The Role of Memory in Stochastic Optimization

Antonio Orvieto[*]
Department of Computer Science
ETH Zürich, Switzerland

Jonas Kohler
Department of Computer Science
ETH Zürich, Switzerland

Aurelien Lucchi
Department of Computer Science
ETH Zürich, Switzerland

Abstract

The choice of how to retain information about past gradients dramatically affects the convergence properties of state-of-the-art stochastic optimization methods, such as Heavy-ball, Nesterov's momentum, RMSprop and Adam. Building on this observation, we use stochastic differential equations (SDEs) to explicitly study the role of memory in gradient-based algorithms. We first derive a general continuous-time model that can incorporate arbitrary types of memory, for both deterministic and stochastic settings. We provide convergence guarantees for this SDE for weakly-quasi-convex and quadratically growing functions. We then demonstrate how to discretize this SDE to get a flexible discrete-time algorithm that can implement a board spectrum of memories ranging from short- to long-term. Not only does this algorithm increase the degrees of freedom in algorithmic choice for practitioners but it also comes with better stability properties than classical momentum in the convex stochastic setting. In particular, no iterate averaging is needed for convergence. Interestingly, our analysis also provides a novel interpretation of Nesterov's momentum as stable gradient amplification and highlights a possible reason for its unstable behavior in the (convex) stochastic setting. Furthermore, we discuss the use of long term memory for second-moment estimation in adaptive methods, such as Adam and RMSprop. Finally, we provide an extensive experimental study of the effect of different types of memory in both convex and nonconvex settings.

[*] Correspondence to [orvietoa@ethz.ch].

1 INTRODUCTION

Our object of study is the classical problem of minimizing finite-sum objective functions:

$$x^* = \arg\min_{x\in\mathbb{R}^d} f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x). \tag{P}$$

Accelerated gradient methods play a fundamental role in optimizing such losses, providing optimal rates of convergence for certain types of function classes such as the ones being convex [Nesterov, 2018]. The two most popular momentum methods are Heavy-ball (HB) [Polyak, 1964] and Nesterov's accelerated gradient (NAG) [Nesterov, 1983]. They are based on the fundamental idea of augmenting gradient-based algorithms with a momentum term that uses *previous gradient directions* in order to accelerate convergence, which yields the following type of iterative updates:

$$x_{k+1} = x_k + \beta_k(x_k - x_{k-1}) - \eta\nabla f(x_k), \tag{HB}$$

with β_k an iteration dependent *momentum parameter*[2] and η a positive number called *learning rate* (a.k.a. *stepsize*).

Although both HB and NAG have received a lot of attention in the literature, the idea of acceleration is still not entirely well understood. For instance, a series of recent works [Su et al., 2016, Wibisono et al., 2016, Yang et al., 2018] has studied these methods from a physical perspective, which yields a connection to damped linear oscillators. Arguably, the insights provided by these works are mostly descriptive and have so far not been able to help with the design of conceptually new algorithms. Furthermore, the resulting analysis often cannot be easily translated to stochastic optimization settings, where stability of momentum methods may actually be reduced due to inexact gradient information [Jain et al., 2018, Kidambi et al., 2018].

[2] Gradient Descent [Cauchy, 1847] can be seen as a special case of HB for $\beta_k = 0$.

This lack of theoretical understanding is rather unsatisfying. Why is it that acceleration able to provide faster rates of convergence for convex functions but fails when used on non-convex functions or in a stochastic setting? This question is especially relevant given that momentum techniques (such as Adam [Kingma and Ba, 2014]) are commonly used in machine learning in order to optimize non-convex objective functions that arise when training deep neural networks.

In order to address this issue, we here exploit an alternative view on the inner workings of momentum methods which is not physically-inspired but instead builds upon the theoretical work on memory gradient diffusions developed by [Cabot et al., 2009] and [Gadat and Panloup, 2014]. In order to leverage this analogy, we first rewrite HB as follows:

$$x_{k+1} = x_k - \eta \sum_{j=0}^{k-1} \left(\prod_{h=j+1}^{k} \beta_h \right) \nabla f(x_j) - \eta \nabla f(x_k) \qquad \text{(HB-SUM)}$$

where $x_0 = x_{-1}$ is assumed. That is, at each iteration k the next step is computed using a *weighted average of past gradients* : $x_{k+1} = x_k + \eta \sum_{j=0}^{k} w(j,k) \nabla f(x_k))$. In particular, if β_h is constant across all iterations, the memory — which is controlled by the weights — vanishes exponentially fast (short-term memory). Such averaging provides a cheap way to 1) adapt to the geometry of ill-conditioned problems (also noted in [Sutskever et al., 2013]) and 2) denoise stochastic gradients if the underlying true gradients are changing slowly. Similarly, adaptive methods [Duchi et al., 2011, Kingma and Ba, 2014] use memory of past square gradients to automatically adjust the learning rate during training. For the latter task, it has been shown [Reddi et al., 2018] that some form of long-term memory is convenient both for in theory (to ensure convergence) and in practice, since it has been observed that, in a mini-batch setting, *large gradients are not common and might be quite informative*.

In summary — most modern stochastic optimization methods can be seen as composition of memory systems. Inspired by this observation and by the undeniable importance of shining some light on the acceleration phenomenon, we make the following contributions.

1. Following previous work from [Cabot et al., 2009], we generalize the continuous-time limit of HB to an interpretable ODE that can implement various types of gradient forgetting (Sec. 3.1). Next, we extend this ODE to the stochastic setting (Sec. 3.2).
2. By comparing the resulting SDE to the model for Nesterov momentum developed in [Su et al., 2016], we provide a novel interpretation of acceleration as gradient amplification and give some potential answers regarding the source of instability of stochastic momentum methods (Sec. 3.3).
3. We study the convergence guarantees of our continuous-time memory system and show that, in the convex setting, long-term (polynomial) memory is more stable than classical momentum (Sec. 4).
4. We discretize this memory system and derive an algorithmic framework that can incorporate various types of gradient forgetting efficiently. Crucially, we show the discretization process preserves the convergence guarantees.
5. We run several experiments to support our theory with empirical evidence in both deterministic and stochastic settings (Sec. 5).
6. We propose a modification of Adam which uses long-term memory of gradient second moments to adaptively choose the learning rates (Sec. 6).

We provide an overview of our notation in App. A.

2 RELATED WORK

Momentum in deterministic settings. The first accelerated proof of convergence for the deterministic setting dates back to [Polyak, 1964] who proved a local linear rate of convergence for Heavy-ball (with constant momentum) for twice continuously differentiable, μ-strongly convex and L-smooth functions (with a constant which is faster than gradient descent). [Ghadimi et al., 2015] derived a proof of convergence of the same method for convex functions with Lipschitz-continuous gradients, for which the Cesàro average of the iterates converges in function value like $\mathcal{O}(1/k)$ (for small enough η and β).

A similar method, Nesterov's Accelerated Gradient (NAG), was introduced by [Nesterov, 1983]. It achieves the optimal $\mathcal{O}(1/k^2)$ rate of convergence for convex functions and, with small modifications, an accelerated (with respect to gradient desacent) linear convergence rate for smooth and strongly-convex functions.

Momentum in stochastic settings. Prior work has shown that the simple momentum methods discussed above lack stability in stochastic settings, where the evaluation of the gradients is affected by noise (see motivation in [Allen-Zhu, 2017] for the Katyusha method). In particular, for quadratic costs, [Polyak, 1987] showed that stochastic Heavy-ball does not achieve any accelerated rate but instead matches the rate of SGD. More general results are proved in [Yang et al., 2016] for these methods, both for convex and for smooth functions, requiring a decreasing learning rate, bounded noise and bounded subgradients (see Tb.1). For strongly-convex functions, [Yuan et al., 2016] also studied the mean-square

Function	Gradient	Rate	Reference
μ-strongly-convex, L-smooth	Deterministic	$f(x_k) - f(x^*) \leq \mathcal{O}(q^k)$	[Polyak, 1964]
Convex, L-smooth	Deterministic	$f(\bar{x}_k) - f(x^*) \leq \mathcal{O}(1/k)$	[Ghadimi et al., 2015]
Convex	Stochastic	$\mathbb{E}\left(f(\bar{x}_k) - f(x^*)\right) \leq \mathcal{O}(1/\sqrt{k})$	[Yang et al., 2016] (*)
Non-convex, L-smooth	Stochastic	$\min_{i \leq k} \mathbb{E}\left[\|\|\nabla f(x_i)\|\|^2\right] \leq \mathcal{O}(1/\sqrt{k})$	[Yang et al., 2016] (*)

Table 1: Existing convergence rate for Heavy-ball for general functions (special cases for quadratic functions are mentioned in the main text). The term $\bar{x}_k$ denotes the Cesaro average of the iterates. The constant q is defined as $q = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$. (*) The results of [Yang et al., 2016] also require bounded noise and bounded gradients as well as a step size decreasing as $1/\sqrt{k}$.

error stability and showed that convergence requires small (constant) learning rates. Furthermore, the rate is shown to be equivalent to SGD and therefore the theoretical benefits of acceleration in the deterministic setting do not seem to carry over to the stochastic setting.

Continuous-time perspective. The continuous time ODE model of NAG for convex functions presented in [Su et al., 2016] led to the developments of several variants of Nesterov-inspired accelerated methods in the deterministic setting (e.g. [Krichene et al., 2015] and [Wilson et al., 2016]). In this line of research, interesting insights often come from a numerical analysis and discretization viewpoint [Zhang et al., 2018, Betancourt et al., 2018]. Similarly, in stochastic settings, guided by SDE models derived from Nesterov's ODE in [Su et al., 2016] and by the variational perspective in [Wibisono et al., 2016], [Xu et al., 2018a] and [Xu et al., 2018b] proposed an interpretable alternative to AC-SA (an accelerated stochastic approximation algorithm introduced in [Lan, 2012] and [Ghadimi and Lan, 2012]). This is a sophisticated momentum method that in expectation achieves a $\mathcal{O}(L/k^2 + \varsigma_*^2 d/(\mu k))$ rate[3] for μ-strongly convex and L-smooth functions and $\mathcal{O}(L/k^2 + \varsigma_*^2 d/\sqrt{k})$ for convex L-smooth functions. These rates are nearly optimal, since in the deterministic limit $\varsigma_* \to 0$ they still capture acceleration.

Unlike [Xu et al., 2018a, Xu et al., 2018b], we focus on how the memory of past gradients relates to the *classical* and most widely used momentum methods (HB, NAG) and, with the help of the SDE models, show that the resulting insights can be used to design building blocks for new optimization methods.

3 MEMORY GRADIENT SDE

In his 1964 paper, Polyak motivated HB as the discrete time analogue of a second order ODE:

$$\ddot{X}(t) + a(t)X(t) + \nabla f(X(t)) = 0, \qquad \text{(HB-ODE)}$$

[3] ς_*^2 bounds the stochastic gradient variance in each direction.

which can be written in phase-space as

$$\begin{cases} \dot{V}(t) = -a(t)V(t) - \nabla f(X(t)) \\ \dot{X}(t) = V(t) \end{cases}. \qquad \text{(HB-ODE-PS)}$$

This connection can be made precise: in App. B.1 we show that HB is indeed the result of semi-implicit Euler integration[4] on HB-ODE-PS.

3.1 MEMORY AND GRADIENT FORGETTING

If the viscosity parameter $\alpha = a(t)$ is time-independent, HB-ODE, with initial condition $\dot{X}(0) = 0$ and $X(0) = x_0$, can be cast into an integro-differential equation[5]:

$$\dot{X}(t) = -\int_0^t e^{-\alpha\cdot(t-s)} \nabla f(X(s)) ds. \qquad \text{(HB-ODE-INT-C)}$$

Bias correction. Notice that the instantaneous update direction of HB-ODE-INT-C is a weighted average of the past gradients, namely $\int_0^t w(s,t)\nabla f(X(s))ds$ with $w(s,t) := e^{\alpha(t-s)}$. However, the weights do not integrate to one. Indeed, for all t, we have $\int_0^t w(s,t)ds = (1 - e^{-\alpha t})/\alpha$, which goes to $1/\alpha$ as $t \to \infty$. As a result, in the constant gradient setting, the previous sum is a *biased estimator* of the actual gradient. This fact suggests a simple modification of HB-ODE-INT-C, for $t > 0$:

$$\dot{X}(t) = -\frac{\alpha}{1 - e^{-\alpha t}} \int_0^t e^{-\alpha\cdot(t-s)} \nabla f(X(s)) ds. \quad (1)$$

which we write as $\dot{X} = -\frac{\alpha}{e^{\alpha t}-1}\int_0^t e^{\alpha s}\nabla f(X(s))ds$. We note that this normalization step follows exactly the same motivation as bias correction in Adam; we provide an overview of this method in App. B.2. If we define $\mathfrak{m}(t) := e^{\alpha t} - 1$, the previous formula takes the form:

[4] [Hairer et al., 2006] for an introduction.

[5] By computing $\ddot{X}$ from HB-ODE-INT-C using the fundamental theorem of calculus and plugging in $\dot{X}(0) = 0$.

$$\dot{X}(t) = -\int_0^t \frac{\dot{\mathfrak{m}}(s)}{\mathfrak{m}(t)} \nabla f(X(s)) ds. \tag{MG-ODE-INT}$$

This memory-gradient integro-differential equation (MG-ODE-INT) provides a generalization of HB-ODE-INT-C, with bias correction. The following crucial lemma is consequence of the fundamental theorem of calculus.

Lemma 3.1. *For any $\mathfrak{m} \in \mathcal{C}^1(\mathbb{R}, \mathbb{R})$ s.t. $\mathfrak{m}(0) = 0$, MG-ODE-INT is normalized : $\int_0^t \frac{\dot{\mathfrak{m}}(s)}{\mathfrak{m}(t)} ds = 1$, for all $t > 0$.*

Proof. Since $\mathfrak{m}(0) = 0$, $\int_0^t \dot{\mathfrak{m}}(s) ds = \mathfrak{m}(t)$. □

Based on Lemma 3.1, we will always set $\mathfrak{m}(0) = 0$. What other properties shall a general $\mathfrak{m}(\cdot)$ have? Requiring $\dot{\mathfrak{m}}(s) \neq 0$ for all $s \geq 0$ ensures that there does not exist a time instant where the gradient is systematically discarded. Hence, since $\mathfrak{m}(0) = 0$, $\mathfrak{m}(\cdot)$ is either monotonically decreasing and negative or monotonically increasing and positive. In the latter case, without loss of generality, we can flip its sign. This motivates the following definition.

Definition. $\mathfrak{m} \in \mathcal{C}^1(\mathbb{R}_+, \mathbb{R})$ is a **memory function** if it is non-negative, strictly increasing and s.t. $\mathfrak{m}(0) = 0$.

For example, $e^{\alpha t} - 1$, from which we started our discussion, is a valid memory function. Crucially, we note that $\dot{\mathfrak{m}}(\cdot)$ plays the important role of controlling the speed at which we *forget* previously observed gradients. For instance, let $\mathfrak{m}(t) = t^3$; since $\dot{\mathfrak{m}}(s) = 3s^2$, the system forgets past gradients *quadratically fast*. In contrast, $\mathfrak{m}(t) = e^{\alpha t} - 1$ leads to *exponential forgetting*. Some important memory functions are listed in Tb. 2, and their respective influence on past gradients is depicted in Fig. 1. We point out that, in the limit $\alpha \to \infty$, the weights $w(s,t) = \frac{\dot{\mathfrak{m}}(s)}{\mathfrak{m}(t)}$ associated with exponential forgetting converge to a Dirac distribution $\delta(t-s)$. Hence, we recover the Gradient Descent ODE [Mertikopoulos and Staudigl, 2018]: $\dot{X}(t) = -\nabla f(X(t))$. For the sake of comparability, we will refer to this as *instantaneous forgetting*.

Finally, notice that MG-ODE-INT can be written as a second order ODE. Too see this, we just need to compute the second derivative. For $t > 0$ we have that

$$\ddot{X}(t) = \frac{\dot{\mathfrak{m}}(t)}{\mathfrak{m}(t)^2} \int_0^t \dot{\mathfrak{m}}(s) \nabla f(X(s)) ds - \frac{\dot{\mathfrak{m}}(t)}{\mathfrak{m}(t)} \nabla f(X(t)).$$

Plugging in the definition of $\dot{X}$ from the integro-differential equation, we get the memory-gradient ODE:

$$\ddot{X}(t) + \frac{\dot{\mathfrak{m}}(t)}{\mathfrak{m}(t)} \dot{X}(t) + \frac{\dot{\mathfrak{m}}(t)}{\mathfrak{m}(t)} \nabla f(X(t)) = 0. \tag{MG-ODE}$$

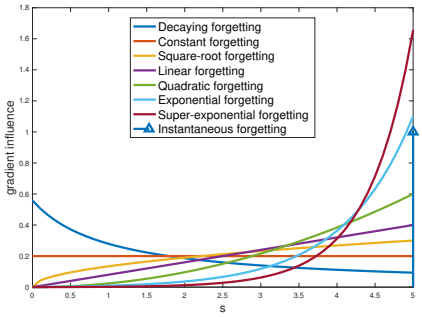

Figure 1: Illustration of the influence of past gradients on $\dot{X}(6)$ (i.e. the right hand side of equation MG-ODE-INT with $t = 5$). The corresponding memory function can be found in Tb. 2. The influence is computed as $\dot{\mathfrak{m}}(s)/\mathfrak{m}(6)$. By Lemma 3.1, the area under all curves is 1.

Forgetting	**Memory $\mathfrak{m}$**	**ODE Coeff. $\dot{\mathfrak{m}}/\mathfrak{m}$**
Decaying	$\log(1+t)$	$1/(t\log(t+1))$
Constant	t	$1/t$
Square-root	$t^{1.5}$	$1.5/t$
Linear	t^2	$2/t$
Quadratic	t^3	$3/t$
Exponential	$e^{\alpha t} - 1$	$\alpha e^{\alpha t}/\left(e^{\alpha t} - 1\right)$
Super-exp	$e^{t^\alpha} - 1$	$\alpha t^{\alpha-1} e^{t^\alpha}/\left(e^{t^\alpha} - 1\right)$
Instantaneous	$-$	$-$

Table 2: Some important examples of memory functions.

Equivalently, we can transform this second order ODE into a system of two first order ODEs by introducing the variable $V(t) := \dot{X}(t)$ and noting that $\dot{V}(t) = -\frac{\dot{\mathfrak{m}}(t)}{\mathfrak{m}(t)} V(t) - \frac{\dot{\mathfrak{m}}(t)}{\mathfrak{m}(t)} \nabla f(X(t))$. This is called the *phase-space representation* of MG-ODE, which we use in Sec. 3.2 to provide the extension to the stochastic setting. Also, for the sake of comparison with recent literature (e.g. [Wibisono et al., 2016]), we provide a variational interpretation of MG-ODE in App. C.2.

Existence and uniqueness. Readers familiar with ODE theory probably realized that, since by definition $\mathfrak{m}(0) = 0$, the question of existence and uniqueness of the solution to MG-ODE is not trivial. This is why we stressed its validity for $t > 0$ multiple times during the derivation. Indeed, it turns out that such a solution may not exist globally on $[0, \infty)$ (see App. C.1). Nevertheless, if we allow to start integration from *any* $\epsilon > 0$ and assume $f(\cdot)$ to be L-smooth, standard ODE theory [Khalil and Grizzle, 2002] ensures that the sought solution exists and is unique on $[\epsilon, \infty)$. Since ϵ can be made as small as we like (in our simulations in App. F we use $\epsilon = 10^{-16}$) this apparent issue can be regarded an artifact of the model. Also, we point out that the integral formulation MG-ODE-INT is well defined for every

$t > 0$. Therefore, in the theoretical part of this work, we act as if integration starts at 0 but we highlight in the appendix that choosing the initial condition $\epsilon > 0$ induces only a negligible difference (see Remarks C.2 and D.1).

3.2 INTRODUCING STOCHASTIC GRADIENTS

In this section we introduce stochasticity in the MG-ODE model. As already mentioned in the introduction, at each step k, iterative stochastic optimization methods have access to an estimate $\mathcal{G}(x_k)$ of $\nabla f(x_k)$: the so called *stochastic gradient*. This information is used and possibly combined with previous gradient estimates $\mathcal{G}(x_0), \dots, \mathcal{G}(x_{k-1})$, to compute a new approximation x_{k+1} to the solution x^*. There are many ways to design $\mathcal{G}(k)$: the simplest [Robbins and Monro, 1951] is to take $\mathcal{G}_{\text{MB}}(x_k) := \nabla f_{i_k}(x_k)$, where $i_k \in \{1, \dots, n\}$ is a uniformly sampled datapoint. This gradient estimator is trivially unbiased (conditioned on past iterates) and we denote its covariance matrix at point x by $\Sigma(x) = \frac{1}{n}\sum_{i=1}^{n}(\nabla f_i(x) - \nabla f(x))(\nabla f_i(x) - \nabla f(x))^T$.

Following [Krichene and Bartlett, 2017] we model such stochasticity adding a volatility term in MG-ODE.

$$\begin{cases} dX(t) = V(t)dt \\ dV(t) = -\frac{\dot{\mathfrak{m}}(t)}{\mathfrak{m}(t)}V(t)dt \\ \qquad\quad -\frac{\dot{\mathfrak{m}}(t)}{\mathfrak{m}(t)}\left[\nabla f(X(t))dt + \sigma(X(t))dB(t)\right] \end{cases} \tag{MG-SDE}$$

where $\sigma(X(t)) \in \mathbb{R}^{d\times d}$ and $\{B(t)\}_{t\geq 0}$ is a standard Brownian Motion. Notice that this system of equations reduces to the phase-space representation of MG-ODE if $\sigma(X(t))$ is the null matrix. The connection from $\sigma(x)$ to the gradient estimator covariance matrix $\Sigma(x)$ can be made precise: [Li et al., 2017] motivate the choice $\sigma(x) = \sqrt{h\Sigma(x)}$, where $\sqrt{\cdot}$ denotes the principal square root and h is the discretization stepsize.

The proof of existence and uniqueness to the solution of this SDE[6] relies on the same arguments made for MG-ODE in Sec. 3.1, with one additional crucial difference: [Orvieto and Lucchi, 2018] showed that $f(\cdot)$ needs to additionally be three times continuously differentiable with bounded third derivative (i.e. $f \in \mathcal{C}_b^3(\mathbb{R}^d, \mathbb{R})$), in order for $\sigma(\cdot)$ to be Lipschitz continuous. Hence, we will assume this regularity and refer the reader to [Orvieto and Lucchi, 2018] for further details.

3.3 THE CONNECTION TO NESTEROV'S SDE

[Su et al., 2016] showed that the continuous-time limit of NAG for convex functions is HB-ODE with time-dependent viscosity $3/t$: $\ddot{X}(t) + \frac{3}{t}\dot{X}(t) + \nabla f(X(t)) = 0$, which we refer to as *Nesterov's ODE*. Using Bessel functions, the authors were able to provide a new insightful description and analysis of this mysterious algorithm. In particular, they motivated how the vanishing viscosity is essential for acceleration[7]. Indeed, the solution to the equation above is s.t. $f(X(t)) - f(x^*) \leq \mathcal{O}(1/t^2)$; in contrast to the solution to the GD-ODE $\dot{X}(t) = -\nabla f(X(t))$, which only achieves a rate $\mathcal{O}(1/t)$.

A closer look at Tb. 2 reveals that the choice $3/t$ is related to MG-ODE with quadratic forgetting, that is $\ddot{X}(t) + \frac{3}{t}\dot{X}(t) + \frac{3}{t}\nabla f(X(t)) = 0$. However, it is necessary to note that in MG-SDE also the gradient term is premultiplied by $3/t$. Here we analyse the effects of this intriguing difference and its connection to acceleration.

Gradient amplification. A naïve way to speed up the convergence of the GD-ODE $\dot{X}(t) = -\nabla f(X(t))$ is to consider $\dot{X}(t) = -t\nabla f(X(t))$. This can be seen by means of the Lyapunov function $\mathcal{E}(x,t) = t^2(f(x) - f(x^*)) + \|x - x^*\|^2$. Using convexity of $f(\cdot)$, we have $\dot{\mathcal{E}}(X(t),t) = -t^2\|\nabla f(X(t))\|^2 \leq 0$ and therefore, the solution is s.t. $f(X(t)) - f(x^*) \leq \mathcal{O}(1/t^2)$. However, the Euler discretization of this ODE is the gradient-descent-like recursion $x_{k+1} = x_k - \eta k \nabla f(x_k)$ — which is *not* accelerated. Indeed, this *gradient amplification* by a factor of t is effectively changing the Lipschitz constant of the gradient field from L to kL. Therefore, each step is going to yield a descent only if[8] $\eta \leq \frac{1}{kL}$. Yet, this iteration dependent learning rate effectively cancels out the gradient amplification, which brings us back to the standard convergence rate $\mathcal{O}(1/k)$. It is thus natural to ask: "*Is the mechanism of acceleration behind Nesterov's ODE related to a similar gradient amplification?*"

In App. C.4 we show that $\{X_N(t), V_N(t)\}_{t\geq 0}$, the solution to Nesterov's SDE[9], is s.t. the infinitesimal update direction $V_N(t)$ of the position $X_N(t)$ can be written as

$$V_N(t) = -\int_0^t \frac{s^3}{t^3}\nabla f(X(s))ds + \zeta_N(t), \tag{2}$$

where $\zeta_N(t)$ is a random vector with $\mathbb{E}[\zeta_N(t)] = 0$ and $\text{Cov}[\zeta_N(t)] = \frac{1}{7}t\sigma\sigma^T$. In contrast, the solution $\{X_{\mathfrak{m}2}(t), V_{\mathfrak{m}2}(t)\})_{t\geq 0}$ of MG-SDE with quadratic forgetting satisfies

$$V_{\mathfrak{m}2}(t) = -\int_0^t \frac{3s^2}{t^3}\nabla f(X(s))ds + \zeta_{\mathfrak{m}2}(t), \tag{3}$$

[6] See e.g. Thm. 5.2.1 in [Øksendal, 2003], which gives sufficient conditions for (strong) existence and uniqueness.

[7] Acceleration is not achieved for a viscosity of e.g. $2/t$.

[8] See e.g. [Bottou et al., 2018].

[9] Nesterov's SDE is defined, as for MG-SDE by augmenting the phase space representation with a volatility term. The resulting system is then : $dX(t) = V(t)dt$; $dV(t) = -3/tV(t)dt - \sigma(X(t))dB(t)$.

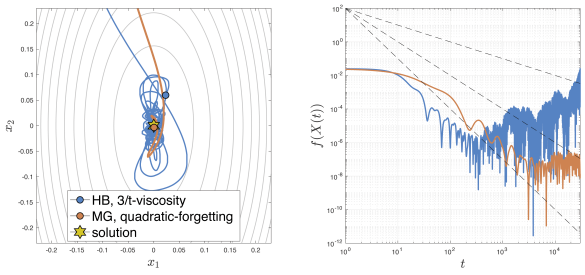

Figure 2: HB-SDE with $\alpha(t) = 3/t$ (i.e. Nesterov's SDE) compared to MG-SDE with quadratic forgetting. Setting as in [Su et al., 2016]: $f(x) = 2 \times 10^{-2} x_1^2 + 5 \times 10^{-3} x_2^2$ starting from $X_0 = (1, 1)$ and $\dot{X}(0) = (0, 0)$. Both systems are exposed to the same noise volatility. Simulation using the Milstein scheme [Mil'shtejn, 1975] with stepsize 10^{-3}.

where $\zeta_{\mathfrak{m}2}(t)$ is a random vector with $\mathbb{E}[\zeta_{\mathfrak{m}2}(t)] = 0$ but $\mathrm{Cov}[\zeta_{\mathfrak{m}2}(t)] = \frac{9}{5t}\sigma\sigma^T$. Even though the reader might already have spotted an important difference in the noise covariances, to make our connection to gradient amplification even clearer, we consider the simpler setting of constant gradients: in this case, we have $V_N(t) = -\frac{1}{4}t\nabla f(X(t)) + \zeta_N(t)$, $V_{\mathfrak{m}2}(t) = -\nabla f(X(t)) + \zeta_{\mathfrak{m}2}(t)$. That is, stochastic algorithms with increasing momentum (i.e. decreasing[10] viscosity, like the Nesterov's SDE) *are systematically amplifying the gradients over time*. Yet, at the same time they also linearly amplify the noise variance (see Fig. 7 in the appendix). This argument can easily be extended to the non-constant gradient case by noticing that $\mathbb{E}[V_{\mathfrak{m}2}(t)]$ is a weighted average of gradients where the weights integrate to 1 for all $t \geq 0$ (Lemma 3.1) . In contrast, in $\mathbb{E}[V_N(t)]$ these weights integrate to $t/4$. This behaviour is illustrated in Fig. 2: While the Nesterov's SDE is faster compared to MG-SDE with $\mathfrak{m}(t) = t^3$ at the beginning, it quickly becomes unstable because of the increasing noise in velocity and hence position.

This gives multiple insights on the behavior of Nesterov's accelerated method for convex functions, both in for deterministic and the stochastic gradients:

1. Deterministic gradients get linearly amplified overtime, which counteracts the slow-down induced by the vanishing gradient problem around the solution. Interestingly Eq. (2) reveals that this amplification is *not* performed directly on the local gradient but on past history, with *cubic* forgetting. It is this feature that makes the discretization stable compared to the naïve approach $\dot{X} = -t\nabla f(X(t))$.
2. Stochasticity corrupts the gradient amplification by an increasing noise variance (see Eq. (2)), which makes Nesterov's SDE unstable and hence not converging. This finding is in line with [Allen-Zhu, 2017].

[10] See the connection between α and β in Thm. B.1.

Furthermore, our analysis also gives an intuition as to why a constant momentum cannot yield acceleration. Indeed, we saw already that HB-ODE-INT-C does not allow such persistent amplification, but at most a constant amplification inversely proportional to the (constant) viscosity. Yet, as we are going to see in Sec. 4, this feature makes the algorithm more stable under stochastic gradients.

To conclude, we point the reader to App. C.4.2, where we extend the last discussion from the constant gradient case to the quadratic cost case and get a close form for the (exploding) covariance of Nesterov's SDE (which backs up theoretically the unstable behavior shown in Fig. 2). Nonetheless, we remind that this analysis still relies on continuous-time models; hence, the results above can only be considered as insights and further investigation is needed to translate them to the actual NAG algorithm.

Time warping of linear memory. Next, we now turn our attention to the following question: "*How is the gradient amplification mechanism of NAG related to its — notoriously wiggling*[11]*— path?*". Even though Nesterov's ODE and MG-ODE with quadratic forgetting are described by similar formulas, we see in Fig. 2 that the trajectories are very different, even when the gradients are large. The object of this paragraph is to show that Nesterov's path has a strong link to — surprisingly — linear forgetting. Consider speeding-up the linear forgetting ODE $\ddot{X}(t) + \frac{2}{t}\dot{X}(t) + \frac{2}{t}\nabla f(X(t))$ by introducing the time change $\tau(t) = t^2/8$ and let $Y(t) = X(\tau(t))$ be the *accelerated* solution to linear forgetting. By the chain rule, we have $\dot{Y}(t) = \dot{\tau}(t)\dot{X}(\tau(t))$ and $\ddot{Y}(t) = \ddot{\tau}(t)\dot{X}(\tau(t)) + \dot{\tau}(t)^2\ddot{X}(\tau(t))$. It can easily be verified that we recover $\ddot{Y}(t) + \frac{3}{t}\dot{Y}(t) + \nabla f(Y(t))$. However, in the stochastic setting, the behaviour is still quite different: as predicted by the theory, in Fig. 3 we see that — when gradients are large — the trajectory of the two sample paths are almost identical[11] (yet, notice that Nesterov moves faster); however, as we approach the solution, Nesterov diverges while linear forgetting stably proceeds towards the minimizer along the Nesterov's ODE path, but at a different speed, until convergence to a neighborhood of the solution, as proved in Sec. 4. Furthermore, in App. C.3 we prove that there are no other time changes which can cast MG-ODE into HB-ODE, which yields the following interesting conclusion: the *only* way to translate a memory system into a momentum method is by using a time change $\tau(t) = \mathcal{O}(t^2)$.

4 ANALYSIS AND DISCRETIZATION

In this section we first analyze the convergence properties of MG-SDE under different memory functions. Next,

[11] Detailed simulations in App. F.

Forgetting	Assumption	Rate	Reference
Instantaneous	**(H0c)**, **(H1)**	$\mathbb{E}[f(\bar{X}(t)) - f(x^*)] \leq C_i/t + \ d\,\sigma_*^2/2$	[Mertikopoulos and Staudigl, 2018]
Exponential	**(H0c)**, **(H1)**	$\mathbb{E}[f(\bar{X}(t)) - f(x^*)] \leq C_e/t + \ d\,\sigma_*^2/2$	App. D, Thm. D.4
Polynomial	**(H0c)**, **(H1)**, $p \geq 2$	$\mathbb{E}[f(X(t)) - f(x^*)] \leq C_p/t + p\,d\,\sigma_*^2/2$	App. D, Thm. D.3

Table 3: Rates of MG-SDE on convex smooth functions . $\bar{X}(t) = \int_0^t X(s)ds$ and C_i, C_e, C_p can be found in the references.

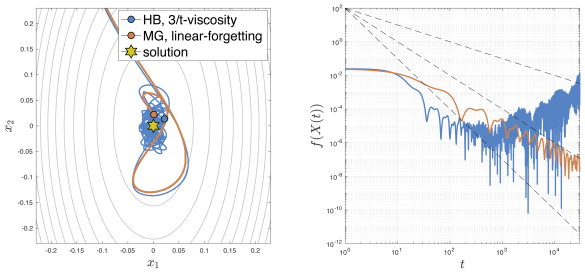

Figure 3: Nesterov's ODE compared to MG-SDE with linear forgetting (i.e. $\dot{\mathfrak{m}}(t)/\mathfrak{m}(t) = 2/t$). Same settings as Fig. 2.

we use the Lyapunov analysis carried out in continuous-time to derive an iterative discrete-time method which implements polynomial forgetting and has provable convergence guarantees. We state a few assumptions:

(H0c) $f \in \mathcal{C}_b^3(\mathbb{R}^d, \mathbb{R})$, $\sigma_*^2 := \sup_x \|\sigma(x)\sigma(x)^T\|_s < \infty$.

The definition of σ_*^2 nicely decouples the measure of noise magnitude to the problem dimension d (which will then, of course, appear explicitly in all our rates).

(H1) The cost $f(\cdot)$ is L-smooth and convex.

(H2) The cost $f(\cdot)$ is L-smooth and μ-strongly convex.

We provide the proofs (under the less restrictive assumptions of weak-quasi-convexity and quadratic growth[12]), as well as an introduction to stochastic calculus, in App. D.

4.1 EXPONENTIAL FORGETTING

If $\mathfrak{m}(t) = e^{\alpha t} - 1$, then $\dot{\mathfrak{m}}(t)/\mathfrak{m}(t) = \frac{\alpha e^{\alpha t}}{e^{\alpha t}-1}$ which converges to α exponentially fast. To simplify the analysis and for comparison with the literature on HB-SDE (which is usually analyzed under constant volatility [Shi et al., 2018]) we consider here MG-SDE with the approximation $\dot{\mathfrak{m}}(t)/\mathfrak{m}(t) \simeq \alpha$. In App. D we show that, under **(H1)**, the rate of convergence of $f(\cdot)$, evaluated at the Cesàro average $\bar{X}(t) = \int_0^t X(s)ds$ is sublinear (see Tb. 3) to a ball[13] around x^* of size $d\,\sigma_*^2/2$, which is in line with known results for SGD [Bottou et al., 2018][14]. Note that the size of this ball would change if we were to study a stochastic version of HB-ODE with constant volatility (i.e. $\ddot{X} + \alpha\dot{X} + \nabla f(X)$). In particular, it would depend on the normalization constant in Eq. (1). Also, in App. D, under **(H2)**, we provide a *linear* convergence rate of the form $f(X(t)) - f(x^*) \leq \mathcal{O}(e^{-\gamma t})$ to a ball (γ depends on μ and α). Our result generalizes the analysis in [Shi et al., 2018] to work with *any viscosity* and with stochastic gradients.

Discretization. As shown in Sec. 3.1, the discrete equivalent of MG-SDE with exponential forgetting is Adam without adaptive stepsizes (see App. B.2). As for the continuous-time model we just studied, for a sufficiently large iteration, exponential forgetting can be approximated with the following recursive formula:

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

which is exactly HB with learning rate $(1-\beta)\eta$. Hence, the corresponding rates can be derived from Tb. 1.

4.2 POLYNOMIAL FORGETTING

The insights revealed in Sec. 3.3 highlight the importance of the choice $\mathfrak{m}(t) = t^p$ in this paper. In contrast to instantaneous [Mertikopoulos and Staudigl, 2018] and exponential forgetting, the rate we prove in App. D for this case under **(H1)** does not involve a Cesàro average — *but holds for the last time point* (see Tb. 3). This stability property is directly linked to our discussion in Sec. 3.3 and shows that different types of memory may react to noise very differently. Also, we note that the size of the ball we found is now also proportional to p; this is not surprising since, as the memory becomes more focused on recent past, we get back to the discussion in the previous subsection and we need to consider a Cesàro average.

Discretization. Finally, we consider the burning question "*Is it possible to discretize MG-SDE — with polyno-*

[12] τ-weak-quasiconvexity is implied by convexity and has been shown to be of chief importance in the context of learning dynamical systems [Hardt et al., 2018]. Strong convexity implies quadratic growth with a unique minimizer [Karimi et al., 2016] as well as τ-weak-quasiconvexity. More details in the appendix.

[13] Note that the term "ball" might be misleading: indeed, the set $\mathcal{N}_\epsilon(x^*) = \{x \in \mathbb{R}^d, f(x) - f(x^*) \leq \epsilon\}$ is not compact in general if $f(\cdot)$ is convex but not strongly convex.

[14] Note that, by definition of $\sigma(\cdot)$ (see discussion after MG-SDE), σ_*^2 is proportional both to the learning rate and to the largest eigenvalue of the stochastic gradient covariance

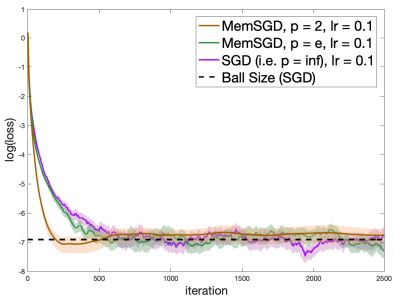

Figure 4: Synthetic example: $f(x_1, x_2) = 0.8 \times x_1^4 + 0.4 \times x_2^4$ with Gaussian noise. Displayed is linear forgetting (i.e. MemSGD-2), exponential forgetting (denoted p=e) with $\beta = 0.8$ and instantaneous forgetting. Average and 95% confidence interval for 150 runs starting from $(1, 1)$.

mial forgetting — to derive a cheap iterative algorithm with similar properties?". In App. E, we build this algorithm in a non-standard way: we reverse-engineer the proof of the rate for MG-SDE to get a method which is able to mimic each step of the proof. Starting from $x_{-1} = x_0$, it is described by the following recursion

$$x_{k+1} = x_k + \frac{k}{k+p}(x_k - x_{k-1}) - \frac{p}{k+p}\eta \nabla f(x_k). \quad \text{(MemSGD-p)}$$

As a direct result of our derivation, we show in Thm. E.2 (App. E) that *this algorithm preserves exactly the rate* of its continuous-time model in the stochastic setting[15]:

$$\mathbb{E}[f(x_k) - f(x^*)] \leq \frac{(p-1)^2 \|x_0 - x^*\|^2}{2\eta p(k+p-1)} + \frac{1}{2} p d \eta \varsigma_*^2.$$

We also show that MemSGD-p can be written as $x_{k+1} = x_k - \eta \sum_{j=0}^k w(j,k) \nabla f(x_k)$, where $\sum_{j=0}^k w(j,k) = 1$ (in analogy to the bias correction in Adam) and with $w(\cdot, k)$ increasing as a polynomial of order $p-1$ for all k, again in complete analogy with the model. Fig. 4 shows the behaviour of different types of memory in a simple convex setting; as predicted, polynomial (in this case linear) forgetting has a much smoother trajectory then both exponential and instantaneous forgetting. Also, the reader can verify that the asymptotic noise level for MemSGD (p=2) is slightly higher, as just discussed.

For ease of comparison with polynomial memory, we will often write MemSGD (p=e) to denote exponential forgetting (i.e. Adam without adaptive steps) and SGD (p=inf) to stress that SGD implements instantaneous forgetting.

[15]Required assumptions: **(H1)**, $p \geq 2$, $\eta \leq \frac{p-1}{pL}$ and ς_*^2 bounds the gradient variance in each direction of $\mathbb{R}^d$.

5 LARGE SCALE EXPERIMENTS

In order to assess the effect of different types of memory in practical settings, we benchmark MemSGD with different memory functions: from instantaneous to exponential, including various types of polynomial forgetting. As a reference point, we also run vanilla HB with constant momentum as stated in the introduction. To get a broad overview of the performance of each method, we run experiments on a convex logistic regression loss as well as on non-convex neural networks in both a mini- and full-batch setting. Details regarding algorithms, datasets and architectures can be found in App. G.1.

Results and discussion. Fig. 5 summarizes our results in terms of training loss. While it becomes evident that no method is best on all problems, we can nevertheless draw some interesting conclusions.

First, we observe that while long-term memory (especially $p = 2$) is faster than SGD in the convex case, it does not provide any empirical gain in the neural network settings. This is not particularly surprising since past gradients may quickly become outdated in non-convex landscapes. Short term memory is at least as good as SGD in all cases except for the CIFAR-10 CNN, which represents the most complex of our loss landscapes in terms of curvature.

Secondly, we find that the best stepsize for HB is always strictly smaller than the one for SGD in the non-convex setting. MemSGD, on the other hand, can run on stepsizes as large as SGD which reflects the gradient amplification of HB as well as the unbiasedness of MemSGD. Interestingly, however, a closer look at Fig. 15 (appendix) reveals that HB (with best stepsize) actually takes much smaller steps than SGD for almost all iterations. While this makes sense from the perspective that memory averages past gradients, it is somewhat counter-intuitive given the inertia interpretation of HB which should make the method travel further than SGD. Indeed, both [Sutskever et al., 2013] and [Goodfellow et al., 2016] attribute the effectiveness of HB to its increased velocity along consistent directions (especially early on in the optimization process). However, our observation, together with the fact that MemSGD with fast forgetting ($p = e$ and $p = 100$) is as good as HB, suggests that there is actually more to the success of taking past gradients into account and that this must lie in the altered directions that adapt better to the underlying geometry of the problem.[16]

Finally, we draw two conclusions that arise when comparing the mini- and full batch setting. First, the superiority of HB and fast forgetting MemSGD over vanilla SGD in the deterministic setting is indeed reduced as soon as

[16]Note that we find the exact opposite in the convex case, where HB does take bigger steps and converges faster.

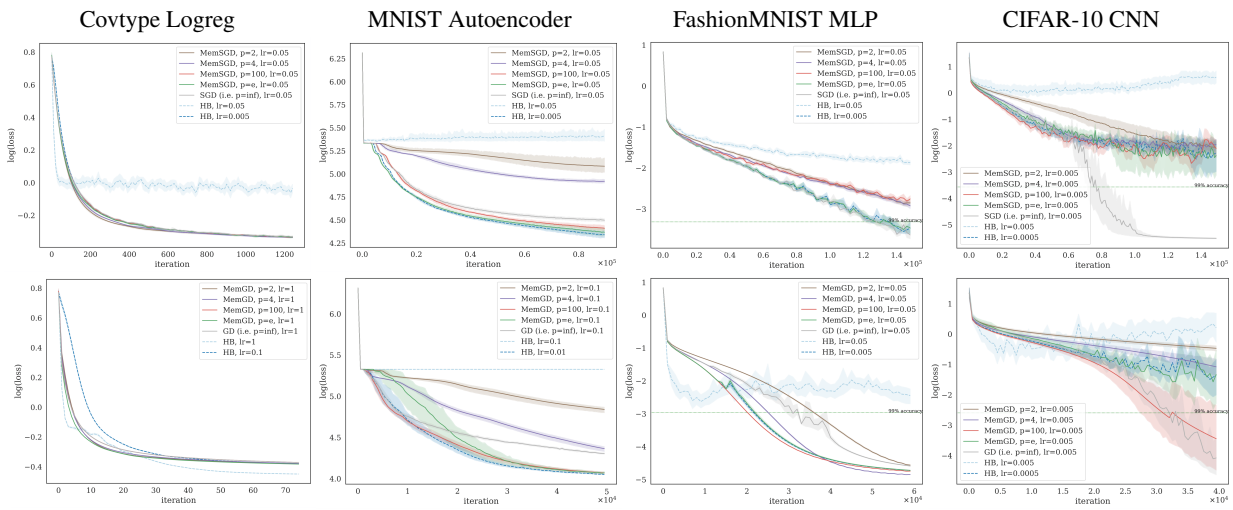

Figure 5: Log loss over iterations in mini- (top) and full-batch (bottom) setting. Average and 95% CI of 10 random initializations.

stochastic gradients come into play (this is in line with the discussion in Sec. 3.3). Second, we find that stochasticity per se is not needed to optimize the neural networks in the sense that all methods eventually reach very similar methods of suboptimality. That is, not even the full batch methods get stuck in any elevated local minima including the saddle found in the MNIST autoencoder which they nicely escape (given the right stepsize).

6 MEMORY IN ADAPTIVE METHODS

While the main focus of this paper is the study of the effect of different types of memory on the *first moment* of the gradients, past gradient information is also commonly used to adapt stepsizes. This is the case for Adagrad and Adam which both make use of the second moment of past gradients to precondition their respective update steps.

Of course, the use of polynomial memory generalizes directly to the second moment estimates and we thus consider a comprehensive study of the effect of long- versus short-term memory in adaptive preconditioning an exciting direction of future research. In fact, as shown in [Reddi et al., 2018] the non-convergence issue of Adam can be fixed by making the method forget past gradients less quickly. For that purpose the authors propose an algorithm called AdamNC that essentially differs from Adam by the choice of $\beta_2 = 1 - 1/k$, which closely resembles Adagrad with constant memory. Interestingly, the memory framework introduced in this paper allows to interpolate between the two extremes of constant- and exponential memory (i.e. Adagrad and Adam) in a principled way. Indeed, by tuning the additional parameter p — which specifies the degree of the polynomial memory function — one can equip Adam with any degree of short- to long-term memory desired. As a proof of concept, Fig. 6 shows that Adam equipped with a polynomial memory of the squared gradients (PolyAdam) can in fact be faster than both Adam and Adagrad.

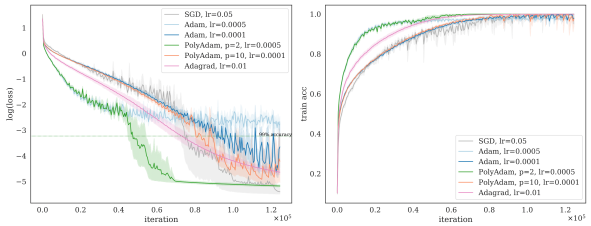

Figure 6: Cifar-10 CNN: Log loss over iterations (left) and training accuracy (right). Average and 95% confidence interval of 10 runs with random initialization.

7 CONCLUSION

We undertook an extensive theoretical study of the role of memory in (stochastic) optimization. We provided convergence guarantees for memory systems as well as for novel algorithms based on such systems. This study led us to derive novel insights on momentum methods. We complemented these findings with empirical results, both on simple functions as well as more complex functions based on neural networks. There, long- and short-term memory methods exhibit a different behaviour, which suggests further investigation is needed to better understand the interplay between the geometry of neural networks losses, memory and gradient stochasticity. On a more theoretical side, an interesting direction of future work is the study of the role of memory in state-of-the art momentum methods such as algorithms that include primal averaging, increasing gradient sensitivity or decreasing learning rates (see e.g. [Krichene and Bartlett, 2017]).

References

[Allen-Zhu, 2017] Allen-Zhu, Z. (2017). Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1200–1205. ACM.

[Arnol'd, 2013] Arnol'd, V. I. (2013). *Mathematical methods of classical mechanics*, volume 60. Springer Science & Business Media.

[Betancourt et al., 2018] Betancourt, M., Jordan, M. I., and Wilson, A. C. (2018). On symplectic optimization. *arXiv preprint arXiv:1802.03653*.

[Bottou et al., 2018] Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311.

[Cabot et al., 2009] Cabot, A., Engler, H., and Gadat, S. (2009). On the long time behavior of second order differential equations with asymptotically small dissipation. *Transactions of the American Mathematical Society*, 361:5983–6017.

[Cauchy, 1847] Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

[Gadat and Panloup, 2014] Gadat, S. and Panloup, F. (2014). Long time behaviour and stationary regime of memory gradient diffusions. In *Annales de l'IHP Probabilités et statistiques*, volume 50, pages 564–601.

[Ghadimi et al., 2015] Ghadimi, E., Feyzmahdavian, H. R., and Johansson, M. (2015). Global convergence of the heavy-ball method for convex optimization. In *Control Conference (ECC), 2015 European*, pages 310–315. IEEE.

[Ghadimi and Lan, 2012] Ghadimi, S. and Lan, G. (2012). Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. *SIAM Journal on Optimization*, 22(4):1469–1492.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Hairer et al., 2006] Hairer, E., Lubich, C., and Wanner, G. (2006). *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media.

[Hardt et al., 2018] Hardt, M., Ma, T., and Recht, B. (2018). Gradient descent learns linear dynamical systems. *The Journal of Machine Learning Research*, 19(1):1025–1068.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

[Jain et al., 2018] Jain, P., Kakade, S. M., Kidambi, R., Netrapalli, P., and Sidford, A. (2018). Accelerating stochastic gradient descent for least squares regression. In *Conference On Learning Theory*, pages 545–604.

[Karimi et al., 2016] Karimi, H., Nutini, J., and Schmidt, M. (2016). 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.

[Khalil and Grizzle, 2002] Khalil, H. K. and Grizzle, J. (2002). *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ.

[Kidambi et al., 2018] Kidambi, R., Netrapalli, P., Jain, P., and Kakade, S. (2018). On the insufficiency of existing momentum schemes for stochastic optimization. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–9. IEEE.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Krichene and Bartlett, 2017] Krichene, W. and Bartlett, P. L. (2017). Acceleration and averaging in stochastic descent dynamics. In *Advances in Neural Information Processing Systems*, pages 6796–6806.

[Krichene et al., 2015] Krichene, W., Bayen, A., and Bartlett, P. L. (2015). Accelerated mirror descent in continuous and discrete time. In *Advances in neural information processing systems*, pages 2845–2853.

[Lan, 2012] Lan, G. (2012). An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1-2):365–397.

[Li et al., 2017] Li, Q., Tai, C., et al. (2017). Stochastic modified equations and adaptive stochastic gradient algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2101–2110. JMLR. org.

[Ma and Yarats, 2018] Ma, J. and Yarats, D. (2018). Quasi-hyperbolic momentum and adam for deep learning. *arXiv preprint arXiv:1810.06801*.

[Mao, 2007] Mao, X. (2007). *Stochastic differential equations and applications*. Elsevier.

[Mertikopoulos and Staudigl, 2018] Mertikopoulos, P. and Staudigl, M. (2018). On the convergence of gradient-like flows with noisy gradient input. *SIAM Journal on Optimization*, 28(1):163–197.

[Mil'shtejn, 1975] Mil'shtejn, G. (1975). Approximate integration of stochastic differential equations. *Theory of Probability & Its Applications*, 19(3):557–562.

[Nesterov, 2018] Nesterov, Y. (2018). *Lectures on convex optimization*. Springer.

[Nesterov, 1983] Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547.

[Øksendal, 2003] Øksendal, B. (2003). Stochastic differential equations. In *Stochastic differential equations*, pages 65–84. Springer.

[Orvieto and Lucchi, 2018] Orvieto, A. and Lucchi, A. (2018). Continuous-time models for stochastic optimization algorithms. *arXiv preprint arXiv:1810.02565*.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

[Polyak, 1964] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

[Polyak, 1987] Polyak, B. T. (1987). Introduction to optimization. optimization software. *Inc., Publications Division, New York*, 1.

[Reddi et al., 2018] Reddi, S., Kale, S., and Kumar, S. (2018). On the convergence of adam and beyond. In *International Conference on Learning Representations*.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[Shi et al., 2018] Shi, B., Du, S. S., Jordan, M. I., and Su, W. J. (2018). Understanding the acceleration phenomenon via high-resolution differential equations. *arXiv preprint arXiv:1810.08907*.

[Shi et al., 2019] Shi, B., Du, S. S., Su, W. J., and Jordan, M. I. (2019). Acceleration via symplectic discretization of high-resolution differential equations. *arXiv preprint arXiv:1902.03694*.

[Su et al., 2016] Su, W., Boyd, S., and Candes, E. J. (2016). A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. *Journal of Machine Learning Research*, 17:1–43.

[Sutskever et al., 2013] Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR.

[Wibisono et al., 2016] Wibisono, A., Wilson, A. C., and Jordan, M. I. (2016). A variational perspective on accelerated methods in optimization. *proceedings of the National Academy of Sciences*, 113(47):E7351–E7358.

[Wilson et al., 2016] Wilson, A. C., Recht, B., and Jordan, M. I. (2016). A lyapunov analysis of momentum methods in optimization. *arXiv preprint arXiv:1611.02635*.

[Xu et al., 2018a] Xu, P., Wang, T., and Gu, Q. (2018a). Accelerated stochastic mirror descent: From continuous-time dynamics to discrete-time algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 1087–1096.

[Xu et al., 2018b] Xu, P., Wang, T., and Gu, Q. (2018b). Continuous and discrete-time accelerated stochastic mirror descent for strongly convex functions. In *International Conference on Machine Learning*, pages 5488–5497.

[Yang et al., 2018] Yang, L., Arora, R., Zhao, T., et al. (2018). The physical systems behind optimization algorithms. In *Advances in Neural Information Processing Systems*, pages 4377–4386.

[Yang et al., 2016] Yang, T., Lin, Q., and Li, Z. (2016). Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *arXiv preprint arXiv:1604.03257*.

[Yuan et al., 2016] Yuan, K., Ying, B., and Sayed, A. H. (2016). On the influence of momentum acceleration on online learning. *The Journal of Machine Learning Research*, 17(1):6602–6667.

[Zhang et al., 2018] Zhang, J., Mokhtari, A., Sra, S., and Jadbabaie, A. (2018). Direct runge-kutta discretization achieves acceleration. In *Advances in Neural Information Processing Systems*, pages 3904–3913.