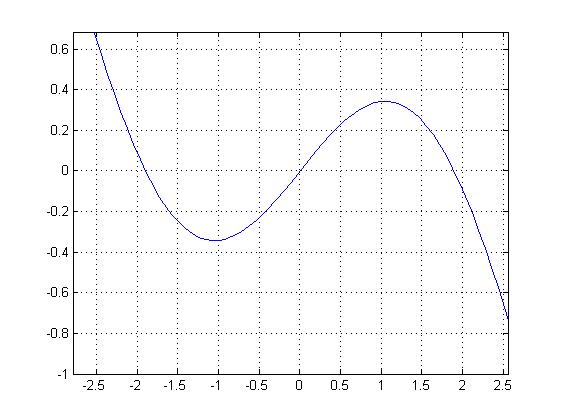
**Задание 7**

>> fzero(f, -4)

ans =

7.0000

**Задание 8**



Из графика видно отрезки с корнями: [-2; -1.5], [-0.1;0.1], [1.5, 2]

По Ньютону:

х1 = -1.8955;

х2 = 0;

х3 = 1.8955;

По fzero:

x1 = -1.8955;

x2 = 0;

x3 = 1.8955;

**Контрольные вопросы**

1. Из каких соображений, и какими методами можно локализовать искомый корень?

2. Каким образом реализуется заданная точность поиска в методе половинного деления и в методе Ньютона? Чем различаются условия прекращения итераций?

3. Почему с помощью метода половинного деления не удаётся находить корни двойной кратности?

4. Сравните (перечислите преимущества и недостатки) методов Ньютона и половинного деления.

5. Назовите условия применимости метода Ньютона.

6. Оцените скорость сходимости в методе Ньютона при поиске корней одинарной и двойной кратности.

7. Назовите условия сходимости метода простых итераций.

8. Каким образом можно априорно вычислить примерное количество итераций, требуемых для нахождения корня с заданной точностью, для всех изученных методов?

**Ответы:**

1. Для априорной оценки количества итераций, требуемых для нахождения корня с заданной точностью, можно использовать различные методы, такие как оценка скорости сходимости метода, аналитический анализ функции и применение теоретических результатов. Однако точное количество итераций заранее предсказать сложно, поэтому часто используется итеративный подход, в котором вычисления продолжаются до достижения заданной точности или выполнения других условий остановки. Локализацию искомого корня в MATLAB можно выполнить с помощью различных методов и функций:
   * Использование графического метода с помощью функции **fplot** для визуализации функции и определения интервала, где функция меняет знак.
   * Применение метода половинного деления с использованием функции **fzero**, которая находит корень функции на заданном интервале.
   * Использование метода Ньютона с помощью функции **fzero**, которая также может использовать метод Ньютона для нахождения корня.
2. В методе половинного деления в MATLAB заданная точность достигается путем контроля длины интервала, в котором находится корень. Условие прекращения итераций - достижение заданной точности или достижение максимального числа итераций (задается параметром **MaxIter**).

В методе Ньютона в MATLAB заданная точность обычно реализуется путем контроля разницы между последовательными приближениями к корню. Условие прекращения итераций - достижение заданной точности или достижение максимального числа итераций (задается параметром **MaxIter**).

1. Метод половинного деления не может находить корни двойной кратности, потому что он основан на поиске изменения знака функции на отрезке, и для корней двойной кратности нет смены знака в окрестности корня.
2. Преимущества метода половинного деления:
   * Гарантирует нахождение корня, если функция непрерывна и имеет разные знаки на концах отрезка.
   * Прост в реализации и не требует производных функции.

Недостатки метода половинного деления:

* + Скорость сходимости медленная, особенно для больших отрезков.

Преимущества метода Ньютона:

* + Быстрая сходимость, особенно вблизи корня.
  + Высокая скорость вычислений.

Недостатки метода Ньютона:

* + Требуется производная функции.
  + Может сойтись к неправильному корню или не сойтись вообще, если начальное приближение выбрано неправильно или функция имеет особенности (вершины, разрывы, асимптоты).

1. Метод Ньютона применим, если:
   * Функция дифференцируема в окрестности корня.
   * Начальное приближение выбрано достаточно близко к корню.
2. Скорость сходимости метода Ньютона:
   * Для корней одинарной кратности метод Ньютона сходится квадратично, что означает, что каждая итерация удваивает количество правильных цифр.
   * Для корней двойной кратности метод Ньютона сходится линейно, что означает, что каждая итерация добавляет одну правильную цифру.
3. Условия сходимости метода простых итераций:
   * Функция должна быть непрерывной в некоторой окрестности корня.
   * Производная функции должна быть ограничена в этой окрестности.
4. Для априорной оценки количества итераций, требуемых для нахождения корня с заданной точностью, можно использовать различные методы, такие как оценка скорости сходимости метода, аналитический анализ функции и применение теоретических результатов. Однако точное количество итераций заранее предсказать сложно, поэтому часто используется итеративный подход, в котором вычисления продолжаются до достижения заданной точности или выполнения других условий остановки.