Lecture 6

Numerical methods for solving optimization problems

Subgradient descent method

The subgradient descent method is an extension of the traditional gradient descent method.

Vector $g \in \mathbb{R}^n$ called the **subgradient** of the function $f(x)$ in point $x_0 \in \mathbb{R}^n$, if:

$$f(x) - f(x_0) \geq \langle g, x - x_0 \rangle, \forall x \epsilon \mathbb{R}^n$$

Set of subgradients of a function $f(x)$ in point x_0 called a subdifferential. This set is limited and closed.

Subgradient descent method

Consider a problem of the form:

$$f(x) \to min, x \epsilon \mathbb{R}^n$$

The function f(x) must be convex and may be non-differentiable. There must also be some constant $M > 0$: $\forall x \epsilon \mathbb{R}^n$ and $\forall g(x) \in \partial f(x)$

$$\|g(x)\|_2 \leq M.$$

For differentiable functions, the gradient coincides with the subgradient. The difference between this method and the previous one is that in this method a subgradient is selected as the direction of descent, that is, the transition from the previous point to the next one is also carried out according to the rule:

$$x^{k+1} = x^k - \alpha g^k$$

Proximal gradient method

This method is a generalization of projective gradient methods that are used to solve non-differentiable convex programming problems. This method is used to solve problems with regularization of the form:

$$F(x) = f(x) + R(x) \to \min_{x \epsilon \mathbb{R}^n}$$

where $f(x)$: $\mathbb{R}^n \to R - L -$ smooth convex function; $R(x)$: $\mathbb{R}^n \to R \cup \{+\infty\} -$ regular convex closed function. Also for R(x) we will require the ability to calculate the proximal operator.

Proximal gradient method

Let the function $R(x)$: $\mathbb{R}^n \to R \cup \{+\infty\} -$ is a regular convex closed function, then we call the mapping from $\mathbb{R}^n$ in $\mathbb{R}^n$:

$$prox_R(x) = arg \min_{y \epsilon \mathbb{R}^n} \left\{ R(y) + \frac{1}{2} \|y - x\|_2^2 \right\}$$

Thus, an algorithm for the proximal gradient method can be formulated. This algorithm will be similar to the gradient descent method algorithm, only now the calculation of the proximal operator will be added for each step.

Proximal gradient method

1. Set the step size $\gamma > 0$, starting point x^0 and number of iterations *N*.
2. In a loop for k from 0 to $N - 1$:
    1. Find $\nabla f(x^k)$
    2. Moving on to the next point $x^{k+1} = prox_{\gamma R}(x^k - \gamma \nabla f(x^k))$

As a result of executing this algorithm, we obtain the point x^N, delivering the minimum objective function.

Theorem. Let $f(x) -$ $L -$ smooth convex function, $R(x)$: $\mathbb{R}^n \to R \cup \{+\infty\} -$ regular convex closed function, $\gamma = \frac{1}{L}$. Then for any N>0

$$f(x^N) - f(x^*) \leq \frac{L\|x^0 - x^*\|_2^2}{2N}$$

Proximal gradient method

This method for convex functions can be accelerated. In the accelerated gradient method (FISTA), the algorithm is as follows:

1. Set the starting point x^0 and number of iterations N. Assume $y^0 = x^0, t_0 = 1$.

2. In a loop for k from 0 to $N - 1$:

    1. Calculate $\nabla f(y^k)$

    2. Moving on to the next point $x^{k+1} = prox_{\frac{1}{L}R}\left(y^k - \frac{1}{L}\nabla f(y^k)\right)$

    3. $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$

    4. $y^{k+1} = x^{k+1} + \frac{t_k - 1}{t_{k+1}}(x^{k+1} - x^k)$

Stochastic gradient descent (SGD) method

The stochastic gradient descent method differs from the gradient descent method by using the function instead of the gradient $g(x, \theta)$, such that its mathematical expectation by random variable θ $E_\theta(x, \theta) = \nabla f(x)$. That is, the transition from the previous point to the next one is carried out according to the rule:

$$x^{k+1} = x^k - \alpha g(x^k, \theta_k)$$

where θ_k – some random parameter that we do not influence, but on average we go in the direction opposite to the direction of the gradient vector.

The penalty function method

Consider an optimization problem of the form:

$$\begin{cases} f(x) \to min \\ g_i(x) \leq 0, i = 1, \dots, m \end{cases}$$

The main idea of the **penalty function method** is to move from a constrained optimization problem to an unconstrained optimization problem by changing the original objective function due to restrictions. This method proposes adding a certain penalty to the objective function, which will increase as it moves away from the minimum point. Initially, this penalty is small or equal to zero if the restrictions are met (or satisfied to within ε).

The penalty function method

Let's move on to the unconstrained optimization problem with an added penalty:

$$F(x, r) = f(x) + \sum_{i=1}^{m} P_i(x, r)$$

where the function $P_i, i = 1, \dots, m$ is responsible for the penalty for the i-th constraint, and the coefficient r regulates the size of the penalty.

The penalty function method

The following types of fines are distinguished:

1. The quadratic penalty is used to take into account equality constraints:

$$P_i\left(x, r^k\right) = r^k (g_i(x))^2$$

Note that the penalty is zero if the desired point satisfies the constraints of the problem.

2. Fine type cut square:

$$P_i\left(x, r^k\right) = r^k (g_i^+(x))^2$$

where $g_i^+(x) = max\{0, g_i(x)\}^2 = \begin{cases} g_i(x), if\ g_i(x) > 0 \\ 0, if\ g_i(x) \leq 0 \end{cases}$

The penalty function method

Example.

$$f(x) = x_1^2 + x_2^2 \to min$$
$$x_1 + x_2 = 1$$

The quadratic penalty is used to take into account equality constraints:

$$P_i\left(x, r^k\right) = r^k (g_i(x))^2$$

We can use a quadratic penalty: $P_i\left(x, r^k\right) = r^2 (x_1 + x_2 - 1\ )^2$

The penalty function method

Now our new objective function will look like this:

$$F(x) = x_1^2 + x_2^2 + r^2(x_1 + x_2 - 1\ )^2$$

Our goal now is to minimize $F(x)$ without considering the constraints.

The penalty function method

Let us formulate the algorithm of the penalty function method:

1. Create a penalty function and an objective function F. Set the penalty size r (for example, 0.01), the initial reference point x^0, and calculate the partial derivatives at this point for each argument. The starting point, as a rule, is taken such that the restrictions are violated (therefore, this method is sometimes called the external point method).

2. Using any numerical method for solving the unconditional minimization problem, find the minimum x^i of the function F.

The penalty function method

3. If the penalty value is less than the specified accuracy, then write x^i as the answer, otherwise, increase the penalty several times (optimally increase from 4 to 10 times) and go to step 2.

The method of barrier functions

Let's consider the **method of barrier functions**.

This method, like the method of penalty functions, reduces the problem of conditional minimization to the problem of unconditional minimization. This occurs by adding an auxiliary function formed from constraints to the objective function. **This function is a barrier that does not allow you to go beyond the permissible area**. Therefore, the starting point must also be selected from the feasible region. As the value approaches the barrier, the value of the objective function should increase. **At the barrier itself, as a rule, it tends to infinity.**

Method of penalty and barrier functions

The following types of barrier functions are distinguished:

1. Logarithmic penalty:

$$P_i\left(x, r^k\right) = -r^k ln\left(g_i^+(x)\right)$$

This function is undefined at invalid points.

2. Inverse function:

$$P_i\left(x, r^k\right) = -r^k \frac{1}{g_i(x)}$$

This function loses its properties if restrictions are violated.

Method of penalty and barrier functions

Example.

$$f(x) = x_1^2 + x_2^2 \to min$$
$$g_1(x) = x_1 + x_2 - 1 \leq 0$$
$$g_2(x) = -x_1 \leq 0$$
$$g_3(x) = -x_2 \leq 0$$

To apply the barrier method, we introduce a barrier function that tends to infinity as it approaches the boundary of the feasible set. For instance, we can use a logarithmic barrier function:

$$P_i\left(x, r^k\right) = -r^k ln\left(g_i^+(x)\right)$$

Method of penalty and barrier functions

$$P\left(x, r^k\right) = -r^k(ln(g_1(x)) + \ln(g_2(x)) + \ln(g_3(x)))$$

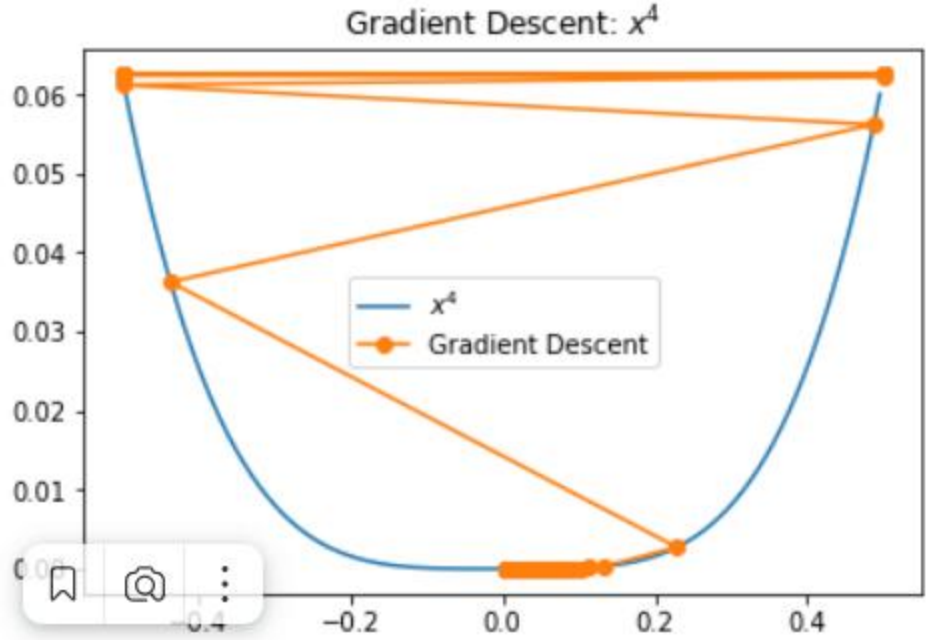
Then the new objective function becomes:

$$F(x) = x_1^2 + x_2^2 + P\left(x, r^k\right)$$

Now the problem reduces to minimizing $F(x)$ as the parameter k increases. As k grows larger, the influence of the barrier function diminishes, allowing us to approach the boundary of admissible values but not cross it.

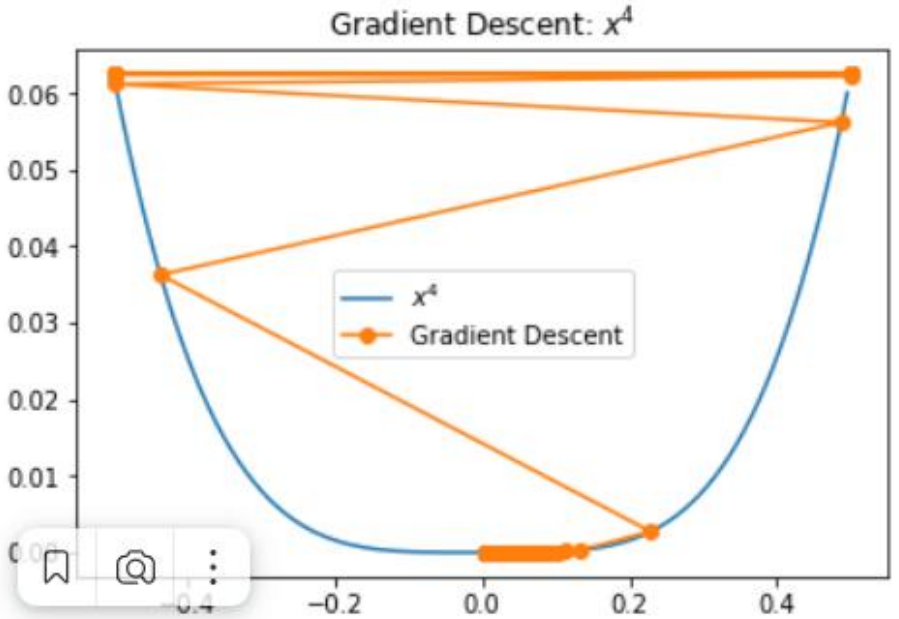
The heavy ball method. Example

The function $f(x) = x^4$ has very different curvature in different regions of data: we will see that a large learning rate leads to bigger steps than we would like in high curvature regions, while a small learning rate leads to smaller steps than we would like in low curvature regions.



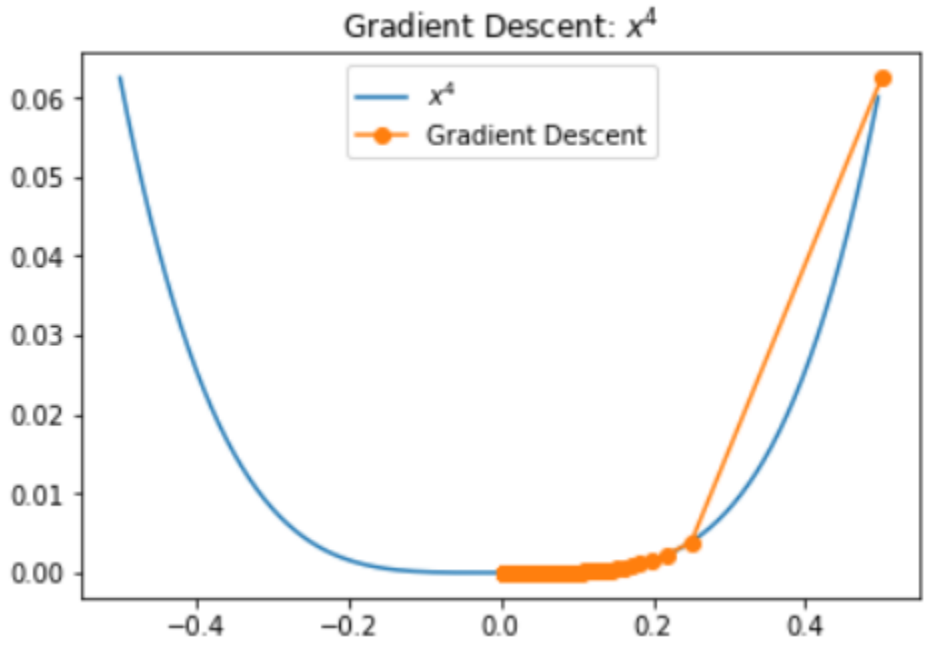
The heavy ball method. Example

We see that we bounce along the walls of the function for some iterations, making bigger steps than we would like. While in the particular case the amount of bouncing is not large, this bouncing across walls in regions of high curvature is a common issue for gradient descent in higher dimensions.



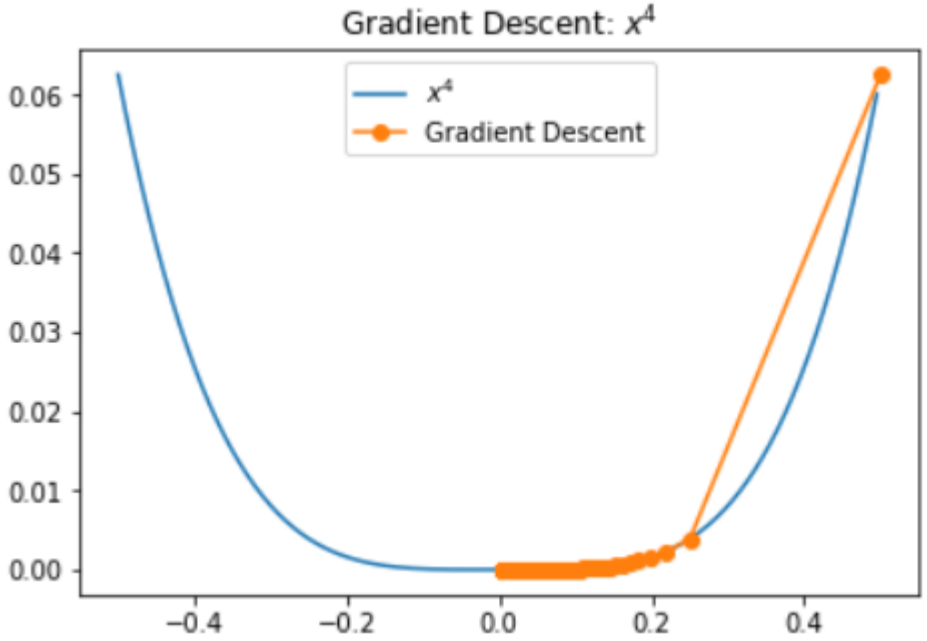
The heavy ball method. Example

We could potentially reduce this bouncing by lowering the learning rate. If we set it to $\alpha = 0.5$ we obtain



The heavy ball method. Example

We no longer bounce along the walls and make fast progress in the region of high curvature, but then in the region of low curvature we drastically increase the number of iterations required due to very small steps towards the optimum.



The heavy ball method. Example

Based on the above results, we want two things:

- To make smaller steps in regions of high curvature to dampen oscillations.
- To make larger steps and accelerate in regions of low curvature.

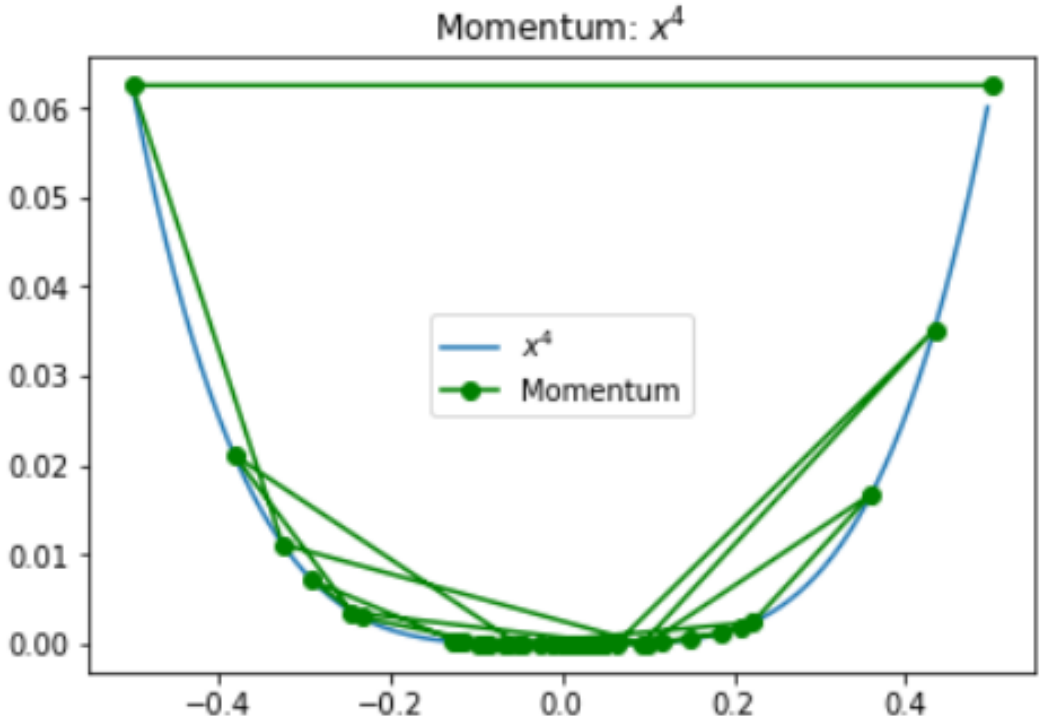
One way to do both is to guide the next steps towards the previous direction. We can achieve this via a simple tweak to the gradient descent update rule:

$$x_{t+1} = x_t - \alpha \nabla f(x_t) + \beta(x_t - x_{t-1})$$

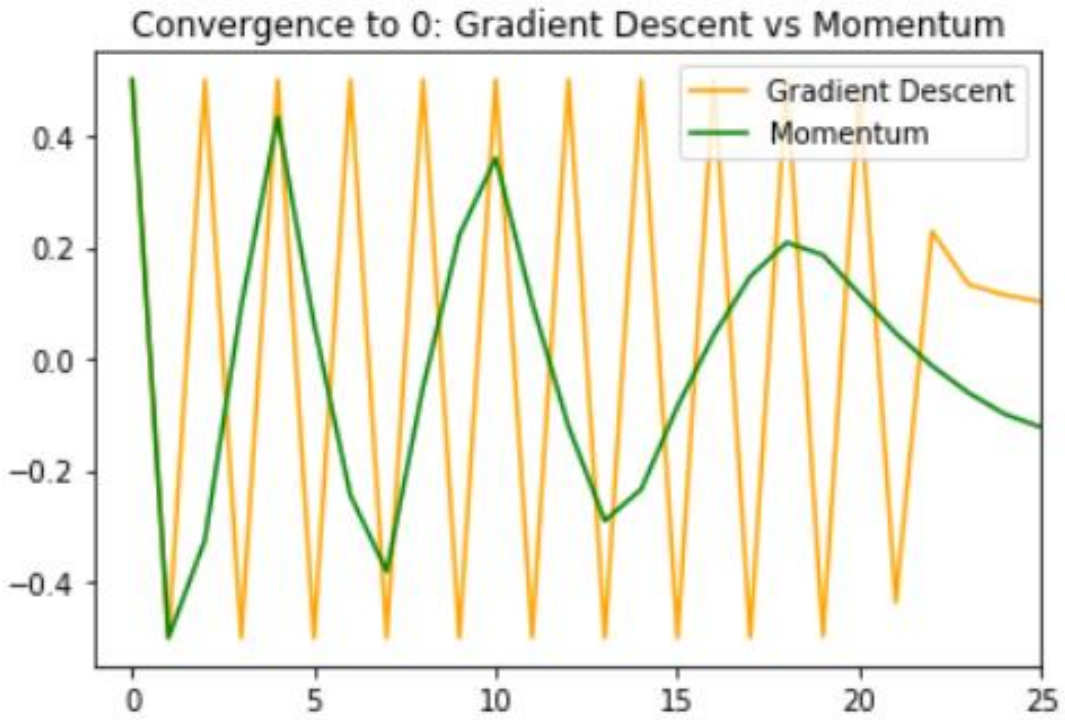
The heavy ball method. Example

This idea comes from Polyak and is also called **the heavy ball method**. Intuitively, a heavier ball will bounce less and move faster through regions of low curvature than a lighter ball due to **momentum**.
Let's set $\beta = 0.825$.

The heavy ball method. Example

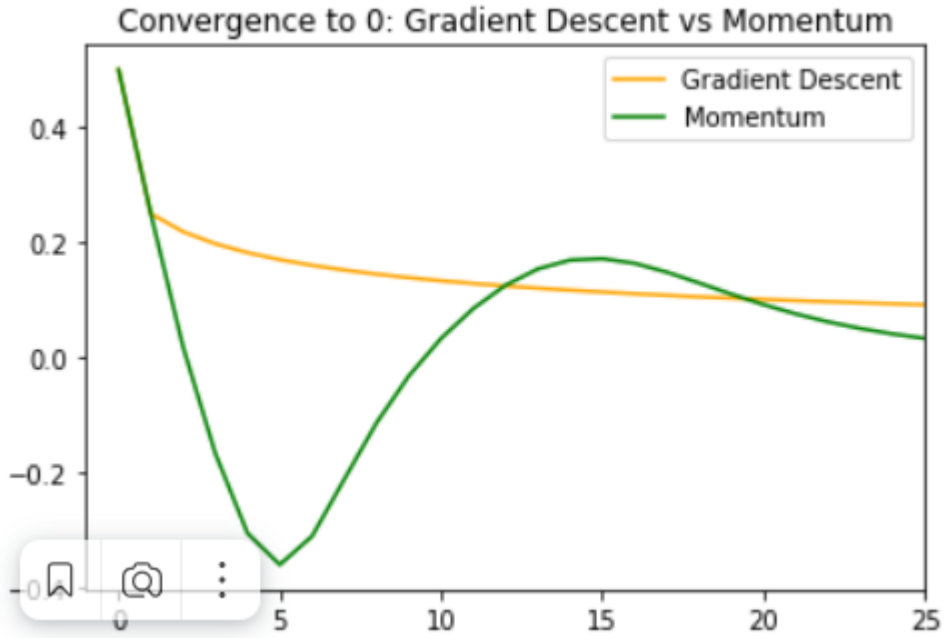


The heavy ball method. Example



The heavy ball method. Example

Next we can look at the example with the smaller learning rate of $\alpha = 0.5$. We set $\beta = 0.8$. We can make the same 'zoomed in' convergence to 0 plot



The heavy ball method. Example

Algorithm 1. Heavy-Ball method (HB)

Input: starting points x_0, x_1 (by default $x_0 = x_1$), number of iterations N, stepsize $\alpha > 0$, momentum parameter $\beta \in [0, 1]$

1: **for** $k = 0, \ldots, N-1$ **do**

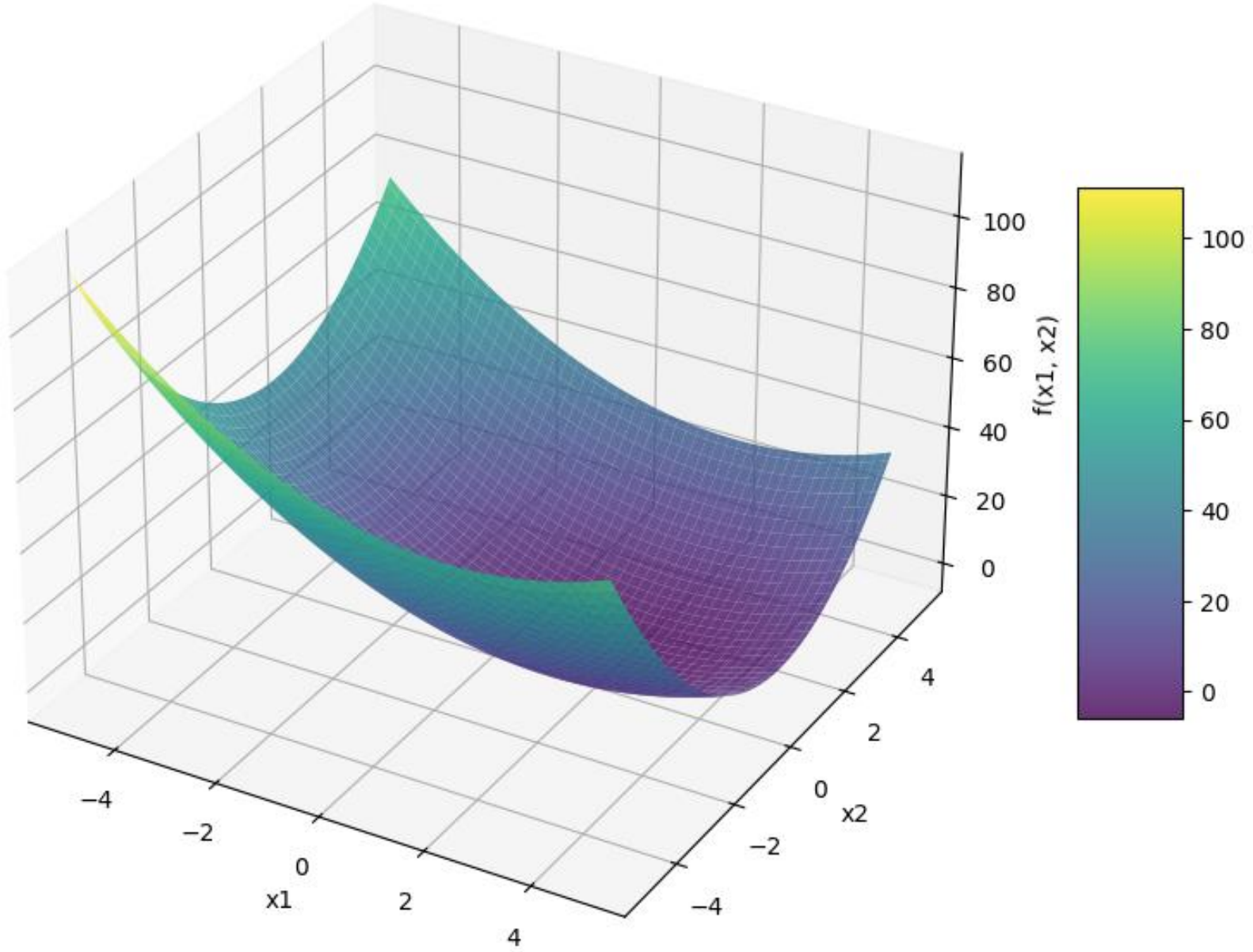
2: $\quad x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k+1})$

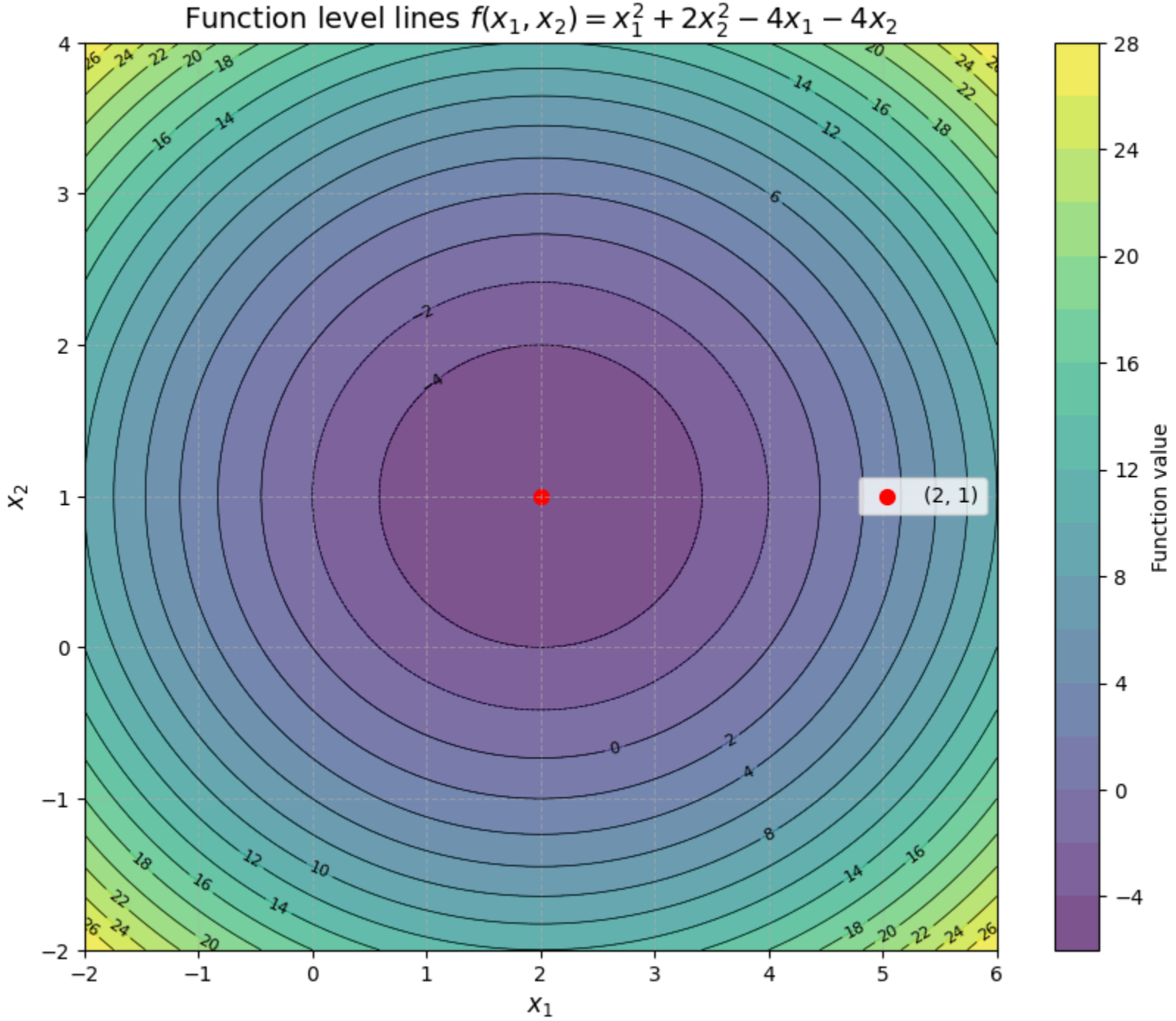
3: **end for**

Output: x_k

The heavy ball method. Example

Example. Find the minimal value of the function $f(x_1, x_2) = x_1^2 + 2x_2^2 - 4x_1 - 4x_2$.


100
80
60
40
20
0
f(x1, x2)
100
80
60
40
20
0
−4
−2
0
2
4
x1
4
2
0
−2
−4
x2


Function level lines $f(x_1, x_2) = x_1^2 + 2x_2^2 - 4x_1 - 4x_2$
x_1
x_2
(2, 1)
Function value

The heavy ball method. Example

Example. Find the minimal value of the function $f(x_1, x_2) = x_1^2 + 2x_2^2 - 4x_1 - 4x_2$.

1. We introduce $x_0 = (\mathbf{0}; \mathbf{0}), x_1 = (0; 0)$, and calculate the criterion values $f_0 = f(0; 0) = 0$

2. We calculate the value of the derivative $\nabla f(x) = (2x_1 - 4; 4x_2 - 4)$ $\nabla f(x_0) = (-4; -4)$. Let's use value of momentum $\beta = 0{,}9$, value of step $\alpha = 0{,}01$ and accuracy $\varepsilon = 0{,}01$.

If the function is strongly convex and smooth, you can take β ≈ 0.9. If there is noise or sharp changes in the gradient, you can reduce β (for example, to 0.5). In deep learning, β = 0.9 is often used.

The heavy ball method. Example

General recommendations for choosing α:

For smooth and strongly convex functions (e.g. quadratic) optimal step: $\alpha = \frac{4}{(1+\sqrt{\beta})^2 \cdot L}$ where L is the Lipschitz constant of the gradient, β is the moment parameter.

For non-smooth or non-convex problems (e.g. deep learning) empirically choose α in the range 10^{-5} to 0,1.

Relationship with momentum (β): The larger β (closer to 1), the smaller α should be to avoid oscillations.

For example, for β=0.9 you can take α=0.01, and for β=0.99 α=0.001.

The heavy ball method. Example

3. We calculate $x_1 = x_0 - \alpha\nabla f(x_0) + \beta(x_1 - x_0) = (0;0) - 0{,}01(-4;-4) + 0{,}9(0;0) = (\mathbf{0,04};\mathbf{0,04})$. We calculate $f(x_1) = -0{,}3152$. $f(x_1) < f(x_0)$.

Let's find the absolute value of the difference between the current value of the function and its value at the previous step: $|f(x_1) - f(x_0)| > \varepsilon$.

The stopping criterion was not met, so we continue the solution and move on to step 2.

4. We calculate the value of the derivative $\nabla f(x) = (2x_1 - 4; 4x_2 - 4)$ $\nabla f(x_1) = (-3{,}92; -3{,}84)$.

The heavy ball method. Example

5. We calculate $x_2 = x_1 - \alpha\nabla f(x_1) + \beta(x_1 - x_0) = (\mathbf{0,1152}; \mathbf{0,1144})$. We calculate $f(x_2) = -0{,}878954$. $f(x_2) < f(x_1)$.

Let's find the absolute value of the difference between the current value of the function and its value at the previous step: $|f(x_1) - f(x_0)| > \varepsilon$.

The stopping criterion was not met, so we continue the solution and move on to step 2.

The heavy ball method. Example

6. We calculate the value of the derivative $\nabla f(x) = (2x_1 - 4; 4x_2 - 4)$ $\nabla f(x_2) = (-3{,}78; -3{,}54)$.

7. We calculate $x_3 = x_2 - \alpha \nabla f(x_2) + \beta(x_2 - x_1) = (\mathbf{0,2201}; \mathbf{0,2168})$. We calculate $f(x_3) = -1{,}60515$. $f(x_3) < f(x_2)$.

Let's find the absolute value of the difference between the current value of the function and its value at the previous step: $|f(x_1) - f(x_0)| > \varepsilon$.

The stopping criterion was not met, so we continue the solution and move on to step 2.

The heavy ball method. Example

8. We calculate the value of the derivative $\nabla f(x) = (2x_1 - 4; 4x_2 - 4)$ $\nabla f(x_3) = (-3{,}56; -3{,}13)$.

7. We calculate $x_4 = x_3 - \alpha \nabla f(x_3) + \beta(x_3 - x_2) = (\mathbf{0,3501}; \mathbf{0,3403})$. We calculate $f(x_3) = -2{,}40742$. $f(x_4) < f(x_3)$.

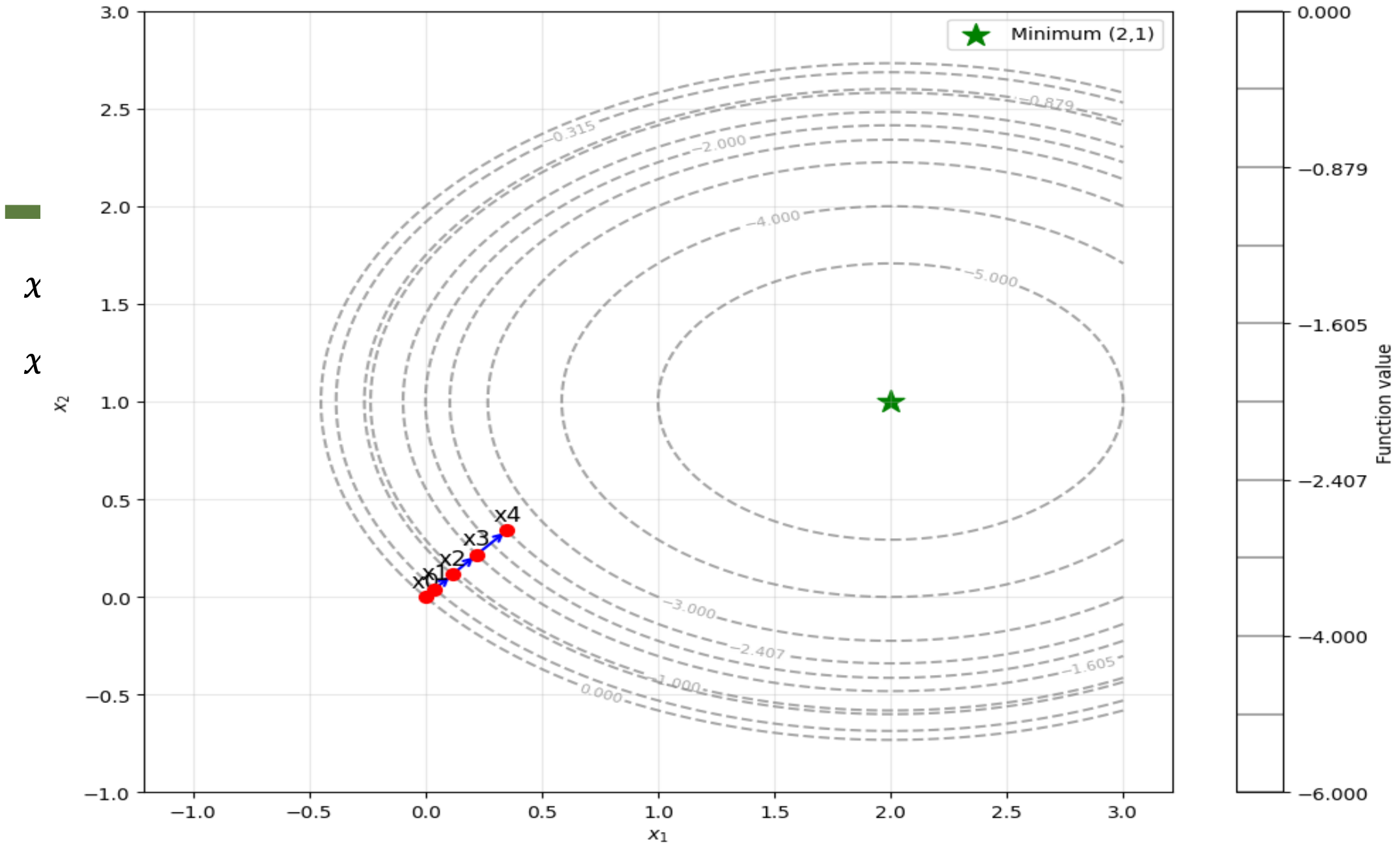
Let's find the absolute value of the difference between the current value of the function and its value at the previous step: $|f(x_1) - f(x_0)| > \varepsilon$.

The stopping criterion was not met, so we continue the solution and move on to step 2.

The heavy ball method. Example

$x_0 = (\mathbf{0};\mathbf{0}), x_1 = (\mathbf{0,04};\mathbf{0,04}), x_2 = (\mathbf{0,1152};\mathbf{0,1144}),$

$x_3 = (\mathbf{0,2201};\mathbf{0,2168}), x_4 = (\mathbf{0,3501};\mathbf{0,3403})\ldots$


Level lines of $f(x_1, x_2) = x_1^2 + 2x_2^2 - 4x_1 - 4x_2$
and optimization trajectory
Minimum (2,1)
x0
x1
x2
x3
x4
−0.315
−0.879
−2.000
−4.000
−5.000
−3.000
−2.407
−1.605
−1.000
0.000
x_1
x_2
Function value
0.000
−0.879
−1.605
−2.407
−4.000
−6.000

Nesterov's Accelerated Gradient Descent

Polyak's momentum algorithm can fail to converge for some carefully built convex optimization problems. Leveraging the idea of momentum introduced by Polyak, Nesterov introduced a slightly altered update rule that has been shown to converge not only for quadratic functions, but for general convex functions.

Algorithm 1 Nesterov's Accelerated Gradient Descent

parameters: number of iterations T, step size α, momentum β and initial condition x_0.
initialize: $v_0 \leftarrow 0$
for $t = 0, \ldots, T-1$ **do**
  $v_{t+1} \leftarrow \beta v_t - \alpha \nabla f(x_t + \beta v_t)$
  $x_{t+1} \leftarrow x_t + v_{t+1}$
end
return x_T

Nesterov's Accelerated Gradient Descent

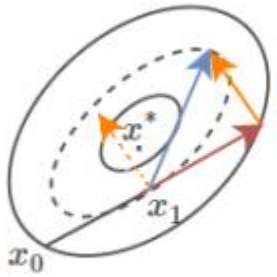
The quantity v_{t+1} can be thought of as the velocity at time t or as the displacement vector: it is calculated by applying the momentum β to the previous v_t displacement and subtracting the gradient step to $x_t + \beta v_t$ which is the point where the momentum term leads from x_t. Rewriting these two sequences as one yields:

$$x_{t+1} = x_t + \beta(x_t - x_{t-1}) - \alpha \nabla f(x_t + \beta(x_t - x_{t-1}))$$

Nesterov's Accelerated Gradient Descent

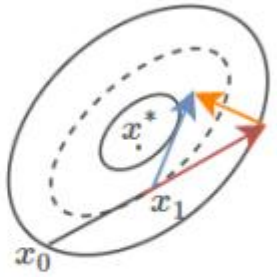
So Polyak's method evaluates the gradient before adding momentum, whereas Nesterov's algorithm evaluates it after applying momentum, which intuitively brings us closer to the minimum x^*.

Polyak's Momentum



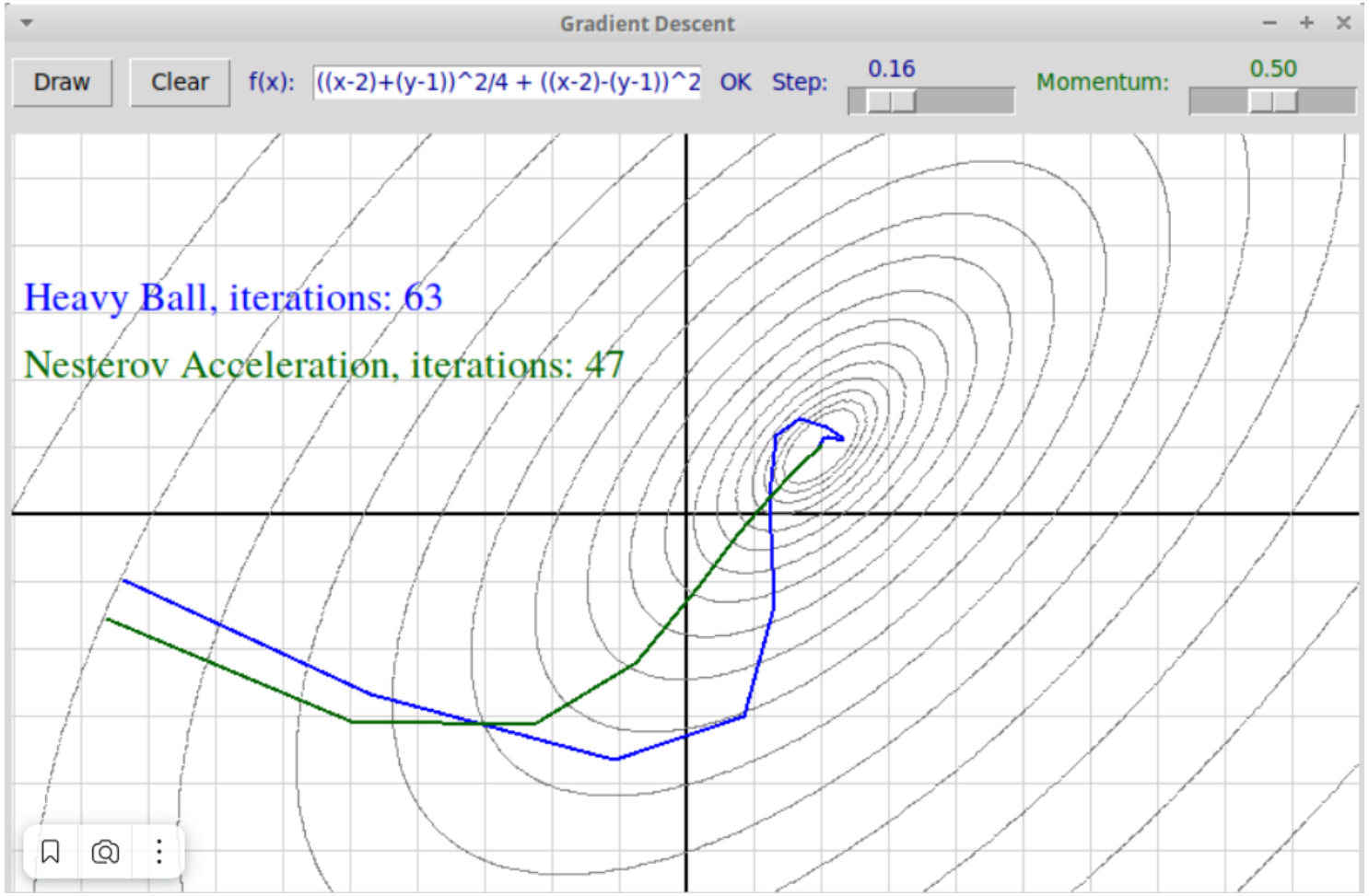
$$x_{t+1} = x_t - \alpha \nabla f(x_t) + \mu(x_t - x_{t-1})$$

Nesterov's Momentum

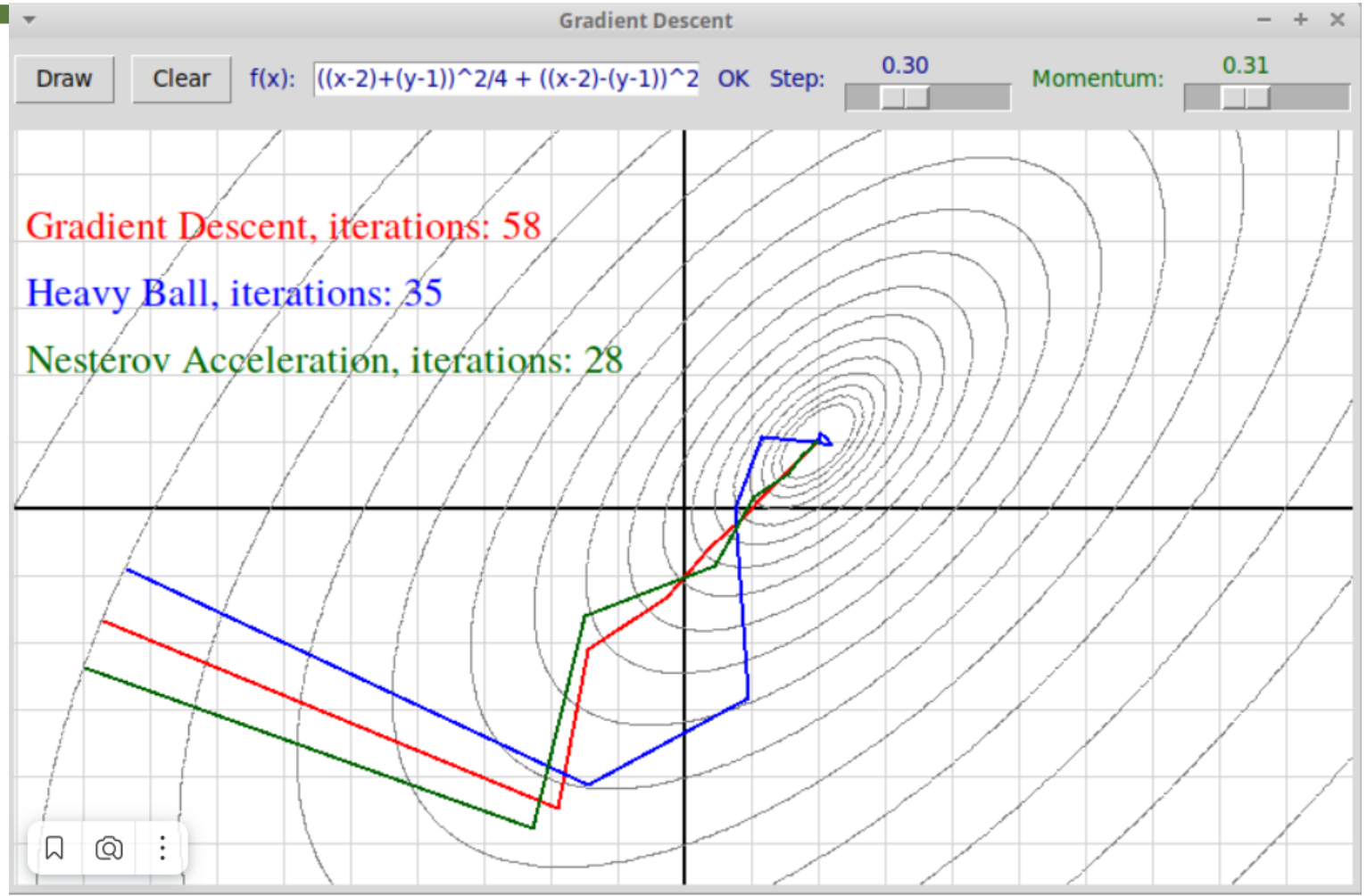


$$x_{t+1} = x_t + \mu(x_t - x_{t-1}) - \gamma \nabla f(x_t + \mu(x_t - x_{t-1}))$$

Nesterov's Accelerated Gradient Descent



Nesterov's Accelerated Gradient Descent



Nesterov's Accelerated Gradient Descent

Polyak's momentum algorithm can fail to converge for some carefully built convex optimization problems. Leveraging the idea of momentum introduced by Polyak, Nesterov introduced a slightly altered update rule that has been shown to converge not only for quadratic functions, but for general convex functions.

Algorithm 1 Nesterov's Accelerated Gradient Descent

parameters: number of iterations T, step size α, momentum β and initial condition x_0.
initialize: $v_0 \leftarrow 0$
for $t = 0, \ldots, T-1$ **do**
  $v_{t+1} \leftarrow \beta v_t - \alpha \nabla f(x_t + \beta v_t)$
  $x_{t+1} \leftarrow x_t + v_{t+1}$
end
return x_T

Nesterov's Accelerated Gradient Descent. Example

Example. Find the minimal value of the function $f(x_1, x_2) = x_1^2 + 2x_2^2 - 4x_1 - 4x_2$.

1. We introduce $x_0 = (\mathbf{0}; \mathbf{0})$, and $v_0 = 0$. Let's use value of momentum $\beta = 0{,}9$, value of step $\alpha = 0{,}01$ and accuracy $\varepsilon = 0{,}01$.

2. We calculate the value of $v_1 = \beta v_0 - \alpha \nabla f(x_0 + \beta v_0) = 0{,}9 \cdot 0 - 0{,}01 \cdot (-4; -4) = (0{,}04; 0{,}04)$; $x_1 = x_0 + v_1 = (\mathbf{0,04}; \mathbf{0,04})$

Let's find the absolute value of the difference between the current value of the function and its value at the previous step: $|f(x_1) - f(x_0)| > \varepsilon$.

The stopping criterion was not met, so we continue the solution and move on to step 2.

Nesterov's Accelerated Gradient Descent. Example

3. We calculate the value of $v_2 = \beta v_1 - \alpha \nabla f(x_1 + \beta v_1) = (0{,}07448; 0{,}07296)$; $x_2 = x_1 + v_2 = (\mathbf{0, 11448; 0, 11296})$

Let's find the absolute value of the difference between the current value of the function and its value at the previous step: $|f(x_1) - f(x_0)| > \varepsilon$.

The stopping criterion was not met, so we continue the solution and move on to step 2.

Nesterov's Accelerated Gradient Descent. Example

3. We calculate the value of $v_3 = \beta v_2 - \alpha \nabla f(x_2 + \beta v_2) = (0{,}1034; 0{,}0985)$; $x_3 = x_2 + v_3 = (\mathbf{0,217882}; \mathbf{0,21146})$

Let's find the absolute value of the difference between the current value of the function and its value at the previous step: $|f(x_1) - f(x_0)| > \varepsilon$.

The stopping criterion was not met, so we continue the solution and move on to step 2.

Nesterov's Accelerated Gradient Descent. Example

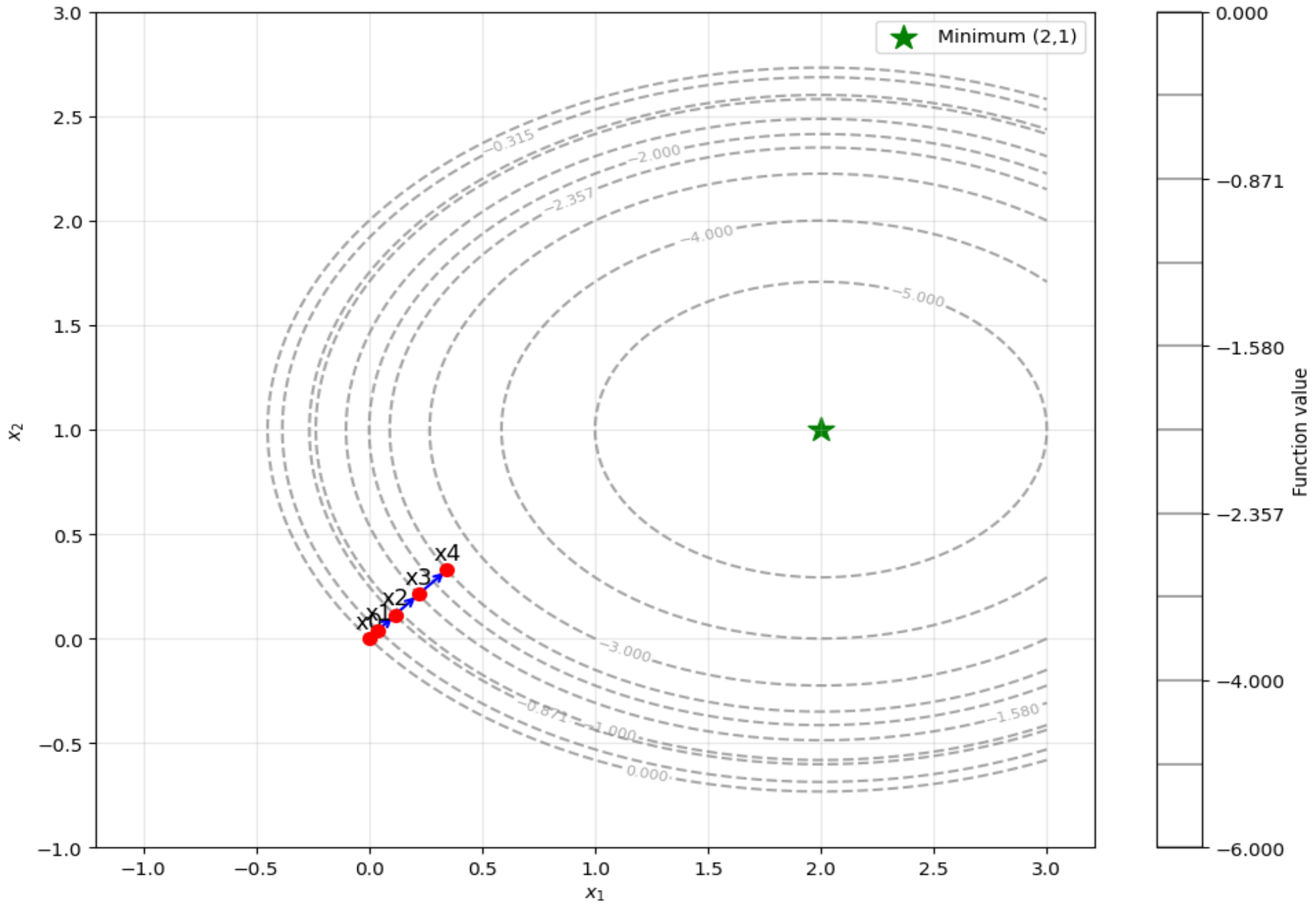
3. We calculate the value of $v_4 = \beta v_3 - \alpha \nabla f(x_3 + \beta v_3) = (0{,}1268; 0{,}1166)$; $x_4 = x_3 + v_4 = (\mathbf{0,344682}; \mathbf{0,32806})$

Let's find the absolute value of the difference between the current value of the function and its value at the previous step: $|f(x_1) - f(x_0)| > \varepsilon$.

The stopping criterion was not met, so we continue the solution and move on to step 2.

Nesterov's Accelerated Gradient Descent. Example

$$x_0 = (0; 0);$$
$$x_1 = (0,04; 0,04);$$
$$x_2 = (0,11448; 0,11296);$$
$$x_3 = (0,217882; 0,21146);$$
$$x_4 = (0,344682; 0,32806)$$


Level lines of $f(x_1, x_2) = x_1^2 + 2x_2^2 - 4x_1 - 4x_2$
and optimization trajectory
Minimum (2,1)
x0
x1
x2
x3
x4
−0.315
−2.000
−2.357
−4.000
−5.000
−3.000
−0.871
−1.000
−1.580
0.000
x_1
x_2
Function value
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.000
−0.871
−1.580
−2.357
−4.000
−6.000