Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Университет ИТМО»

Факультет информационных технологий и программирования

Лабораторная работа № 1

Выполнили студенты группы M3435:

Дорохов Ярослав Юрьевич

Бизяев Илья Владимирович

Пономарев Павел Николаевич

Санкт-Петербург

2021

Задание 1.

Были реализованы методы одномерного поиска: метод дихотомии, метод золотого сечения, метод Фибоначчи.

Метод дихотомии
Реализация алгоритма приведена по ссылке в функции `dichotomy_method`.
Алгоритму на вход подается функция f, отрезок [a, b], на котором необходимо найти минимум, желаемая точность eps, а также ограничение по числу итераций.
Точки x_1 и x_2, в которых находятся значения функции, вычисляются как
$(a_i + b_i) / 2 \pm delta$.
Длина отрезка $[a_{i+1}, b_{i+1}] \approx$ ½ длине отрезка $[a_i, b_i]$. Алгоритм заканчивается в тот момент, когда: либо число итераций превысило заданное ограничение, либо когда длина отрезка $[a_n, b_n]$ становится не больше чем $2 * eps$.
Алгоритм возвращает найденную точку минимума $(a_n + b_n) / 2$, количество совершенных итераций и количество вызовов переданной функции f.

Метод золотого сечения
Реализация алгоритма приведена по ссылке в функции `golden_section_method`.
Алгоритму на вход подается функция f, отрезок [a, b], на котором необходимо найти минимум, желаемая точность eps, а также ограничение по числу итераций.
Точки x_1 и x_2, в которых находятся значения функции, находятся симметрично относительно середины отрезка, делят его в отношении золотого сечения. Это позволяет переиспользовать ранее вычисленные значения функции, так как на следующей итераций одна из точек x_1, x_2 попадет в новый отрезок и уже его будет делить в отношении золотого сечения. За одну итерацию отрезок уменьшается в примерно 1.618 раз. Алгоритм заканчивается в тот момент, когда: либо число итераций превысило заданное ограничение, либо когда длина отрезка $[a_n, b_n]$ становится не больше чем $2 * eps$.
Алгоритм возвращает найденную точку минимума, количество совершенных итераций и количество вызовов переданной функции f.

Метод Фибоначчи
Реализация алгоритма приведена по ссылке в функции `fibonacci_method`.
Алгоритму на вход подается функция f, отрезок [a, b], на котором необходимо найти минимум, желаемая точность eps.
По заданной точности алгоритм определяет количество итераций n, которое ему необходимо совершить, а также считает необходимые числа Фибоначчи. Точки x_1,x_2 на итерации k вычисляются в соответствии с формулами: $a_k + F_{(n - k + 1)} / F_{(n - k + 3)} * (b_k - a_k)$ и $a_k + F_{(n - k + 2)} / F_{(n - k + 3)} * (b_k - a_k)$.
Длина отрезка $[a_k,b_k]$ вычисляется как $|[a,b]| * F_{n - k + 3} / F_{n + 2}$.
Алгоритм заканчивается в тот момент, когда: либо число итераций достигло

вычисленного n. Алгоритм возвращает найденную точку минимума $(a_n + b_n) / 2$, количество совершенных итераций и количество вызовов переданной функции f.

Тесты для реализованных методов приведены по данной ссылке.

Также было приведено сравнение методов по количество итераций и вызовов функции в зависимости от заданной точности. Все результаты можно посмотреть по ссылке.
Код, использующийся для сравнения доступен здесь.
В отчете приводим полученные результаты для следующего случая:

```
f(x) = (x - 1) * (x - 23) a=-300, b=400.
```

Сравнение количества итераций:

precision	dich_iters	gold_iters	fib_iters
0.1	13	19	18
0.01	17	24	22
0.001	20	28	27
0.0001	23	33	32
1E-05	27	38	37
1E-06	30	43	41
1E-07	33	48	46
1E-08	36	52	51

Сравнение количества вычислений функции:

precision	dich_calls	gold_calls	fib_calls
0.1	26	20	19
0.01	34	25	23
0.001	40	29	28
0.0001	46	34	33
1E-05	54	39	38
1E-06	60	44	42
1E-07	66	49	47
1E-08	72	53	52

Наименьшее **число итераций** совершает метод дихотомии. Количество итераций необходимое методам золотого сечения и Фибоначчи сопоставимо, однако несколько меньшее количество необходимо методу первому.
Наименьшее **число вызовов функции** совершает метод Фибоначчи. Сопоставимое с ним количество (но все же немного большее) совершает метод золотого сечения. Количество вызовов функции необходимое методу дихотомию больше примерно в 1.32 раза.

Задание 2.

Был реализован метод градиентного спуска, стратегии выбора шага, а также критерии останова.
Реализация метода градиентного спуска приведена по ссылке в функции

`gradient_descent`. Функция в качестве аргументов принимает:

- `f` — минимизируемая функция;
- `f_grad` — градиент минимизируемой функции;
- `start` — точка старта;
- `step_strategy` — стратегия выбора шага градиентного спуска;
- `stop_criteria` — критерий останова градиентного спуска;
- `eps_strategy` — точность для стратегии выбора шага (если необходима);
- `eps_criteria` — точность для критерия останова (если необходима);
- `max_iterations_strategy` — максимальное количество итераций в стратегии выбора шага (если необходимо);
- `max_iterations_criteria` — максимальное количество итераций в критерии останова (если необходимо).

Реализованные **стратегии выбора шага**:

- Постоянный шаг (`ConstantStepStrategy`)
- Дробление: условие Арно (`DivideStepStrategy`)
- Метод наискорейшего спуска
  - Дихотомия (`DichotomyStepStrategy`)
  - Золотое сечение (`GoldenSectionStepStrategy`)
  - Фибоначчи (`FibonacciStepStrategy`)

Реализацию данных стратегий можно посмотреть по ссылке.

Реализованные **критерии останова**:

- Сходимость по аргументу (`ArgumentStopCriteria`)
- Сходимость по функции (`FunctionStopCriteria`)
- Сходимость по градиенту (`GradStopCriteria`)
- Максимальное количество итераций (`MaxIterationCriteria`)

Реализацию данных критериев можно посмотреть по ссылке.

Тесты градиентного спуска с использованием различных критериев останова и стратегий выбора шага можно найти по ссылке.

Было проведено сравнение градиентных спусков, использующих различные реализации методов наискорейшего спуска. Результаты доступны по ссылке, реализацию сравнения можно посмотреть здесь.
Полученные результаты демонстрируют, что количество итераций практически не зависит от выбранной реализации. При этом, все методы наискорейшего спуска демонстрируют себя лучше, чем другие (постоянный шаг, дробление).

Задание 3.

Для анализа траекторий градиентного спуска были выбраны 3 квадратичные двумерные функции:

- $f(x, y) = x^2+y^2$
  - *число обусловленности: 1*
- $f(x, y) = 10x^2+y^2$

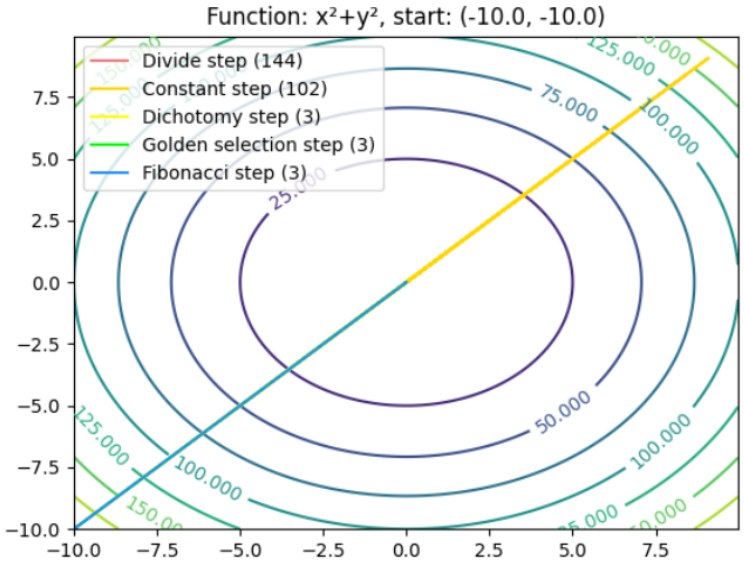
    - *число обусловленности: 10*
- $f(x, y) = x^2-xy+2y^2$
    - *число обусловленности: ≈2,78*

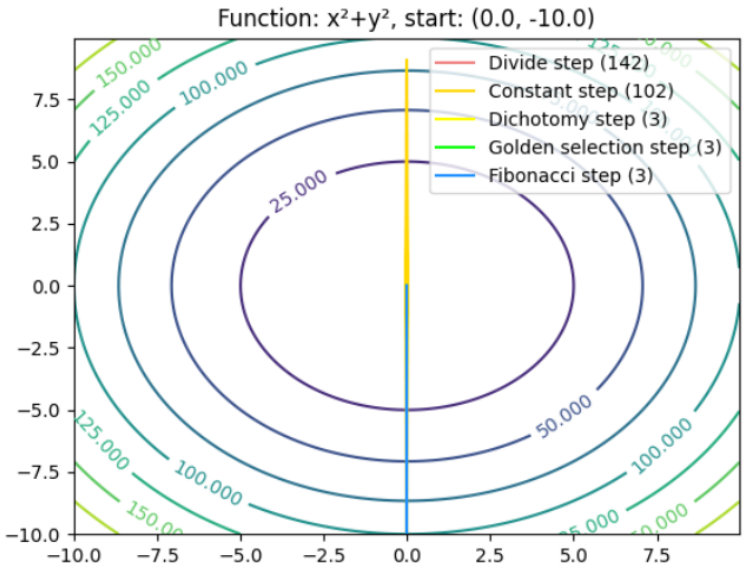
С использованием библиотеки Matplotlib были построены графики с линиями уровня функций и траекториями метода градиентного спуска. Для каждой функции рассматривались 3 начальных точки: левая нижняя, средняя нижняя и правая верхняя точки границ участка.

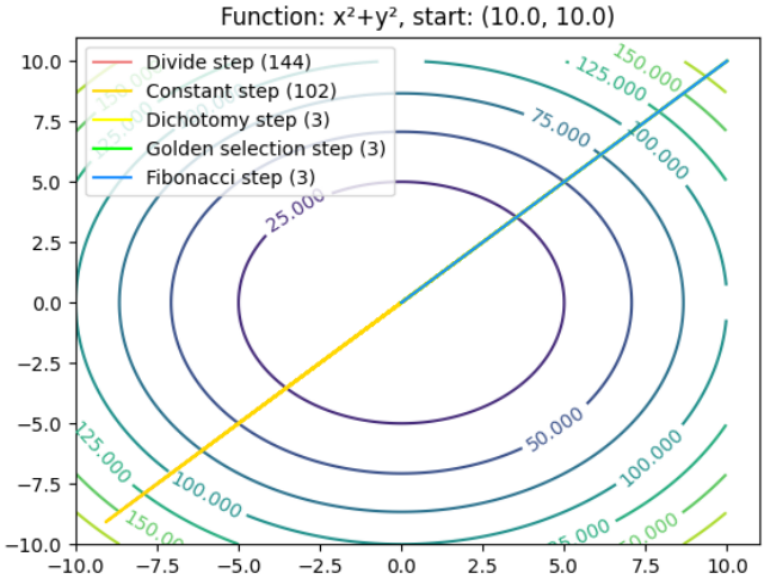
Реализация приведена по ссылке.

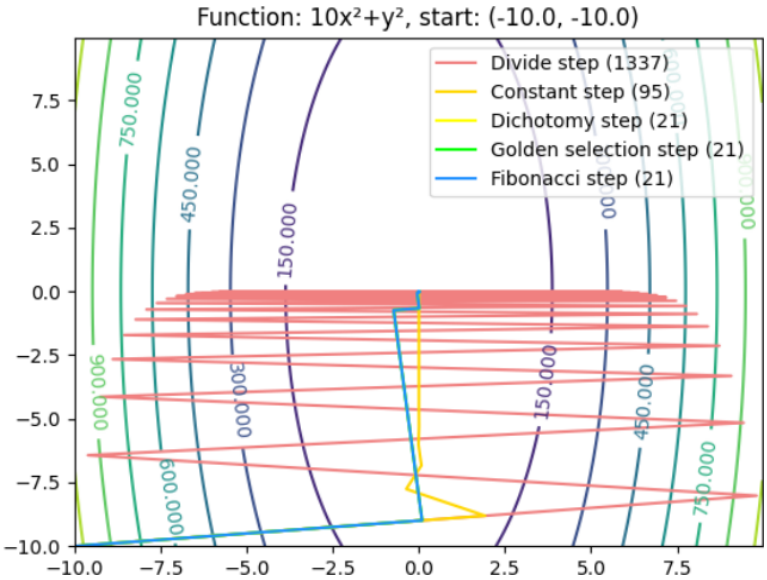
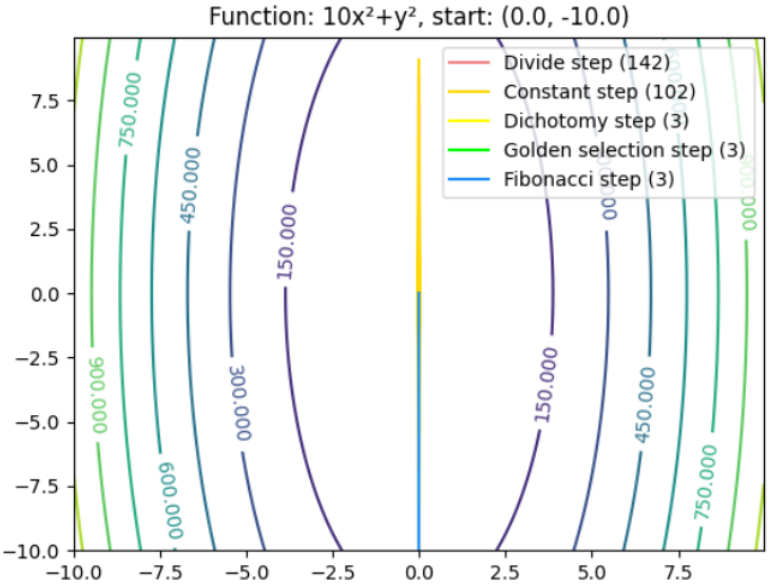
В легенде графиков указаны используемые стратегии выбора шагов и итоговое число шагов до выполнения условия остановки.

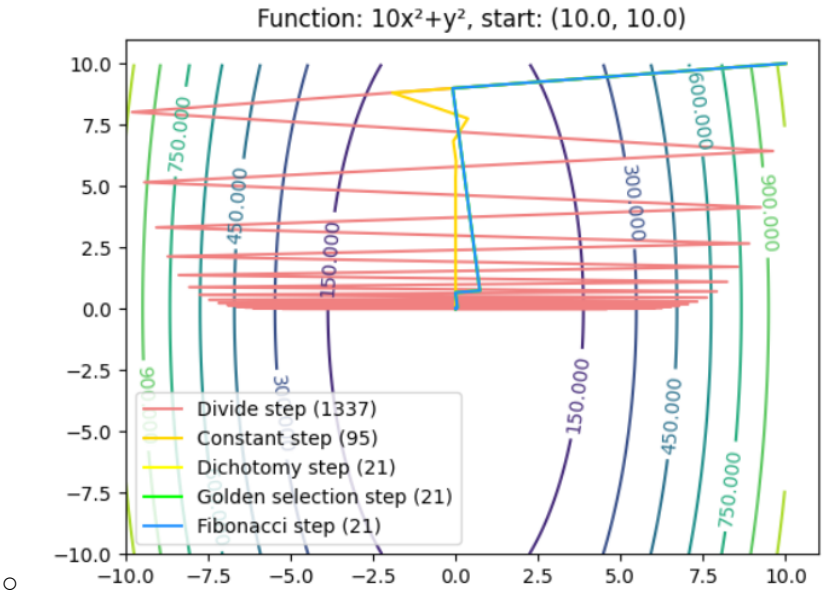
Результаты:

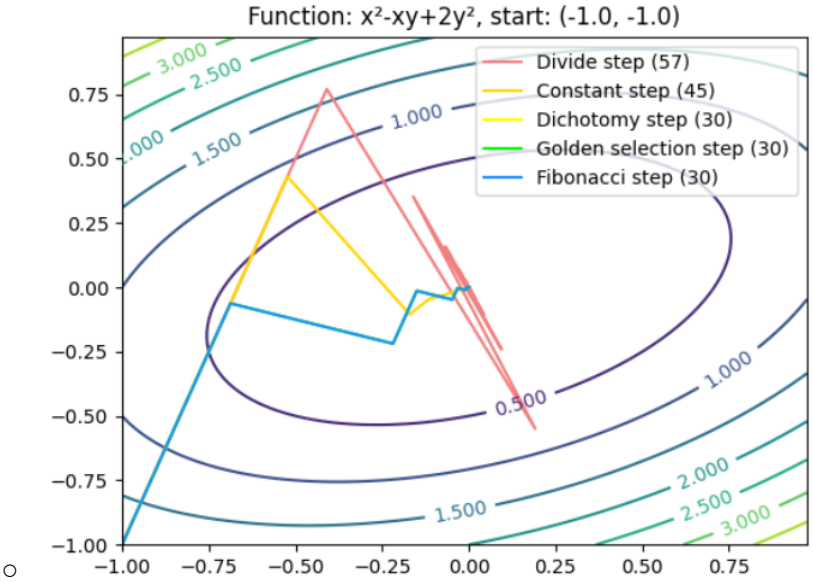
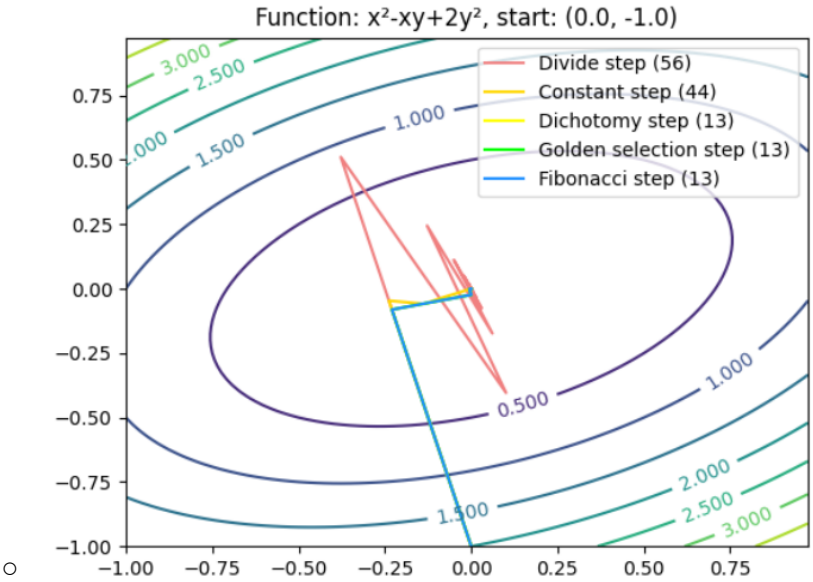
- **$f(x, y) = x^2+y^2$**
    - 

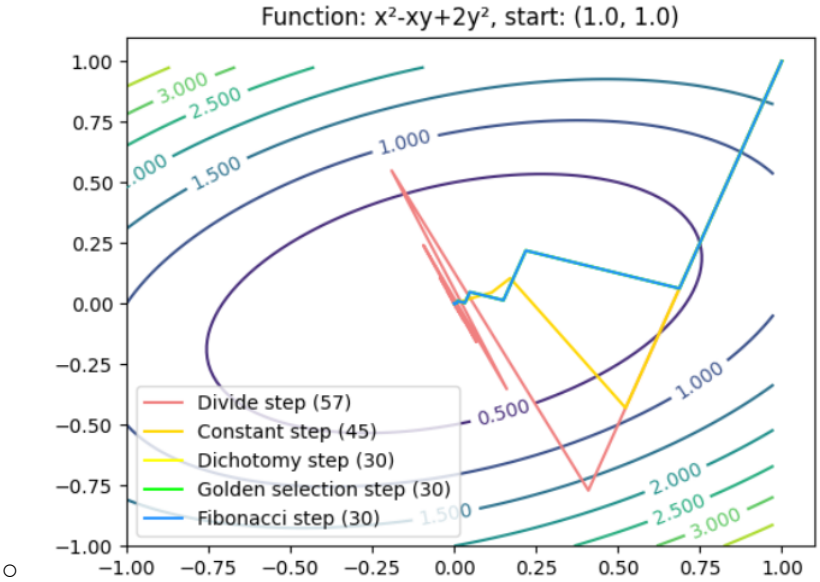
    - 

- 
Function: x²+y², start: (10.0, 10.0)
Divide step (144)
Constant step (102)
Dichotomy step (3)
Golden selection step (3)
Fibonacci step (3)

- **f(x, y) = 10x²+y²**
  - 
Function: 10x²+y², start: (-10.0, -10.0)
Divide step (1337)
Constant step (95)
Dichotomy step (21)
Golden selection step (21)
Fibonacci step (21)
  - 
Function: 10x²+y², start: (0.0, -10.0)
Divide step (142)
Constant step (102)
Dichotomy step (3)
Golden selection step (3)
Fibonacci step (3)

- 
Function: 10x²+y², start: (10.0, 10.0)
Divide step (1337)
Constant step (95)
Dichotomy step (21)
Golden selection step (21)
Fibonacci step (21)

- **f(x, y) = x²-xy+2y²**
  - 
Function: x²-xy+2y², start: (-1.0, -1.0)
Divide step (57)
Constant step (45)
Dichotomy step (30)
Golden selection step (30)
Fibonacci step (30)
  - 
Function: x²-xy+2y², start: (0.0, -1.0)
Divide step (56)
Constant step (44)
Dichotomy step (13)
Golden selection step (13)
Fibonacci step (13)



Вывод

На примере построенных графиков можно заметить, для функций с бóльшим числом обусловленности градиентный метод сходится медленнее. Если число обусловленности >> 1, то функции называют плохо обусловленными или овражными.

Наименьшее число шагов во всех случаях потребовалось стратегиям выбора шага, основанным на методах золотого сечения, дихотомии и Фибоначчи, наибольшее — дроблению.

Для симметричной первой функции выбор начальной точки не оказал значительного влияния на число шагов до остановки; в случае второй функции, для всех стратегий поиск завершился быстрее для начальной точки в овраге; для третьей, методы наискорейшего спуска получили результат быстрее при старте с более крутого склона.

Задание 4.

Было приведено исследование числа итераций необходимых градиентному спуску для сходимости, в зависимости от следующих параметров:

- число обусловленности k оптимизируемой функции;
- размерность пространство n оптимизируемых переменных.

Для этого была написана функция `generate_optimization_task`, которая по заданным n и k случайным образом в соответствии с аргументами генерирует квадратичную форму (функцию и её градиент).

Число обусловленности симметрической матрицы можно вычислить как $k = |\lambda_{max}| / |\lambda_{min}|$.

По спектральной теореме квадратная матрица A может быть разложена следующим образом: $A = VDV^{-1}$, где D — диагональная матрица с соответствующими собственными значениями на главной диагонали, а V — матрица, столбцы которой являются собственными векторами матрицы. В случае, если матрица A — вещественная симметрическая, то V — ортогональная. $V^{-1} = V^T$.

Таким образом по заданным n и k, мы можем сгенерировать матрицу D, а затем привести сгенерировать ортогональную матрицу V. Так мы сможем получить симметрическую матрицу M (гессиан квадратичной формы) которая будет задавать некоторую квадратичную форму с числом обусловленности k и размерностью пространства n. По матрице (гессиану) мы можем получить функцию и градиент. Реализацию данной логики можно посмотреть по ссылке.

Код, написанный для сравнения можно найти здесь, результаты доступны по ссылке.

Можно заметить, что количество итераций необходимых методу градиентного спуска, в целом увеличивается с ростом числа обусловленности.