Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

Лабораторная работа № 2, 3

Методы оптимизации

Вариант №9

Студент

Карагодина Ксения Андреевна

группа Р3215

Преподаватель

Селина Елена Георгиевна

Санкт - Петербург 2025 год

**Заданная функция**

**Заданные интервалы**

**Заданная точность**

**Метод деления отрезка пополам**

Задаются a, b и погрешность

* Берем две точки вблизи интервала [a, b]
* Сравниваем :

Если

* Повторяем шаги, пока длина отрезка не станет меньше

**Ручное решение**

Шаг №1: интервал

Шаг №2: интервал

Шаг №3: интервал

Шаг №4: интервал

Результат четырех итераций

Интервал

**Программное решение**

def f(x):  
 return (1/3) \* x\*\*3 - 5 \* x + x \* math.log(x)  
  
def bisectionMethod(a, b, e):  
 while (b - a) > 2 \* e:  
 x1 = (a + b - e) / 2  
 x2 = (a + b + e) / 2  
  
 y1 = f(x1)  
 y2 = f(x2)  
  
 if y1 > y2:  
 a = x1  
 else:  
 b = x2  
  
 xm = (a + b) / 2  
 return xm

def main():

result = bisectionMethod(a, b, e)  
print(f"Минимум функции находится в точке: {result}")  
print(f"Значение функции в этой точке: {f(result)}")

Результат работы программы

Минимум функции находится в точке: 1.8410767517089845

Значение функции в этой точке: -6.001532556320019

**Метод золотого сечения**

**Алгоритм:**

1. На первом шаге точки вычисляются по формулам:
2. Вычисляются значение функции в этих точках и возможно 2 случая:
3. Если На второй итерации полагаем равным , а вычисляем по формуле .

Значение функции вычисляется только в точке , так как значение функции в уже было вычислено на предыдущем шаге.

1. Если На второй итерации полагаем равным , а вычисляем по формуле .

Значение функции вычисляется только в точке , так как значение функции в уже было вычислено на предыдущем шаге.

1. Вычисления продолжают до тех пор, пока длина интервала не станет меньше требуемой точности.

**Ручное решение**

Шаг №1: интервал

Шаг №2: интервал

Шаг №3: интервал

Шаг №4: интервал

Интервал

**Программное решение**

def goldenSectionMethod(a, b, e):  
 golden\_ratio = 0.618  
 x1 = a + (1 - golden\_ratio) \* (b - a)  
 x2 = a + golden\_ratio \* (b - a)  
  
 f\_x1 = f(x1)  
 f\_x2 = f(x2)  
  
 while abs(b - a) > e:  
 if f\_x1 < f\_x2:  
 b = x2  
 x2 = x1  
 f\_x2 = f\_x1  
 x1 = a + (1 - golden\_ratio) \* (b - a)  
 f\_x1 = f(x1)  
 else:  
 a = x1  
 x1 = x2  
 f\_x1 = f\_x2  
 x2 = a + golden\_ratio \* (b - a)  
 f\_x2 = f(x2)  
  
 return (a + b) / 2

def main():

result = goldenSectionMethod(a, b, e)  
print(f"Минимум функции находится в точке: {result}")  
print(f"Значение функции в этой точке: {f(result)}")

Результат работы программы

Минимум функции находится в точке: 1.8411219763447724

Значение функции в этой точке: -6.00153255587944

**Метод хорд**

1. Находим , вычисляем
2. Проверка на окончание поиска:

Если и завершить поиск, иначе пункт 3

1. Переход к новому отрезку. Если , то положить b = , иначе положить a = . Перейти к шагу 1.

**Ручное решение**

Шаг №1: интервал

*(*

Шаг №2: интервал

*(*

*(*

Шаг №3: интервал

*(*

Интервал [1.5, ]

**Программное решение**

def chord\_method(a, b, epsilon):  
 # Проверяем, что f'(a) и f'(b) имеют разные знаки  
 if f\_derivative(a) \* f\_derivative(b) >= 0:  
 raise ValueError("Производная на концах интервала должна иметь разные знаки.")  
  
 while True:  
 # Вычисляем точку пересечения хорды с осью OX  
 x\_hat = a - (f\_derivative(a) \* (a - b)) / (f\_derivative(a) - f\_derivative(b))  
  
 # Вычисляем значение производной в новой точке  
 f\_derivative\_x\_hat = f\_derivative(x\_hat)  
  
 # Проверяем условие завершения  
 if abs(f\_derivative\_x\_hat) <= epsilon:  
 return x\_hat # Возвращаем найденную точку минимума  
  
 # Обновляем интервал  
 if f\_derivative\_x\_hat > 0:  
 b = x\_hat  
 else:  
 a = x\_hat

Результат работы программы

Точка минимума: 1.841086056900067

Значение функции в точке минимума: -6.001532556935503

**Метод Ньютона**

1. Высчитываем точку
2. На (k+1)-м шаге по найденной на предыдущем шаге точке
3. Вычисления производятся до тех пор, пока не выполнится неравенство

, после чего полагаем

**Ручное решение**

Пусть

*4.0714*

*= 1.8428*

Условие не выполняется, продолжаем итерации

*4.2282*

*= 1.8411*

Условие выполнено

**Программное решение**

def newton\_method(x0, epsilon):  
 x\_k = x0 # Начальное приближение  
 while True:  
 f\_prime = f\_derivative(x\_k) # Значение первой производной в точке x\_k  
 f\_double\_prime = f\_second\_derivative(x\_k) # Значение второй производной в точке x\_k  
  
 # Проверка на окончание итераций  
 if abs(f\_prime) <= epsilon:  
 return x\_k # Возвращаем найденную точку минимума  
  
 # Обновление приближения  
 x\_k = x\_k - f\_prime / f\_double\_prime

def main():

x0 = (a + b) / 2

x\_min = newton\_method(x0, e)  
print(f"Точка минимума: {x\_min}")  
print(f"Значение функции в точке минимума: {f(x\_min)}")

Результат работы программы

Точка минимума: 1.8410976526905583

Значение функции в точке минимума: -6.001532557190462

**Метод квадратной аппроксимации**

1. Задать начальную точку , величину шага по оси X, – малые положительные значения, характеризующие точность.
2. Вычислить вторую точку:
3. Вычислить значения функции в точках
4. Сравнить точки
5. Если , положить
6. Если
7. Вычислить
8. Найти
9. По точкам вычислить точку минимума квадратичного интерполяционного полинома:

И величину функции

(Если знаменатель в формуле для на некоторой итерации обращается в ноль, то результатом итерации является прямая. В этом случае обозначается и переход ко 2 шагу)

1. Условие окончания расчета:
2. Если оба условия выполняются, конец поиска:
3. Если хотя бы одно из условий не выполняется и , выбрать наименьшую точку и две точки по обе стороны от нее. Обозначить эти точки в обычном порядке и перейти к шагу 6
4. Если хотя бы одно из условий не выполняется и то положить точку и перейти к шагу 2.

**Ручное решение**

Шаг №1:

Минимум:

Проверим условия

Если =0.002, то < 0.002, условие не выполняется.

Если =0.02, то < 0.02, условие не выполняется.

не находится в интервале  =.

Шаг №2:

Минимум:

Проверим условия

Если =0.002, то  < 0.002, условие выполняется.

Если =0.02, то < 0.02, условие выполняется.

Так как условия выполнены, то

Конец поиска.

**Программное решение**

def quadratic\_approximation(x1, delta\_x, epsilon1=0.0001, epsilon2=0.0001, max\_iter=100):  
 x2 = x1 + delta\_x  
 f1, f2 = f(x1), f(x2)  
  
 if f1 > f2:  
 x3 = x1 + 2 \* delta\_x  
 else:  
 x3 = x1 - delta\_x  
  
 f3 = f(x3)  
  
 for \_ in range(max\_iter):  
 F\_min = min(f1, f2, f3)  
 x\_min = x1 if f1 == F\_min else (x2 if f2 == F\_min else x3)  
  
 denominator = (x2 - x3) \* f1 + (x3 - x1) \* f2 + (x1 - x2) \* f3  
 if denominator == 0:  
 x1 = x\_min  
 x2 = x1 + delta\_x  
 f1, f2 = f(x1), f(x2)  
 if f1 > f2:  
 x3 = x1 + 2 \* delta\_x  
 else:  
 x3 = x1 - delta\_x  
 f3 = f(x3)  
 continue  
  
 x\_bar = 0.5 \* ((x2\*\*2 - x3\*\*2) \* f1 + (x3\*\*2 - x1\*\*2) \* f2 + (x1\*\*2 - x2\*\*2) \* f3)  
 x\_bar /= denominator  
 f\_bar = f(x\_bar)  
  
 if abs((F\_min - f\_bar) / f\_bar) < epsilon1 and abs((x\_min - x\_bar) / x\_bar) < epsilon2:  
 return x\_bar  
  
 if x\_bar < x1 or x\_bar > x3:  
 x1 = x\_bar  
 x2 = x1 + delta\_x  
 f1, f2 = f(x1), f(x2)  
 if f1 > f2:  
 x3 = x1 + 2 \* delta\_x  
 else:  
 x3 = x1 - delta\_x  
 f3 = f(x3)  
 else:  
 if f\_bar < F\_min:  
 x\_min = x\_bar  
 F\_min = f\_bar  
  
 if x\_min < x2:  
 x3 = x2  
 x2 = x\_min  
 else:  
 x1 = x2  
 x2 = x\_min  
  
 f1, f2, f3 = f(x1), f(x2), f(x3)  
  
 return x\_bar

def main():

x1 = 1.5  
delta\_x = 0.1  
min\_x = quadratic\_approximation(x1, delta\_x)  
print(f"Точка минимума: {min\_x}")  
print(f"Значение функции в точке минимума: {f(min\_x)}")

Результат работы программы

Точка минимума: 1.8404246320739788

Значение функции в точке минимума: -6.001531602016654