Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

УНИВЕРСИТЕТ ИТМО **УЧЕБНЫЙ ЦЕНТР ОБЩЕЙ ФИЗИКИ ФТФ**

Группа: М32101

Студент: Косовец Роман / Дмитрий Надеждин

Преподаватель: Дорофеева Юлия Александровна

Рабочий протокол и отчет по лабораторной работе № 1

«Алгоритмы одномерной минимизации функции»

1. Цель работы:

Реализовать алгоритмы одномерной минимизации функции без производной, а именно:

- Метод дихотомии
- Метод золотого сечения
- Метод Фибоначчи
- Метод парабол (SPI)
- Метод Брента

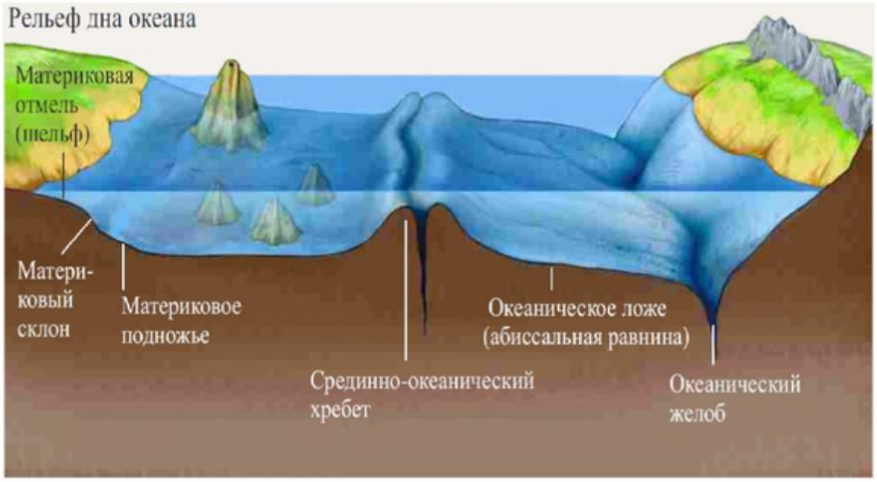
2. Задачи, решаемые при выполнении работы:

1) Сравнить методы по количеству итераций и вычислений функции в зависимости от разной точности
2) Проанализировать данные алгоритмы.

3. Условие задачи:

Вариант 2

Океаническое дно



Исследовательская экспедиция на батискафе решила исследовать флору и фауну океанического желоба вблизи западного бережья Южной Америки. Ранее при помощи акустического профилирования была получена картина океанического дна на пути следования батискафа. Профиль дна описывается следующей функцией на заданном промежутке (в относительных единицах):

$$y(x) = \sin(x)x^3$$

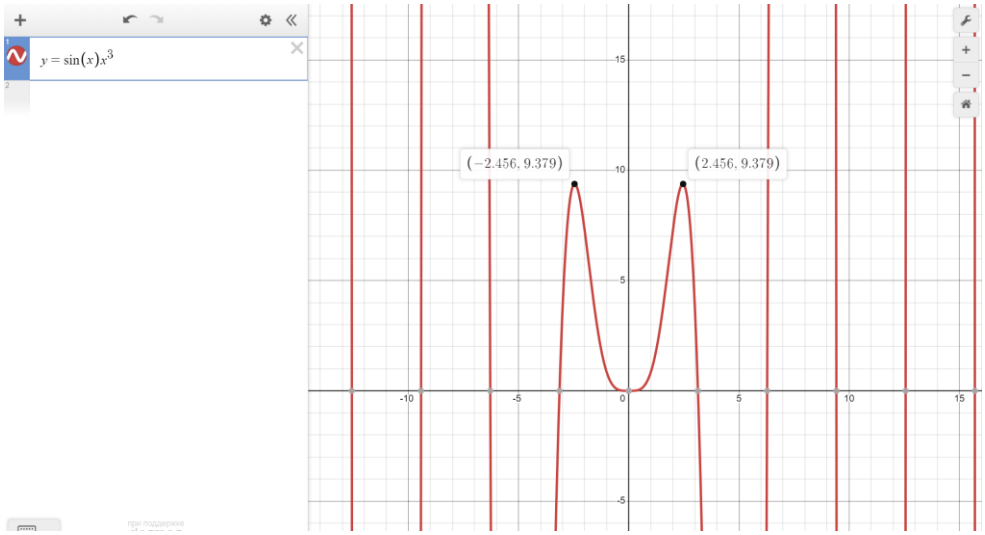
. Для построения точного курса исследователям необходимо извлечь из этих данных координату самой глубокой точки. Именно она будет являться океаническим желобом, который они так жаждут исследовать.

4. Решение задачи:

Для дальнейшего решения задачи и описания математических методов необходимо определить промежуток, на котором данная функция $y = \sin(x)x^3$ является унимодальной.

Это связано с особенностью метода дихотомии, который работает на основе разделения промежутка поиска на две равные части. Если функция на промежутке не является унимодальной (то есть имеет несколько минимумов), то разделение промежутка может привести к тому, что одна из частей промежутка будет содержать искомый минимум, а другая - нет. В этом случае некоторые методы могут не дать правильного результата.

Для выполнения данного этапа воспользуемся приложением Desmos, построим график и найдем нужный промежуток.

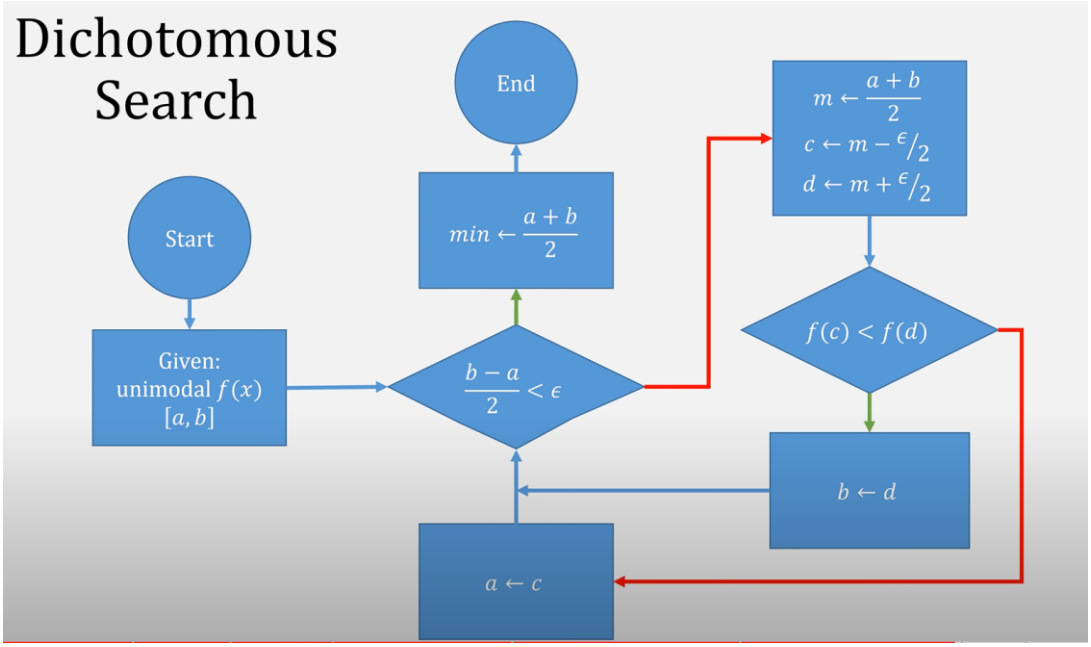


Как видно из графика, на данном промежутке уравнение имеет только один экстремум (минимум), поэтому функция $y = \sin(x)x^3$ унимодальная на промежутке [-2.45; 2.45].

Таким образом, промежуток функции будет равен: [-2.45; 2.45]

5. Метод Дихотомии:

Блок-схема алгоритма



Метод дихотомии — это метод численной оптимизации, который использует деление отрезка пополам для нахождения минимума функции на заданном отрезке.

Алгоритм метода дихотомии, следующий:

1. Задаем начальный отрезок [a, b], на котором ищем минимум функции.
2. Пока длина текущего отрезка больше заданной точности, выполняем следующие действия:
    - Выбираем середину отрезка mid = (a + b) / 2
    - Вычисляем ближайшие точки c и d путем прибавления к mid или отнимания заданной alpha
    - Вычисляем значения функции в точках: f(c), f(d)
    - Если f(c) <f(d), то минимум функции находится на отрезке [a, c], иначе минимум функции находится на отрезке [c, b]
    - Заменяем текущий отрезок на выбранный в предыдущем шаге отрезок и повторяем шаги до тех пор, пока длина отрезка не станет меньше заданной точности.

Алгоритм (Python)
https://github.com/RomanKosovets/PriMat/blob/main/Lab_1/1_Method_Dichotomy.py

Преимущества метода дихотомии:

1. Простота реализации. Метод дихотомии - один из самых простых методов поиска минимума функции
2. Гарантированная сходимость. Метод дихотомии гарантированно находит минимум функции на заданном промежутке

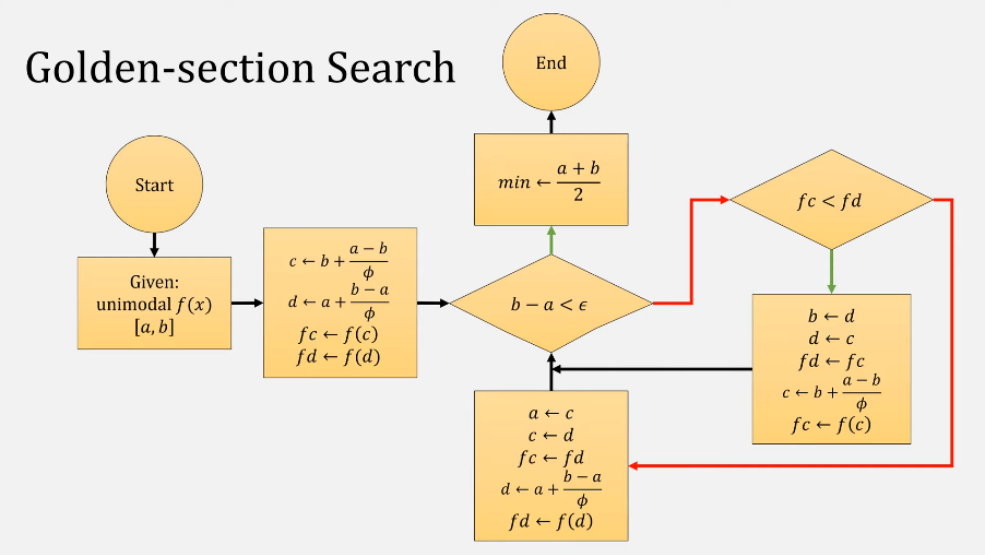
Недостатки метода дихотомии:

1. Низкая скорость сходимости. Метод дихотомии сходится медленно, особенно на промежутках большой длины
2. Зависимость от заданной точности. Для достижения заданной точности метод дихотомии может потребовать большое число итераций

   В целом, метод дихотомии — это простой и надежный метод поиска минимума функции на заданном промежутке, но его эффективность может сильно зависеть от свойств функции на этом промежутке. Если функция удовлетворяет условиям монотонности и гладкости, то метод дихотомии может быть эффективным инструментом для поиска минимума функции.

6. Метод золотого сечения:

Блок-схема алгоритма



Метод золотого сечения – это метод, основанный на принципе деления отрезка в пропорции золотого сечения, что позволяет уменьшать длину интервала постепенно и приближаться к точке минимума.

Алгоритм метода золотого сечения, следующий:

1. Задаем начальный интервал [a, b], точность epsilon и значение золотого сечения phi = (sqrt (5) - 1) / 2.
2. Вычисляем две точки на интервале: x1 = b - (b - a) * phi и x2 = a + (b - a) * phi.
3. Вычисляем значения функции в точках x1 и x2.
4. Если значение функции в точке x1 меньше, чем в x2, то новый интервал будет [a, x2], иначе [x1, b].
5. Повторяем шаги 2-4 до тех пор, пока длина интервала не станет меньше epsilon.

Алгоритм (Python)

https://github.com/RomanKosovets/PriMat/blob/main/Lab_1/2_Golden_Ratio.py

Преимущества метода золотого сечения:

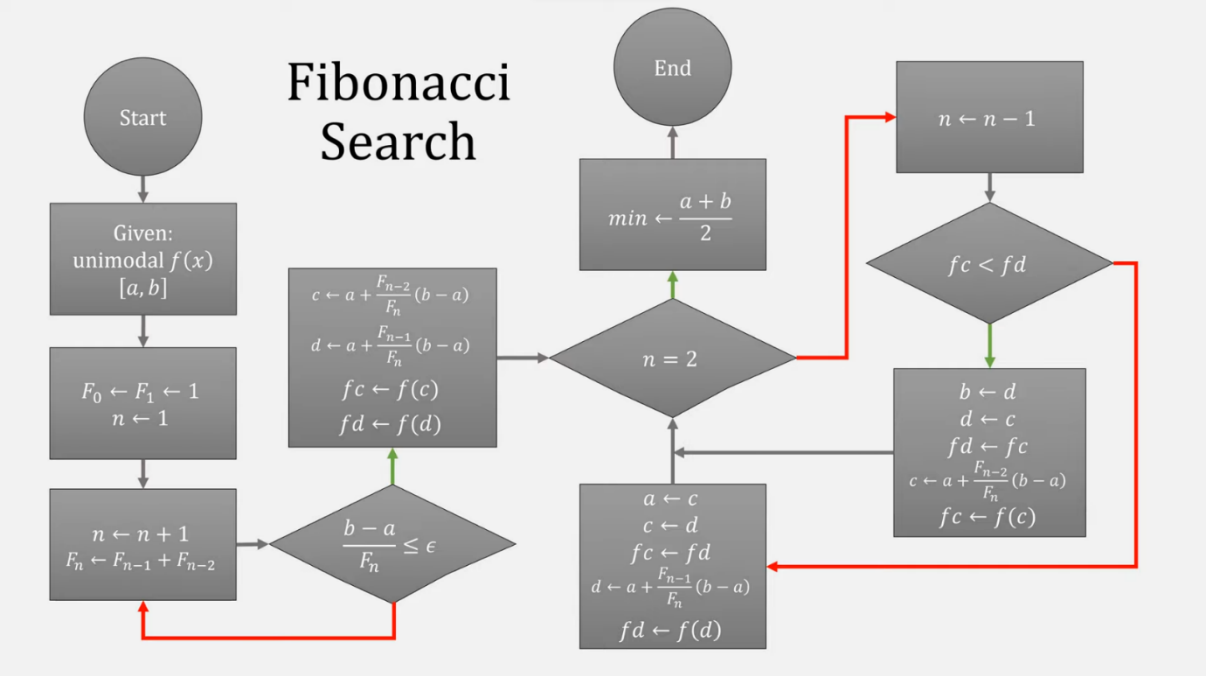
1. Высокая точность. Метод золотого сечения позволяет достигать высокой точности при нахождении минимума функции
2. Простота реализации. Алгоритм метода золотого сечения прост в реализации и не требует сложных вычислений
3. Гарантированная сходимость. Метод золотого сечения гарантирует нахождение минимума функции на заданном интервале

Недостатки метода золотого сечения:

1. Медленная сходимость – линейная скорость
2. Неэффективен при многомерной оптимизации. Метод золотого сечения неэффективен при многомерной оптимизации, так как требует разбиения каждого измерения на интервалы и выполнения алгоритма для каждого измерения

7. Метод Фибоначчи:

Блок-схема



Метод Фибоначчи – метод который использует последовательность чисел Фибоначчи для поиска оптимального значения функции на заданном интервале. Идея метода заключается в том, чтобы разбить исходный интервал на более мелкие интервалы с помощью золотого сечения, а затем выбрать новый интервал для поиска минимума, используя последовательность чисел Фибоначчи.

Алгоритм метода Фибоначчи, следующий:

1. Определяем числа Фибоначчи: первые два числа равны 0 и 1, а каждое последующее число равно сумме двух предыдущих чисел.
2. Задаем точность epsilon для минимизации функции и выбираем числа n так, чтобы F(n+1)> = (b - a) / epsilon, где F(n) – это n-ое число Фибоначчи.
3. Далее выбираем две точки на интервале:
   x1 = a + (b - a) * F (n - 1) / F (n + 1)
   x2 = a + (b - a) * F (n) / F (n + 1)
4. Вычисляем значения функций в этих точках:
   f1 = f (x1)
   f2 = f (x2)
5. Сравниваем значения функции в этих точках и выбираем новый интервал для поиска минимума:
   если f1 <f2, то новый интервал будет [a, x2]
   если f1> f2, то новый интервал будет [x1, b]

Алгоритм (Python)

https://github.com/RomanKosovets/PriMat/blob/main/Lab_1/3_Method_Fibonacci.py

Преимущества метода Фибоначчи:

1. Гарантируемая сходимость
2. Сравнительно прост в реализации и не требует большого количества вычислений
3. Заранее известно количество итераций

Недостатки метода Фибоначчи:

1. Линейная скорость сходимости
2. Требует подсчета чисел Фибоначчи и числа n
3. Может быть медленным для некоторых функций, особенно если выбрано большое число n

8. Метод парабол:

Метод парабол - этот метод основан на использовании параболы для аппроксимации функции в трех точках и нахождении минимума параболы.

Алгоритм метода Парабол, следующий:

1. Задаем начальный интервал [a, b], точность epsilon, а также выбираем три начальные точки
   x1 = a
   x2 = (a + b) / 2
   x3 = b
2. Находим значения функции f(x1), f(x2) и f(x3)
3. Находим точку минимума параболы u, проходящей через точки
   (x1, f(x1))
   (x2, f(x2))
   (x3, f(x3))
4. Если точка минимума параболы лежит внутри интервала [x1, x3] и ее расстояние до x2 меньше epsilon, то возвращаем эту точку как минимум функции.
5. Иначе выбираем новую точку u, ближайшую к x2 среди точек x1, x2 и x3, такую что она даёт меньшее значение функции f.
   - Если новая точка u лежит правее x2 и значение функции f(u) < f(x2), то новый интервал [x2, u] становится текущим интервалом, а точки x1, x2 и x3 обновляются.
   - Если новая точка u лежит левее x2 и значение функции f(u) < f(x2), то новый интервал [u, x2] становится текущим интервалом, а точки x1, x2 и x3 обновляются.
   - Если новая точка u не удовлетворяет условиям из пунктов 7 и 8, то новый интервал [x1, x3] становится текущим интервалом, а точки x1, x2 и x3 обновляются.
6. Повторяем шаги, пока не выполнится условие пункта 4

Преимущества метода Парабол:

1. Суперлинейная скорость сходимости. Однако она гарантируется только в малой окрестности точки минимума x_{min}
2. Находит экстремум функции

Недостатки метода Парабол:

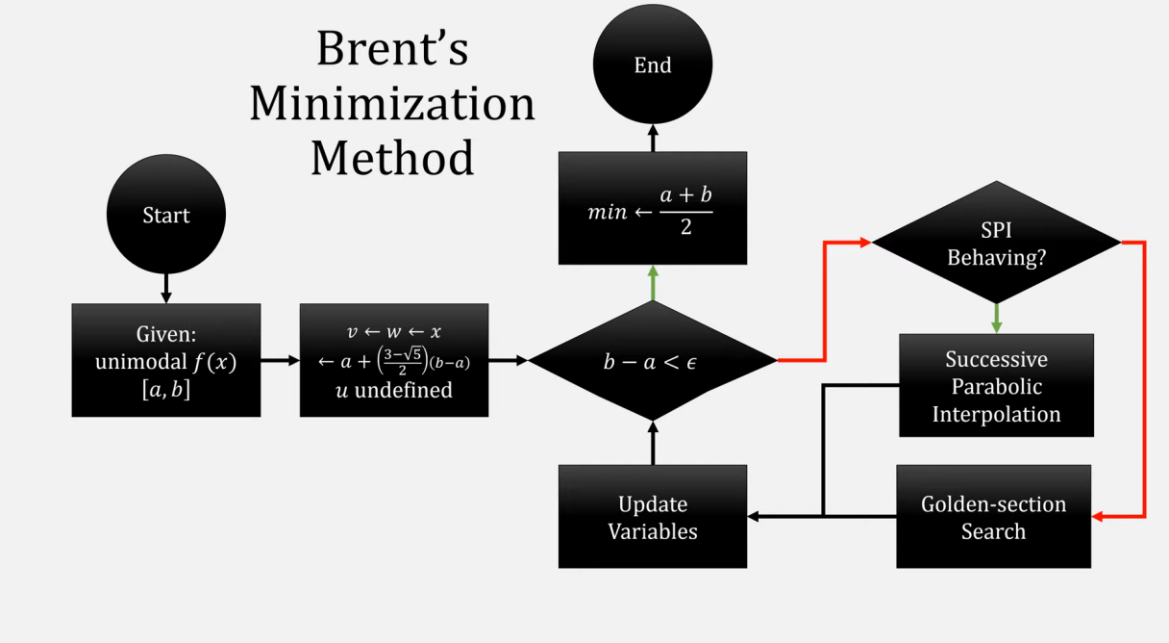
1. Нет гарантии сходимости. Если функция имеет несколько локальных минимумов, то итерационный метод может сойтись к любому из них.
2. Метод может работать неэффективно или дать неверный результат, если начальные точки выбраны неправильно или если функция имеет слишком сложную форму. Необходимо чтобы функция была достаточно гладкой и монотонной вблизи точки минимума.

Алгоритм (Python)

https://github.com/RomanKosovets/PriMat/blob/main/Lab_1/4_Method_Parabol.py

9. Комбинированный метод Брента:

Блок-схема



Алгоритм Брента - идея заключается в том, чтобы совместить метод золотого сечения и метод парабол, для получения скорости от метода парабол и надёжности от метода золотого сечения. Для этого в алгоритме предусмотрены условия, в зависимости от которых на текущей итерации будет выбран тот или иной алгоритм.

Алгоритм Брента, следующий:

1. Задаем интервал [a, b], точность результата epsilon и количество итераций steps, а также константу для метода золотого сечения
2. После устанавливаются начальные значения переменных

x – точка с наименьшим значением функции
w – точка со вторым наименьшим значением
v – предыдущее значение
u – последняя вычисленная точка

3. Запускаем цикл, вычисляем середину интервала xm и значение epsilon2
4. Выбирается точка u на интервале [a, b], которая является оптимальной точкой для следующей итерации и вычисляется значение функции в этой точке
5. Если точность достигнута, то алгоритм завершается и возвращается минимум функции и соответствующий аргумент минимума. В противном случае, переходим к следующей итерации, и используя значения x, w, v, u с помощью метода парабол вычисляем минимум

Алгоритм (Python)

https://github.com/RomanKosovets/PriMat/blob/main/Lab_1/5_Method_Brent.py

Преимущества алгоритма Брента:

1. Сочетание нескольких методов оптимизации позволяет достичь высокой скорости сходимости и точности результата.
2. Как правило суперлинейная скорость сходимости
3. Гарантированная сходимость

Недостатки алгоритма Брента:

1. Алгоритм требует больше вычислительных ресурсов, чем простые методы оптимизации, такие как метод золотого сечения.
2. Сложность имплементации (Однако можно использовать библиотеки, в которых уже предустановлен метод Брента)

10. Вывод:

Данные алгоритмы минимизации хорошо ведут себя на унимодальных функциях и справляются с задачей минимизации. Однако, для многомодальных функций они не подходят.

Каждый алгоритм обладает своими преимуществами и недостатками. Стоит отметить алгоритм Брента, который почти во всём превосходит остальные алгоритмы, потому что берёт лучшее из методов золотого сечения и парабол. Если стоит задача минимизировать унимодальную функцию без использования производных, то стоит рассмотреть использование именно этого алгоритма.

Название Метода	Кол-во итераций
Дихотомии	**27** $n \geq \frac{\ln((b_0 - a_0)/\varepsilon)}{\ln 2}$
Фибоначчи	**38** $\frac{b-a}{\epsilon} = F_n$
Золотого сечения	**38** $n \geq \frac{\ln \frac{b_1 - a_1}{\varepsilon}}{\ln \frac{\sqrt{5}-1}{2}}$
Парабол	**1**
Комбинированный метод Брента	**4**