МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Костромской государственный университет»

(КГУ)

ИАСТ

Кафедра автоматизированных систем и технологий

09.03.02

Направление подготовки/Специальность Информационные системы и технологии

Дисциплина Численные методы

# Лабораторная №3.

# Решение поиска минимума функции (Вариант 19).

Выполнил студент

Копосов Лев Владимирович

Группа 22-ИСбо-1б

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_

Кострома

**Постановка задачи.**

Найти минимум функции с помощью методов: второй способ дихотомии, золотого сечения, квадратичной интерполяции с точностью до EPS = 0.001. Сравнить точность и сходимость методов.

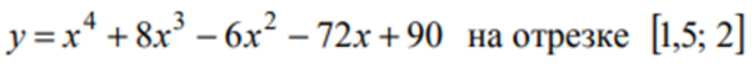
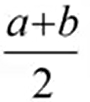
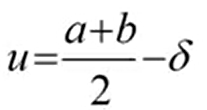
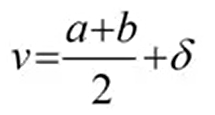
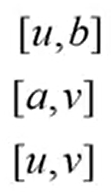
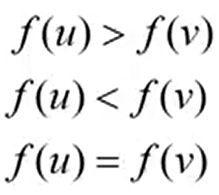
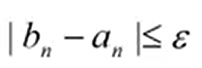
****

рис. 1. Задача N19

**Краткая теория используемых методов.**

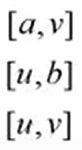
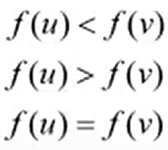
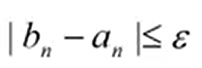
1. Второй способ дихотомии.

Метод второго способа дихотомии является численным методом оптимизации, который используется для поиска минимума функции на отрезке.

Основная идея метода заключается в том, что функция минимизируется путем последовательного уменьшения интервала, содержащего минимум функции. Метод второго способа дихотомии заключается в последовательном делении отрезка пополам , получая точки , , где - малое значение, и выборе той части , которая удовлетворяет условию . Потом выбираем начальное приближение  и продолжаем процесс итерации, пока не будет достигнута необходимая точность. Решение будет считаться приближенным, если выполняется условие , где bn - значение на оси OX в правой части промежутка, an - значение на оси OX в левой части промежутка, ε - заданная точность.

1. Золотое сечение.

Метод золотого сечения - это численный метод оптимизации, который используется для поиска минимума функции на отрезке.

Основная идея метода заключается в том, что он основан на принципе деления отрезка “золотым сечением”. Метод золотого сечения заключается в последовательном делении отрезка на константу (1.618), получая точки, и выборе той части , которая удовлетворяет условию . Потом выбираем начальное приближение  и продолжаем процесс итерации, пока не будет достигнута необходимая точность. Решение будет считаться приближенным, если выполняется условие , где bn - значение на оси OX в правой части промежутка, an - значение на оси OX в левой части промежутка, ε - заданная точность.

1. Квадратичная интерполяция.

Метод квадратичной интерполяции - это численный метод оптимизации, который используется для поиска минимума функции.

Основная идея метода заключается в том, что он основан на аппроксимации функции квадратичным полиномом в области около текущего приближения минимума. Метод квадратичной интерполяции заключается в выборе 3 точек на заданном интервале, вычислением значений функции в этих точках и проведением интерполяции в этих точках, “строим” параболу по этим точкам и находим минимум в ней . Новую полученную точку принимаем за опорную.

Потом выбираем начальное приближение  и продолжаем процесс итерации, пока не будет достигнута необходимая точность. Решение будет считаться приближенным, если выполняется условие , где A - значение на оси OX промежутка,  - минимум интерполяционного полинома, ε - заданная точность.

**Алгоритм решения поиска минимума функции.**

1. **Метод второго способа дихотомии.**

Шаг 1. Задаем исходную функцию, отрезок [a, b], малое значение h и точность вычисления ε;

Шаг 2. Присваиваем новое значение c = (a + b) / 2;

Шаг 3. Если |b - a| < ε, то выводим корень решения функции x = c, Конец.

Шаг 4. Вычисляем значения u = c + h и v = c - h;

Шаг 5. Если f(u) < f(v), то присваиваем новое значение a = u；

Иначе присваиваем новое значение b = v;

Шаг 6. Переходим к Шаг 2.

|  |
| --- |

рис. 2. Метод второго способа дихотомии

1. **Метод золотого сечения.**

Шаг 1. Задаем исходную функцию, отрезок [a, b], малое значение h и точность вычисления ε;

Шаг 2. Указываем константу GOLDEN\_RATIO = 1.618;

Шаг 3. Если |b - a| < ε, то выводим корень решения функции x = (a + b) / 2, Конец.

Шаг 4. Вычисляем значения u = a + (b - a) / GOLDEN\_RATIO, v = b - (b - a) / GOLDEN\_RATIO;

Шаг 5. Если f(u) < f(v), то присваиваем новое значение a = v；

Иначе присваиваем новое значение b = u;

Шаг 6. Переходим к Шаг 3.

|  |
| --- |

рис. 3. Метод золотого сечения

1. **Метод квадратичной интерполяции.**

Шаг 1. Задаем исходную функцию, отрезок [a, b], переменную A = a, малое значение h и точность вычисления ε;

Шаг 2. Вычисляем точку b = A + h, и значение функции f\_a и f\_b в точке A и b соответственно.

Шаг 3. Если f\_a < f\_b, то c = A - h;

Иначе c = A + 2 \* h;

Шаг 4. Вычисляем значение функции f\_c в точке c;

Шаг 5. Вычисляем точку t = find\_new\_point(), где параметры A, b, c соответственно и значение функции в этой точке f\_t;

Шаг 6. Если |f(A) - f\_t| < ε, то выводим корень решения функции x = min(a, b, t), Конец.

Шаг 7. Находим максимальное значение в функции f\_t, f(A), f(b), f\_c;

Шаг 8. Если есть значение в списке points, где points{A, b, c, t}

Иначе Переходим к Шаг 14;

Шаг 9. Если f(point) != max\_f, где point - точка из массива points, то Переходим к Шаг 14;

Шаг 10. Если |point - t| > h, то удаляем точку point из списка points, Переходим к Шаг n;

Шаг 11. Если есть значение в списке points, где points{A, b, c, t}

Иначе Переходим к Шаг 14;

Шаг 12. Вычисляем значение diff = |t - \_point|, где \_point - точка из массива points{A, b, c, t};

Шаг 13. Если diff > max\_diff, то присваиваем max\_diff = diff и max\_point = \_point;

14. Сортируем список точек points и присваиваем переменным A, b, c новые значения из списка points соответственно, Переходим к Шаг 5;

Алгоритм функции find\_new\_point().

Шаг 1. Вычисляем значения функции y1, y2, y3 в точках x1, x2, x3, где x1, x2, x3 - параметры функции find\_new\_point;

Шаг 2. Вычисляем значение delta = (x1 - x2) \* (x2 - x3) \* (x3 - x1);

Шаг 3. Вычисляем значения a = ((x3 - x2) \* y1 + (x1 - x3) \* y2 + (x2 - x1) \* y3) / delta, b = ((x2\*\*2 - x3\*\*2) \* y1 + (x3\*\*2 - x1\*\*2) \* y2 + (x1\*\*2 - x2\*\*2) \* y3) / delta, c = (x2 \* x3 \* (x3 - x2) \* y1 + x3 \* x1 \* (x1 - x3) \* y2 + x1 \* x2 \* (x2 - x1) \* y3) / delta;

Шаг 4. Если a != 0, то вычисляем x\_min = -b / (2 \* a);

Шаг 5. Выводим решения функции x = x\_min, Конец.

|  |
| --- |

рис. 4. Метод квадратичной интерполяции

**Вывод результата решения задачи.**

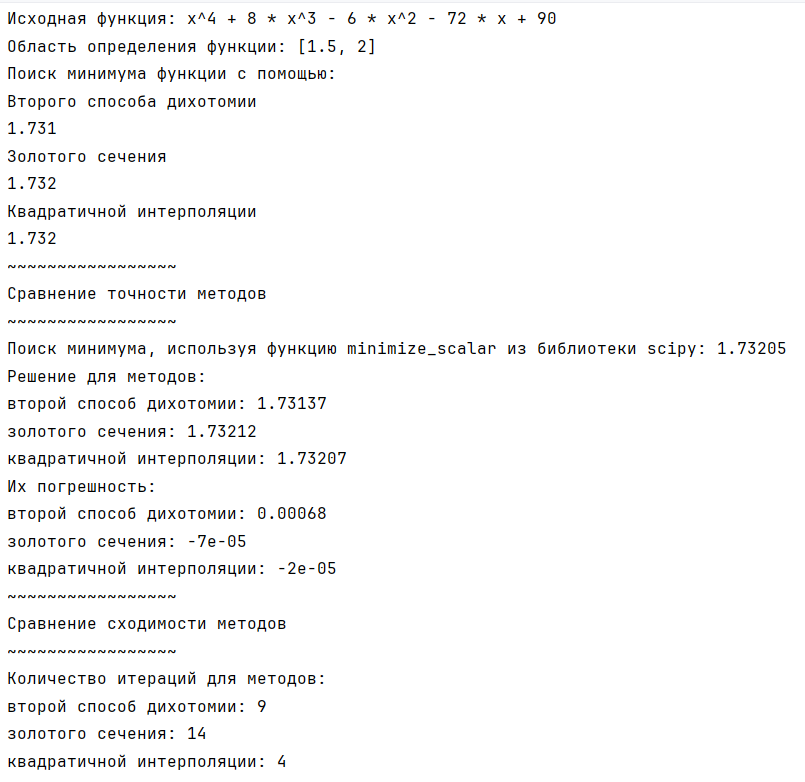
****

рис. 5. Вывод результата задачи

**Проверка правильности решения.**

Правильность решений была проверена с помощью unit тестов.

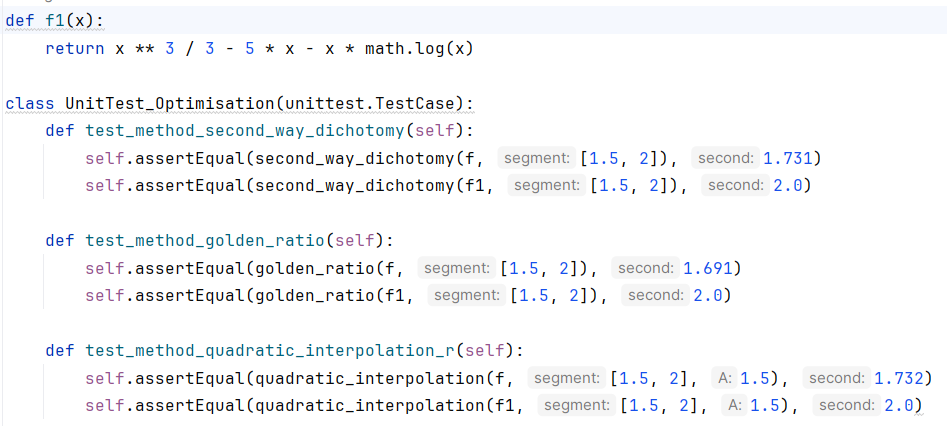


рис. 6. Проверка корректности программы с помощью unit тестов

**Выводы.**

Нашли минимум функции с помощью методов: второй способ дихотомии, золотого сечения, квадратичной интерполяции с точностью до EPS = 0.001. Сравнили точность и сходимость методов. Проверили решение с помощью unit тестов.

**Приложение: код программы.**

| **import math**  **import scipy.optimize**  **import scipy.optimize as sc**  **def f(x):**  ***"""Исходная функция"""***  **return x \*\* 4 + 8 \* x \*\* 3 - 6 \* x \*\* 2 - 72 \* x + 90**  **result\_arr = [float] \* 3**  **iterc\_arr = [int] \* 3**  **def second\_way\_dichotomy(func, segment, h = 1e-3,**  **accuracy = 1e-3, MAX\_ITER = 100):**  ***"""***  ***Поиск минимума функции с помощью***  ***метода второй способ дихотомии***  ***"""***  **c = None**  **a, b = segment**  **iteration\_count = 1**  **while iteration\_count < MAX\_ITER:**  **c = (a + b) / 2**  **if abs(b - a) < accuracy:**  **break**  **u = c + h**  **v = c - h**  **if func(u) < func(v):**  **a = u**  **else:**  **b = v**  **iteration\_count += 1**  **result\_arr[0] = c**  **iterc\_arr[0] = iteration\_count**  **return round(c, 3)**  **def golden\_ratio(func, segment,**  **accuracy = 1e-3, MAX\_ITER = 100):**  ***"""***  ***Поиск минимума функции с помощью***  ***метода золотого сечения***  ***"""***  **GOLDEN\_RATIO = 1.618**  **a, b = segment**  **x = None**  **iteration\_count = 1**  **while iteration\_count < MAX\_ITER:**  **if abs(b - a) < accuracy:**  **x = (a + b) / 2**  **break**  **u = a + (b - a) / GOLDEN\_RATIO**  **v = b - (b - a) / GOLDEN\_RATIO**  **if func(u) < func(v):**  **a = v**  **else:**  **b = u**  **iteration\_count += 1**  **result\_arr[1] = x**  **iterc\_arr[1] = iteration\_count**  **return round(x, 3)**  **def quadratic\_interpolation(func, segment, A,**  **h=1e-3, E=1e-3, MAX\_ITER=100):**  ***"""***  ***Поиск минимума функции с помощью***  ***метода квадратичной интерполяции***  ***"""***  **x = None**  **a, b = segment**  **t = None**  **f\_a = func(A)**  **b = A + h**  **f\_b = func(b)**  **if f\_a < f\_b:**  **c = A - h**  **else:**  **c = A + 2 \* h**  **f\_c = func(c)**  **iteration\_count = 1**  **while MAX\_ITER > iteration\_count:**  **t = find\_new\_point(func, A, b, c)**  **f\_t = func(t)**  **if abs(func(A) - f\_t) < E:**  **x = min(max(t, a), b)**  **break**  **else:**  **points = [A, b, c, t]**  **max\_f = max(f\_t, func(A), func(b), f\_c)**  **for point in points:**  **if func(point) == max\_f:**  **if abs(point - t) <= h:**  **max\_point = max([(\_point, abs(t - \_point)) for \_point in points], key=lambda x: x[1])**  **points.remove(max\_point[0])**  **else:**  **points.remove(point)**  **break**  **points = sorted(points)**  **A, b, c = points**  **iteration\_count += 1**  **result\_arr[2] = x**  **iterc\_arr[2] = iteration\_count**  **return round(x, 3)**  **def find\_new\_point(func, x1, x2, x3):**  **y1, y2, y3 = func(x1), func(x2), func(x3)**  **delta = (x1 - x2) \* (x2 - x3) \* (x3 - x1)**  **a = ((x3 - x2) \* y1 + (x1 - x3) \* y2 + (x2 - x1) \* y3) / delta**  **b = ((x2\*\*2 - x3\*\*2) \* y1 + (x3\*\*2 - x1\*\*2) \* y2 + (x1\*\*2 - x2\*\*2) \* y3) / delta**  **c = (x2 \* x3 \* (x3 - x2) \* y1 + x3 \* x1 \* (x1 - x3) \* y2 + x1 \* x2 \* (x2 - x1) \* y3) / delta**  **if a == 0:**  **raise ZeroDivisionError("Деление на ноль")**  **x\_min = -b / (2 \* a)**  **return x\_min**  **def check\_accuracy(res\_arr):**  ***"""Сравнение точности"""***  **min\_main = sc.minimize\_scalar(f, bounds=(1.5, 2), method="bounded")**  **correct\_result = round(min\_main.x, 5)**  **print(f"Поиск минимума, используя функцию minimize\_scalar "**  **f"из библиотеки scipy: {round(correct\_result, 5)}")**  **print(f"Решение для методов:\n"**  **f"второй способ дихотомии: {round(res\_arr[0], 5)}\n"**  **f"золотого сечения: {round(res\_arr[1], 5)}\n"**  **f"квадратичной интерполяции: {round(res\_arr[2], 5)}")**  **print(f"Их погрешность:\n"**  **f"второй способ дихотомии: {round(correct\_result - res\_arr[0], 5)}\n"**  **f"золотого сечения: {round(correct\_result - res\_arr[1], 5)}\n"**  **f"квадратичной интерполяции: {round(correct\_result - res\_arr[2], 5)}")**  **def check\_convergence(count\_iter):**  ***"""Сравнение сходимости"""***  **print(f"Количество итераций для методов:\n"**  **f"второй способ дихотомии: {count\_iter[0]}\n"**  **f"золотого сечения: {count\_iter[1]}\n"**  **f"квадратичной интерполяции: {count\_iter[2]}")**  **if \_\_name\_\_ == "\_\_main\_\_":**  **print("Исходная функция: x^4 + 8 \* x^3 - 6 \* x^2 - 72 \* x + 90")**  **a, b = [1.5, 2]**  **print(f"Область определения функции: [{a}, {b}]")**  **print("Поиск минимума функции с помощью:")**  **print("Второго способа дихотомии")**  **res1 = second\_way\_dichotomy(f, [a, b])**  **print(res1)**  **print("Золотого сечения")**  **res2 = golden\_ratio(f, [a, b])**  **print(res2)**  **print("Квадратичной интерполяции")**  **res3 = quadratic\_interpolation(f, [a, b], a)**  **print(res3)**  **print(f"~~~~~~~~~~~~~~~~~\n"**  **f"Сравнение точности методов\n"**  **f"~~~~~~~~~~~~~~~~~")**  **check\_accuracy(result\_arr)**  **print(f"~~~~~~~~~~~~~~~~~\n"**  **f"Сравнение сходимости методов\n"**  **f"~~~~~~~~~~~~~~~~~")**  **check\_convergence(iterc\_arr)** |
| --- |