Zero order methods. Line search. Gradient Descent. Accelerated Gradient Descent

Daniil Merkulov

Applied Math for Data Science. Sberuniversity.

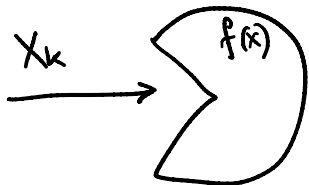$$f(x) : \mathbb{R} \to \mathbb{R}$$

$$\min_{x \in [a,b]} f(x)$$

Optimization without gradients
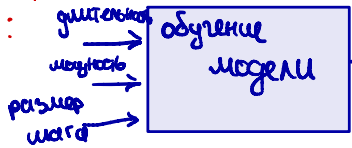
The curse of dimensionality for zero-order methods

$$\min_{x \in \mathbb{R}^n} f(x)$$

методы нулевого порядка

- метод имитации отжига
- эволюционный алгоритм

$f(x_k)$ нулевой порядок

X_k

$f(x)$

$f(x_k)$

$f(x_k), \nabla f(x_k)$

$f(x_k), \nabla f(x_k), \nabla^2 f(x_k)$

подбор гиперпараметров ML-моделей:

длительность
мощность
размер шага

обучение модели

test accuracy

$acc(\alpha, N, epochs) \to \max$

$\dfrac{\partial acc}{\partial \alpha}$

α
N
epochs

The curse of dimensionality for zero-order methods

проклятие размерности

чем больше размерность, тем стремительнее хуже

$$\min_{x \in \mathbb{R}^n} f(x)$$

GD: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$ Zero order GD: $x_{k+1} = x_k - \alpha_k G,$

where G is a 2-point or multi-point estimator of the gradient.

пусть есть

$d = 100$ мерная задача

Кол-во итераций
N_{GD} (400)

Кол-во итераций
N_{ZO} (1000)

$d = 10^6$ N_{GD} (400) $10^4 \cdot N_{ZO}$ 10^7

The curse of dimensionality for zero-order methods

$$\min_{x \in \mathbb{R}^n} f(x)$$

GD: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$ Zero order GD: $x_{k+1} = x_k - \alpha_k G,$

where G is a 2-point or multi-point estimator of the gradient.

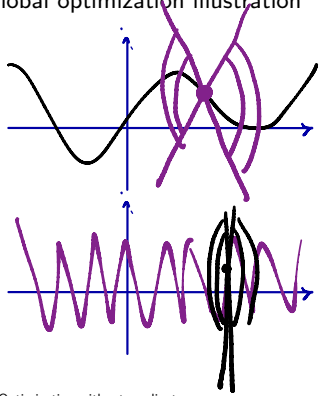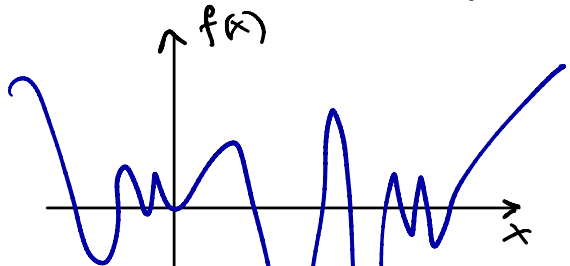|	$f(x)$ - smooth	$f(x)$ - smooth and convex	$f(x)$ - smooth and strongly convex
GD	$\|\nabla f(x_k)\|^2 \approx \mathcal{O}\left(\frac{1}{k}\right)$	$f(x_k) - f^* \approx \mathcal{O}\left(\frac{1}{k}\right)$	$\|x_k - x^*\|^2 \approx \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$
Zero order GD	$\|\nabla f(x_k)\|^2 \approx \mathcal{O}\left(\frac{n}{k}\right)$	$f(x_k) - f^* \approx \mathcal{O}\left(\frac{n}{k}\right)$	$\|x_k - x^*\|^2 \approx \mathcal{O}\left(\left(1 - \frac{\mu}{nL}\right)^k\right)$

Global optimization

$$\left| f(x) - f(y) \right| \le L \cdot \|x - y\|$$

$$f : \mathbb{R}^n \to \mathbb{R}$$

- 🐍Global optimization illustration

глобальная оптимизация

L_3

константа Липшица
показывает
насколько быстро может изменяться функция

$L_2 > L_1$

L_2

Global optimization

- 🐍 Global optimization illustration
- 🐍 Nevergrad illustration

метод иммитации отжига

$$\min f(x)$$

1. Генерируем случ. X

если $f(X) < f(X_{k-1})$, то

$$X_k = X$$

если $f(X) \geq f(X_{k-1})$

$P(T)$ ↓↓ $1-P(T)$ ↑↑

принимаем   отвергаем

$X_k = X$ генерируем нов. X

Global optimization

- 🔗 Global optimization illustration
- 🔗 Nevergrad illustration
- 🔗 Introduction to Optuna

коммивояжер

$$X = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 2 \\ 5 \end{pmatrix} \quad 5$$

→ легко посчитать дистанцию

$n!$ вариантов

особь

Генетический алгоритм

1. Сгенерируем N особей ← популяция

$N=100$

$f(x_1)$ ← длина пути

$f(x_N)$

2. Скрещивание

$$bread(X_i, X_j) = \widetilde{X}_i \quad f(\widetilde{X}_i)$$

получилось 50 потомков

3. Мутация

$$mut(X_i) = \overline{X}_i$$

получилось 50 мутантов

4. Селекция

TOP N

$$X_{k+1} = X_k - \alpha_k \cdot \nabla f(X_k)$$

шаг

КАК выбрать?

Line search

$$d_k = \arg\min_{\alpha} f(X_{k+1}) =$$

$$= \arg\min_{\alpha} f(X_k - \alpha \nabla f(X_k))$$

$$\varphi(\alpha) = f(X_k - \alpha \nabla f(X_k)) \leftarrow \text{„качество" шага } \alpha$$

$$\min_{\alpha \in \mathbb{R}^+}$$

Problem

Suppose, we have a problem of minimization of a function $f(x) : \mathbb{R} \to \mathbb{R}$ of scalar variable:

$$f(x) \to \min_{x \in \mathbb{R}}$$

Problem

Suppose, we have a problem of minimization of a function $f(x) : \mathbb{R} \to \mathbb{R}$ of scalar variable:

$$f(x) \to \min_{x \in \mathbb{R}}$$

Sometimes, we refer to the similar problem of finding minimum on the line segment $[a, b]$:

$$f(x) \to \min_{x \in [a,b]}$$

Problem

Suppose, we have a problem of minimization of a function $f(x) : \mathbb{R} \to \mathbb{R}$ of scalar variable:

$$f(x) \to \min_{x \in \mathbb{R}}$$

Sometimes, we refer to the similar problem of finding minimum on the line segment $[a, b]$:

$$f(x) \to \min_{x \in [a,b]}$$

---

i **Example**

выбор шага (гиперпараметра) метода обучения модели.

Typical example of line search problem is selecting appropriate stepsize for gradient descent algorithm:

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$
$$\alpha = \text{argmin } f(x_{k+1})$$

Problem

Suppose, we have a problem of minimization of a function $f(x) : \mathbb{R} \to \mathbb{R}$ of scalar variable:

$$f(x) \to \min_{x \in \mathbb{R}}$$

Sometimes, we refer to the similar problem of finding minimum on the line segment $[a, b]$:

$$f(x) \to \min_{x \in [a,b]}$$

> **i** Example
>
> Typical example of line search problem is selecting appropriate stepsize for gradient descent algorithm:
>
> $$x_{k+1} = x_k - \alpha \nabla f(x_k)$$
> $$\alpha = \text{argmin } f(x_{k+1})$$

The line search is a fundamental optimization problem that plays a crucial role in solving complex tasks. To simplify the problem, let's assume that the function, $f(x)$, is *unimodal*, meaning it has a single peak or valley.

Unimodal function

> **ℹ Definition**
>
> Function $f(x)$ is called **unimodal** on $[a, b]$, if there is $x_* \in [a, b]$, that $f(x_1) > f(x_2) \quad \forall a \leq x_1 < x_2 < x_*$ and $f(x_1) < f(x_2) \quad \forall x_* < x_1 < x_2 \leq b$
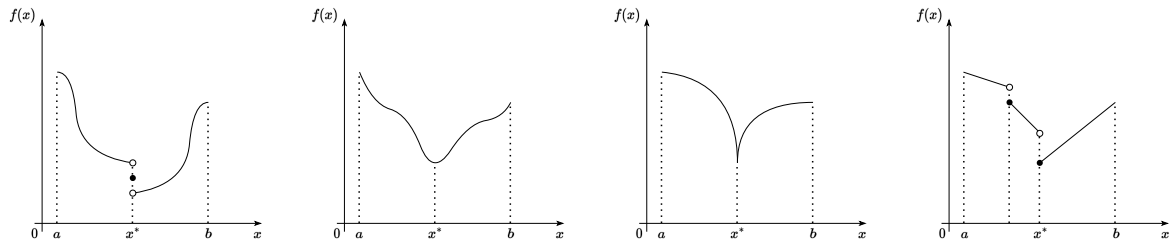
Unimodal function

← одна мода (не обяз выпуклая)

> **ⓘ Definition**
>
> Function $f(x)$ is called **unimodal** on $[a, b]$, if there is $x_* \in [a, b]$, that $f(x_1) > f(x_2)$ $\forall a \le x_1 < x_2 < x_*$ and $f(x_1) < f(x_2)$ $\forall x_* < x_1 < x_2 \le b$

Figure 1: Examples of unimodal functions

Key property of unimodal functions

Let $f(x)$ be unimodal function on $[a, b]$. Than if $x_1 < x_2 \in [a, b]$, then:

- if $f(x_1) \leq f(x_2) \rightarrow x_* \in [a, x_2]$

Key property of unimodal functions

Let $f(x)$ be unimodal function on $[a, b]$. Than if $x_1 < x_2 \in [a, b]$, then:

- if $f(x_1) \leq f(x_2) \to x_* \in [a, x_2]$
- if $f(x_1) \geq f(x_2) \to x_* \in [x_1, b]$

Key property of unimodal functions

Let $f(x)$ be unimodal function on $[a, b]$. Than if $x_1 < x_2 \in [a, b]$, then:

- if $f(x_1) \leq f(x_2) \rightarrow x_* \in [a, x_2]$
- if $f(x_1) \geq f(x_2) \rightarrow x_* \in [x_1, b]$

Key property of unimodal functions

Let $f(x)$ be unimodal function on $[a, b]$. Than if $x_1 < x_2 \in [a, b]$, then:

- if $f(x_1) \leq f(x_2) \rightarrow x_* \in [a, x_2]$
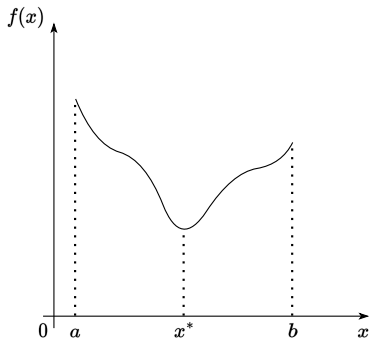- if $f(x_1) \geq f(x_2) \rightarrow x_* \in [x_1, b]$

Proof Let's prove the first statement. On the contrary, suppose that $f(x_1) \leq f(x_2)$, but $x^* > x_2$. Then necessarily $x_1 < x_2 < x^*$ and by the unimodality of the function $f(x)$ the inequality: $f(x_1) > f(x_2)$ must be satisfied. We have obtained a contradiction.

Key property of unimodal functions

Let $f(x)$ be unimodal function on $[a, b]$. Than if $x_1 < x_2 \in [a, b]$, then:

- if $f(x_1) \leq f(x_2) \rightarrow x_* \in [a, x_2]$
- if $f(x_1) \geq f(x_2) \rightarrow x_* \in [x_1, b]$

Proof Let's prove the first statement. On the contrary, suppose that $f(x_1) \leq f(x_2)$, but $x^* > x_2$. Then necessarily $x_1 < x_2 < x^*$ and by the unimodality of the function $f(x)$ the inequality: $f(x_1) > f(x_2)$ must be satisfied. We have obtained a contradiction.

Key property of unimodal functions

Let $f(x)$ be unimodal function on $[a, b]$. Than if $x_1 < x_2 \in [a, b]$, then:

- if $f(x_1) \leq f(x_2) \rightarrow x_* \in [a, x_2]$
- if $f(x_1) \geq f(x_2) \rightarrow x_* \in [x_1, b]$

Proof Let's prove the first statement. On the contrary, suppose that $f(x_1) \leq f(x_2)$, but $x^* > x_2$. Then necessarily $x_1 < x_2 < x^*$ and by the unimodality of the function $f(x)$ the inequality: $f(x_1) > f(x_2)$ must be satisfied. We have obtained a contradiction.
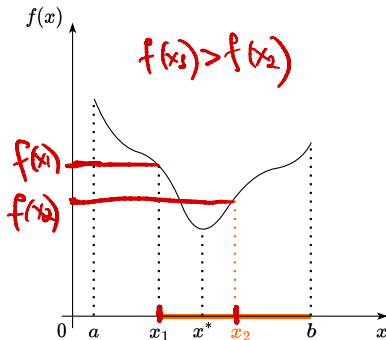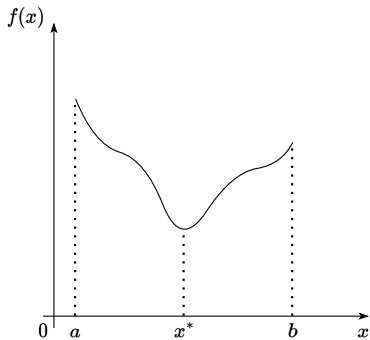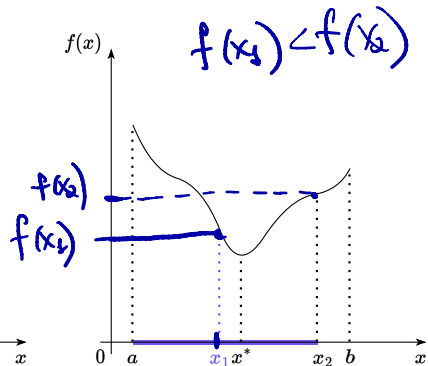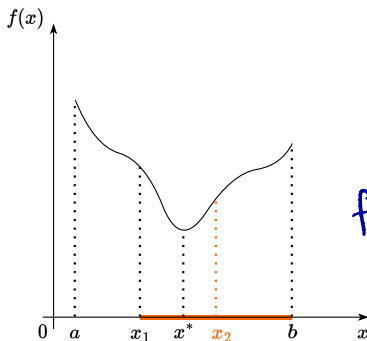
Key property of unimodal functions

Let $f(x)$ be unimodal function on $[a, b]$. Than if $x_1 < x_2 \in [a, b]$, then:

- if $f(x_1) \leq f(x_2) \rightarrow x_* \in [a, x_2]$
- if $f(x_1) \geq f(x_2) \rightarrow x_* \in [x_1, b]$

Proof Let's prove the first statement. On the contrary, suppose that $f(x_1) \leq f(x_2)$, but $x^* > x_2$. Then necessarily $x_1 < x_2 < x^*$ and by the unimodality of the function $f(x)$ the inequality: $f(x_1) > f(x_2)$ must be satisfied. We have obtained a contradiction.
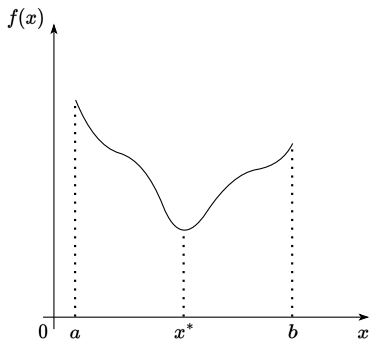
Dichotomy method

We aim to solve the following problem:

$$f(x) \to \min_{x \in [a,b]}$$

We divide a segment into two equal parts and choose the one that contains the solution of the problem using the values of functions, based on the key property described above. Our goal after one iteration of the method is to halve the solution region.

Figure 2: Dichotomy method for unimodal function

Dichotomy method

We measure the function value at the middle of the line segment

Figure 3: Dichotomy method for unimodal function

Dichotomy method

In order to apply the key property we perform another measurement.

Figure 4: Dichotomy method for unimodal function

Dichotomy method

We select the target line segment. And in this case we are lucky since we already halved the solution region. But that is not always the case.

Figure 5: Dichotomy method for unimodal function
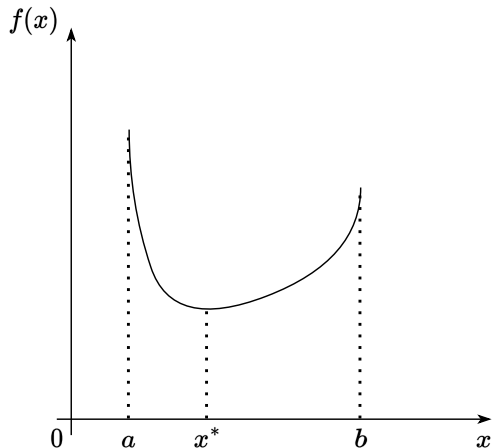
Dichotomy method

Let's consider another unimodal function.

Figure 6: Dichotomy method for unimodal function

Dichotomy method

Measure the middle of the line segment.

Figure 7: Dichotomy method for unimodal function

Dichotomy method

Get another measurement.

Figure 8: Dichotomy method for unimodal function

Dichotomy method

Select the target line segment. You can clearly see, that the obtained line segment is not the half of the initial one. It is $\frac{3}{4}(b-a)$. So to fix it we need another step of the algorithm.

Figure 9: Dichotomy method for unimodal function

Dichotomy method

After another additional measurement, we will surely get
$\frac{2}{3}\frac{3}{4}(b-a) = \frac{1}{2}(b-a)$

Figure 10: Dichotomy method for unimodal function

Dichotomy method

To sum it up, each subsequent iteration will require at most two function value measurements.

Figure 11: Dichotomy method for unimodal function

Dichotomy method. Algorithm

```python
def binary_search(f, a, b, epsilon):
    c = (a + b) / 2
    while abs(b - a) > epsilon:
        y = (a + c) / 2.0
        if f(y) <= f(c):
            b = c
            c = y
        else:
            z = (b + c) / 2.0
            if f(c) <= f(z):
                a = y
                b = z
            else:
                a = c
                c = z
    return c
```

(handwritten annotations)

- c = (a + b) / 2 ← серединка
- y = (a + c) / 2.0 ← левая четверть
- здесь
- z = (b + c) / 2.0 ← правая четверть
- здесь
- здесь

1 вызов f
2 вызова f
2 вызова f

$f \to \min\limits_{x,y,z}$ Line search

Dichotomy method. Bounds

The length of the line segment on $k + 1$-th iteration:

$$\Delta_{k+1} = b_{k+1} - a_{k+1} = \frac{1}{2^k}(b - a)$$

Dichotomy method. Bounds

The length of the line segment on $k+1$-th iteration:

$$\Delta_{k+1} = b_{k+1} - a_{k+1} = \frac{1}{2^k}(b-a)$$

For unimodal functions, this holds if we select the middle of a segment as an output of the iteration x_{k+1}:

$$|x_{k+1} - x_*| \leq \frac{\Delta_{k+1}}{2} \leq \frac{1}{2^{k+1}}(b-a) \leq (0.5)^{k+1} \cdot (b-a)$$

Dichotomy method. Bounds

The length of the line segment on $k + 1$-th iteration:

$$\Delta_{k+1} = b_{k+1} - a_{k+1} = \frac{1}{2^k}(b - a)$$

For unimodal functions, this holds if we select the middle of a segment as an output of the iteration x_{k+1}:

$$|x_{k+1} - x_*| \leq \frac{\Delta_{k+1}}{2} \leq \frac{1}{2^{k+1}}(b - a) \leq (0.5)^{k+1} \cdot (b - a)$$

Note, that at each iteration we ask oracle no more, than 2 times, so the number of function evaluations is $N = 2 \cdot k$, which implies:

$$|x_{k+1} - x_*| \leq (0.5)^{\frac{N}{2}+1} \cdot (b - a) \leq (0.707)^N \frac{b - a}{2}$$

скорость сходимости

геом.
прогрессии

чем
меньше,

тем
лучше

Dichotomy method. Bounds

The length of the line segment on $k + 1$-th iteration:

$$\Delta_{k+1} = b_{k+1} - a_{k+1} = \frac{1}{2^k}(b - a)$$

For unimodal functions, this holds if we select the middle of a segment as an output of the iteration x_{k+1}:

$$|x_{k+1} - x_*| \leq \frac{\Delta_{k+1}}{2} \leq \frac{1}{2^{k+1}}(b - a) \leq (0.5)^{k+1} \cdot (b - a)$$

Note, that at each iteration we ask oracle no more, than 2 times, so the number of function evaluations is $N = 2 \cdot k$, which implies:

$$|x_{k+1} - x_*| \leq (0.5)^{\frac{N}{2}+1} \cdot (b - a) \leq (0.707)^N \frac{b - a}{2}$$

By marking the right side of the last inequality for ε, we get the number of method iterations needed to achieve ε accuracy:

$$K = \left\lceil \log_2 \frac{(b - a)}{\varepsilon} - 1 \right\rceil$$

Golden selection

The idea is quite similar to the dichotomy method. There are two golden points on the line segment (left and right) and the insightful idea is, that on the next iteration one of the points will remain the golden point.

Iteration k

Iteration $k+1$

r_{k+1} l_{k+1}

$$\frac{1}{1-x} = \frac{1-x}{x}$$

$$x = (1-x)^2$$

$$x^2 - 2x + 1 - x = 0$$

$$x^2 - 3x + 1 = 0$$

$$1 - x = 0.62$$

$$x \approx 0.38$$

Figure 12: Key idea, that allows us to decrease function evaluations

Golden selection. Algorithm

```python
def golden_search(f, a, b, epsilon):
    tau = (sqrt(5) + 1) / 2
    y = a + (b - a) / tau**2
    z = a + (b - a) / tau
    while b - a > epsilon:
        if f(y) <= f(z):
            b = z
            z = y
            y = a + (b - a) / tau**2
        else:
            a = y
            y = z
            z = a + (b - a) / tau
    return (a + b) / 2
```

Golden selection. Bounds

$$|x_{k+1} - x_*| \le b_{k+1} - a_{k+1} = \left(\frac{1}{\tau}\right)^{N-1} (b-a) \approx 0.618^k (b-a),$$

where $\tau = \frac{\sqrt{5}+1}{2}$.

Хуже, чем дихотомия по итерации

- The geometric progression constant **more** than the dichotomy method - 0.618 worse than 0.5

Golden selection. Bounds

$$|x_{k+1} - x_*| \le b_{k+1} - a_{k+1} = \left(\frac{1}{\tau}\right)^{N-1}(b-a) \approx 0.618^k(b-a),$$

where $\tau = \frac{\sqrt{5}+1}{2}$.

- The geometric progression constant **more** than the dichotomy method - 0.618 worse than 0.5
- The number of function calls **is less** than for the dichotomy method - 0.707 worse than 0.618 - (for each iteration of the dichotomy method, except for the first one, the function is calculated no more than 2 times, and for the gold method - no more than one)

Successive parabolic interpolation

Sampling 3 points of a function determines unique parabola. Using this information we will go directly to its minimum. Suppose, we have 3 points $x_1 < x_2 < x_3$ such that line segment $[x_1, x_3]$ contains minimum of a function $f(x)$. Then, we need to solve the following system of equations:

Successive parabolic interpolation

Sampling 3 points of a function determines unique parabola. Using this information we will go directly to its minimum. Suppose, we have 3 points $x_1 < x_2 < x_3$ such that line segment $[x_1, x_3]$ contains minimum of a function $f(x)$. Then, we need to solve the following system of equations:

$$ax_i^2 + bx_i + c = f_i = f(x_i), i = 1, 2, 3$$

Note, that this system is linear, since we need to solve it on a, b, c. Minimum of this parabola will be calculated as:

комбинация золотого сечения + метода парабол. интерпол.

метод Брента

Successive parabolic interpolation

Sampling 3 points of a function determines unique parabola. Using this information we will go directly to its minimum. Suppose, we have 3 points $x_1 < x_2 < x_3$ such that line segment $[x_1, x_3]$ contains minimum of a function $f(x)$. Then, we need to solve the following system of equations:

$$ax_i^2 + bx_i + c = f_i = f(x_i), i = 1, 2, 3$$

Note, that this system is linear, since we need to solve it on a, b, c. Minimum of this parabola will be calculated as:

$$u = -\frac{b}{2a} = x_2 - \frac{(x_2 - x_1)^2(f_2 - f_3) - (x_2 - x_3)^2(f_2 - f_1)}{2\left[(x_2 - x_1)(f_2 - f_3) - (x_2 - x_3)(f_2 - f_1)\right]}$$

Note, that if $f_2 < f_1, f_2 < f_3$, than u will lie in $[x_1, x_3]$

Successive parabolic interpolation. Algorithm [1]

```python
def parabola_search(f, x1, x2, x3, epsilon):
    f1, f2, f3 = f(x1), f(x2), f(x3)
    while x3 - x1 > epsilon:
        u = x2 - ((x2 - x1)**2*(f2 - f3) - (x2 - x3)**2*(f2 - f1))/(2*((x2 - x1)*(f2 - f3) - (x2 - x3)*(f2 - f1)))
        fu = f(u)

        if x2 <= u:
            if f2 <= fu:
                x1, x2, x3 = x1, x2, u
                f1, f2, f3 = f1, f2, fu
            else:
                x1, x2, x3 = x2, u, x3
                f1, f2, f3 = f2, fu, f3
        else:
            if fu <= f2:
                x1, x2, x3 = x1, u, x2
                f1, f2, f3 = f1, fu, f2
            else:
                x1, x2, x3 = u, x2, x3
                f1, f2, f3 = fu, f2, f3
    return (x1 + x3) / 2
```

---

[1] The convergence of this method is superlinear, but local, which means, that you can take profit from using this method only near some neighbour of optimum. *Here* is the proof of superlinear convergence of order 1.32.

Inexact line search

Sometimes it is enough to find a solution, which will approximately solve out problem. This is very typical scenario for mentioned stepsize selection problem

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$
$$\alpha = \mathsf{argmin}\ f(x_{k+1})$$

Inexact line search

Sometimes it is enough to find a solution, which will approximately solve out problem. This is very typical scenario for mentioned stepsize selection problem

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$
$$\alpha = \text{argmin } f(x_{k+1})$$

Consider a scalar function $\phi(\alpha)$ at a point x_k:

$$\boxed{\phi(\alpha) = f(x_k - \alpha \nabla f(x_k)), \alpha \geq 0}$$

значение $f(x_{k+1})$
после шага α изт. x_k

Inexact line search

Sometimes it is enough to find a solution, which will approximately solve out problem. This is very typical scenario for mentioned stepsize selection problem

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$
$$\alpha = \arg\min f(x_{k+1})$$

Consider a scalar function $\phi(\alpha)$ at a point x_k:

$$\phi(\alpha) = f(x_k - \alpha \nabla f(x_k)), \alpha \geq 0$$

The first-order approximation of $\phi(\alpha)$ near $\alpha = 0$ is:

$$\phi(\alpha) \approx f(x_k) - \alpha \nabla f(x_k)^\top \nabla f(x_k)$$

$$\frac{\partial \phi}{\partial \alpha} = \frac{\partial \phi}{\partial f} \cdot \frac{\partial f}{\partial x} \cdot \frac{\partial x}{\partial \alpha} (-\partial f)$$

$$= 1 \cdot \nabla f(x_k) \cdot \nabla f(x_k)$$

как будто никуда не выходили

α слишк. больш

касат.
в т. α=0
угол наклона

лучше знач. шага

$$\frac{\partial \phi}{\partial \alpha}$$

Пусть есть $\varphi(\alpha)$

$$\varphi_0^I(\alpha) = \varphi(0) + \frac{\partial \varphi}{\partial \alpha} \cdot (\alpha - 0) =$$

$$= f(x_k) - \|\nabla f(x_k)\|^2 \cdot \alpha$$

$$f(x_k) - \alpha \|\nabla f(x_k)\|_2^2$$

Figure 13: Illustration of Taylor approximation of $\phi_0^I(\alpha)$

Inexact line search. Sufficient Decrease

The inexact line search condition, known as the Armijo condition, states that α should provide sufficient decrease in the function f, satisfying:

$$f(x_k - \alpha \nabla f(x_k)) \leq f(x_k) - c_1 \cdot \alpha \nabla f(x_k)^\top \nabla f(x_k)$$

Inexact line search. Sufficient Decrease

The inexact line search condition, known as the Armijo condition, states that α should provide sufficient decrease in the function f, satisfying:

$$f(x_k - \alpha\nabla f(x_k)) \leq f(x_k) - c_1 \cdot \alpha\nabla f(x_k)^\top \nabla f(x_k)$$

for some constant $c_1 \in (0,1)$. Note that setting $c_1 = 1$ corresponds to the first-order Taylor approximation of $\phi(\alpha)$. However, this condition can accept very small values of α, potentially slowing down the solution process. Typically, $c_1 \approx 10^{-4}$ is used in practice.

Inexact line search. Sufficient Decrease

The inexact line search condition, known as the Armijo condition, states that α should provide sufficient decrease in the function f, satisfying:

$$f(x_k - \alpha \nabla f(x_k)) \leq f(x_k) - c_1 \cdot \alpha \nabla f(x_k)^\top \nabla f(x_k)$$

for some constant $c_1 \in (0,1)$. Note that setting $c_1 = 1$ corresponds to the first-order Taylor approximation of $\phi(\alpha)$. However, this condition can accept very small values of α, potentially slowing down the solution process. Typically, $c_1 \approx 10^{-4}$ is used in practice.

ℹ **Example**

If $f(x)$ represents a cost function in an optimization problem, choosing an appropriate c_1 value is crucial. For instance, in a machine learning model training scenario, an improper c_1 might lead to either very slow convergence or missing the minimum.

Figure 14: Illustration of sufficient decrease condition with coefficient c_1

Inexact line search. Goldstein Conditions

Consider two linear scalar functions $\phi_1(\alpha)$ and $\phi_2(\alpha)$:

$$\phi_1(\alpha) = f(x_k) - c_1\alpha\|\nabla f(x_k)\|^2$$

$$\phi_2(\alpha) = f(x_k) - c_2\alpha\|\nabla f(x_k)\|^2$$

Inexact line search. Goldstein Conditions

Consider two linear scalar functions $\phi_1(\alpha)$ and $\phi_2(\alpha)$:

$$\phi_1(\alpha) = f(x_k) - c_1\alpha\|\nabla f(x_k)\|^2$$

$$\phi_2(\alpha) = f(x_k) - c_2\alpha\|\nabla f(x_k)\|^2$$

The Goldstein-Armijo conditions locate the function $\phi(\alpha)$ between $\phi_1(\alpha)$ and $\phi_2(\alpha)$. Typically, $c_1 = \rho$ and $c_2 = 1 - \rho$, with $\rho \in (0.5, 1)$.

Figure 15: Illustration of Goldstein conditions

Inexact line search. Curvature Condition

To avoid excessively short steps, we introduce a second criterion:

$$-\nabla f(x_k - \alpha \nabla f(x_k))^\top \nabla f(x_k) \geq c_2 \nabla f(x_k)^\top (-\nabla f(x_k))$$

Inexact line search. Curvature Condition

To avoid excessively short steps, we introduce a second criterion:

$$-\nabla f(x_k - \alpha \nabla f(x_k))^\top \nabla f(x_k) \geq c_2 \nabla f(x_k)^\top (-\nabla f(x_k))$$

for some $c_2 \in (c_1, 1)$. Here, c_1 is from the Armijo condition. The left-hand side is the derivative $\nabla_\alpha \phi(\alpha)$, ensuring that the slope of $\phi(\alpha)$ at the target point is at least c_2 times the initial slope $\nabla_\alpha \phi(\alpha)(0)$. Commonly, $c_2 \approx 0.9$ is used for Newton or quasi-Newton methods. Together, the sufficient decrease and curvature conditions form the Wolfe conditions.

Figure 16: Illustration of curvature condition

Inexact line search. Wolfe Condition

$$-\nabla f(x_k - \alpha \nabla f(x_k))^\top \nabla f(x_k) \geq c_2 \nabla f(x_k)^\top (-\nabla f(x_k))$$

Together, the sufficient decrease and curvature conditions form the Wolfe conditions.

Figure 17: Illustration of Wolfe condition

Backtracking Line Search

Backtracking line search is a technique to find a step size that satisfies the Armijo condition, Goldstein conditions, or other criteria of inexact line search. It begins with a relatively large step size and iteratively scales it down until a condition is met.

Backtracking Line Search

Backtracking line search is a technique to find a step size that satisfies the Armijo condition, Goldstein conditions, or other criteria of inexact line search. It begins with a relatively large step size and iteratively scales it down until a condition is met.

Algorithm:

1. Choose an initial step size, α_0, and parameters $\beta \in (0, 1)$ and $c_1 \in (0, 1)$.

Backtracking Line Search

Backtracking line search is a technique to find a step size that satisfies the Armijo condition, Goldstein conditions, or other criteria of inexact line search. It begins with a relatively large step size and iteratively scales it down until a condition is met.

Algorithm:

1. Choose an initial step size, α_0, and parameters $\beta \in (0,1)$ and $c_1 \in (0,1)$.
2. Check if the chosen step size satisfies the chosen condition (e.g., Armijo condition).
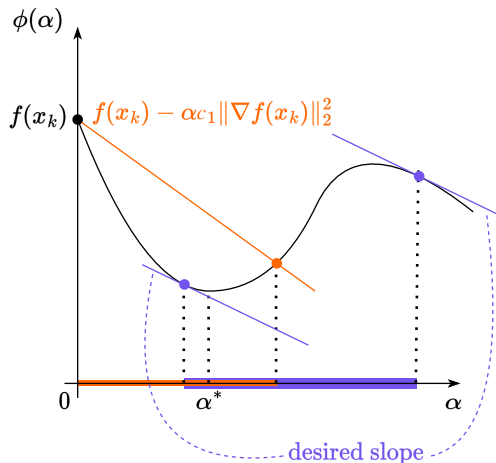
Backtracking Line Search

Backtracking line search is a technique to find a step size that satisfies the Armijo condition, Goldstein conditions, or other criteria of inexact line search. It begins with a relatively large step size and iteratively scales it down until a condition is met.

Algorithm:
1. Choose an initial step size, α_0, and parameters $\beta \in (0, 1)$ and $c_1 \in (0, 1)$.
2. Check if the chosen step size satisfies the chosen condition (e.g., Armijo condition).
3. If the condition is satisfied, stop; else, set $\alpha := \beta\alpha$ and repeat step 2.

Backtracking Line Search

Backtracking line search is a technique to find a step size that satisfies the Armijo condition, Goldstein conditions, or other criteria of inexact line search. It begins with a relatively large step size and iteratively scales it down until a condition is met.

Algorithm:

1. Choose an initial step size, α_0, and parameters $\beta \in (0, 1)$ and $c_1 \in (0, 1)$.
2. Check if the chosen step size satisfies the chosen condition (e.g., Armijo condition).
3. If the condition is satisfied, stop; else, set $\alpha := \beta\alpha$ and repeat step 2.

Backtracking Line Search

Backtracking line search is a technique to find a step size that satisfies the Armijo condition, Goldstein conditions, or other criteria of inexact line search. It begins with a relatively large step size and iteratively scales it down until a condition is met.

Algorithm:
1. Choose an initial step size, α_0, and parameters $\beta \in (0, 1)$ and $c_1 \in (0, 1)$.
2. Check if the chosen step size satisfies the chosen condition (e.g., Armijo condition).
3. If the condition is satisfied, stop; else, set $\alpha := \beta\alpha$ and repeat step 2.

The step size α is updated as

$$\alpha_{k+1} := \beta\alpha_k$$

in each iteration until the chosen condition is satisfied.

> **i** Example
>
> In machine learning model training, the backtracking line search can be used to adjust the learning rate. If the loss doesn't decrease sufficiently, the learning rate is reduced multiplicatively until the Armijo condition is met.

Numerical illustration

Figure 18: Comparison of different line search algorithms

Open In Colab ♣

Gradient Descent

Direction of local steepest descent

Let's consider a linear approximation of the
differentiable function f along some direction
$h, \|h\|_2 = 1$:

Direction of local steepest descent

Let's consider a linear approximation of the differentiable function f along some direction $h, \|h\|_2 = 1$:

$$f(x + \alpha h) = f(x) + \alpha \langle f'(x), h \rangle + o(\alpha)$$

Direction of local steepest descent

Let's consider a linear approximation of the differentiable function f along some direction $h, \|h\|_2 = 1$:

$$f(x + \alpha h) = f(x) + \alpha \langle f'(x), h \rangle + o(\alpha)$$

We want h to be a decreasing direction:

$$f(x + \alpha h) < f(x)$$

$$f(x) + \alpha \langle f'(x), h \rangle + o(\alpha) < f(x)$$

Direction of local steepest descent

Let's consider a linear approximation of the differentiable function f along some direction $h, \|h\|_2 = 1$:

$$f(x + \alpha h) = f(x) + \alpha \langle f'(x), h \rangle + o(\alpha)$$

We want h to be a decreasing direction:

$$f(x + \alpha h) < f(x)$$

$$f(x) + \alpha \langle f'(x), h \rangle + o(\alpha) < f(x)$$

and going to the limit at $\alpha \to 0$:

$$\langle f'(x), h \rangle \leq 0$$

Direction of local steepest descent

Let's consider a linear approximation of the differentiable function f along some direction $h, \|h\|_2 = 1$:

$$f(x + \alpha h) = f(x) + \alpha \langle f'(x), h \rangle + o(\alpha)$$

We want h to be a decreasing direction:

$$f(x + \alpha h) < f(x)$$

$$f(x) + \alpha \langle f'(x), h \rangle + o(\alpha) < f(x)$$

and going to the limit at $\alpha \to 0$:

$$\langle f'(x), h \rangle \leq 0$$

Also from Cauchy–Bunyakovsky–Schwarz inequality:

$$|\langle f'(x), h \rangle| \leq \|f'(x)\|_2 \|h\|_2$$
$$\langle f'(x), h \rangle \geq -\|f'(x)\|_2 \|h\|_2 = -\|f'(x)\|_2$$

Direction of local steepest descent

Let's consider a linear approximation of the differentiable function f along some direction $h, \|h\|_2 = 1$:

$$f(x + \alpha h) = f(x) + \alpha \langle f'(x), h \rangle + o(\alpha)$$

We want h to be a decreasing direction:

$$f(x + \alpha h) < f(x)$$

$$f(x) + \alpha \langle f'(x), h \rangle + o(\alpha) < f(x)$$

and going to the limit at $\alpha \to 0$:

$$\langle f'(x), h \rangle \leq 0$$

Also from Cauchy–Bunyakovsky–Schwarz inequality:

$$|\langle f'(x), h \rangle| \leq \|f'(x)\|_2 \|h\|_2$$
$$\langle f'(x), h \rangle \geq -\|f'(x)\|_2 \|h\|_2 = -\|f'(x)\|_2$$

Thus, the direction of the antigradient

$$h = -\frac{f'(x)}{\|f'(x)\|_2}$$

gives the direction of the **steepest local** decreasing of the function f.

Direction of local steepest descent

Let's consider a linear approximation of the differentiable function f along some direction $h, \|h\|_2 = 1$:

$$f(x + \alpha h) = f(x) + \alpha\langle f'(x), h\rangle + o(\alpha)$$

We want h to be a decreasing direction:

$$f(x + \alpha h) < f(x)$$

$$f(x) + \alpha\langle f'(x), h\rangle + o(\alpha) < f(x)$$

and going to the limit at $\alpha \to 0$:

$$\langle f'(x), h\rangle \leq 0$$

Also from Cauchy–Bunyakovsky–Schwarz inequality:

$$|\langle f'(x), h\rangle| \leq \|f'(x)\|_2\|h\|_2$$
$$\langle f'(x), h\rangle \geq -\|f'(x)\|_2\|h\|_2 = -\|f'(x)\|_2$$

Thus, the direction of the antigradient

$$h = -\frac{f'(x)}{\|f'(x)\|_2}$$

gives the direction of the **steepest local** decreasing of the function f. The result of this method is

$$x_{k+1} = x_k - \alpha f'(x_k)$$

Convergence of Gradient Descent algorithm

Heavily depends on the choice of the learning rate α:

Loss value 0.87

w_1 0.50, w_2 3.00

Exact line search aka steepest descent

$$\alpha_k = \arg \min_{\alpha \in \mathbb{R}^+} f(x_{k+1}) = \arg \min_{\alpha \in \mathbb{R}^+} f(x_k - \alpha \nabla f(x_k))$$

More theoretical than practical approach. It also allows you to analyze the convergence, but often exact line search can be difficult if the function calculation takes too long or costs a lot. An interesting theoretical property of this method is that each following iteration is orthogonal to the previous one:

$$\alpha_k = \arg \min_{\alpha \in \mathbb{R}^+} f(x_k - \alpha \nabla f(x_k))$$

метод наискорейшего спуска

Exact line search aka steepest descent

$$\alpha_k = \arg \min_{\alpha \in \mathbb{R}^+} f(x_{k+1}) = \arg \min_{\alpha \in \mathbb{R}^+} f(x_k - \alpha \nabla f(x_k))$$

More theoretical than practical approach. It also allows you to analyze the convergence, but often exact line search can be difficult if the function calculation takes too long or costs a lot. An interesting theoretical property of this method is that each following iteration is orthogonal to the previous one:

$$\alpha_k = \arg \min_{\alpha \in \mathbb{R}^+} f(x_k - \alpha \nabla f(x_k))$$

Optimality conditions:

Exact line search aka steepest descent

$$\alpha_k = \arg \min_{\alpha \in \mathbb{R}^+} f(x_{k+1}) = \arg \min_{\alpha \in \mathbb{R}^+} f(x_k - \alpha \nabla f(x_k))$$

More theoretical than practical approach. It also allows you to analyze the convergence, but often exact line search can be difficult if the function calculation takes too long or costs a lot. An interesting theoretical property of this method is that each following iteration is orthogonal to the previous one:

$$\underset{\text{necessary}}{\text{neобx}} \quad \alpha_k = \arg \min_{\alpha \in \mathbb{R}^+} f(\overset{x^{k+1}}{\overbrace{x_k - \alpha \nabla f(x_k)}})$$

Optimality conditions:

$$\frac{\partial f(x_{k+1})}{\partial \alpha} = 0 \quad \frac{\partial f}{\partial x_{k+1}} \cdot \frac{\partial x_{k+1}}{\partial \alpha} = 0$$

$$\boxed{\nabla f(x_{k+1})^\top \nabla f(x_k) = 0}$$

$$\frac{\partial (x_{k+1})}{\partial \alpha} = \frac{\partial (x_k - \alpha \nabla f(x_k))}{\partial \alpha} =$$

$$= - \nabla f(x_k)$$

Figure 19: Steepest Descent

Open In Colab ♣

пусть есть $f(x) = \frac{1}{2} x^T A x - b^T x$ ← кв. функция (лин. рег.)

$A \in S_+^n$

$\nabla f = Ax - b$

$\nabla^2 f = A$

$\nabla f(x^*) = 0$

$Ax^* = b$

шаг GD:

Strongly convex quadratics

$x_{k+1} = x_k - \alpha \cdot \nabla f(x_k) = x_k - \alpha \cdot (A x_k - b) = (I - \alpha A) x_k + \alpha b$

наискорейший спуск:

$f(x_{k+1}) \to \min\limits_{\alpha \in \mathbb{R}^+}$

$f\big((I - \alpha A) x_k + \alpha b\big) \to \min\limits_{\alpha \in \mathbb{R}}$

$$\alpha = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k)^T \nabla f(x_k) \nabla f(x_k)}$$

заметим, что: $= g$

$\nabla f(x_{k+1})^T \nabla f(x_k) = 0$

$\Rightarrow (A x_{k+1} - b)^T g = 0$

$g^T \big(A(x_k - \alpha g) - b\big) = 0$

$g^T \Big(\underbrace{A x_k - b}_{g} - \alpha A g\Big) = 0$

$g^T g - \alpha g^T A g = 0$

$\alpha = \dfrac{g^T g}{g^T A g}$

Coordinate shift

Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.$$

Coordinate shift

Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.

Coordinate shift

Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.
- Secondly, we have a spectral decomposition of the matrix A:

$$A = Q \Lambda Q^T$$

Coordinate shift

Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}_{++}^d.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.
- Secondly, we have a spectral decomposition of the matrix A:

$$A = Q \Lambda Q^T$$

- Let's show, that we can switch coordinates to make an analysis a little bit easier. Let $\hat{x} = Q^T(x - x^*)$, where x^* is the minimum point of initial function, defined by $Ax^* = b$. At the same time $x = Q\hat{x} + x^*$.

Coordinate shift
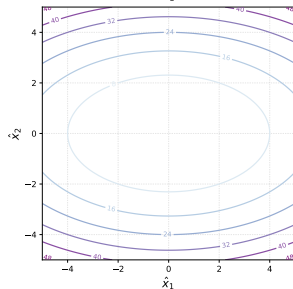
Consider the following quadratic optimization problem:
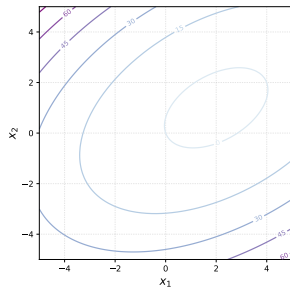
$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}_{++}^d.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.

- Secondly, we have a spectral decomposition of the matrix A:

$$A = Q \Lambda Q^T$$

- Let's show, that we can switch coordinates to make an analysis a little bit easier. Let $\hat{x} = Q^T(x - x^*)$, where x^* is the minimum point of initial function, defined by $Ax^* = b$. At the same time $x = Q\hat{x} + x^*$.

$$f(\hat{x}) = \frac{1}{2}(Q\hat{x} + x^*)^\top A(Q\hat{x} + x^*) - b^\top(Q\hat{x} + x^*)$$
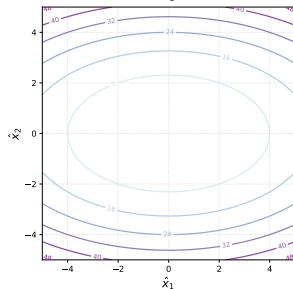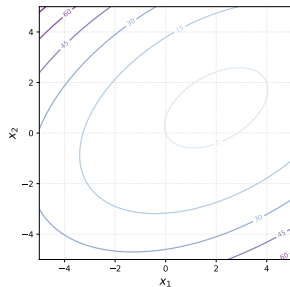
Coordinate shift

Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}_{++}^d.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.
- Secondly, we have a spectral decomposition of the matrix A:

$$A = Q \Lambda Q^T$$

- Let's show, that we can switch coordinates to make an analysis a little bit easier. Let $\hat{x} = Q^T(x - x^*)$, where x^* is the minimum point of initial function, defined by $Ax^* = b$. At the same time $x = Q\hat{x} + x^*$.

$$f(\hat{x}) = \frac{1}{2}(Q\hat{x} + x^*)^\top A (Q\hat{x} + x^*) - b^\top (Q\hat{x} + x^*)$$
$$= \frac{1}{2}\hat{x}^T Q^T A Q \hat{x} + (x^*)^T A Q \hat{x} + \frac{1}{2}(x^*)^T A (x^*)^T - b^T Q \hat{x} - b^T x^*$$
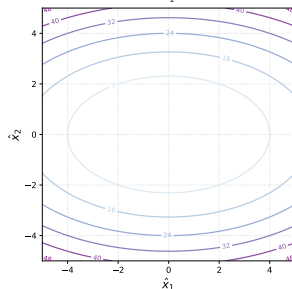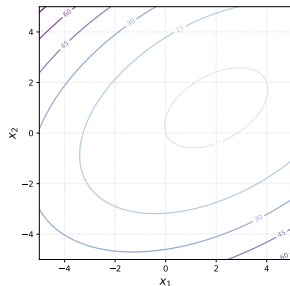
Coordinate shift

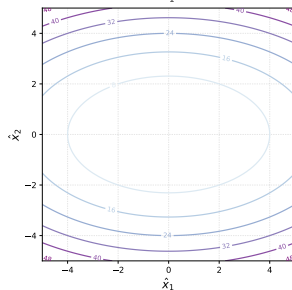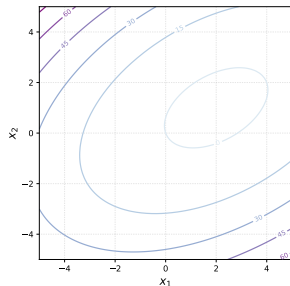Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}_{++}^d.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.
- Secondly, we have a spectral decomposition of the matrix A:

$$A = Q \Lambda Q^T$$

- Let's show, that we can switch coordinates to make an analysis a little bit easier. Let $\hat{x} = Q^T(x - x^*)$, where x^* is the minimum point of initial function, defined by $Ax^* = b$. At the same time $x = Q\hat{x} + x^*$.

$$
\begin{aligned}
f(\hat{x}) &= \frac{1}{2}(Q\hat{x} + x^*)^\top A(Q\hat{x} + x^*) - b^\top(Q\hat{x} + x^*) \\
&= \frac{1}{2}\hat{x}^T Q^T A Q \hat{x} + (x^*)^T A Q \hat{x} + \frac{1}{2}(x^*)^T A (x^*)^T - b^T Q \hat{x} - b^T x^* \\
&= \frac{1}{2}\hat{x}^T \Lambda \hat{x}
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})x_{(i)}^k \text{ For } i\text{-th coordinate}$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})x_{(i)}^k \text{ For } i\text{-th coordinate}$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})x_{(i)}^k \text{ For } i\text{-th coordinate}$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda)x^k \\
x_{(i)}^{k+1} &= (1 - \alpha^k \lambda_{(i)})x_{(i)}^k \text{ For } i\text{-th coordinate} \\
x_{(i)}^{k+1} &= (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$
|1 - \alpha\mu| < 1
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})x_{(i)}^k \text{ For } i\text{-th coordinate}$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:
$$\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1$$
$$-1 < 1 - \alpha\mu < 1$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})x^k_{(i)} \text{ For } i\text{-th coordinate}$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:
$$\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1$$
$$-1 < 1 - \alpha\mu < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})x^k_{(i)} \text{ For } i\text{-th coordinate}$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:
$$\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1 \qquad\qquad\qquad |1 - \alpha L| < 1$$
$$-1 < 1 - \alpha\mu < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})x^k_{(i)} \text{ For } i\text{-th coordinate}$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:
$$\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1 \qquad\qquad |1 - \alpha L| < 1$$
$$-1 < 1 - \alpha\mu < 1 \qquad\qquad -1 < 1 - \alpha L < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda) x^k$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate}$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:
$$\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1 \qquad\qquad |1 - \alpha L| < 1$$
$$-1 < 1 - \alpha\mu < 1 \qquad\qquad -1 < 1 - \alpha L < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0 \qquad\qquad \alpha < \frac{2}{L} \qquad \alpha L > 0$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})x_{(i)}^k \text{ For } i\text{-th coordinate}$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:
$$\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1 \qquad\qquad |1 - \alpha L| < 1$$
$$-1 < 1 - \alpha\mu < 1 \qquad\qquad -1 < 1 - \alpha L < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0 \qquad\qquad \alpha < \frac{2}{L} \qquad \alpha L > 0$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2} x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda) x^k$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)}) x_{(i)}^k \text{ For } i\text{-th coordinate}$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:
$$\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1 \qquad\qquad |1 - \alpha L| < 1$$
$$-1 < 1 - \alpha\mu < 1 \qquad\qquad -1 < 1 - \alpha L < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0 \qquad\qquad \alpha < \frac{2}{L} \qquad \alpha L > 0$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2} x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda) x^k$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)}) x_{(i)}^k \text{ For } i\text{-th coordinate}$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0$$

Now we would like to tune α to choose the best (lowest) convergence rate

$$\rho^* = \min_\alpha \rho(\alpha)$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1 \qquad\qquad |1 - \alpha L| < 1$$
$$-1 < 1 - \alpha\mu < 1 \qquad\qquad -1 < 1 - \alpha L < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0 \qquad\qquad \alpha < \frac{2}{L} \qquad \alpha L > 0$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})x^k_{(i)} \text{ For } i\text{-th coordinate}$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

Now we would like to tune α to choose the best (lowest) convergence rate

$$\rho^* = \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha \lambda_{(i)}|$$

$$|1 - \alpha\mu| < 1 \qquad\qquad |1 - \alpha L| < 1$$
$$-1 < 1 - \alpha\mu < 1 \qquad -1 < 1 - \alpha L < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0 \qquad\qquad \alpha < \frac{2}{L} \qquad \alpha L > 0$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})x_{(i)}^k \text{ For } i\text{-th coordinate}$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

Now we would like to tune α to choose the best (lowest) convergence rate

$$\rho^* = \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha\lambda_{(i)}|$$
$$= \min_\alpha \{|1 - \alpha\mu|, |1 - \alpha L|\}$$

$$\begin{array}{ll}
|1 - \alpha\mu| < 1 & |1 - \alpha L| < 1 \\
-1 < 1 - \alpha\mu < 1 & -1 < 1 - \alpha L < 1 \\
\alpha < \dfrac{2}{\mu} \quad \alpha\mu > 0 & \alpha < \dfrac{2}{L} \quad \alpha L > 0
\end{array}$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})x^k_{(i)} \text{ For } i\text{-th coordinate}$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1 \qquad\qquad |1 - \alpha L| < 1$$
$$-1 < 1 - \alpha\mu < 1 \qquad\qquad -1 < 1 - \alpha L < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0 \qquad\qquad \alpha < \frac{2}{L} \qquad \alpha L > 0$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$\rho^* = \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha\lambda_{(i)}|$$
$$= \min_\alpha \{|1 - \alpha\mu|, |1 - \alpha L|\}$$
$$\alpha^*: \quad 1 - \alpha^*\mu = \alpha^* L - 1$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})x_{(i)}^k \text{ For } i\text{-th coordinate}$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:
$$\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$\begin{array}{ll} |1 - \alpha\mu| < 1 & \quad |1 - \alpha L| < 1 \\ -1 < 1 - \alpha\mu < 1 & \quad -1 < 1 - \alpha L < 1 \\ \alpha < \dfrac{2}{\mu} \quad \alpha\mu > 0 & \quad \alpha < \dfrac{2}{L} \quad \alpha L > 0 \end{array}$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$\rho^* = \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha\lambda_{(i)}|$$
$$= \min_\alpha \{|1 - \alpha\mu|, |1 - \alpha L|\}$$
$$\alpha^*: \quad 1 - \alpha^*\mu = \alpha^* L - 1$$
$$\alpha^* = \frac{2}{\mu + L}$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})x_{(i)}^k \text{ For } i\text{-th coordinate}$$
$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:
$$\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1 \qquad\qquad |1 - \alpha L| < 1$$
$$-1 < 1 - \alpha\mu < 1 \qquad -1 < 1 - \alpha L < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0 \qquad \alpha < \frac{2}{L} \qquad \alpha L > 0$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$\rho^* = \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha\lambda_{(i)}|$$
$$= \min_\alpha \{|1 - \alpha\mu|, |1 - \alpha L|\}$$
$$\alpha^*: \quad 1 - \alpha^*\mu = \alpha^* L - 1$$
$$\alpha^* = \frac{2}{\mu + L} \quad \rho^* = \frac{L - \mu}{L + \mu}$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2} x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} = x^k - \alpha^k \nabla f(x^k) &= x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k
\end{aligned}
$$

$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)}) x_{(i)}^k \text{ For } i\text{-th coordinate}$$

$$x_{(i)}^{k+1} = (1 - \alpha^k \lambda_{(i)})^k x_{(i)}^0$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$
\begin{array}{ll}
|1 - \alpha \mu| < 1 & |1 - \alpha L| < 1 \\
-1 < 1 - \alpha \mu < 1 & -1 < 1 - \alpha L < 1 \\
\alpha < \dfrac{2}{\mu} \quad \alpha \mu > 0 & \alpha < \dfrac{2}{L} \quad \alpha L > 0
\end{array}
$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$
\begin{aligned}
\rho^* = \min_\alpha \rho(\alpha) &= \min_\alpha \max_i |1 - \alpha \lambda_{(i)}| \\
&= \min_\alpha \{|1 - \alpha \mu|, |1 - \alpha L|\}
\end{aligned}
$$

$$\alpha^*: \quad 1 - \alpha^* \mu = \alpha^* L - 1$$

$$\alpha^* = \frac{2}{\mu + L} \quad \rho^* = \frac{L - \mu}{L + \mu}$$

$$x^{k+1} = \left(\frac{L - \mu}{L + \mu} \right)^k x^0$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$
$$= (I - \alpha^k \Lambda)x^k$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})x^k_{(i)} \text{ For } i\text{-th coordinate}$$
$$x^{k+1}_{(i)} = (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:
$$\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1$$

Remember, that $\lambda_{\min} = \mu > 0, \lambda_{\max} = L \geq \mu$.

$$|1 - \alpha\mu| < 1 \qquad\qquad |1 - \alpha L| < 1$$
$$-1 < 1 - \alpha\mu < 1 \qquad -1 < 1 - \alpha L < 1$$
$$\alpha < \frac{2}{\mu} \qquad \alpha\mu > 0 \qquad \alpha < \frac{2}{L} \qquad \alpha L > 0$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$\rho^* = \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha\lambda_{(i)}|$$
$$= \min_\alpha \{|1 - \alpha\mu|, |1 - \alpha L|\}$$
$$\alpha^*: \quad 1 - \alpha^*\mu = \alpha^* L - 1$$
$$\alpha^* = \frac{2}{\mu + L} \quad \rho^* = \frac{L - \mu}{L + \mu}$$
$$x^{k+1} = \left(\frac{L - \mu}{L + \mu}\right)^k x^0 \quad f(x^{k+1}) = \left(\frac{L - \mu}{L + \mu}\right)^{2k} f(x^0)$$

Convergence analysis

So, we have a linear convergence in the domain with rate $\frac{\kappa-1}{\kappa+1} = 1 - \frac{2}{\kappa+1}$, where $\kappa = \frac{L}{\mu}$ is sometimes called *condition number* of the quadratic problem.

κ	ρ	Iterations to decrease domain gap 10 times	Iterations to decrease function gap 10 times
1.1	0.05	1	1
2	0.33	3	2
5	0.67	6	3
10	0.82	12	6
50	0.96	58	29
100	0.98	116	58
500	0.996	576	288
1000	0.998	1152	576

Polyak-Lojasiewicz smooth case

Polyak-Lojasiewicz condition. Linear convergence of gradient descent without convexity

PL inequality holds if the following condition is satisfied for some $\mu > 0$,

$$\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f^*) \quad \forall x$$

It is interesting, that the Gradient Descent algorithm might converge linearly even without convexity.

The following functions satisfy the PL condition but are not convex. 🐍Link to the code

$$f(x) = x^2 + 3\sin^2(x)$$

Function, that satisfies
Polyak- Lojasiewicz condition

Polyak-Lojasiewicz condition. Linear convergence of gradient descent without convexity

PL inequality holds if the following condition is satisfied for some $\mu > 0$,

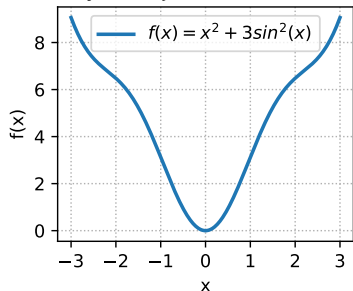$$\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f^*) \quad \forall x$$

It is interesting, that the Gradient Descent algorithm might converge linearly even without convexity.

The following functions satisfy the PL condition but are not convex. 🐍Link to the code

$$f(x) = x^2 + 3\sin^2(x)$$

$$f(x, y) = \frac{(y - \sin x)^2}{2}$$

Function, that satisfies
Polyak- Lojasiewicz condition

Non-convex PL function

Convergence analysis

> **i Theorem**
>
> Consider the Problem
>
> $$f(x) \to \min_{x \in \mathbb{R}^d}$$
>
> and assume that f is μ-Polyak-Lojasiewicz and L-smooth, for some $L \geq \mu > 0$.
> Consider $(x^k)_{k \in \mathbb{N}}$ a sequence generated by the gradient descent constant stepsize algorithm, with a stepsize satisfying $0 < \alpha \leq \frac{1}{L}$. Then:
>
> $$f(x^k) - f^* \leq (1 - \alpha\mu)^k (f(x^0) - f^*).$$

Smooth convex case

Smooth convex case

> **i** Theorem
>
> Consider the Problem
>
> $$f(x) \to \min_{x \in \mathbb{R}^d}$$
>
> and assume that f is convex and L-smooth, for some $L > 0$.
> Let $(x^k)_{k \in \mathbb{N}}$ be the sequence of iterates generated by the gradient descent constant stepsize algorithm, with a stepsize satisfying $0 < \alpha \le \frac{1}{L}$. Then, for all $x^* \in \operatorname{argmin} f$, for all $k \in \mathbb{N}$ we have that
>
> $$f(x^k) - f^* \le \frac{\|x^0 - x^*\|^2}{2\alpha k}.$$

Lower bounds

How optimal is $\mathcal{O}\left(\frac{1}{k}\right)$?

- Is it somehow possible to understand, that the obtained convergence is the fastest possible with this class of problem and this class of algorithms?

How optimal is $\mathcal{O}\left(\frac{1}{k}\right)$?

- Is it somehow possible to understand, that the obtained convergence is the fastest possible with this class of problem and this class of algorithms?
- The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\ \ \vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

How optimal is $\mathcal{O}\left(\frac{1}{k}\right)$?

- Is it somehow possible to understand, that the obtained convergence is the fastest possible with this class of problem and this class of algorithms?
- The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

- Consider a family of first-order methods, where

$$
x^{k+1} \in x^0 + \operatorname{span}\left\{\nabla f(x^0), \nabla f(x^1), \ldots, \nabla f(x^k)\right\} \tag{1}
$$

Smooth convex case

> **i** Theorem
>
> There exists a function f that is L-smooth and convex such that any method 2 satisfies
>
> $$\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(1+k)^2}$$

Smooth convex case

> **ℹ Theorem**
>
> There exists a function f that is L-smooth and convex such that any method 2 satisfies
>
> $$\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(1+k)^2}$$

- No matter what gradient method you provide, there is always a function f that, when you apply your gradient method on minimizing such f, the convergence rate is lower bounded as $\mathcal{O}\left(\frac{1}{k^2}\right)$.

Smooth convex case

> **i Theorem**
>
> There exists a function f that is L-smooth and convex such that any method 2 satisfies
>
> $$\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(1+k)^2}$$

- No matter what gradient method you provide, there is always a function f that, when you apply your gradient method on minimizing such f, the convergence rate is lower bounded as $\mathcal{O}\left(\frac{1}{k^2}\right)$.
- The key to the proof is to explicitly build a special function f.

Recap

Recap

$$\text{Gradient Descent:} \qquad \min_{x \in \mathbb{R}^n} f(x) \qquad x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$	$\|\nabla f(x^k)\|^2 \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$\|x^k - x^*\|^2 \sim \mathcal{O}\left(\left(1 - \dfrac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \dfrac{1}{\varepsilon}\right)$

Recap

$$\text{Gradient Descent:} \qquad \min_{x \in \mathbb{R}^n} f(x) \qquad x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$	$\|\nabla f(x^k)\|^2 \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$\|x^k - x^*\|^2 \sim \mathcal{O}\left(\left(1 - \dfrac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \dfrac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k \left(f(x^0) - f^*\right).$$

Note also, that for any x

$$1 - x \leq e^{-x}$$

Recap

Gradient Descent: $\qquad \min_{x \in \mathbb{R}^n} f(x) \qquad\qquad x^{k+1} = x^k - \alpha^k \nabla f(x^k)$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$	$\|\nabla f(x^k)\|^2 \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$\|x^k - x^*\|^2 \sim \mathcal{O}\left(\left(1 - \dfrac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \dfrac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x^0) - f^*).$$

Note also, that for any x

$$1 - x \leq e^{-x}$$

Finally we have

$$\varepsilon = f(x^{k_\varepsilon}) - f^* \leq \left(1 - \frac{\mu}{L}\right)^{k_\varepsilon} (f(x^0) - f^*)$$

$$\leq \exp\left(-k_\varepsilon \frac{\mu}{L}\right)(f(x^0) - f^*)$$

$$k_\varepsilon \geq \kappa \log \frac{f(x^0) - f^*}{\varepsilon} = \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$$

Recap

$$\text{Gradient Descent:} \qquad \min_{x \in \mathbb{R}^n} f(x) \qquad x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\|\nabla f(x^k)\|^2 \sim \mathcal{O}\left(\frac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$	$\|x^k - x^*\|^2 \sim \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x^0) - f^*).$$

Note also, that for any x

$$1 - x \leq e^{-x}$$

Finally we have

$$\varepsilon = f(x^{k_\varepsilon}) - f^* \leq \left(1 - \frac{\mu}{L}\right)^{k_\varepsilon} (f(x^0) - f^*)$$

$$\leq \exp\left(-k_\varepsilon \frac{\mu}{L}\right) (f(x^0) - f^*)$$

$$k_\varepsilon \geq \kappa \log \frac{f(x^0) - f^*}{\varepsilon} = \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$$

Question: Can we do faster, than this using the first-order information?

Recap

$$\text{Gradient Descent:} \qquad \min_{x \in \mathbb{R}^n} f(x) \qquad x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$	$\|\nabla f(x^k)\|^2 \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$\|x^k - x^*\|^2 \sim \mathcal{O}\left(\left(1 - \dfrac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \dfrac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k \left(f(x^0) - f^*\right).$$

Note also, that for any x

$$1 - x \leq e^{-x}$$

Finally we have

$$\varepsilon = f(x^{k_\varepsilon}) - f^* \leq \left(1 - \frac{\mu}{L}\right)^{k_\varepsilon} \left(f(x^0) - f^*\right)$$

$$\leq \exp\left(-k_\varepsilon \frac{\mu}{L}\right) \left(f(x^0) - f^*\right)$$

$$k_\varepsilon \geq \kappa \log \frac{f(x^0) - f^*}{\varepsilon} = \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$$

Question: Can we do faster, than this using the first-order information? **Yes, we can.**

Lower bounds

Lower bounds

convex (non-smooth)	smooth (non-convex)[2]	smooth & convex[3]	smooth & strongly convex (or PL)
$\mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$	$\mathcal{O}\left(\dfrac{1}{k^2}\right)$	$\mathcal{O}\left(\dfrac{1}{k^2}\right)$	$\mathcal{O}\left(\left(1-\sqrt{\dfrac{\mu}{L}}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\sqrt{\varepsilon}}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\sqrt{\varepsilon}}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\sqrt{\kappa}\log\dfrac{1}{\varepsilon}\right)$

---

[2]Carmon, Duchi, Hinder, Sidford, 2017
[3]Nemirovski, Yudin, 1979

Lower bounds

The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

Lower bounds

The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

Consider a family of first-order methods, where

$$
x^{k+1} \in x^0 + \mathsf{span} \left\{ \nabla f(x^0), \nabla f(x^1), \dots, \nabla f(x^k) \right\} \quad (2)
$$

Lower bounds

The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\ \ \vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

Consider a family of first-order methods, where

$$
x^{k+1} \in x^0 + \text{span}\left\{ \nabla f(x^0), \nabla f(x^1), \dots, \nabla f(x^k) \right\} \quad (2)
$$

> **i Non-smooth convex case**
>
> There exists a function f that is M-Lipschitz and convex such that any first-order method of the form 2 satisfies
>
> $$
> \min_{i \in [1,k]} f(x^i) - f^* \geq \frac{M \|x^0 - x^*\|_2}{2(1 + \sqrt{k})}
> $$

Lower bounds

The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\quad \vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

Consider a family of first-order methods, where

$$
x^{k+1} \in x^0 + \mathsf{span}\left\{ \nabla f(x^0), \nabla f(x^1), \ldots, \nabla f(x^k) \right\} \quad (2)
$$

---

i Non-smooth convex case

There exists a function f that is M-Lipschitz and convex such that any first-order method of the form 2 satisfies

$$
\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{M\|x^0 - x^*\|_2}{2(1 + \sqrt{k})}
$$
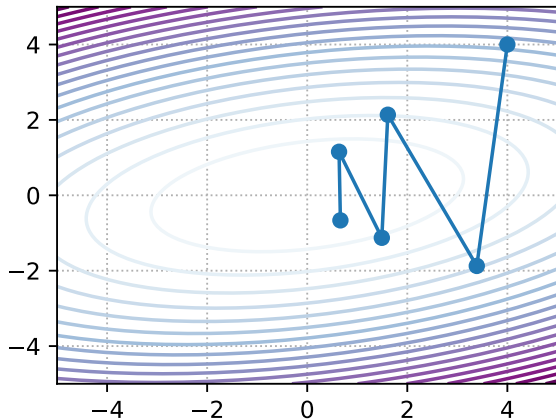
---

i Smooth and convex case

There exists a function f that is L-smooth and convex such that any first-order method of the form 2 satisfies

$$
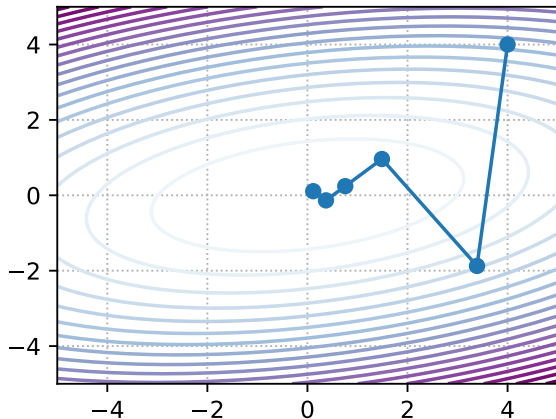\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(1 + k)^2}
$$

Strongly convex quadratic problem

Oscillations and acceleration

Gradient Descent   Heavy Ball

Coordinate shift

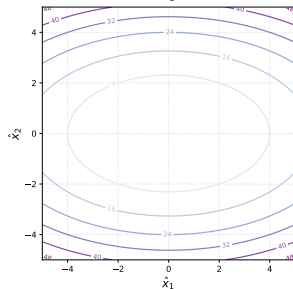Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}_{++}^d.$$

Coordinate shift

Consider the following quadratic optimization problem:
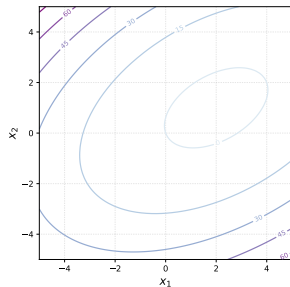
$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}_{++}^d.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.

Coordinate shift

Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}_{++}^d.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.
- Secondly, we have a spectral decomposition of the matrix A:
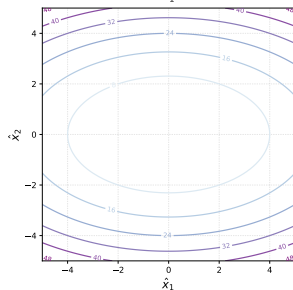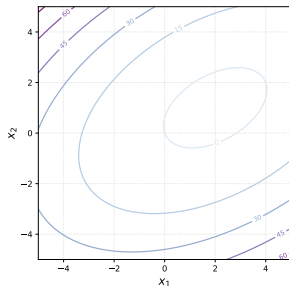
$$A = Q \Lambda Q^T$$

Coordinate shift

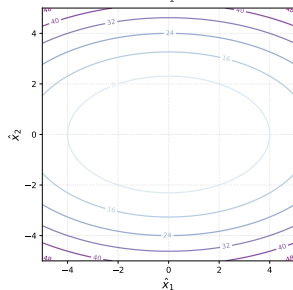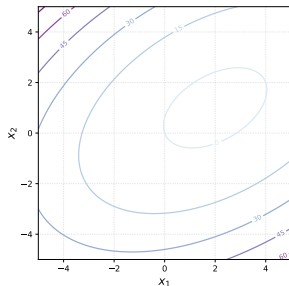Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.

- Secondly, we have a spectral decomposition of the matrix A:

$$A = Q \Lambda Q^T$$

- Let's show, that we can switch coordinates in order to make an analysis a little bit easier. Let $\hat{x} = Q^T(x - x^*)$, where x^* is the minimum point of initial function, defined by $Ax^* = b$. At the same time $x = Q\hat{x} + x^*$.

$$
\begin{aligned}
f(\hat{x}) &= \frac{1}{2}(Q\hat{x} + x^*)^\top A(Q\hat{x} + x^*) - b^\top(Q\hat{x} + x^*) \\
&= \frac{1}{2}\hat{x}^T Q^T A Q \hat{x} + (x^*)^T A Q \hat{x} + \frac{1}{2}(x^*)^T A (x^*)^T - b^T Q \hat{x} - b^T x^* \\
&= \frac{1}{2}\hat{x}^T \Lambda \hat{x}
\end{aligned}
$$

Heavy ball

Polyak Heavy ball method

Trajectories with Contour Plot

Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

$$x^{k+1} = x^k - \alpha \nabla f(x^k) + \beta(x^k - x_{k-1}).$$

Polyak Heavy ball method

Trajectories with Contour Plot

Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

$$x^{k+1} = x^k - \alpha \nabla f(x^k) + \beta(x^k - x_{k-1}).$$

Which is in our (quadratics) case is

$$\hat{x}_{k+1} = \hat{x}_k - \alpha \Lambda \hat{x}_k + \beta(\hat{x}_k - \hat{x}_{k-1}) = (I - \alpha \Lambda + \beta I)\hat{x}_k - \beta \hat{x}_{k-1}$$

Trajectories with Contour Plot

Polyak Heavy ball method

Trajectories with Contour Plot
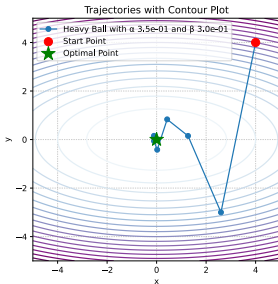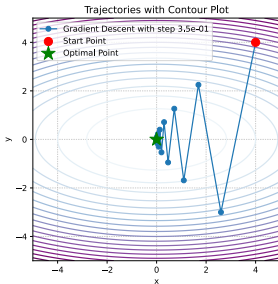
Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

$$x^{k+1} = x^k - \alpha \nabla f(x^k) + \beta(x^k - x_{k-1}).$$

Which is in our (quadratics) case is

$$\hat{x}_{k+1} = \hat{x}_k - \alpha \Lambda \hat{x}_k + \beta(\hat{x}_k - \hat{x}_{k-1}) = (I - \alpha \Lambda + \beta I)\hat{x}_k - \beta \hat{x}_{k-1}$$

This can be rewritten as follows

$$\hat{x}_{k+1} = (I - \alpha \Lambda + \beta I)\hat{x}_k - \beta \hat{x}_{k-1},$$
$$\hat{x}_k = \hat{x}_k.$$

Polyak Heavy ball method

Trajectories with Contour Plot

Trajectories with Contour Plot

Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

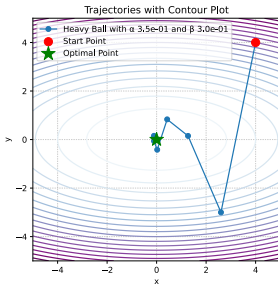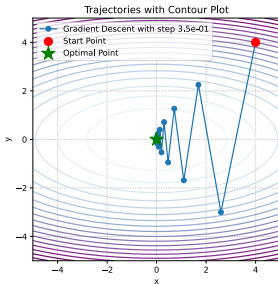$$x^{k+1} = x^k - \alpha \nabla f(x^k) + \beta(x^k - x_{k-1}).$$

Which is in our (quadratics) case is

$$\hat{x}_{k+1} = \hat{x}_k - \alpha \Lambda \hat{x}_k + \beta(\hat{x}_k - \hat{x}_{k-1}) = (I - \alpha \Lambda + \beta I)\hat{x}_k - \beta \hat{x}_{k-1}$$

This can be rewritten as follows

$$\hat{x}_{k+1} = (I - \alpha \Lambda + \beta I)\hat{x}_k - \beta \hat{x}_{k-1},$$
$$\hat{x}_k = \hat{x}_k.$$

Let's use the following notation $\hat{z}_k = \begin{bmatrix} \hat{x}_{k+1} \\ \hat{x}_k \end{bmatrix}$. Therefore $\hat{z}_{k+1} = M\hat{z}_k$, where the iteration matrix M is:

Polyak Heavy ball method

Trajectories with Contour Plot

Trajectories with Contour Plot

Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

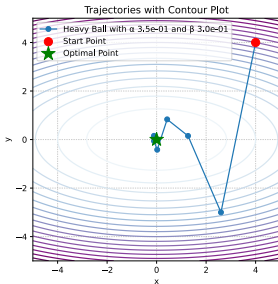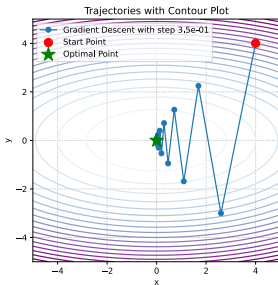$$x^{k+1} = x^k - \alpha \nabla f(x^k) + \beta(x^k - x_{k-1}).$$
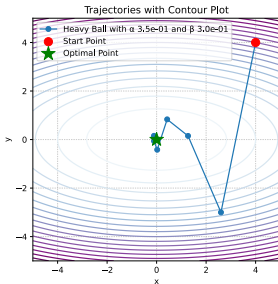
Which is in our (quadratics) case is

$$\hat{x}_{k+1} = \hat{x}_k - \alpha \Lambda \hat{x}_k + \beta(\hat{x}_k - \hat{x}_{k-1}) = (I - \alpha \Lambda + \beta I)\hat{x}_k - \beta \hat{x}_{k-1}$$

This can be rewritten as follows

$$\hat{x}_{k+1} = (I - \alpha \Lambda + \beta I)\hat{x}_k - \beta \hat{x}_{k-1},$$
$$\hat{x}_k = \hat{x}_k.$$

Let's use the following notation $\hat{z}_k = \begin{bmatrix} \hat{x}_{k+1} \\ \hat{x}_k \end{bmatrix}$. Therefore $\hat{z}_{k+1} = M\hat{z}_k$, where the iteration matrix M is:

$$M = \begin{bmatrix} I - \alpha \Lambda + \beta I & -\beta I \\ I & 0_d \end{bmatrix}.$$

Reduction to a scalar case

Note, that M is $2d \times 2d$ matrix with 4 block-diagonal matrices of size $d \times d$ inside. It means, that we can rearrange the order of coordinates to make M block-diagonal in the following form. Note that in the equation below, the matrix M denotes the same as in the notation above, except for the described permutation of rows and columns. We use this slight abuse of notation for the sake of clarity.

Reduction to a scalar case

Note, that M is $2d \times 2d$ matrix with 4 block-diagonal matrices of size $d \times d$ inside. It means, that we can rearrange the order of coordinates to make M block-diagonal in the following form. Note that in the equation below, the matrix M denotes the same as in the notation above, except for the described permutation of rows and columns. We use this slight abuse of notation for the sake of clarity.

Figure 22: Illustration of matrix M rearrangement

$$
\begin{bmatrix} \hat{x}_k^{(1)} \\ \vdots \\ \hat{x}_k^{(d)} \\ \hat{x}_{k-1}^{(1)} \\ \vdots \\ \hat{x}_{k-1}^{(d)} \end{bmatrix} \rightarrow \begin{bmatrix} \hat{x}_k^{(1)} \\ \hat{x}_{k-1}^{(1)} \\ \vdots \\ \hat{x}_k^{(d)} \\ \hat{x}_{k-1}^{(d)} \end{bmatrix} \qquad M = \begin{bmatrix} M_1 & & & \\ & M_2 & & \\ & & \cdots & \\ & & & M_d \end{bmatrix}
$$

where $\hat{x}_k^{(i)}$ is i-th coordinate of vector $\hat{x}_k \in \mathbb{R}^d$ and M_i stands for 2×2 matrix. This rearrangement allows us to study the dynamics of the method independently for each dimension. One may observe, that the asymptotic convergence rate of the $2d$-dimensional vector sequence of $\hat{z}_k$ is defined by the worst convergence rate among its block of coordinates. Thus, it is enough to study the optimization in a one-dimensional case.

Reduction to a scalar case

For i-th coordinate with λ_i as an i-th eigenvalue of matrix W we have:

$$M_i = \begin{bmatrix} 1 - \alpha\lambda_i + \beta & -\beta \\ 1 & 0 \end{bmatrix}.$$

Reduction to a scalar case

For i-th coordinate with λ_i as an i-th eigenvalue of matrix W we have:

$$M_i = \begin{bmatrix} 1 - \alpha\lambda_i + \beta & -\beta \\ 1 & 0 \end{bmatrix}.$$

The method will be convergent if $\rho(M) < 1$, and the optimal parameters can be computed by optimizing the spectral radius

$$\alpha^*, \beta^* = \arg\min_{\alpha, \beta} \max_{\lambda \in [\mu, L]} \rho(M) \quad \alpha^* = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2}; \quad \beta^* = \left(\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}} \right)^2.$$

Reduction to a scalar case

For i-th coordinate with λ_i as an i-th eigenvalue of matrix W we have:

$$M_i = \begin{bmatrix} 1 - \alpha\lambda_i + \beta & -\beta \\ 1 & 0 \end{bmatrix}.$$

The method will be convergent if $\rho(M) < 1$, and the optimal parameters can be computed by optimizing the spectral radius

$$\alpha^*, \beta^* = \arg\min_{\alpha,\beta} \max_{\lambda \in [\mu, L]} \rho(M) \quad \alpha^* = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2}; \quad \beta^* = \left(\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}\right)^2.$$

It can be shown, that for such parameters the matrix M has complex eigenvalues, which forms a conjugate pair, so the distance to the optimum (in this case, $\|z_k\|$), generally, will not go to zero monotonically.

Heavy ball quadratic convergence

We can explicitly calculate the eigenvalues of M_i:

$$\lambda_1^M, \lambda_2^M = \lambda \left(\begin{bmatrix} 1 - \alpha\lambda_i + \beta & -\beta \\ 1 & 0 \end{bmatrix} \right) = \frac{1 + \beta - \alpha\lambda_i \pm \sqrt{(1 + \beta - \alpha\lambda_i)^2 - 4\beta}}{2}.$$

Heavy ball quadratic convergence

We can explicitly calculate the eigenvalues of M_i:

$$\lambda_1^M, \lambda_2^M = \lambda \left(\begin{bmatrix} 1 - \alpha\lambda_i + \beta & -\beta \\ 1 & 0 \end{bmatrix} \right) = \frac{1 + \beta - \alpha\lambda_i \pm \sqrt{(1 + \beta - \alpha\lambda_i)^2 - 4\beta}}{2}.$$

When α and β are optimal (α^*, β^*), the eigenvalues are complex-conjugated pair $(1 + \beta - \alpha\lambda_i)^2 - 4\beta \leq 0$, i.e. $\beta \geq (1 - \sqrt{\alpha\lambda_i})^2$.

Heavy ball quadratic convergence

We can explicitly calculate the eigenvalues of M_i:

$$\lambda_1^M, \lambda_2^M = \lambda\left(\begin{bmatrix} 1 - \alpha\lambda_i + \beta & -\beta \\ 1 & 0 \end{bmatrix}\right) = \frac{1 + \beta - \alpha\lambda_i \pm \sqrt{(1 + \beta - \alpha\lambda_i)^2 - 4\beta}}{2}.$$

When α and β are optimal (α^*, β^*), the eigenvalues are complex-conjugated pair $(1 + \beta - \alpha\lambda_i)^2 - 4\beta \leq 0$, i.e. $\beta \geq (1 - \sqrt{\alpha\lambda_i})^2$.

$$\mathsf{Re}(\lambda_1^M) = \frac{L + \mu - 2\lambda_i}{(\sqrt{L} + \sqrt{\mu})^2}; \quad \mathsf{Im}(\lambda_1^M) = \frac{\pm 2\sqrt{(L - \lambda_i)(\lambda_i - \mu)}}{(\sqrt{L} + \sqrt{\mu})^2}; \quad |\lambda_1^M| = \frac{L - \mu}{(\sqrt{L} + \sqrt{\mu})^2}.$$

Heavy ball quadratic convergence

We can explicitly calculate the eigenvalues of M_i:

$$\lambda_1^M, \lambda_2^M = \lambda \left(\begin{bmatrix} 1 - \alpha\lambda_i + \beta & -\beta \\ 1 & 0 \end{bmatrix} \right) = \frac{1 + \beta - \alpha\lambda_i \pm \sqrt{(1 + \beta - \alpha\lambda_i)^2 - 4\beta}}{2}.$$

When α and β are optimal (α^*, β^*), the eigenvalues are complex-conjugated pair $(1 + \beta - \alpha\lambda_i)^2 - 4\beta \leq 0$, i.e. $\beta \geq (1 - \sqrt{\alpha\lambda_i})^2$.

$$\mathsf{Re}(\lambda_1^M) = \frac{L + \mu - 2\lambda_i}{(\sqrt{L} + \sqrt{\mu})^2}; \quad \mathsf{Im}(\lambda_1^M) = \frac{\pm 2\sqrt{(L - \lambda_i)(\lambda_i - \mu)}}{(\sqrt{L} + \sqrt{\mu})^2}; \quad |\lambda_1^M| = \frac{L - \mu}{(\sqrt{L} + \sqrt{\mu})^2}.$$

And the convergence rate does not depend on the stepsize and equals to $\sqrt{\beta^*}$.

Heavy Ball quadratics convergence

> **i** Theorem
>
> Assume that f is quadratic μ-strongly convex L-smooth quadratics, then Heavy Ball method with parameters
>
> $$\alpha = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2}, \beta = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$$
>
> converges linearly:
>
> $$\|x_k - x^*\|_2 \leq \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right) \|x_0 - x^*\|$$

Heavy Ball Global Convergence [4]

> **ℹ Theorem**
>
> Assume that f is smooth and convex and that
>
> $$\beta \in [0, 1), \quad \alpha \in \left(0, \frac{2(1-\beta)}{L}\right).$$
>
> Then, the sequence $\{x_k\}$ generated by Heavy-ball iteration satisfies
>
> $$f(\overline{x}_T) - f^\star \leq \begin{cases} \frac{\|x_0 - x^\star\|^2}{2(T+1)}\left(\frac{L\beta}{1-\beta} + \frac{1-\beta}{\alpha}\right), & \text{if } \alpha \in \left(0, \frac{1-\beta}{L}\right], \\ \frac{\|x_0 - x^\star\|^2}{2(T+1)(2(1-\beta)-\alpha L)}\left(L\beta + \frac{(1-\beta)^2}{\alpha}\right), & \text{if } \alpha \in \left[\frac{1-\beta}{L}, \frac{2(1-\beta)}{L}\right), \end{cases}$$
>
> where $\overline{x}_T$ is the Cesaro average of the iterates, i.e.,
>
> $$\overline{x}_T = \frac{1}{T+1}\sum_{k=0}^{T} x_k.$$

---

[4] Global convergence of the Heavy-ball method for convex optimization, Euhanna Ghadimi et.al.

Heavy Ball Global Convergence [5]

> **i** Theorem
>
> Assume that f is smooth and strongly convex and that
>
> $$\alpha \in (0, \frac{2}{L}), \quad 0 \leq \beta < \frac{1}{2}\left(\frac{\mu\alpha}{2} + \sqrt{\frac{\mu^2\alpha^2}{4} + 4(1 - \frac{\alpha L}{2})} \right).$$
>
> where $\alpha_0 \in (0, 1/L]$. Then, the sequence $\{x_k\}$ generated by Heavy-ball iteration converges linearly to a unique optimizer $x^\star$. In particular,
>
> $$f(x_k) - f^\star \leq q^k(f(x_0) - f^\star),$$
>
> where $q \in [0, 1)$.

---

[5] Global convergence of the Heavy-ball method for convex optimization, Euhanna Ghadimi et.al.

Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems

Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.

Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.
- Recently was proved, that there is no global accelerated convergence for the method.

Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.
- Recently was proved, that there is no global accelerated convergence for the method.
- Method was not extremely popular until the ML boom

Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.
- Recently was proved, that there is no global accelerated convergence for the method.
- Method was not extremely popular until the ML boom
- Nowadays, it is de-facto standard for practical acceleration of gradient methods, even for the non-convex problems (neural network training)

Nesterov accelerated gradient

The concept of Nesterov Accelerated Gradient method

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \qquad x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}) \qquad \begin{cases} y_{k+1} = x_k + \beta(x_k - x_{k-1}) \\ x_{k+1} = y_{k+1} - \alpha \nabla f(y_{k+1}) \end{cases}$$

The concept of Nesterov Accelerated Gradient method

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \qquad x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}) \qquad \begin{cases} y_{k+1} = x_k + \beta(x_k - x_{k-1}) \\ x_{k+1} = y_{k+1} - \alpha \nabla f(y_{k+1}) \end{cases}$$

Let's define the following notation

$$x^+ = x - \alpha \nabla f(x) \qquad \text{Gradient step}$$
$$d_k = \beta_k(x_k - x_{k-1}) \qquad \text{Momentum term}$$

Then we can write down:

$$x_{k+1} = x_k^+ \qquad \text{Gradient Descent}$$
$$x_{k+1} = x_k^+ + d_k \qquad \text{Heavy Ball}$$
$$x_{k+1} = (x_k + d_k)^+ \qquad \text{Nesterov accelerated gradient}$$

NAG convergence for quadratics

General case convergence

> **ℹ Theorem**
>
> Let $f : \mathbb{R}^n \to \mathbb{R}$ is convex and L-smooth. The Nesterov Accelerated Gradient Descent (NAG) algorithm is designed to solve the minimization problem starting with an initial point $x_0 = y_0 \in \mathbb{R}^n$ and $\lambda_0 = 0$. The algorithm iterates the following steps:
>
> $$\textbf{Gradient update:} \qquad y_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$$
>
> $$\textbf{Extrapolation:} \qquad x_{k+1} = (1 - \gamma_k)y_{k+1} + \gamma_k y_k$$
>
> $$\textbf{Extrapolation weight:} \qquad \lambda_{k+1} = \frac{1 + \sqrt{1 + 4\lambda_k^2}}{2}$$
>
> $$\textbf{Extrapolation weight:} \qquad \gamma_k = \frac{1 - \lambda_k}{\lambda_{k+1}}$$
>
> The sequences $\{f(y_k)\}_{k\in\mathbb{N}}$ produced by the algorithm will converge to the optimal value f^* at the rate of $\mathcal{O}\left(\frac{1}{k^2}\right)$, specifically:
>
> $$f(y_k) - f^* \leq \frac{2L\|x_0 - x^*\|^2}{k^2}$$

General case convergence

> **i** Theorem
>
> Let $f : \mathbb{R}^n \to \mathbb{R}$ is μ-strongly convex and L-smooth. The Nesterov Accelerated Gradient Descent (NAG) algorithm is designed to solve the minimization problem starting with an initial point $x_0 = y_0 \in \mathbb{R}^n$ and $\lambda_0 = 0$. The algorithm iterates the following steps:
>
> $$\text{Gradient update:} \qquad y_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$$
>
> $$\text{Extrapolation:} \qquad x_{k+1} = (1 - \gamma_k)y_{k+1} + \gamma_k y_k$$
>
> $$\text{Extrapolation weight:} \qquad \gamma_k = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$$
>
> The sequences $\{f(y_k)\}_{k \in \mathbb{N}}$ produced by the algorithm will converge to the optimal value f^* linearly:
>
> $$f(y_k) - f^* \leq \frac{\mu + L}{2}\|x_0 - x^*\|_2^2 \exp\left(-\frac{k}{\sqrt{\kappa}}\right)$$