# Chapter 6

**In our attempt to match the lower bounds for gradient descent, in the previous chapter we "cheated" by using information beyond the first-order gradient to achieve up to a *quadratic* convergence rate. But the question of whether we can match the initial stated lower bounds by just using gradients remains open.**

**In this chapter, we will discuss one way to match these lower bounds using only gradient information, closing this gap. This is achieved with the notion of acceleration/momentum, where we will discuss the Heavy Ball method by Polyak and Nesterov's optimal methods.**

Momentum | Heavy Ball method | Nesterov's acceleration | Adaptive restarts and noise in acceleration

We remind once again what are the limits of gradient descent-based methods, under convex assumptions.

- For convex objective functions with Lipschitz continuous gradients, with constant $L$, we can prove that there exists an instance $f$ such that first-order methods cannot be better than:

$$f(x_T) - f(x^\star) \geq \frac{3L\|x_0 - x^\star\|_2^2}{32(T+1)^2} = O\left(\tfrac{1}{T^2}\right).$$

Under this assumption, and only using gradients, we cannot achieve better than the above.

- For convex objectives functions with both Lipschitz continuous gradients and strong convexity, a similar argument holds. I.e., there is a strongly convex function $f$ such that gradient descent-based methods cannot be better than:

$$\|x_T - x^\star\|_2^2 \geq \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2T} \|x_0 - x^\star\|_2^2.$$

where $\kappa = L/\mu > 1$. Here we observe that, while we have achieved the same convergence rate with respect to the exponent—i.e., in both cases we have $c^T$, for $c < 1$—in the lower bound case, we see $\sqrt{\kappa}$ instead of $\kappa$.

**Gradient descent and acceleration.** We will focus on two *multi-step* gradient descent methods: the Heavy Ball method and (one of) Nesterov's accelerated methods. These methods are called multi-step since they take into account the history of points computed, in order to prove convergence. In its most generic form (and abstractly denoting the algorithm as a function $\varphi(\cdot)$), these methods can be written as:

$$x_{t+1} = \varphi(x_t, x_{t-1}, \dots, x_{t-\ell}),$$

where $\ell$ here represents the time window in the past from which we take information in order to accelerate the process.

In a sense, gradient methods—and even second order methods—are one-step methods with $\ell = 0$.

**Heavy-ball method.** We will start with the Heavy ball method, which can be described by the following recursion:

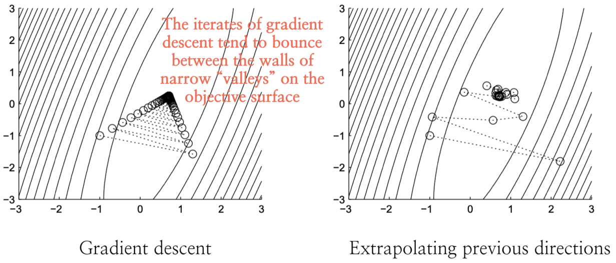
$$x_{t+1} = \underbrace{x_t - \eta\nabla f(x_t)}_{\text{Gradient step}} + \underbrace{\beta(x_t - x_{t-1})}_{\text{Momentum step}}.$$

Here, $x_t$ is the current estimate, $\eta$ is the step size, similar to standard gradient descent, and $\beta$ is the momentum parameter. Observe that, following the discussion above, this recursion belongs to the case:

$$x_{t+1} = \varphi(x_t, x_{t-1}).$$

*What is the motivation for using such a method?* A key issue in gradient descent is pathological curvature. When curvature in different regions and/or directions is very different, for a fixed learning rate gradient descent will make slow progress in one of either the high or low curvature regions/directions. For pathological curvature, we want to make smaller steps in regions of high curvature to dampen oscillations and make larger steps and accelerate in regions of low curvature.

Further, we will try to answer this question through some plots. See the figures that follow: instead of unnecessarily zig-zagging in the case of gradient descent updates, momentum uses past information in order to be "biased", and thus achieves a more *direct* trajectory towards the (local or global) stationary point.



Gradient descent

Extrapolating previous directions

**Fig. 38.** Motivation for using acceleration in gradient descent. Borrowed from Boyd's and Vanderberghe book on "Convex optimization".

Momentum is inspired by some physical analogy: Consider we have a ball that moves along a curved surface (*that's why the method is called heavy-ball*). The motion of the ball in a potential field, under the force of friction, is described by a second-order differential equation:

$$\mu \cdot \tfrac{\partial^2 x(t)}{\partial t^2} = -\nabla f(x(t)) - b\tfrac{\partial x(t)}{\partial t}.$$

Observe that the intuition of the heavy ball method comes from the continuous space, where gradient descent is actually known as gradient flow. (*The field that studies how we move from phenomena that happen in the continuous space to the discrete space is an active research area in optimization and machine learning*). One way to discretize the above continuous differential equation is to obtain:

$$\mu \cdot \tfrac{x_{t+\Delta t} - 2x_t + x_{t-\Delta t}}{\Delta t^2} = -\nabla f(x_t) - b\tfrac{x_t - x_{t-\Delta t}}{\Delta t},$$

which results into:

$$x_{t+\Delta t} = x_t - \tfrac{\Delta t^2}{\mu}\nabla f(x_t) + \left(1 - \tfrac{b\Delta t}{\mu}\right)(x_t - x_{t-\Delta t}).$$

This resembles to the discrete Heavy-ball description above.

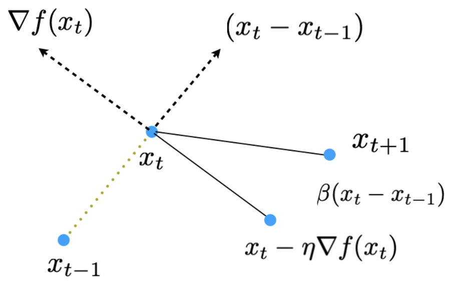


**Fig. 39.** Motions of heavy-ball method. If current gradient step is in the same direction as previous step, then move a little further in that direction.

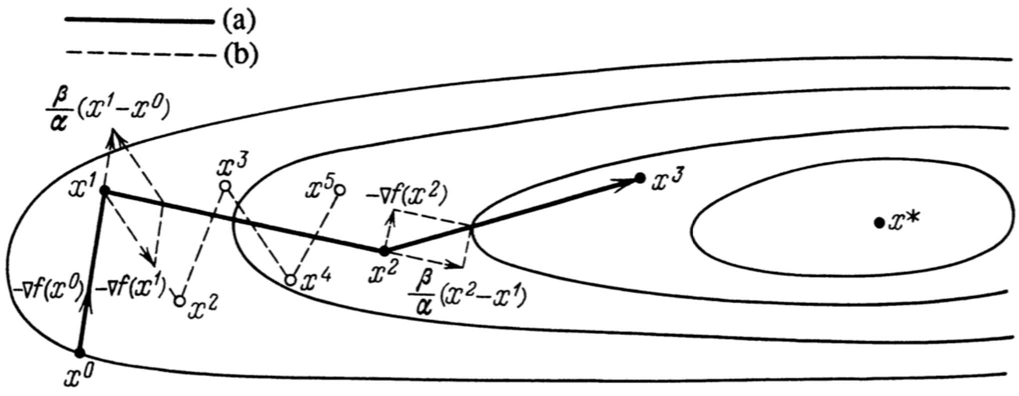


**Fig. 40.** Motivation for using acceleration in gradient descent. Borrowed from Polyak's book on "Introduction to optimization". (a) is Gradient descent, (b) is Heavy ball method.

Locally, at a point $x_t$, the Heavy ball method "makes decisions" according to the figure above.

*But how does it perform in theory?* Let us first make the assumption that we use the heavy-ball method for convex functions $f$.

**Theorem 5.** *Consider the heavy-ball recursion, with step size $\eta$ and momentum parameter $\beta$. Let $f$, the objective function, be convex, with L-Lipschitz continuous gradients. Further, assume that $f$ is strongly convex with parameter $\mu$, and with a unique global minimum $x^\star$. Then, for step size and momentum parameters satisfying:*

$$\eta = \tfrac{4}{(\sqrt{\mu}+\sqrt{L})^2}, \text{ and } \beta = \tfrac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}},$$

*the heavy ball recursion gives an estimate $x_T$ after $T$ iterations, such that:*

$$\|x_T - x^\star\|_2 \leq \left(\tfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^T \|x_0 - x^\star\|_2.$$

Before we provide the proof, compare this with the lower bounds provided at the beginning of the chapter: *the Heavy-ball method achieves the lower bounds, by just using the value of the estimates from the previous iteration!* I.e., we do not compute or store something extraordinarily large, such as keeping a long history of gradients or computing the Hessian.

*Proof:* In contrast to gradient method, we will focus on the behavior of two consecutive distances, $\|x_{t+1}-x^\star\|_2$, $\|x_t-x^\star\|_2$:

$$\begin{aligned} &\left\| \begin{bmatrix} x_{t+1}-x^\star \\ x_t-x^\star \end{bmatrix} \right\|_2 \\ &= \left\| \begin{bmatrix} x_t+\beta\left(x_t-x_{t-1}\right)-x^\star \\ x_t-x^\star \end{bmatrix} - \eta \begin{bmatrix} \nabla f\left(x_t\right) \\ 0 \end{bmatrix} \right\|_2 \\ &= \left\| \begin{bmatrix} (1+\beta)I & -\beta I \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} x_t-x^\star \\ x_{t-1}-x^\star \end{bmatrix} - \eta \begin{bmatrix} \nabla^2 f\left(z_t\right)\left(x_t-x^\star\right) \\ 0 \end{bmatrix} \right\|_2 \end{aligned}$$

For the last equality, we use the generalization of the *mean value theorem*, according to which, for a function $f : [\alpha, \beta] \rightarrow \mathbb{R}$, differentiable, there exists $\gamma \in (\alpha, \beta)$ such that:

$$f'(\gamma) = \tfrac{f(\beta)-f(\alpha)}{\beta-\alpha}.$$

This leads to the following equation for our case: $\nabla f(x_t) = \nabla^2 f(z_t)(x_t - x^\star)$, with $z_t$ in the space between $x_t$ and $x^\star$. (*To see this, consider the substitutions $f'(\cdot) \rightarrow \nabla^2 f(\cdot)$, $f(\cdot) \rightarrow \nabla f(\cdot)$, and the fact that $\nabla f(x^\star) = 0$.*) Continuing the above recursion, we have:

$$\begin{aligned} &\left\| \begin{bmatrix} x_{t+1}-x^\star \\ x_t-x^\star \end{bmatrix} \right\|_2 \\ &= \left\| \begin{bmatrix} (1+\beta)I - \eta\nabla^2 f\left(z_t\right) & -\beta I \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} x_t-x^\star \\ x_{t-1}-x^\star \end{bmatrix} \right\|_2 \\ &\leq \left\| \begin{bmatrix} (1+\beta)I - \eta\nabla^2 f\left(z_t\right) & -\beta I \\ I & 0 \end{bmatrix} \right\|_2 \cdot \left\| \begin{bmatrix} x_t-x^\star \\ x_{t-1}-x^\star \end{bmatrix} \right\|_2 \end{aligned}$$

where in the last step we apply the Cauchy-Schwarz inequality.

Let us focus on the contraction matrix:

$$\left\| \begin{bmatrix} (1+\beta)I - \eta\nabla^2 f\left(z_t\right) & -\beta I \\ I & 0 \end{bmatrix} \right\|_2$$

We know that $\nabla^2 f(\cdot) \succ 0$ by strong convexity, and it has an eigenvalue decomposition:

$$\nabla^2 f\left(z_t\right) = U\Lambda U^\top,$$

where $U$ is an orthonormal matrix, and $\Lambda$ is a diagonal matrix, with the eigenvalues of $\nabla^2 f(\cdot)$ on its diagonal. Since $\nabla^2 f(\cdot) \succ 0$, observe that all the eigenvalues are positive. Let us denote the eigenvalues as $\lambda_i$. Then, for simplicity of our arguments, we will get the following equalities, under proper assumptions:

$$\begin{aligned} &\left\| \begin{bmatrix} (1+\beta)I - \eta\nabla^2 f\left(z_t\right) & -\beta I \\ I & 0 \end{bmatrix} \right\|_2 \\ &= \left\| \begin{bmatrix} U^\top & 0 \\ 0 & U^\top \end{bmatrix} \cdot \begin{bmatrix} (1+\beta)I - \eta U\Lambda U^\top & -\beta I \\ I & 0 \end{bmatrix} \cdot \begin{bmatrix} U & 0 \\ 0 & U \end{bmatrix} \right\|_2 \\ &= \left\| \begin{bmatrix} (1+\beta)U^\top IU - \eta U^\top U\Lambda U^\top U & -\beta U^\top IU \\ U^\top IU & 0 \end{bmatrix} \right\|_2 \\ &= \left\| \begin{bmatrix} (1+\beta)I - \eta\Lambda & -\beta I \\ I & 0 \end{bmatrix} \right\|_2 \\ &= \max_i \left\| \begin{bmatrix} 1+\beta-\eta\lambda_i & -\beta \\ 1 & 0 \end{bmatrix} \right\|_2 \end{aligned}$$

I.e., the maximum value is equivalent to finding the maximum eigenvalue of many $2 \times 2$ matrices. To compute the eigenvalues of such matrices, we need to find the roots of the equation:

$$\xi^2 - (1+\beta-\eta\lambda_i)\xi + \beta = 0.$$

Observe that for $\beta \geq \left(1-\sqrt{\eta\lambda_i}\right)^2$,the roots of the characteristic equations are imaginary, and both have magnitude $\sqrt{\beta}$. By $L$-smoothness and strong convexity assumptions,

$$\left(1-\sqrt{\eta\lambda_i}\right)^2 \leq \max\left\{|1-\sqrt{\eta\mu}|^2,\ |1-\sqrt{\eta L}|^2\right\}.$$

Then, by letting $\beta = \max\left\{|1-\sqrt{\eta\mu}|^2,\ |1-\sqrt{\eta L}|^2\right\}$, we have:

$$\left\|\begin{bmatrix}(1+\beta)I-\eta\nabla^2 f\left(z_t\right) & -\beta I\\ I & 0\end{bmatrix}\right\|_2 \leq \max\left\{|1-\sqrt{\eta\mu}|,\ |1-\sqrt{\eta L}|\right\}.$$

Now, by letting $\eta = \frac{4}{(\sqrt{\mu}+\sqrt{L})^2}$, we have:

$$\beta = \tfrac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}},\ \text{and}\ \max\left\{|1-\sqrt{\eta\mu}|,\ |1-\sqrt{\eta L}|\right\} = \tfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}.$$

This leads finally to:

$$\left\|\begin{bmatrix}x_{t+1}-x^\star\\ x_t-x^\star\end{bmatrix}\right\|_2 \leq \left(\tfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)\left\|\begin{bmatrix}x_t-x^\star\\ x_{t-1}-x^\star\end{bmatrix}\right\|_2.$$

Unfolding this recursion, and focusing on the top row, we obtain:

$$\|x_T-x^\star\|_2 \leq \left(\tfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^T \|x_0-x^\star\|_2.$$

□

Thus, heavy-ball method converges linearly, but, in Big-Oh notation and given that the factor $\kappa$ is an important one, its iteration complexity is $O(\sqrt{\kappa}\log\frac{1}{\varepsilon})$, as compared to $O(\kappa\log\frac{1}{\varepsilon})$ of standard gradient descent. The corresponding `iPython Notebook` compares the convergence of gradient descent and the heavy ball method.

*What about using the heavy-ball method for convex but just L-smooth functions? Can we still prove convergence or, even better, the acceleration?* In our thus-far discussion on the heavy-ball method, we made the following assumptions, on top of convexity and $L$-smoothness:

- $f$ is also strongly convex with parameter $\mu$.
- $f$ is twice differentiable.

There are some surprising results when we start dropping some of these assumptions. (*The research on these questions is still active currently; thus, if you find any results that disprove any of the statements below, please let me know.*) Zavriev and Kostyuk in [45] prove that the heavy-ball method trajectories to converge to a stationary point, with sufficient conditions, when the function $f$ is just $L$-smooth, but not necessarily convex. It turns out that current state of the art results for just $L$-smooth, *and convex* functions $f$ is the following theorem by Ghadimi, Feyzmahdavian and, Johansson [46].

**Theorem 6.** *Let $f$ be a convex function that has $L$-Lipschitz continuous gradients. Consider the heavy-ball recursion with momentum parameter and step size satisfying: $\beta \in [0,1)$, $\eta \in \left(0, \frac{2(1-\beta)}{L}\right)$. Then,*

$$f(\bar{x}_T) - f(x^\star) = O\left(\tfrac{1}{T}\right),$$

*where $\bar{x}_T = \frac{1}{T+1}\sum_{t=0}^T x_t$.*

*Sketch of proof:* The proof uses the following steps:

- Define $p_t = \frac{\beta}{1-\beta}(x_t - x_{t-1})$, which leads to heavy-ball recursion: $x_{t+1}+p_{t+1} = x_t + p_t - \frac{\eta}{1-\beta}\nabla f(x_t)$.
- Compute $\|x_{t+1}+p_{t+1}-x^\star\|_2^2$ by substituting the quantity $x_{t+1}+p_{t+1}$ and unrolling the square identity.
- Using standard $L$-smoothness identities, we get to:

$$\begin{aligned}&\tfrac{2\eta\lambda}{(1-\beta)}\sum_{t=0}^T \left(f\left(x_t\right)-f(x^\star)\right)\\ &+\sum_{t=0}^T\left(\tfrac{2\eta\beta}{(1-\beta)^2}\left(f\left(x_t\right)-f(x^\star)\right)+\|x_{t+1}+p_{t+1}-x^\star\|^2\right)\\ &\leq \sum_{t=0}^T\left(\tfrac{2\eta\beta}{(1-\beta)^2}\left(f\left(x_{t-1}\right)-f(x^\star)\right)+\|x_t+p_t-x^\star\|^2\right)\end{aligned}$$

  for some auxiliary variable $\lambda \in (0,1]$.
- This implies that:

$$\tfrac{2\eta\lambda}{(1-\beta)}\sum_{t=0}^T\left(f\left(x_t\right)-f(x^\star)\right) \leq \tfrac{2\eta\beta}{(1-\beta)^2}\left(f\left(x_0\right)-f(x^\star)\right)+\|x_0-x^\star\|^2$$

- Given convexity of $f$, we have by Jensen's inequality that:

$$(T+1)f(\bar{x}_T) \leq \sum_{t=0}^T f(x_t).$$

- The above lead to:

$$\begin{aligned}&f\left(\bar{x}_T\right)-f(x^\star)\\ &\quad\leq \tfrac{1}{T+1}\left(\tfrac{\beta}{\lambda(1-\beta)}\left(f\left(x_0\right)-f(x^\star)\right)+\tfrac{1-\beta}{2\eta\lambda}\|x_0-x^\star\|^2\right)\\ &\quad= O(\tfrac{1}{T})\end{aligned}$$

□

The above result denotes that the average of all estimates actually drops with rate $O(\frac{1}{T})$; i.e., the current proof for heavy-ball is no better than that of simple gradient descent method! One can use per-iteration specific values for $\eta_t$ and $\beta_t$, which further leads to:

$$f\left(x_T\right)-f(x^\star) = O(\tfrac{1}{T}),$$

according to Ghadimi, Feyzmahdavian and, Johansson [46]. *However, still there is a gap between our current theory and the possibly achievable lower bounds!*

What is more interesting is the following fact: So far, we focused on the $L$-smoothness assumption; if we assume also strong convexity, but we drop the assumption that $f$ is twice differentiable, there are cases where the heavy-ball method does not necessarily converge, even using Polyak's stability conditions!

**Nesterov's accelerated method.** In our discussion so far, for both theory and practice, we made the following choices:

- Practically, heavy-ball method satisfies the recursion $x_{t+1} = x_t - \eta\nabla f(x_t) + \beta(x_t - x_{t-1})$, where the gradient is computed at the current point $x_t$.
- Theoretically, heavy-ball method was shown to achieve the lower bounds for the case of $L$-smooth and $\mu$-strongly convex case.

Nesterov, in his seminal paper [47] in 1983, he proved that a slightly different version of the heavy-ball method can achieve the lower bounds of $O(\frac{1}{T^2})$ for first-order methods under $L$-smoothness assumption; a result that is currently missing for the simple heavy-ball method.

First, let us describe Nesterov's proposal. The idea is based on the following observation: The Heavy-ball method

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

can be equivalently written as a two-step procedure:

$$\begin{aligned}\widetilde{x}_t &= x_t - \eta \nabla f(x_t)\\ x_{t+1} &= \widetilde{x}_t + \beta(x_t - x_{t-1}).\end{aligned}$$

In a way, in Heavy-ball, we end up to $x_{t+1}$ after computing the gradient of $f$ at $x_t$, and performing the momentum step. But what if we compute the gradient at a point that looks *more similar* to the motions we perform, even after the gradient calculation in heavy-ball? This leads to Nesterov's suggestion where we compute:

$$\begin{aligned}\widetilde{x}_t &= x_t - \eta \nabla f(x_t + \beta(x_t - x_{t-1}))\\ x_{t+1} &= \widetilde{x}_t + \beta(x_t - x_{t-1}).\end{aligned}$$

Locally, at a point $x_t$, the Nesterov's method "makes decisions" according to the following figure.

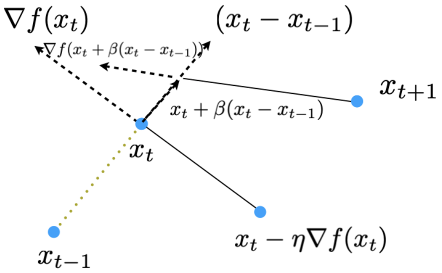

$\nabla f(x_t)$
$(x_t - x_{t-1})$
$\nabla f(x_t + \beta(x_t - x_{t-1}))$
$x_{t+1}$
$x_t + \beta(x_t - x_{t-1})$
$x_t$
$x_{t-1}$
$x_t - \eta \nabla f(x_t)$

**Fig. 41.** Motions of Nesterov's accelerated method. If current gradient step is in same direction as previous step, then move a little further in that direction. Compare this figure with previous Figure.

The above can be written in the following form, which is more recognizable as Nesterov's recursion:

$$\begin{aligned}x_{t+1} &= y_t - \eta \nabla f(y_t)\\ y_{t+1} &= x_{t+1} + \beta(x_{t+1} - x_t)\end{aligned}$$

What was revolutionary is the fact that Nesterov proposed specific, time-dependent, values for $\beta_t$—that are at the same time practical—which lead provably to acceleration! One such schedule for the momentum parameters $\beta_t$ satisfies:

$$\theta_0 = 1, \; \theta_{t+1} = \tfrac{1+\sqrt{1+4\theta_t^2}}{2}, \; \beta_t = \tfrac{\theta_t - 1}{\theta_{t+1}}$$

Let us first consider the case where $f$ is convex and $L$-smooth.

**Theorem 7.** *Let $f$ be a convex function with $L$-Lipschitz continuous gradients. Then, Nesterov's recursion with $\beta_t$ as defined above, and $\eta = \frac{1}{L}$ satisfies:*

$$f(x_T) - f(x^\star) \leq \tfrac{2L\|x_0 - x^\star\|_2^2}{T^2} = O\left(\tfrac{1}{T^2}\right).$$

I.e., Nesterov's accelerated method achieves the lower bound, for the case of just $L$-smooth convex functions!

Further, for strongly convex functions with parameter $\mu$, one can also show that, similarly to the heavy-ball method, it achieves the complexity $O\left(\sqrt{\kappa}\log\frac{1}{\varepsilon}\right)$; i.e., it also achieves the lower bound for the case of $L$-smooth and $\mu$-strongly convex functions! (*We omit the proof; we also leave the discussion of acceleration in non-convex settings for later.*)

**Interesting facts about acceleration.** Closing this chapter, we will discuss two interesting facts using acceleration.

*Set up:* For the first one, we will need an optimal configuration for Nesterov's accelerated method, when we know exactly the condition number of the convex problem, $\kappa = \frac{L}{\mu}$. The recursion satisfies:

$$\begin{aligned}x_{t+1} &= y_t - \tfrac{1}{L}\nabla f(y_t)\\ y_{t+1} &= x_{t+1} + \beta^\star(x_{t+1} - x_t),\end{aligned}$$

where

$$\beta^\star = \frac{1-\sqrt{\frac{\mu}{L}}}{1+\sqrt{\frac{\mu}{L}}} = \frac{1-\sqrt{\frac{1}{\kappa}}}{1+\sqrt{\frac{1}{\kappa}}}$$

Let us define also $q^\star = \frac{1}{\kappa}$. The above recursion is provably optimal, and the proof is omitted; by optimal, we mean that there is a constant step size along with this momentum parameter that achieves to the lower bounds. However, it requires the exact knowledge of the Lipschitz gradient continuity parameter $L$ and strong convex parameter $\mu$. Also, note that this selection is optimal assuming convexity.

For the second one, we will assume that the gradient calculation step includes some noise. As before, we assume that the function satisfies Lipschitz gradient continuity. One natural way to think of this is to assume that we compute only a *noisy* version of the gradient:

$$\widetilde{\nabla} f(y_t) = \nabla f(y_t) + \xi.$$

For the theory, we will need the following definition of inexact first-order oracle. In the noiseless case, we know that:

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y - x\rangle \leq \tfrac{L}{2}\|x - y\|_2^2$$

Pictorially, at every point $x$ the function can be "sandwiched" between a tangent linear function, $\langle \nabla f(x), y - x\rangle$, and a parabola. For the inexact oracle, we will assume the same inequality holds with some slack $\delta > 0$:

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y - x\rangle \leq \tfrac{L}{2}\|x - y\|_2^2 + \delta$$

Pictorially, it comes with the same illustration, except now there's some slack between the linear approximation and the parabola. Let us know describe these interesting phenomena.

*Acceleration often leads to non-decreasing sequence of function values.* It is common, when running an accelerated method, to have the appearance of ripples in the trace of the objective value; these are seemingly regular increases in the objective. The following figure is borrowed from [48] by O'Donoghue and Candes.

The function we are optimizing here is a simple quadratic function:

$$f(x) = \tfrac{1}{2}x^\top Ax,$$

where $A$ is a positive definite matrix. First, observe that in this case,

$$\min_x f(x)$$

has optimal solution $x^\star = 0$, and $f(x^\star) = 0$. Further, the Lipschitz gradient continuity parameter satisfies $L = \lambda_{\max}(A)$, and the strong convexity parameter satisfies $\mu = \lambda_{\min}(A)$.

Let's extract some information from the plot. The case where $q = 1$ leads to:

$$y_{t+1} = x_{t+1} + \tfrac{1-\sqrt{q}}{1+\sqrt{q}}(x_{t+1} - x_t) = x_{t+1}$$

and thus the accelerated version boils down to:

$$x_{t+1} = x_t - \tfrac{1}{L}\nabla f(x_t),$$

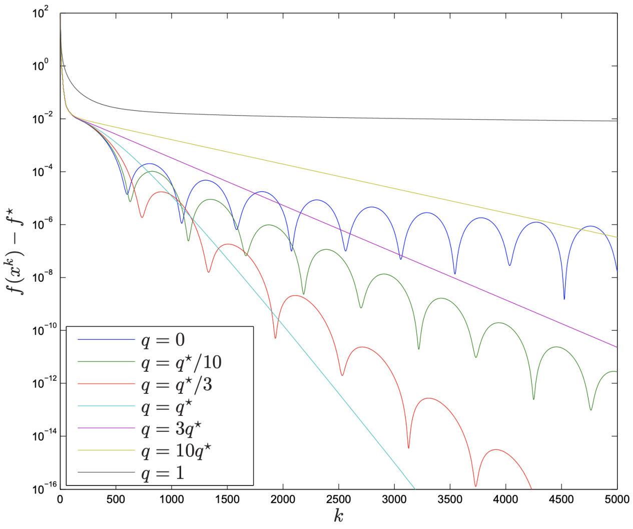


**Fig. 42.** Behavior of optimal's accelerated method for a convex function, where we do not set up the $q$ parameter correctly (in other words, we only approximate the values $L$ and $\mu$).

the gradient descent method. Also, assuming that the momentum parameter takes values in $[0, 1]$, the maximum parameter case is when $q = 0$, where:

$$\beta = \tfrac{1-\sqrt{q}}{1+\sqrt{q}} = 1.$$

Ranging the value of $\beta$, we observe an interesting phenomenon. Starting with $q = 1$ (i.e., $\beta = 0$), we obtain the behavior of gradient descent, which from the figure shows the worst performance (in terms of iteration complexity). On the other end, for $q = 0$ we obtain the maximum $\beta$ value, that definitely “beats” gradient descent, but there are other values of $\beta$, between the values 0 and 1, that give a better performance.

More importantly, we observe these interesting ripples in the plots: the function values do not monotonically decrease as the iterations increase, but rather follow a periodic pattern. However, overall and despite this behavior, the function values decrease faster than plain gradient descent. Of course, as expected the optimal performance—without any ripples—is achieved by $q^\star$.

Overall, slightly over- or under-estimating the optimal value $q$ (or equivalently of $\kappa$) leads to presumably severe detrimental effect on the rate of convergence of the algorithm. Note the clear difference between the cases where we underestimate ($q < q^\star$) and where we overestimate ($q > q^\star$): in the former we observe this rippling behavior in the function traces, while in the latter we observe the classical monotonic convergence.

To understand better what is happening during the ripples, we also provide the following plot from the same paper by O'Donoghue and Candes. It is obvious that the high momentum values cause the trajectory towards the optimum $x^\star$ to overshoot and oscillate around it. This causes a rippling in the function values along the trajectory, as we get closer but then move further away from the optimum.

*What about Nesterov's routines on selecting $\beta_t$?* Someone would wonder “*what happens when we use the routine:*

$$\theta_0 = 1, \; \theta_{t+1} = \tfrac{1+\sqrt{1+4\theta_t^2}}{2}, \; \beta_t = \tfrac{\theta_t - 1}{\theta_{t+1}}''$$

It turns out that, as the iterations increase, the $\beta_t$ values keep increasing towards the maximum value 1, as shown in the plot next. Thus, Nesterov's approach naturally often leads to a rippling behavior, that we observe in practice.

*What could be a solution to this? (Adaptive) restarts of the momentum $\beta$ procedure.* One approach to avoid ripples is to restart the $\beta_t$ computation procedure once in a while. E.g., one natural check we can make is to check at every new point whether the function value starts increasing; in that case, we can reset $\theta_{t+1} = 0$ and compute a new set of $\beta$'s. But, do these techniques work in practice? It turns out they do!

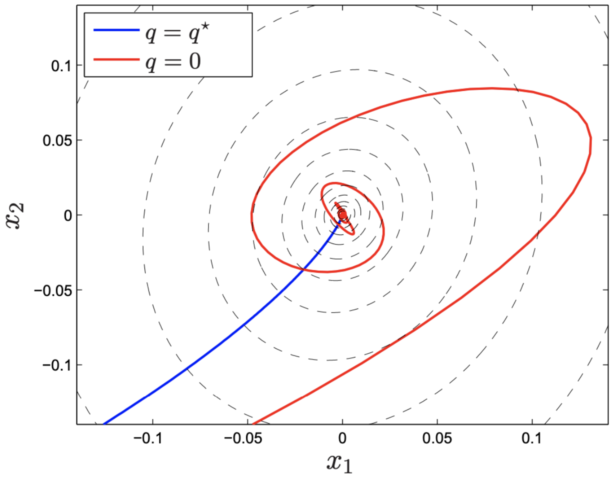


**Fig. 43.** Comparison of behavior between optimal $q^\star$ and maximum momentum parameter ($q = 0$) for a 2-dimensional toy example.

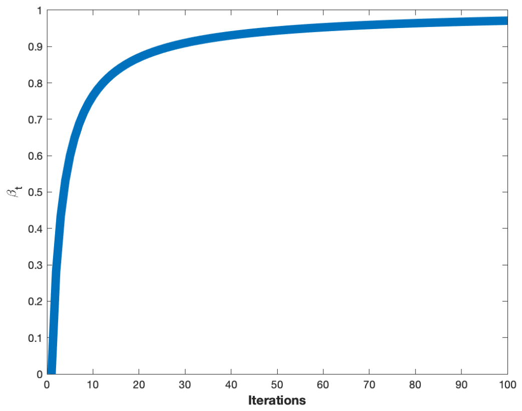


**Fig. 44.** $\beta_t$ values w.r.t. number of iterations, according to the rule $\theta_0 = 1, \; \theta_{t+1} = \frac{1+\sqrt{1+4\theta_t^2}}{2}, \; \beta_t = \frac{\theta_t - 1}{\theta_{t+1}}$.

*Behavior of acceleration under noisy settings.* The point of this subsection is that simple GD is more noise tolerant than accelerated methods. The noise tolerance corresponds to the case where we might not be able to compute exactly the gradient, but have a rough approximation of it.

This statement is based on the work by Devolder, Glineur and Nesterov [49]. The main idea is that, even if accelerated

GD converges faster than the plain GD, it must also accumulate errors faster (linearly) with the number of iterations.

Let us consider a noisy version of the above experiment. In particular, instead of computing exactly $\nabla f(x) = Ax - b$ per iteration, we see $\nabla f(x) + \xi = Ax - b + \xi$ where $\xi$ is a vector sampled from the $n$-dimensional normal distribution. Let us see how this performs in practice.

(*See ipython notebook.*)

But what can we say theoretically for this phenomenon? It turns out that what [49] shows is that, for an inexact first-order oracle, that satisfies the Lipschitz gradient continuity with slack $\delta$, we can hope for:

$$f(x_t) - \min_x f(x) \leq O\left(\tfrac{L}{t}\right) + \delta.$$

I.e., while we know that we decrease the error at a rate $O(\frac{1}{T})$, we cannot “beat” the fact that there is error every step, and we cannot reduce the error more than within a $\delta$ radius around the optimum.

On the other hand, what acceleration provably gives us is:

$$f(x_t) - \min_x f(x) \leq O\left(\tfrac{L}{t^2}\right) + t \cdot \delta.$$

I.e., the same story holds but, at the same time, the error level that we want to “beat” increases with the number of iterations (i.e., $t_1\delta < t_2\delta$ for any $t_1 < t_2$). Thus, acceleration, while converges faster in a noiseless setting, it also accumulates errors faster.

(*See ipython notebook.*)

## Appendix

1. J. Nocedal and S. Wright. Numerical optimization. Springer Science & Business Media, 2006.
2. Y. Nesterov. Introductory lectures on convex optimization: A basic course, volume 87. Springer Science & Business Media, 2013.
3. S. Boyd and L. Vandenberghe. Convex optimization. Cambridge university press, 2004.
4. D. Bertsekas. Convex optimization algorithms. Athena Scientific Belmont, 2015.
5. Sébastien Bubeck. Convex optimization: Algorithms and complexity. Foundations and Trends® in Machine Learning, 8(3-4):231–357, 2015.
6. S. Weisberg. Applied linear regression, volume 528. John Wiley & Sons, 2005.
7. T. Hastie, R. Tibshirani, and M. Wainwright. Statistical learning with sparsity: the lasso and generalizations. CRC press, 2015.
8. J. Friedman, T. Hastie, and R. Tibshirani. The elements of statistical learning, volume 1. Springer series in statistics New York, 2001.
9. M. Paris and J. Rehacek. Quantum state estimation, volume 649. Springer Science & Business Media, 2004.
10. M. Daskin. A maximum expected covering location model: formulation, properties and heuristic solution. Transportation science, 17(1):48–70, 1983.
11. I. Goodfellow, Y. Bengio, and A. Courville. Deep learning. MIT press, 2016.
12. L. Trefethen and D. Bau III. Numerical linear algebra, volume 50. Siam, 1997.
13. G. Strang. Introduction to linear algebra, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.
14. G. Golub. Cmatrix computations. The Johns Hopkins, 1996.
15. A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.
16. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.
17. S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015.
18. T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119, 2013.
19. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.
20. Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1243–1252. JMLR. org, 2017.
21. Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Fifteenth annual conference of the international speech communication association, 2014.
22. Tom Sercu, Christian Puhrsch, Brian Kingsbury, and Yann LeCun. Very deep multilingual convolutional neural networks for LVCSR. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4955–4959. IEEE, 2016.
23. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. page arXiv:1706.03762, 2017.
24. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. page arXiv:1810.04805, 2018.
25. Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and VQA. In AAAI, pages 13041–13049, 2020.
26. Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2020.
27. Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. arXiv preprint arXiv:1909.08053, 2019.
28. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683, 2019.
29. Gary Marcus, Ernest Davis, and Scott Aaronson. A very preliminary analysis of DALL-E 2. arXiv preprint arXiv:2204.13807, 2022.
30. John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. Nature, 596(7873):583–589, 2021.
31. Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
32. Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training nlp models: A concise overview. arXiv preprint arXiv:2004.08900, 2020.
33. H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 795–811. Springer, 2016.
34. Philip Wolfe. Convergence conditions for ascent methods. SIAM review, 11(2):226–235, 1969.
35. Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. Pacific Journal of mathematics, 16(1):1–3, 1966.
36. Stephen Wright and Jorge Nocedal. Numerical optimization. Springer Science, 35(67-68):7, 1999.
37. B. Polyak. Introduction to optimization. Inc., Publications Division, New York, 1, 1987.
38. Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. lecture notes of EE392o, Stanford University, Autumn Quarter, 2004:2004–2005, 2003.
39. Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. Naval research logistics quarterly, 3(1-2):95–110, 1956.
40. M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In Proceedings of the 30th international conference on machine learning, number CONF, pages 427–435, 2013.
41. J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the $\ell_1$-ball for learning in high dimensions. In Proceedings of the 25th international conference on Machine learning, pages 272–279, 2008.
42. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8):30–37, 2009.
43. A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In Advances in neural information processing systems, pages 1257–1264, 2008.
44. T. Booth and J. Gubernatis. Improved criticality convergence via a modified Monte Carlo power iteration method. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.
45. S. Zavriev and F. Kostyuk. Heavy-ball method in nonconvex optimization problems. Computational Mathematics and Modeling, 4(4):336–341, 1993.
46. E. Ghadimi, H. Feyzmahdavian, and M. Johansson. Global convergence of the heavy-ball method for convex optimization. In 2015 European control conference (ECC), pages 310–315. IEEE, 2015.
47. Y. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In Dokl. akad. nauk Sssr, volume 269, pages 543–547, 1983.
48. B. O'Donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. Foundations of computational mathematics, 15(3):715–732, 2015.
49. O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. Mathematical Programming, 146(1-2):37–75, 2014.
50. L. Bottou, F. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. Siam Review, 60(2):223–311, 2018.