Методы оптимизации

Национальный исследовательский университет ИТМО

Факультет информационных технологий и программирования

Лабораторная работа №1

Содержание

методы оптимизации

1. Основное задание

Реализуйте следующие методы:

1. метод градиентного спуска с различными стратегиями выбора шага (learning rate scheduling);
2. любой метод одномерного поиска и градиентный спуск на его основе;
3. изучите возможности библиотеки scipy.optimize. Найдите и используйте для сравнения библиотечные аналоги реализованных Вами методов.

2. метод градиентного спуска с различными стратегиями выбора шага

2.1 Алгоритмы градиентного спуска

В данной лабораторной мы будем реализовывать 2 градиентный спуска

1. С постояным шагом
2. С адаптивным шагом

2.2 Общая схема алгоритма

Для начала определим абстрактный алгоритм градиентного спуска.

Algorithm 1 Псевдо градиентный спуск

Ensure: $f : \mathbb{R}^2 \to \mathbb{R},\ x_0 \in \mathbb{R}^2$
  $y \leftarrow 1$
  $X \leftarrow x$
  $x \leftarrow x_0$
  for $iter \leftarrow 1$ **to** N **do**
    $g \leftarrow grad(f, x)$
    $step \leftarrow next\_step(g)$
    $x_{next} \leftarrow x - learning\_rate \cdot step$
    if $||x_{next} - x|| \leq \varepsilon$ **then** $break$
    end if
    $x \leftarrow x\_next$
  end for

где $grad$ может быть либо $\nabla f(x)$ либо $\dfrac{f(x+\delta) - f(x)}{\delta}$, с некоторой выбранной δ
Соответсвенно нам нужно реализовать $next\_step$ функцию

2.3 Постоянный шаг

Algorithm 2 Постоянный шаг

  Procedure $\text{NEXT\_STEP}(g, \alpha)$
Ensure: g **- Градиент**
Ensure: α **- Постоянный шаг**
  $step \leftarrow g \cdot \alpha$

2.4 Адаптивный шаг

Формула, которая используется в данном алгоритме

$$x_{k+1} = x_k - \frac{A}{\sqrt{G_k + \varepsilon}} \cdot g_k$$

Algorithm 3 Адаптивный шаг

Procedure NEXT_STEP(g, G)
Ensure: G **- Сумма квадратов граидентов**
Ensure: g **- Градиент**
$G \leftarrow G + g^2$
$step \leftarrow \frac{A}{\sqrt{G+\varepsilon}} \cdot g$

3. метод одномерного поиска и градиентный спуск на его основе

Стратегия заключается в преобразовании многомерной задачи оптимизации в последовательность одномерных задач вдоль направления антиградиента.

3.1 Общая схема алгоритма

На каждой итерации

1. Вычисляем направление антиградиента
2. Ищем оптимальны α в этом направление
3. Обновляем параметрый $x_{k+1} = x_k - \alpha \cdot \boldsymbol{\nabla} f(x_k)$

Algorithm 4 Одномерный поиск градиентный спуск

Ensure: f **- функция**
Ensure: x_0 **- начальная точка**
$x \leftarrow x_0$
for $iter$ **in** $range(1, maxIter)$ **do**
  $g \leftarrow grad(f,x)$
  $fd \leftarrow \lambda t.f(x - t \cdot g)$
  $\alpha \leftarrow search(fd,0,1,\varepsilon)$
  $x_{next} \leftarrow x - g \cdot \alpha$
  if $||x_{next} - x|| \leq \varepsilon$ **then**
    break
  end if
  $x \leftarrow x_{next}$
end for

Причем в данном алгоритме все, что нам нужно определить каким образом работает $search$.

3.2 Поиск методом золотого сечения

- Эффективный метод поиска минимума унимодальных функций
- На каждой итерации интервал сокращается в $1/\varphi$ раз
- Гарантированная линейная скорость сходимости
- Требует только вычисления значений функции

Algorithm 5 Метод поиска золотого сечения

Ensure: $a \in \mathbb{R} -$ **Левая граница поиска**
Ensure: $b \in \mathbb{R} -$ **Правая граница поиска**
Ensure: $f -$ **функция, у которой мы ищем**
Ensure: ε **- погрешность поиска**

$gr \leftarrow \frac{1+\sqrt{5}}{2}$
$c \leftarrow b - \frac{b-a}{gr}$
$d \leftarrow a + \frac{b-a}{gr}$
while do $|c-d| \leq \varepsilon$
  if $f(c) \leq f(d)$ **then**
    $b \leftarrow d$
  else
    $a \leftarrow c$
  end if
  $c \leftarrow b - \frac{b-a}{gr}$
  $d \leftarrow a + \frac{b-a}{gr}$
end while
return $\frac{b+a}{2}$

3.3 Методом Дихотомии

- Разделяет текущий интервал пополам на каждой итерации
- Сравнивает значения функции в двух близких точках около центра
- Прост в реализации, но менее эффективен чем золотое сечение
- Требует вдвое больше вычислений функции на итерацию
- Гарантирует сокращение интервала в 2 раза за каждые две итерации

Algorithm 6 Метод поиска дихотомии

Ensure: $a \in \mathbb{R} -$ **Левая граница поиска**
Ensure: $b \in \mathbb{R} -$ **Правая граница поиска**
Ensure: $f -$ **функция, у которой мы ищем**
Ensure: ε **- погрешность поиска**

```
while b − a ≤ ε do
    c ← (b + a)/2
    if f(c − ε) ≤ f(c + ε) then
        b ← c
    else
        a ← c
    end if
end while
return (b + a)/2
```

4. Выбор функция для тестирования

Чтобы протестировать наши алгоритмы неплохо выбрать специфичный квадратичные функции.

4.1 Функция Розенборка

Определение 1 (Функция Розенборка)**.** Функцией розенборка называют $f(x,y) = (1 - x^2)^2 + 100(y - x^2)^2$

У этой функции есть очень хорошие свойства, которые помогут нам исследовать градиентные спуски.

1. Долинная структура (Функция имеет длинное изогнутое плоское дно («долину»), ведущее к минимуму.)
   Это поможет изучить сходимость вдоль долины.
2. Сильная невыпуклость (пораболический овраг внутри долины)
   Направление вдоль долины резко меняется, за счет чего мы проверим устойчивость алгоритма.
3. Плохая обусловность гесиана (высокая чувствительность к шагу).

4.2 Функция Растринга

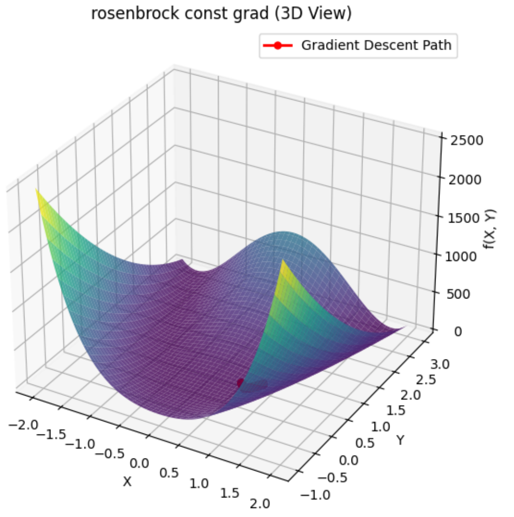
Определение 2 (Функция Растринга)**.** Функцией Растринга называют $f(x,y) = 20 + (x^2 - 10\cos(2\pi x)) + (y^2 - 10\cos(2\pi y))$

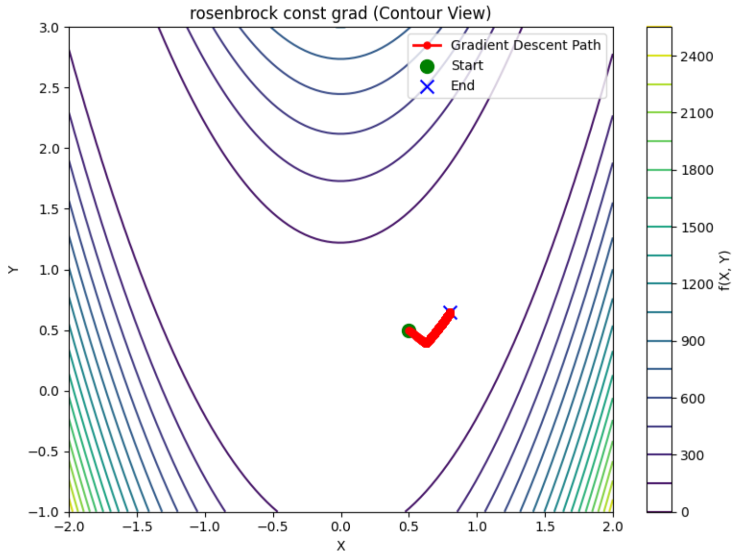
1. Множество локальный минимумовю
   градиентный спуск легко застревает в ближайшем локальном минимуме.
2. Высокочастотные осцилляции
   градиент быстро меняет направление, что вызывает осцилляции.
3. Глобальная структура
   Глобальный минимум расположен в начале координат и окружён "холмами"локальных минимумов.

5. Визуализация и тестирование

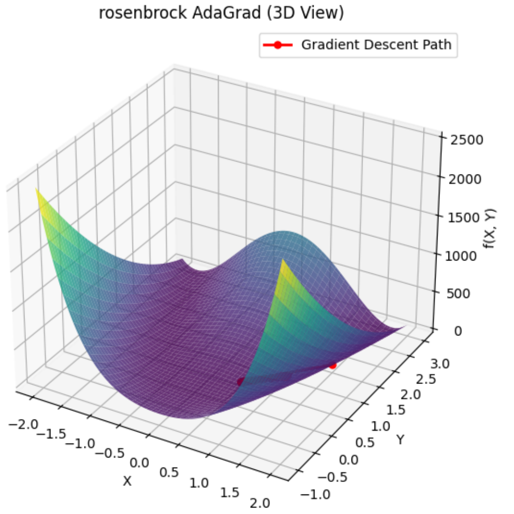
5.1 Визуализация для функции розенброка

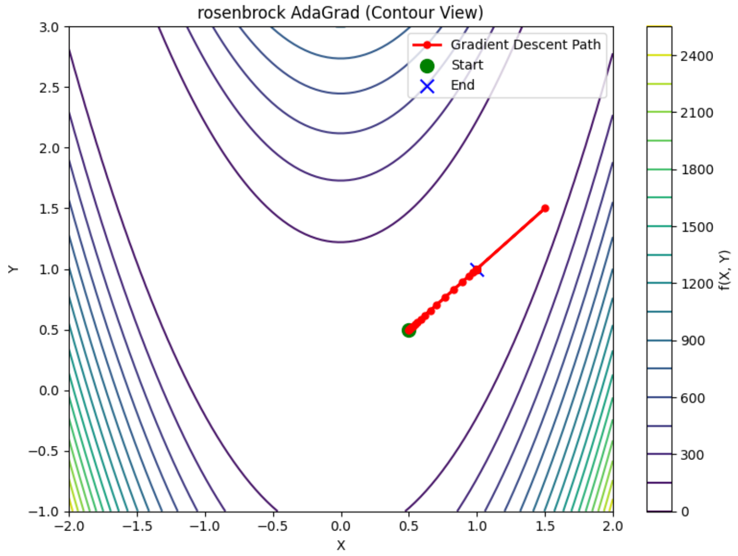
Наша главная гипотеза состоит в том, что для этой функции сходимость методом одномерного поиска будет быстрее.



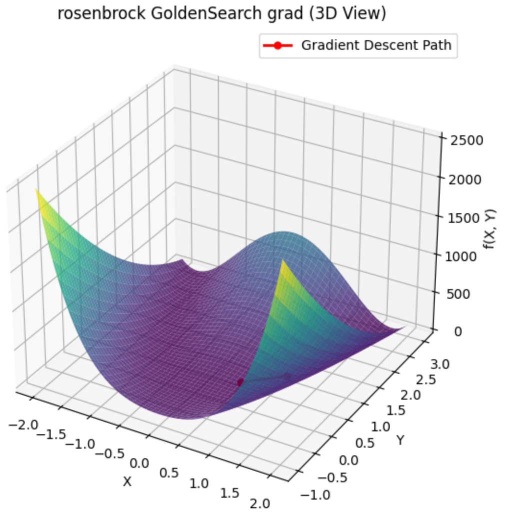


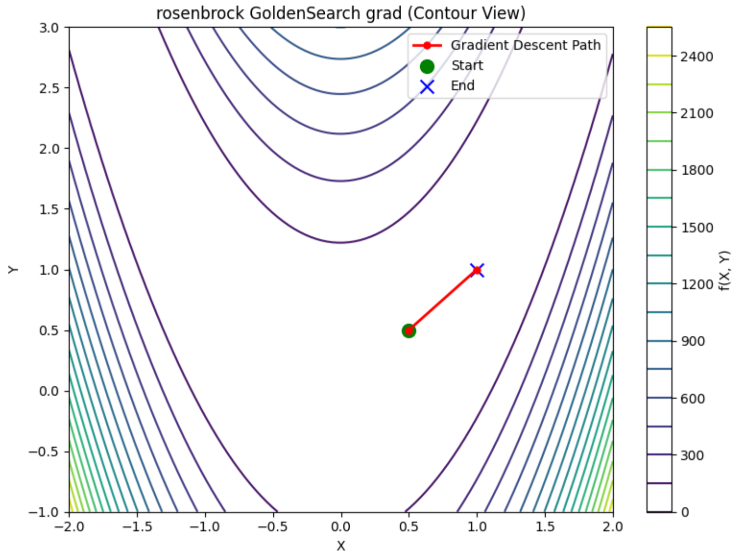
Как видно мы даже не дошли до экстремума в виду маленьких постояных шагов.





В адаптивной реализаци мы уже потратили меньше шагов, а также дошли до экстремума



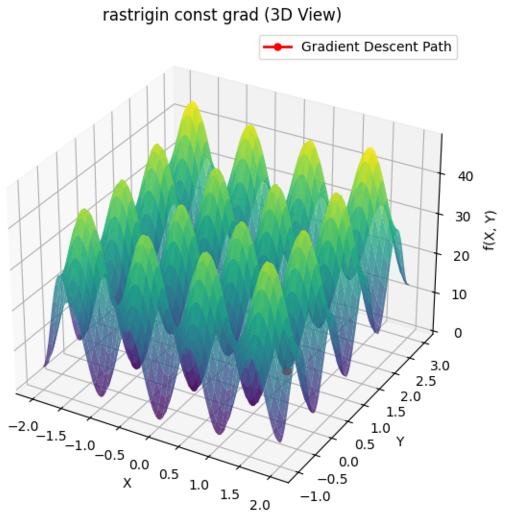


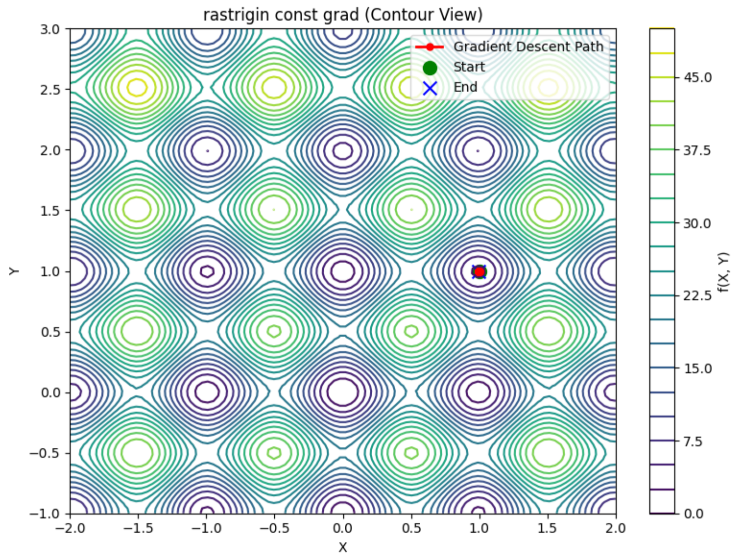
Как и предпологалось одномерным поиском мы потратили меньше шагов.

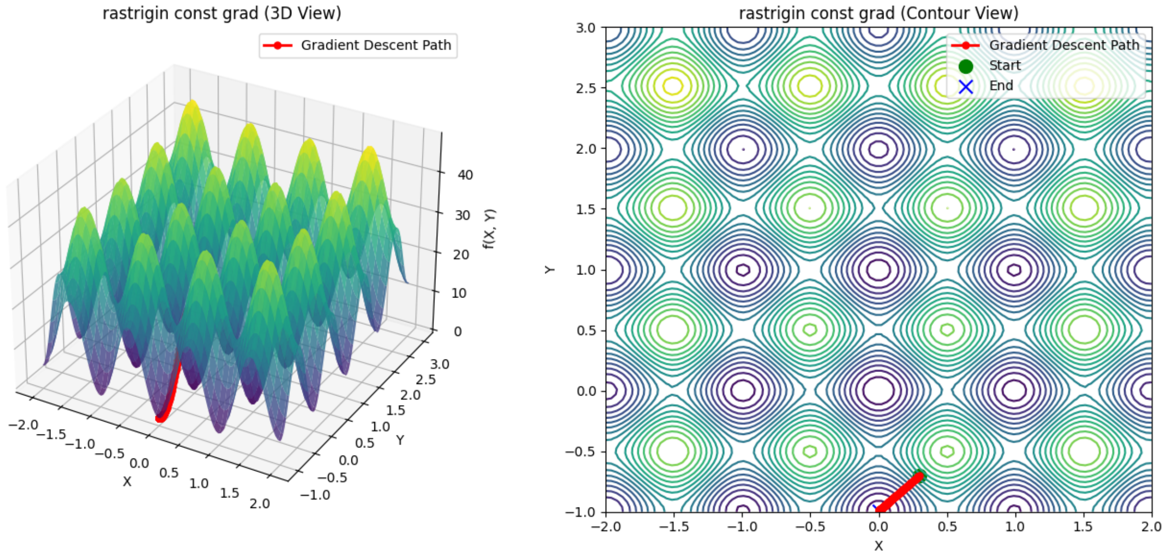
5.2 Визуализация для функции растригина

Главная гипотеза для этой функции - неустойчивость одномерного поиска.

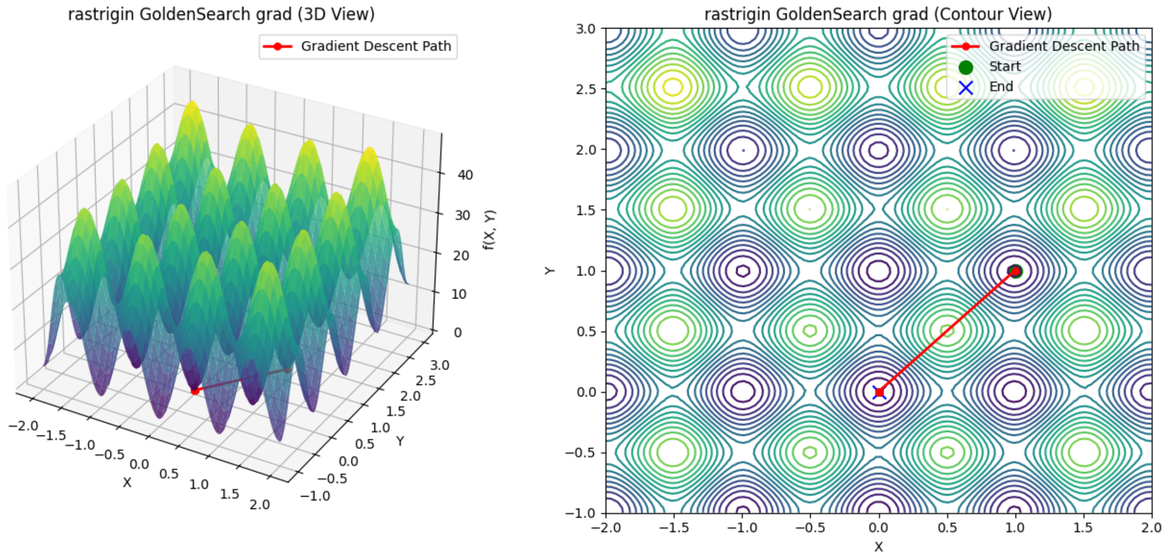
давайте начнем в локальной минимуме.

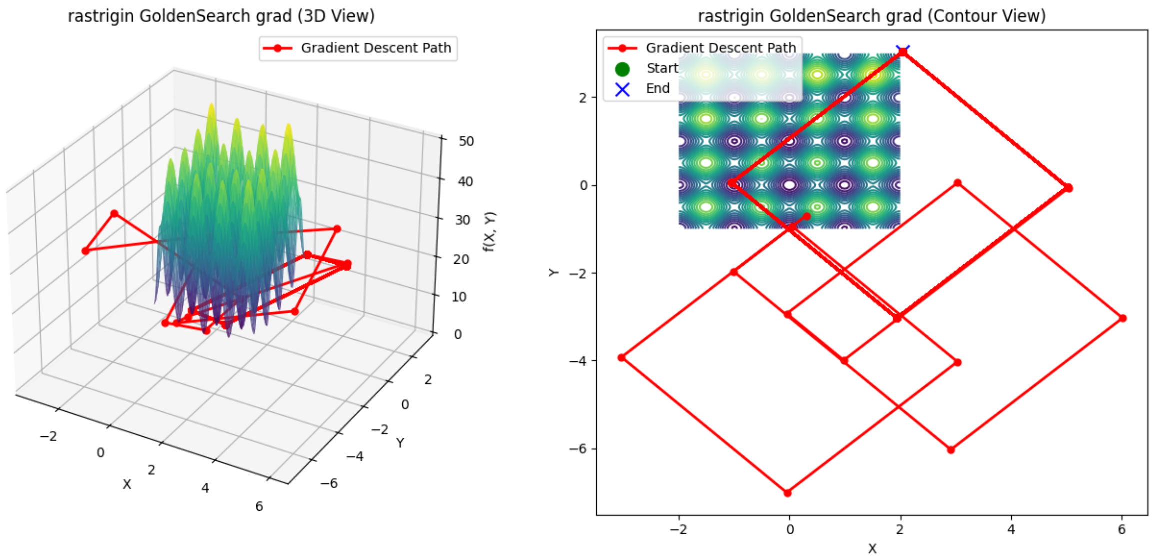






Постоянный шаг не покидает локальный минимум, а также достигает ближайщий локальный минмум.





Как видно поиск неусточивый ввиду покидание локального минимума, а также ей тяжело найти минимум.